Intelligent Application Gateway

Advanced Configuration

December 2006

Version 3.7

© 2006 Whale Communications, a Microsoft subsidiary. All rights reserved.

This manual and the information contained herein are confidential and proprietary to Whale Communications, a Microsoft subsidiary, its affiliates and subsidiaries (hereinafter, the "Company"). All intellectual property rights (including, without limitation, copyrights, trade secrets, trademarks, etc.) evidenced by or embodied in and/or attached/connected/related to this manual, information contained herein and the Product, is and shall be owned solely by the Company. The Company does not convey to you an interest in or to this manual, information contained herein and the Product, but only a limited right of use. Any unauthorized use, disclosure or reproduction is a violation of the licenses and/or the Company's proprietary rights and will be prosecuted to the full extent of the Law.

TRADEMARKS

Application Aware, and Attachment Wiper are service marks, trademarks or registered trademarks of Whale Communications or its subsidiaries in the United States and other countries, or both.
Netscape, and Netscape Navigator are service marks, trademarks or registered trademarks of America Online, Inc. or its subsidiaries in the United States and other countries, or both.
Carbon, Macintosh, Mac OS, and Safari are service marks, trademarks or registered trademarks of Apple Computer, Inc. or its subsidiaries in the United States and other countries, or both.
Citrix , Citrix NFuse, Citrix Presentation Server, Citrix MetaFrame, Citrix SecureGateway, and ICA, are service marks, trademarks or registered trademarks of Citrix Systems, Inc. or its subsidiaries in the United States and other countries, or both.
Debian is a service mark, trademark or registered trademark of Software in the Public Interest, Inc. or its subsidiaries in the United States and other countries, or both.
GNU, and GZip are service marks, trademarks or registered trademarks of Free Software Foundation, Inc. or its subsidiaries in the United States and other countries, or both.
Domino, Lotus, IBM Lotus, iNotes, Lotus iNotes, Lotus Domino, Notes, Sametime, and WebSphere are service marks, trademarks or registered trademarks of IBM Corporation or its subsidiaries in the United States and other countries, or both.
Linux is a service mark, trademark or registered trademark of Linus Torvalds or its subsidiaries in the United States and other countries, or both.
Active Directory, ActiveSync, ActiveX, Excel, Microsoft, Outlook, SharePoint, Visual Basic, Windows Mobile, Windows NT, Windows Server are service marks, trademarks or registered trademarks of Microsoft Corporation or its subsidiaries in the United States and other countries, or both.
Camino, Firefox and Mozilla are service marks, trademarks or registered trademarks of Mozilla Foundation or its subsidiaries in the United States and other countries, or both.
Novell, Novell Directory Services, Novell NetWare, and SUSE are service marks, trademarks or registered trademarks of Novell, Inc. or its subsidiaries in the United States and other countries, or both.
PGP is a service mark, trademark or registered trademark of PGP Corporation or its subsidiaries in the United States and other countries, or both.
Red Hat is a service mark, trademark or registered trademark of Red Hat, Inc. or its subsidiaries in the United States and other countries, or both.
Resonate is a registered trademark of Resonate, Inc. The Resonate logo and Resonate Central Dispatch are trademarks of Resonate, Inc. Resonate Central Dispatch contains technology protected under U.S. Patent 5,774,660.
ACE SecurID, RC4, and RSA SecurID, are service marks, trademarks or registered trademarks of RSA Security Inc. or its subsidiaries in the United States and other countries, or both.

SAP is a service mark, trademark or registered trademark of SAP AG or its subsidiaries in the United States and other countries, or both.
Java, JavaScript, JRE, and Sun are service marks, trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the United States and other countries, or both. Enhanced HAT owned by Sun Microsystems, Inc.
Norton and Symantec are service marks, trademarks or registered trademarks of Symantec Corporation or its subsidiaries in the United States and other countries, or both.
Apache is a service mark, trademark or registered trademark of The Apache Software Foundation or its subsidiaries in the United States and other countries, or both.
Terminal Services is a service mark, trademark or registered trademark of The Regents of the University of California or its subsidiaries in the United States and other countries, or both.
Unix is a service mark, trademark or registered trademark of The Open Group or its subsidiaries in the United States and other countries, or both.
XCompress is a service mark, trademark or registered trademark of XCache Technologies, Inc. or its subsidiaries in the United States and other countries, or both.

All other trademarks, copyrights, product and or service marks mentioned in this manual, whether claimed or registered, are the exclusive property of their respective owners.

DISCLAIMER

The Company has reviewed this manual thoroughly. All statements, technical information, and recommendations in this manual and in any guides or related documents are believed reliable, but the accuracy and completeness thereof are not guaranteed or warranted, and they are not intended to be, nor should they be understood to be, representations or warranties concerning the products described. Further, the Company reserves the right to make changes to the information described in this manual at any time without notice and without obligation to notify any person of such changes.

LIMITATION OF LIABILITY

Neither the Company nor any of its worldwide subsidiaries or distributors or management or employees grants any warranties in respect to any damages or deficiencies resulting from accident, alteration, modification, foreign attachments, misuse, tampering, negligence, improper maintenance, abuse or failure to implement any updates furnished. The Products must be used and maintained in strict compliance with the instructions and safety precautions of the Company contained herein in all supplements thereto or in any other written documents of the Company. The products must not be altered without prior written consent of the Company.

The Company grants no warranties with respect to the Products, either express or implied, including any implied warranties of merchantability or fitness for a particular purpose. The Company will have no liability for any damages whatsoever arising out of or in connection with the delivery, installation, use or performance of the product. In no event shall the Company be liable under any legal theory (including but not limited to contract, negligence, misrepresentation, strict liability in tort or warranty of any kind) for any indirect, special, incidental or consequential damages (including but not limited to loss of profits), even if the Company has notice of the possibility of damages.

Without limiting the effect of the preceding clauses, the Company's maximum liability, if any, for damages (including but not limited to liability arising out of contract, negligence, misrepresentation, strict liability in tort or warranty of any kind) shall not exceed the consideration paid to the Company for the product. The Company shall under no circumstances be liable for damages arising out of any claim (including but not limited to a claim for personal injury or property damage) made by any third person or party.

Document Name	Intelligent Application Gateway Advanced Configuration
Document Revision	3.7
Date	December 2006
Software Version No.	3.7

Contents

Chapter 1: Introduction 15

About This Guide 15

Conventions Used in This Guide 16

Chapter 2: Security Management Tools 19

Overview 19

Creating Encryption Keys 20

Service Policy Manager 22

Accessing the Service Policy Manager 23

Service Policy Manager Window Layout 24

List Pane 25

Toolbar 25

Creating Lists of IP Addresses and Port Numbers 26

Changing the Login Passwords 28

Protecting Web Servers on the IAG 29

Configuration Program 31

Configuration Program 32

Accessing the Configuration Program 32

Configuration Program Layout 33

Configuration Program Toolbar 34

The Create New Trunk Wizard 35

Configuration Files 35

Creating a New Configuration File 35

Opening an Existing Configuration File 35

Configuration Projects 36

Exporting a Configuration Project 36

Importing a Configuration Project 37

Changing the Port Numbers of the IIS Default Web Site 39
SSL Configuration 39
Editor 40
Layout of the Editor 41
Editing a File with the Editor 42
Editing Tips 43
Finding the Application-Type 44
Configuration Messages 44
Messages Filter 45
Chapter 3: Customizing Web Pages 47
Before You Begin 48
Customizing General User Interaction Pages 49
Content Changes in General User Interaction Pages 51
Portal Homepage Configuration 54
Customizing the Default Portal Homepage 54
Content Changes in the Default Portal Homepage 57
Using a Custom Portal Homepage 61
Adding Application Links on a Custom Portal Homepage 63
Defining Scheduled Logoff and Session Timeout on a Custom Homepage 65
Adding Links to IAG Features on a Custom Homepage 66
Customizing Certified Endpoint Enrollment Pages 67
Content Changes in Certified Endpoint Enrollment Pages 70
Changing File Access Language Definitions 71
Customizing the Web Monitor Windows 72
Customizing Web Monitor Preferences 74
Customizing Web Monitor Charts 75
Multi-Language Support 77
Editing Out-Of-The-Box Languages 78
Configuring Support for Additional Languages 78

General Guidelines 79
Strings That Include Special Characters 79
Application-Specific Strings 80

Chapter 4: Access Control 81

Authentication 81
Configuration in the Authentication Tab 82
Configuration of the Authentication Servers 89
Defining a New Server Type 90
Credentials Management by Users 91
Adding Authentication Credentials 92
Changing the Password 93
Change Password Requirements 93
Configuring Cross-Site Single Sign On 94
The Authentication Process 95
Authentication Pages 96
Adding Functionality to the Default Pages 99
Retrieving the Source IP From the Authentication Pages 101
Using Custom Include Pages 101
Using Your Own Authentication Pages 103
Authentication Schemes 104
LDAP Authentication Schemes 105
Overview 105
LDAP Authentication Flow 105
Configuring the LDAP Authentication Server or Servers 106
LDAP Client Certificate Authentication Scheme 107
Overview 107
LDAP Client Certificate Authentication Flow 108
Configuration at the IAG 108
Configuring the LDAP Authentication Server or Servers 110
Preparing the Endpoint Browser 110

NT Domain Authentication Scheme 111
Overview 111
NT Domain Authentication Flow 111
RADIUS Authentication Scheme 112
Overview 112
Secret Key 112
Challenge-Response Modes 112
RADIUS Groups 112
RADIUS Authentication Flow 112
Configuring the RADIUS Authentication Server 114
RSA SecurID Authentication Scheme 114
Overview 114
Next Token Mode 114
New PIN Mode 115
RSA SecurID Authentication Flow 115
Configuring the RSA ACE Authentication Server 117
TACACS Authentication Scheme 118
Overview 118
TACACS Authentication Flow 118
Configuring the TACACS Authentication Server 119
WINHTTP Authentication Scheme 120
Overview 120
WINHTTP Authentication Flow 120
User-Defined Authentication Scheme 121
Authentication Scheme Functions 122
Authentication Scheme Object Fields 122
Server Name Translation 125
Creating and Editing Translation Rules 126
Initial Host Selection 128
Configuring Host Selection in the Authentication Pages 128
Configuring Initial Host Selection in the Server Name Translation Tab 130
Configuring Initial Host Selection for OWA 2000/2003 Applications 131

Chapter 5: Session Settings 133

Session Configuration 133

Default and Privileged Session Settings 137

Chapter 6: Content Inspection 141

Configuring URL Inspection Rules 143

Configuration in the URL Inspection Tab 143

URL Inspection Tab—Valid Methods..... 144

URL Inspection Tab—URL Set Level..... 146

URL Inspection Tab—General Options 146

URL Inspection Tab—Out-Of-The-Box Security Configuration..... 147

Configuration in the Global URL Settings Tab 148

Global URL Settings Tab—Global Parameter List 149

Global URL Settings Tab—Rejected Values..... 152

Global URL Settings Tab—URL Settings 152

Configuring a Ruleset in the URL Set Tab 164

URL Set Tab—URL List..... 165

URL Set Tab—Parameter List 168

URL Set Tab—Unlisted Parameters Area 169

Creating, Editing, and Removing Rules..... 170

Using Variables in URLs 173

Creating a Ruleset 174

Configuring the Use of URL Variables in the Authentication Pages 175

Important Notes 176

Limitations 176

Repetitions 176

Rule Creating Utility 177

Creating Rulesets with the Utility 178

Recording Rules..... 178

Optimizing the Ruleset 181

Exporting and Importing Rulesets 187
Exporting Rulesets 188
Importing Rulesets 189
Checking HTTP Headers 190
The Flow of HTTP Header Checks 191
Blocking the "Negotiate" Authorization Header 193
Configuring and Activating Additional Header Checks 193
HTTP Header Checks Configuration File 193
General Parameters..... 194
Header Parameters 195
Cookie Parameters..... 198
Support of Non-Compliant HTTP Applications 201
Irregular Separation of Header and Data 201
Irregular HTTP Data Chunking 202
Irregular HTTP Header Content-Type 202
Allowing Characters After Escaped Characters in URL Parameters 203

Chapter 7: Application Customizers..... 205

Using Frames 206
Sample Implementations 206
Session Timeout Warning 207
Replacement of the Server Value in Response Headers 207
Customizing Your Web Applications 207
Application Customization Tab 208
Global Content-Type List 210
Handling Compression in Responses 212
Configuring the URL Extension List..... 213
GZip Compression 214
Advanced Configuration of the Templates 215
Functionality of the Templates 215

Microsoft Outlook Web Access 5.5 215
Microsoft Outlook Web Access 2000 SP2/SP3 215
Microsoft Outlook Web Access 2000 POST SP3 216
Microsoft Outlook Web Access 2003 216
Microsoft Outlook Web Access 2003 SP1 216
Domino Webmail 4.6 Interface 216
Domino Webmail 5.0/6.0 Interface 216
Domino iNotes (Single Server) 217
Domino iNotes (Multi Servers) 217
All Domino (Webmail 5.x/6.x/7.x and iNotes) Interfaces 217
Portal Template 218
"OtherWeb" Template 218
Before You Begin 219
Conventions Used by the Application Customization Templates 219
Creating a Custom Application Customization File 220
Syntax of the Application Customization Template 221
Manipulation Elements 222
<ADD_HEADER> *223*
<DATA_CHANGE> *226*
<FRAME_WRAP> *230*
<HEADER_CHANGE> *234*
<URL_CHANGE> *251*

Chapter 8: Optimizing Portal Performance 255

Host Address Translation 255
What Types of Data are Manipulated? 256
Link Manipulation 256
Processing Requests 258
Processing Responses 261
Configuration in the Application Access Portal Tab 263
Skip Body Parsing per Application Server 265
Search and Replace in Content per Application Server 269

Manual URL Replacement 272
Dynamic Forwarding Rules..... 275
Advanced Configuration in the Configuration Files 278
Before You Begin 279
Once You Edit a File..... 280
Configuring the HTML Parser 280
<PARSER_HTML> *280*
Configuring the JavaScript Parser 287
<PARSER_EXCEPTION> *287*
Data Change 289
<DATA_CHANGE> *290*
Cookie Handling 300
<COOKIES_HANDLING> *300*
Header Filtering 314
<HEADER_FILTER> *314*
Excluding Servers from the Manipulation Process 321
<DONT_SIGN_SERVERS> *321*
Application Aware Skip Body Parsing 324
<URLS_WITHOUT_BODY_PARSING_REQ> *324*
<URLS_WITHOUT_BODY_PARSING_RES> *325*
Application Aware Search and Replace in Content 331
<URLS_WITH_SEARCH_AND_REPLACE_RES> *331*
Troubleshooting the Application Access Portal 337
HAT via Proxy 338
Dealing with Unrecognized URLs in Responses 339
Removing Encoding from Encoded Content 340
<REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT> *341*
Rerouting Unrecognized URLs Based on a Referer Header 342
Adding Content-Types to the Content Parsers 343
<PARSERS_CONFIGURATION> *344*
Header Token-to-Token Replacement 347
<HEADER_CHANGE> *348*

Maintaining Original Cookie Names 356
Error Reporting 357
Configuring an Application Access Portal Trace 358
Activating the Dump Utility 359

Chapter 9: Configuring the High Availability Array 361

Defining the High Availability Parameters 362
Preparing the List of IAG Servers 362
Configuring Trunk IP Addresses 364
Deploying the High Availability Configuration 365
Exporting a Configuration Project 365
Importing a Configuration Project 366

Appendix A: Unicode Support 369

Enabling UTF-16 and UTF-32 Encoding 369
Activating the Encoding Filter Extension 370
Creating and Updating the Encoding Configuration File 371

Appendix B: Regex++, Regular Expression Syntax 373

Literals 373
Wildcard 373
Repeats 373
Parenthesis 374
Alternatives 374
Sets 374
Character Classes 375
Sample Set 376
Line Anchors 376
Back References 376
Characters by Code 377

Word Operators 377
Buffer Operators 377
Escape Operator 378
Single Character Escape Sequences 378
Miscellaneous Escape Sequences 379
What Gets Matched? 380

Appendix C: Form Authentication Engine 381

Functional Operation 382
How Login Forms Are Processed 382
How Change Password Forms Are Processed 383
Configuring the Form Authentication Engine 384
Conventions Used in the Engine's Data Definition and Configuration Files 385
Before You Begin 386
Form Authentication Engine Custom Data Definition File 387
Creating a Custom Form Authentication Engine Data Definition File 388
Syntax of the Form Authentication Engine Data Definition File 389
<USER_AGENTS> *389*
<USER_AGENT_GROUPS> *393*
<SCRIPTS> *396*
Form Authentication Engine Custom Configuration File 400
Creating a Custom Form Authentication Engine Configuration File 400
Syntax of the Form Authentication Engine Configuration File 402
<APPLICATION> *402*
Sample Login Form Configuration 427
Autosubmit Script 431
Configuring a Custom Autosubmit Script 432
Sample Autosubmit Script 434
Client-Side Processing 434
Multi-Value Variables 436

Chapter 1

Introduction

The Intelligent Application Gateway (IAG) provides web application security solutions that enable you to rapidly deploy secure web-based access to sensitive web applications throughout your organization, while significantly reducing the deployment cycles. The system protects against known and unknown threats by isolating application servers and tightly controlling application layer access to them. It utilizes an application-centric approach to unite all of the components that protect the application into a single appliance. Encryption, authorization, authentication, HTTP payload screening, pre-defined rule-sets, application customization, and a physical air gap all reside within an integrated platform.

About This Guide

This guide is intended for the system administrator of the IAG. It provides you with detailed information regarding advanced configuration and capabilities of the IAG, described below. This guide provides information on the following topics:

- Chapter 2: **"Security Management Tools"** introduces you to the interfaces you use for advanced configuration of the IAG.
- Chapter 3: **"Customizing Web Pages"** describes how you customize the look-and-feel and other aspects of the HTML pages the user interacts with, for example changing the company logo and the color scheme, and key text items such as the administrator email address and the company site link.
- Chapter 4: **"Access Control"** describes the mechanisms that the IAG applies in order to control users' access to the organization's internal network, including authentication and authorization servers and repositories, server name translation, and initial host selection.
- Chapter 5: **"Session Settings"** describes the configuration of session settings, such as the maximum number of sessions that can be concurrently open, how you define default and privileged sessions, and more.
- Chapter 6: **"Content Inspection"** deals with the IAG's application-level content inspection. It explains how you create and edit inspection rules and rules sets, how you can configure the IAG to

check headers of incoming requests, and how you can, to a limited extent, enable the IAG to communicate with non-compliant HTTP applications.

- Chapter 7: **"Application Customizers"** describes how you can use the customizers to add functionality to web applications, either by adding a frame around the original content, or by seamlessly adding the required functionality to the application content. The Application Customization interface also enables you to configure the system's global content-type and compression handling lists.
- Chapter 8: **"Optimizing Portal Performance"** describes how the SSL VPN portal enables access to multiple web applications by using Host Address Translation (HAT). It provides information on optimizing portal performance and advanced configuration of the Portal trunk, and contains portal troubleshooting tips and procedures.
- Chapter 9: **"Configuring the High Availability Array"** describes how you configure the IAG High Availability option and deploy the High Availability configuration to all IAG servers in the array.
- Appendix A: **"Unicode Support"** describes the steps you need to take if any of the pages users access via the IAG are encoded using the UTF-16 or UTF-32 Unicode encoding formats.
- Appendix B: **"Regex++, Regular Expression Syntax"** describes the Regex++ library, used in the configuration of the IAG.
- Appendix C: **"Form Authentication Engine"** describes how you configure the Form Authentication Engine to handle HTML forms sent by the application to the end-user.

Conventions Used in This Guide

This section explains the conventions used throughout this Guide.

Menu Item	Menu names and menu items.
Button	Buttons that you select with the mouse.
	Icons that you select with the mouse are represented graphically.
Procedure	Title of an operating procedure.
`Computer text`	System files and information that you type in.

Caution

A note advising you that failure to take or avoid a specific action could result in loss of data.

Note

Important information you should note.

Tip

Helpful tips for working with the e-Gap Appliance.

Chapter 2

Security Management Tools

Overview

The Intelligent Application Gateway (IAG) provides you with a number of interfaces that help you to control access to internal resources and to configure parameters for each of the services, trunks, and internal websites that run on the IAG:

- **The Service Policy Manager**. A policy-enforcement tool that can optionally be used by the security administrator to pre-configure and set the organization's security policy. Using the Service Policy Manager, you can maintain a number of policy management and configuration setups, thus easily applying the settings that are best suited for your organization at any given time. This interface is described in "Service Policy Manager" on page 22.
- **The Configuration program**. You use the Configuration program to create trunks and to modify the configuration parameters whenever you wish, in order to comply with the changing needs of your organization, such as an additional network you wish to communicate with, or adding new applications to the AAP. This interface is described in "Configuration Program" on page 31.

Note

The first time you access either the Configuration program or the Service Policy Manager, you are required to create an encryption key and passphrase for the IAG. The key and passphrase serve both IAG applications, so that this action is only required once; when you subsequently access either application, you use the same passphrase.

The process for creating the encryption keys and passphrase is described in "Creating Encryption Keys" on page 20.

- **The Editor**. This interface is used to open and edit the IAG configuration files, which are encrypted during day-to-day operation. It enables you to decrypt and open the files for viewing or editing and to encrypt them after editing, as well as to convert and edit base64-encoded selections, as described in "Editor" on page 40.

Creating Encryption Keys

This section describes how you create an encryption key and passphrase for the IAG. The first time you access either the Configuration program or the Service Policy Manager, you are prompted to create an encryption key and passphrase. This action is only required once; the key and passphrase subsequently serve both IAG applications.

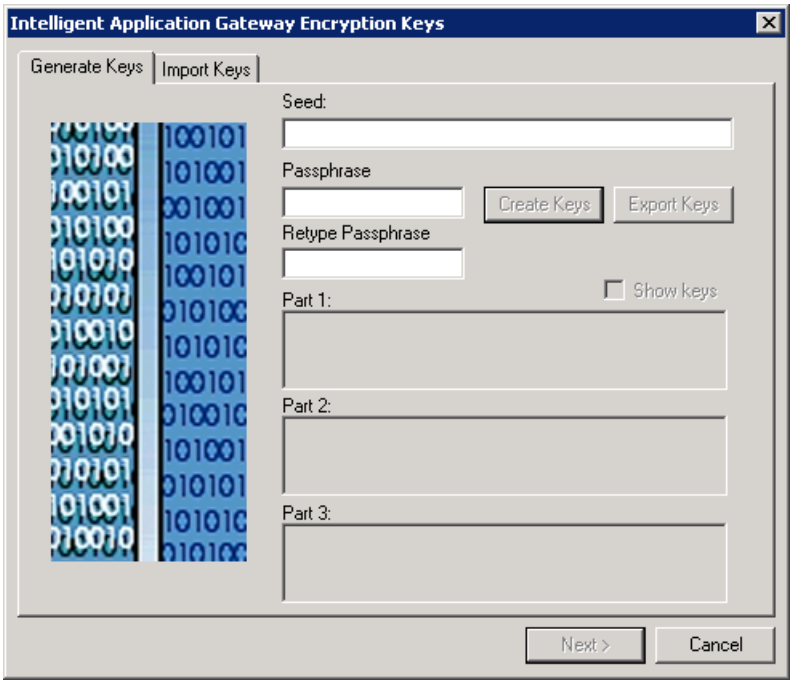
Note

- For more information about the IAG encryption keys, refer to the *Intelligent Application Gateway User Guide*, to "Encryption" on page 21.
- If different keys have already been defined on different IAG servers and you need to unify them, you take these steps:
  - To export the encryption keys, run the following utility in a command line, on one of the IAG servers:
    `…\Whale-Com\e-Gap\common\bin\ExportKeys.exe`
  - To import the keys to the other IAG servers, run the following utility in a command line, on each of the servers:
    `…\Whale-Com\e-Gap\common\bin\ImportKeys.exe`

To create the IAG encryption keys and passphrase:

1. At the IAG, click Start, then point to **Programs > Whale Communications IAG**, and select one of the following:
   - To access the Configuration program, click **Configuration**.
   - To access the Service Policy Manager, in the **Additional Tools** group, click **Service Policy Manager.**

   The Intelligent application Gateway Encryption Keys dialog box is displayed.



Note

This dialog box is only displayed once, the first time you access either the Configuration program or the Service Policy Manager.

2. Enter a string of characters in the Seed field, until you fill the entire field. You can use any combination of letters and digits. The keys you enter, combined with the time interval between these keys, are used to automatically create unique keys.
3. Enter an 8 to 32 alphanumeric passphrase in the Passphrase field. It is very important that you remember the passphrase you enter here. The passphrase is required to configure the IAG once it is installed and running, and for future software upgrades and modifications. You will also use it if you install a High Availability array.
4. Verify the passphrase by entering the same characters in the Retype Passphrase field, then click Create Keys.

   A message box informs you that the keys have been successfully created. When you dismiss this box, the Save As dialog box is displayed.
5. Enter the path and filename to save the new encryption keys, and then click Save.

   A message box informs you that the keys were exported successfully.

> **Note**
> In High Availability arrays, you must use the same encryption keys across all IAG servers in the array. In this case, on each of the subsequent IAG servers you configure in the array, in the Encryption Keys dialog box, access the Import Keys tab, and click Import Keys to import the keys you saved. You must enter the passphrase you defined in the first IAG server.

6. Dismiss the message box, and in the Encryption Keys dialog box, click Next>.

   The application is initiated. For more information, refer to:

   - *"Accessing the Service Policy Manager" on page 23.*
   - *"Accessing the Configuration Program" on page 32.*

Service Policy Manager

The Service Policy Manager is where you can optionally pre-configure the IAG and create, maintain, and enforce the IAG's security policies.

The Service Policy Manager is pre-populated with default values, so that it is possible to create and configure trunks in the Configuration program without pre-configuration in the Service Policy Manager. You can, however, use the Service Policy Manager to control the IP addresses and port numbers which are used to access the internal network. When you do this, in the Configuration program you can only configure the IAG in such a way that is within the policy defined in the Service Policy Manager.

This section describes how you:

- Access the Service Policy Manager, as described in "Accessing the Service Policy Manager" on page 23.
- Define IP addresses and port numbers in the Service Policy Manager, as described in "Creating Lists of IP Addresses and Port Numbers" on page 26.

In addition, you can use the Service Policy Manager to:

- Change the Login password, as described in "Changing the Login Passwords" on page 28.
- Protect your web servers, as described in "Protecting Web Servers on the IAG" on page 29.

Accessing the Service Policy Manager

To access the Service Policy Manager window:

1. At the IAG, click Start, then point to **Programs > Whale Communications IAG > Additional Tools**, and click **Service Policy Manager**.

   The dialog box that is displayed may differ, as follows:

   - *If you have not accessed either the Service Policy Manager or the Configuration program until now, you are prompted to define the IAG encryption keys. For more details, see "Creating Encryption Keys" on page 20.*
   - *If you have previously accessed the Configuration program, but this is the first time you are accessing the Service Policy Manager, you are prompted to create a password for the Service Policy Manager. Proceed to step 2.*

   Note
   The password must contain at least six digits.

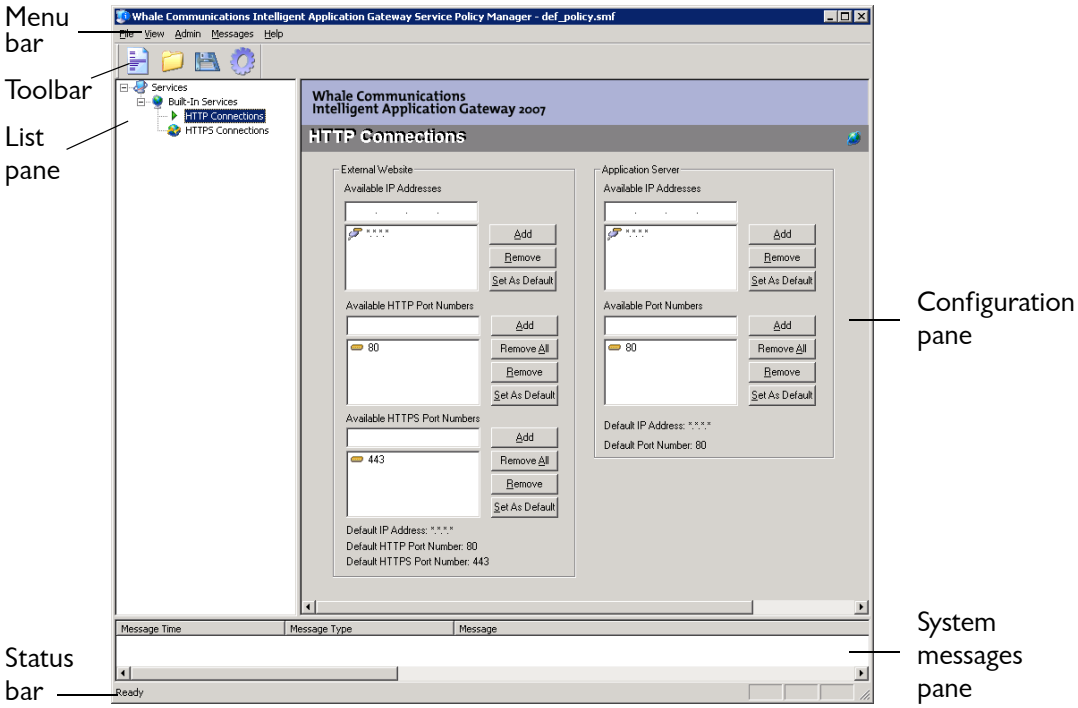
   - *When you access the Service Policy Manager on consecutive sessions, the IAG prompts you to enter the password.*

   Tip
   If you wish to change the password, refer to "Changing the Login Passwords" on page 28.

2. Enter the passphrase where applicable, then enter the password and click OK.

   The Service Policy Manager window is displayed.

Figure 1. Sample Service Policy Manager Window



Service Policy Manager Window Layout

The Service Policy Manager window is divided into the following areas:

- The menu bar and toolbar provide access to the Service Policy Manager functions.
- The List pane displays the existing services, and enables you to add user-defined TCP services.

> **Note**
> You can expand and collapse the Services tree in the List pane, to show and hide the service-types and services.

- The Configuration pane is where you define and edit configuration parameters for the services displayed in the List pane.
- Messages generated by the Service Policy Manager are displayed in the Message pane. You can also set the IAG so that the messages are displayed in the Windows Event Viewer, as described in "Configuration Messages" on page 44.
- The status of the IAG is displayed at the bottom, in the Status bar.

Tip

In the **View** menu, you can select whether to view or hide the following screen elements:

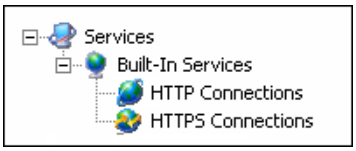
- Toolbar
- Status Bar
- Messages

List Pane

In the List pane, the Built-In Services group is divided into two categories:

- HTTP Connections
- HTTPS Connections.

Figure 2. Service Policy Manager List Pane



Toolbar

The Service Policy Manager toolbar buttons are described in the following table.

Table 1. Service Policy Manager Window—Toolbar Buttons

Button		Description
	New	Creates a new Service Policy Manager file (`*.smf`)
	Open	Opens an existing Service Policy Manager file (`*.smf`)
	Save	Saves the currently open file
	Activate	Saves the file, then activates the Service Policy Manager configuration

Creating Lists of IP Addresses and Port Numbers

You can optionally use the Service Policy Manager to control the IP addresses and port numbers which are used to access the internal network, by creating lists that will enforce the selection of pre-defined IP address and port numbers during trunk configuration in the Configuration program.

The Service Policy Manager is pre-configured as follows:

- The IP addresses for both the External Website and the Application Server, and for both HTTP and HTTPS connections, are defined with wildcards (*.*.*.*), so that you can later choose any IP address during trunk configuration. The wildcards are set as the Default IP Address.

Tip

- You can insert specific IP addresses in the Service Policy Manager in addition to the wildcard definition, so that they are readily available for selection during trunk configuration. You can also use combinations of wildcards and octets, for example: 192.168.*.*
- To restrict the list of IP addresses available for trunk creation and configuration in the Configuration program, enter the IP addresses you want to have access to in the Service Policy Manager, and delete the wildcard value *.*.*.*.

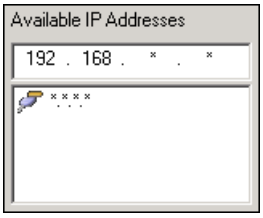
- The Port Numbers are defined as follows:
  - For an HTTP connection, the default port number for both the External Website and the Application Server is 80.
  - For an HTTPS connection, the default port number for the External Website is 443.
  - For an HTTPS connection, the default port number for the Application Server is 80, but ports 443, 6001, 6002 and 80 are also available.

The following procedures describe how you:

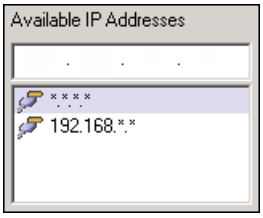
- Add an IP address to the list of available addresses
- Add a port number to the list of available port numbers
- Select a default IP address and default port number
- Remove an IP address or a port number from the list

To add an IP address to the list of available addresses:

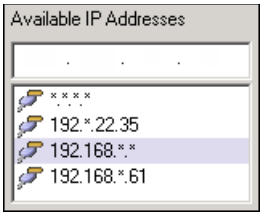
1. Under Available IP Addresses, enter an IP address. You can use the asterisk wildcard (*) instead of any of the four required octets, to be replaced with any number during trunk configuration.



2. Click Add on the right side of the Available IP Addresses list.
   The IP address is added to the Available IP Addresses list.

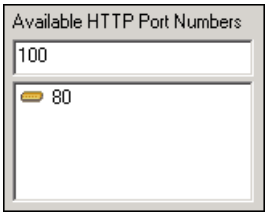


3. Repeat the steps above to add as many IP addresses as you wish to the list.

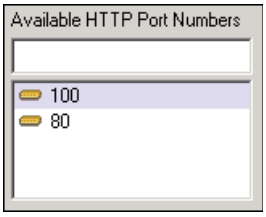


To add a port number to the list of available port numbers:

1. Under Available Port Numbers, enter a port number.



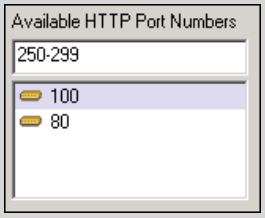
2. Click Add on the right side of the Available Port Numbers list.
   The port number is added to the Available Port Numbers list.



3. Repeat the steps above to add as many port numbers as you wish to the list.

Tip

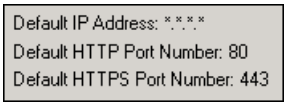
You can add a single port or a range of ports, as demonstrated in the following example:



To select the default IP address and default port number:

- Select the IP address or port number you wish to set as default, and click Set As Default on the right side of the applicable list.

  The selected IP address or port number is set to be the default. It will be displayed by default when the Configuration program is displayed.



To remove an IP address or a port number from the list:

- Select the IP address or port number and click Remove.

  The selected address or number is removed from the list. It will no longer be available for selection in the Configuration program.

Changing the Login Passwords

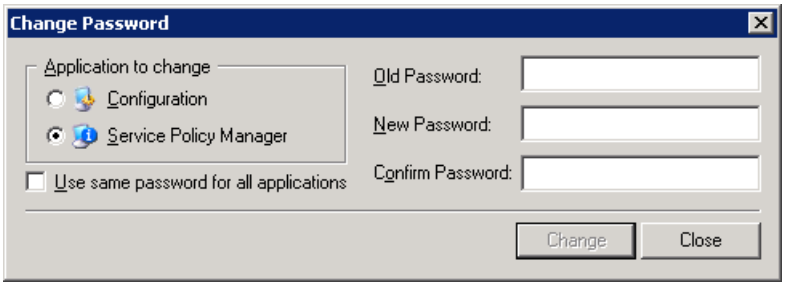
The Configuration program and the Service Policy Manager are protected by a password. When you access these applications for the first time after installation of the IAG, you are prompted to enter a new password, which is then assigned to the application.

At any time after that, you can use the Service Policy Manager to change the passwords, as described below.

To change a password:

1. In the Service Policy Manager window, on the **Admin** menu, click **Change Passwords...**

   The Change Password dialog box is displayed.



2. Under “Application to change”, select the application for which you wish to change the password. If you wish to apply the same password for all the IAG applications, select the option “Use same password for all applications”.
3. At the right, enter the required information: passphrase or old password, and new password.

Note

The password must contain at least six digits.

4. Click Change.

   The selected password or passwords change, accordingly.

Protecting Web Servers on the IAG

During the creation of trunks in the Configuration program, a filter is added to each of the trunks you create; the IAG utilizes Internet Service Application Programming Interface (ISAPI) filters, which run under Windows Server™ IIS (Internet Information Server). For each filter, a website is created under the IIS. If any sites other than the IAG websites reside on the same computer, they are deleted, with the exception of the Default Website. For example: if the application server also resides on the IAG, it will be deleted, unless protected as described in this section.

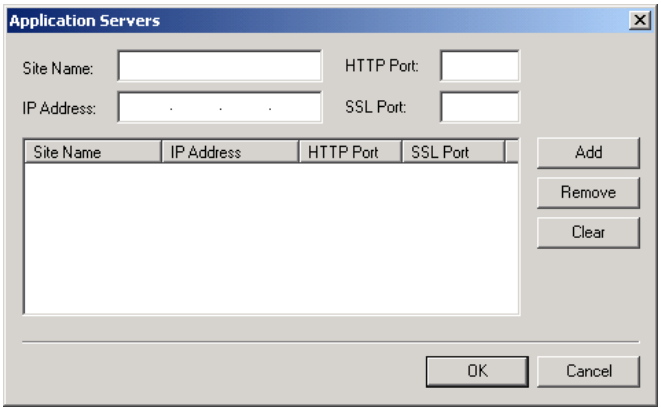
You can use the Service Policy Manager to define the web servers that you wish to protect. Those will not be deleted during the creation of the trunk, even if they reside on the IAG. Also, during trunk creation and configuration in the Configuration program, you can not give a filter either the same site name, or the same combination of IP address and port number as one of the protected web servers. If you try to do so, the Configuration program will prompt you to redefine the relevant trunk parameters.

The following procedure describes the process of defining which web server will be protected.

To define the web servers that will not be deleted:

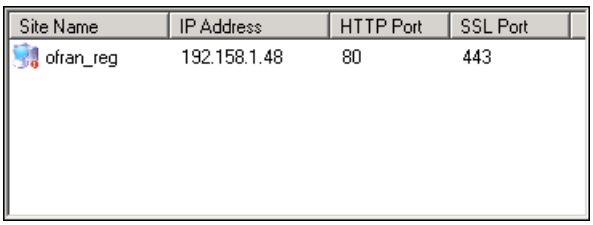
1. In the Service Policy Manager window, in the **Admin** menu, click **Application Server Configuration**.

   The Application Servers dialog box is displayed.



2. Enter the site name, IP address, and HTTP and SSL port numbers, and click Add.

   The server is added to the list.



3. Repeat the above step to add all the servers you wish to protect to the list. Click OK.

   The Application Servers dialog box closes. The servers defined in the list will not be deleted during configuration of HTTP and HTTPS trunks.

To remove application servers from the list:

- Select an item on the list and click Remove to remove a selected server from the list.

Or,

- Click Clear to remove all servers from the list.

Configuration Program

In the Configuration program you create, configure, and edit trunks on the Intelligent Application Gateway. The configuration process defines the parameters that control data transfer to and from the internal network. You define the types of data that can be transferred, and their transfer channels. You also define the acceptance rules that are applicable for each type of service.

Some of the fields in the Configuration program are pre-populated with default values set by the system, or values that you assigned in the Service Policy Manager, if you define the security policy of the service. For example, the default wildcard values for IP addresses (*.*.*.*) enable you to insert any IP address in the Configuration program.

The following sections describe:

- How you access and use the Configuration program, in "Configuration Program" on page 32
- The process of creating and editing Configuration files, in "Configuration Files" on page 35
- How to export and import Configuration projects, in "Configuration Projects" on page 36
- How to change the port number of the IIS Default Web Site, in "Changing the Port Numbers of the IIS Default Web Site" on page 39
- How to configure SSL encryption settings, used to encrypt SSL traffic in HTTPS trunks, as described in "SSL Configuration" on page 39.

Once you familiarize yourself with the Configuration program, you can use it to create and edit trunks, as described in detail in separate chapters, as follows:

- For the creation and editing of Portal trunks, refer to the *Intelligent Application Gateway User Guide*, to Chapter 2: "SSL VPN Portals"
- For the creation and editing of Webmail and Basic trunks, refer to the *Intelligent Application Gateway User Guide*, to Chapter 3: "Single Application Sites".

Configuration Program

The following sections provide instructions on how to access the window, and describe its various elements.

> **Note**
> Before you start the configuration process, log on to Windows with full administrator privileges.

Accessing the Configuration Program

To access the Configuration program:

1. In the Windows desktop of the IAG, click Start, then point to **Programs > Whale Communications IAG** and click **Configuration**.

   The dialog box that is displayed may differ, as follows:

   - *If you have not accessed either the Configuration program or the Service Policy Manager until now, you are prompted to define the IAG encryption keys. For more details, see "Creating Encryption Keys" on page 20.*
   - *If you have previously accessed the Service Policy Manager, but this is the first time you are accessing the Configuration program, you are prompted to create a password for the Configuration program. Proceed to step 2.*

> **Note**
> The password must contain at least six digits.

   - *When you access the Configuration program on consecutive sessions, the IAG prompts you to enter the password.*

> **Tip**
> If you wish to change the password, refer to "Changing the Login Passwords" on page 28.

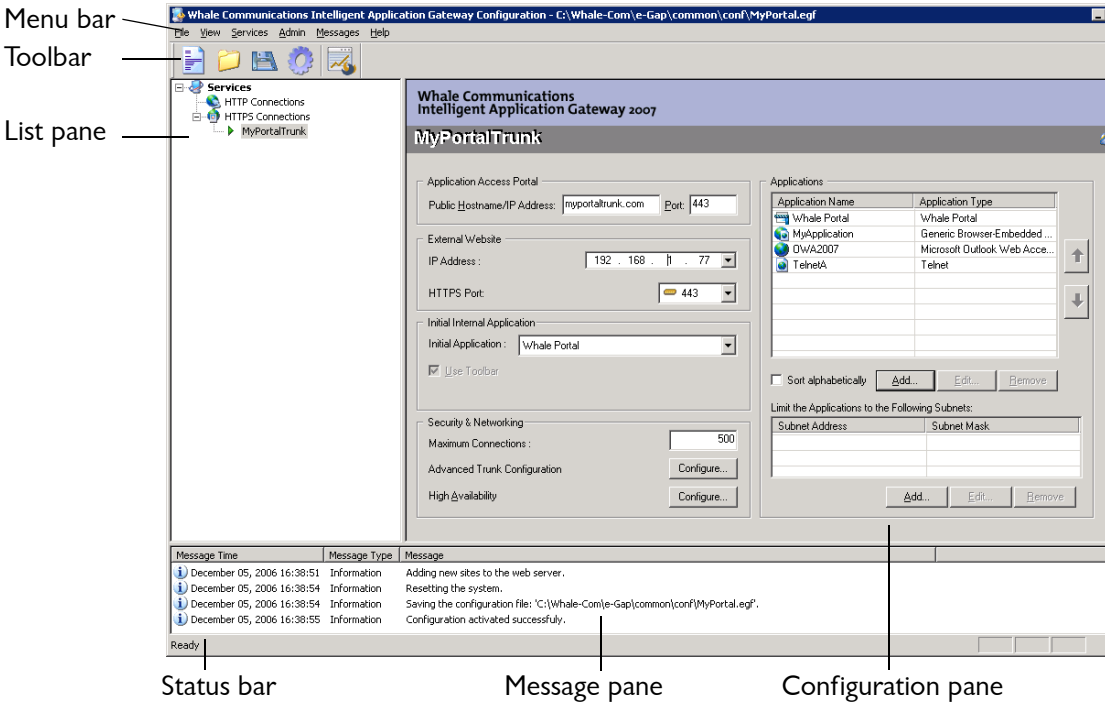
2. Enter the passphrase where applicable, then enter the password as required and click OK.

   The main window of the Configuration program is displayed.

   - *The first time you configure the IAG, a new configuration file is opened, and the main window of the Configuration program displays default parameters.*

- *When you open the Configuration program on consecutive configuration sessions, the last saved file is opened, and the main window of the Configuration program displays the file's configuration parameters. The file name and location are displayed in the Title bar.*

Figure 3. Sample Configuration Program Window, for a Portal Trunk



Configuration Program Layout

The main window of the Configuration program is divided into the following areas:

- The menu bar and toolbar provide access to the configuration functions.
- In the List pane you can see the Services and all the trunks that are defined in the IAG. You can also initialize the Create New Trunk Wizard from here.
- The Configuration pane is where you edit trunk configuration parameters. From this pane you also access additional dialog boxes, such as the Advanced Trunk Configuration and Application Properties dialog boxes.

Note
The appearance and contents of the Configuration pane vary, depending on the trunk type you have selected.

- Messages generated by the Configuration program are displayed in the Message pane. You can also set the IAG so that the messages are displayed in the Windows Event Viewer, as described in "Configuration Messages" on page 44.
- The status of the IAG is displayed at the bottom, in the Status bar.

Tip
In the **View** menu, you can select whether to view or hide the following screen elements:

- Toolbar
- Status Bar
- Messages

Configuration Program Toolbar

The Configuration program toolbar buttons are described in the following table.

Table 2. Configuration Program—Toolbar Buttons

Button		Description
	New	Creates a new configuration file (*.egf)
	Open	Opens an existing configuration file (*.egf)
	Save	Saves the currently open file
	Activate	Saves the file, then activates the configuration and creates supplemental configuration files necessary for the operation of the IAG.
	Web Monitor	Launches the IAG Web Monitor. For more details about the Web Monitor, refer to the *Intelligent Application Gateway User Guide*, to "Web Monitor" on page 258.

The Create New Trunk Wizard

When you create a new trunk, the Configuration program activates the Create New Trunk Wizard for quick auto-completion of the initial trunk setup, external website, authentication, application customization, and URL inspection rules.

- The creation of a Portal trunk is described in the *Intelligent Application Gateway User Guide*, in Chapter 2: "SSL VPN Portals".
- The creation of Webmail and Basic trunks is described in the *Intelligent Application Gateway User Guide*, in Chapter 3: "Single Application Sites".

Configuration Files

A number of files are created during the configuration process. You define, edit, and save the trunks' parameters in one, main configuration file, using the Create New Trunk Wizard and the Configuration program. Additional files are created, depending on the application of the IAG, for example files that hold URL Inspection rulesets.

You can create a number of main configuration files, with different parameter settings. The parameters in the active file are the ones that are applied to the incoming and outgoing data. When you wish to apply different configuration parameters, you can open and activate another configuration file, thus applying its parameters to the system.

The procedures for creating a new configuration file and for opening an existing file are described in the following sections. The name and path of the currently active configuration file are saved in the Windows Registry.

Tip

For information about exporting and importing the configuration settings between a number of IAG servers, refer to "Configuration Projects" on page 36.

Creating a New Configuration File

You create a new configuration file the first time you configure the IAG. You can create as many additional files as you need at any time following the initial configuration.

Opening an Existing Configuration File

You can revert to configuration settings which you previously saved on the current IAG server by opening and activating an existing configuration file, as described below.

> **Note**
> You can not open a configuration file which you saved on another IAG server in this manner. In order to do so, refer to "Configuration Projects" on page 36.

To open a configuration file:

1. In the toolbar of the Configuration program, click , or on the **File** menu, click **Open**.
   The Open Configuration File window is displayed.
2. Select the file you wish to open, enter your passphrase, and click Open.
   The selected file opens. The file name and path are displayed in the title bar.
3. Click to activate the configuration; when prompted, enter your passphrase.
   The Activate Configuration window is displayed.
4. In the Activate Configuration window, click Activate >.
   The configuration file you have loaded is now active, and its parameters determine the IAG's behavior.

Configuration Projects

Once you configure one IAG server, you can export the configuration settings into a configuration project. You can then import the project into other IAG servers, and apply the same configuration settings to multiple IAG servers.

> **Note**
> The same encryption key has to be installed and an identical passphrase must be used for the IAG server from which you export the configuration project, and all the IAG servers into which you import the project. For more information about encryption, refer to the *Intelligent Application Gateway User Guide*, to "Encryption" on page 21.

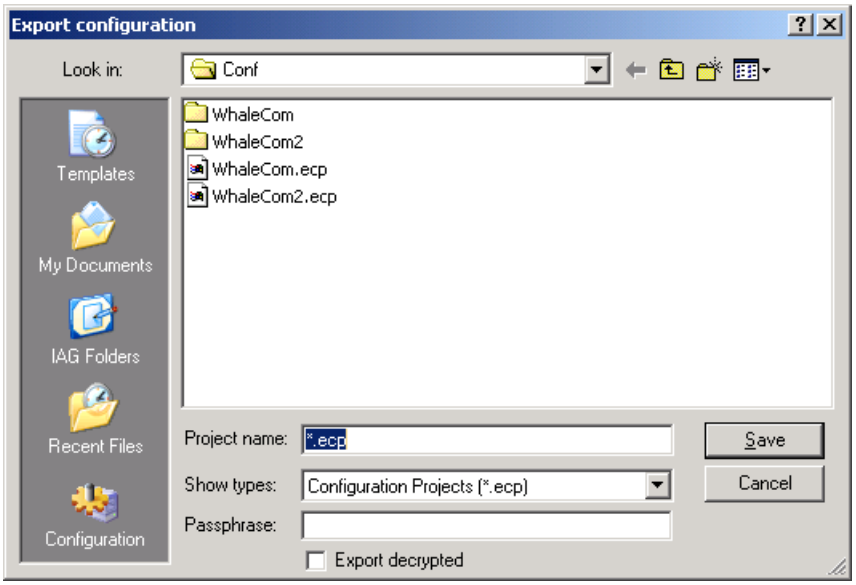
Exporting a Configuration Project

You can export the current configuration settings into a project. When you wish to restore those settings, either on the current IAG server or on other IAG servers, you can import the project, as described in "Importing a Configuration Project" on page 37.

To export the configuration:

1. In the Configuration program, on the **File** menu, click **Export...**

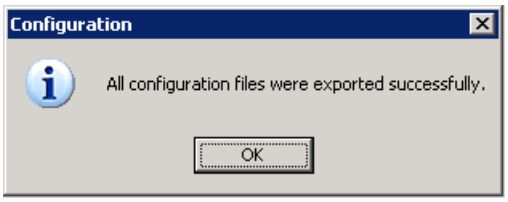
   The Export configuration window is displayed.



2. Enter the following:
   - Select the location to where you wish to export the project, and enter project name.
   - Enter your passphrase.
   - By default, the configuration file is encrypted when it is exported. If you wish the file not to be encrypted, select the "Export Decrypted" option.

   Click Export. When prompted, select Yes to create a new file.

   The Export configuration window closes, and the configuration is exported to the selected location. The following message is displayed:



Importing a Configuration Project

You can import a configuration project that you exported from one IAG server into other IAG servers, as described below.

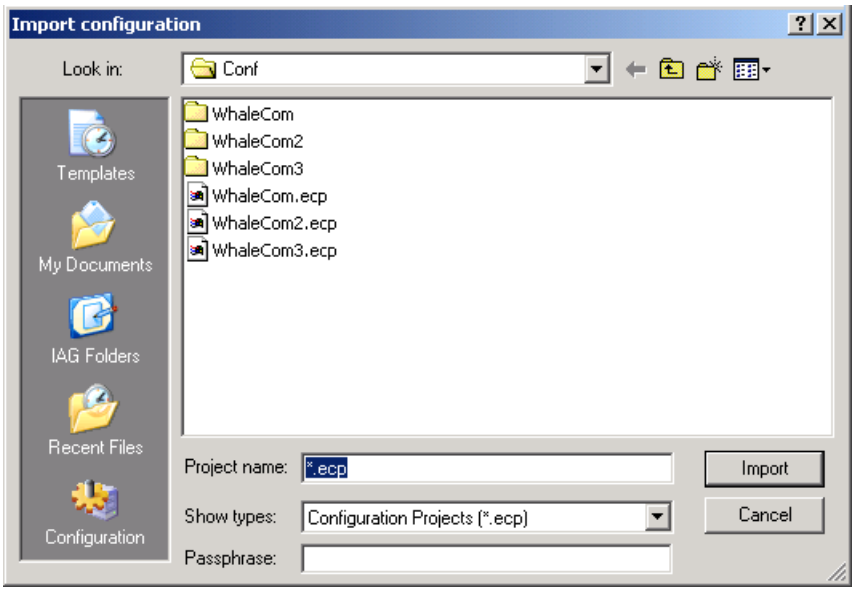
Note

Verify that the same encryption key and passphrase were used for the IAG server from which you export the configuration project, and all the IAG servers into which you import the configuration project.

To import the configuration:

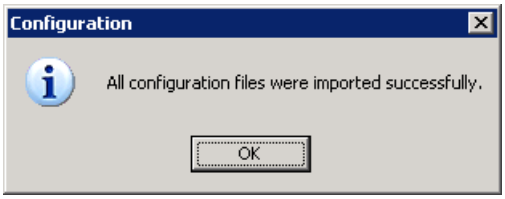
1. In the Configuration program, on the **File** menu, click **Import...**

   The Import configuration window is displayed.



2. Select the configuration file, enter your passphrase, then click Import.

   The Import configuration window closes. The project, including all its configuration settings, is imported into the IAG server. The following message is displayed:



3. Click OK to dismiss the message. In the menu bar of the Configuration program, click and follow the instructions on the screen to save and activate the configuration.

   The configuration is applied to the current IAG server. The configuration parameters of the current IAG server are set according to the configuration settings you have imported.

Changing the Port Numbers of the IIS Default Web Site

The default port numbers assigned for the IIS Default Web Site on the IAG are:

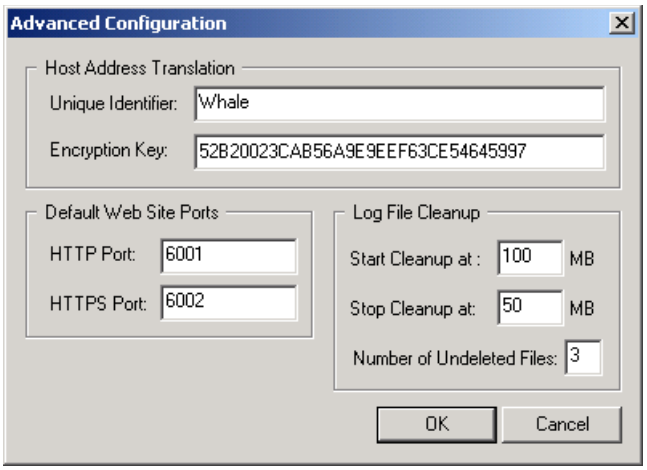
- For HTTP connections–6001
- For HTTPS connections–6002

If you are using one or both of these ports for another application, you can change the port number, as described below.

To change the ports defined for the IIS Default Web Site:

1. In the Configuration program, on the **Admin** menu, click **Advanced Configuration...**

   The Advanced Configuration window is displayed.



2. In the Default Web Site Ports area, replace the default HTTP or HTTPS port numbers, as required, and click OK.
3. Click to save and activate the configuration.

SSL Configuration

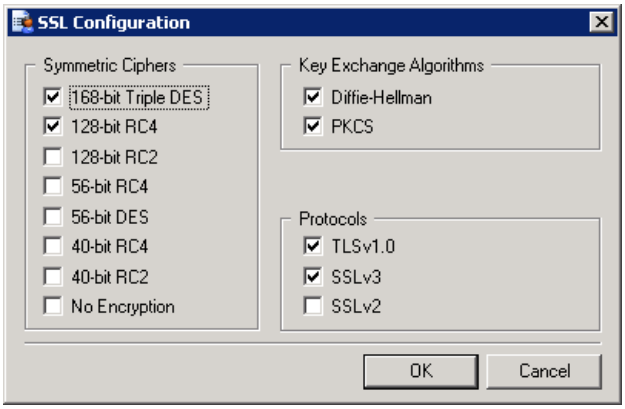
This option enables you to configure the following SSL encryption settings, used to encrypt SSL traffic in HTTPS trunks:

- **Symmetric Ciphers**: key sizes
- **Key Exchange Algorithms**
- **Protocols**: allowed SSL protocols

To configure SSL settings:

1. In the Configuration program, on the **Admin** window, click **SSL Configuration...**

The SSL Configuration dialog box is displayed.



2. Use the SSL Configuration dialog box to make the required changes.

Note

In order for the changes to take effect, you must activate the configuration, then restart the IAG.

Editor

Sometimes you might need to view or edit IAG files, such as configuration files, which are automatically encrypted during day-to-day operation. On other occasions, you might need to create a file which has to be encrypted.

Using the Editor program, you can:

- Open and edit text (ascii) files, for example: `.ini` files
- Decrypt and edit encrypted files
- Use the Sort and Find capabilities to locate the required section within a file
- Convert selections of text within the file to and from Base64 format
- Save and encrypt files
- Open multiple files simultaneously

Layout of the Editor

Figure 4. Layout of the Editor

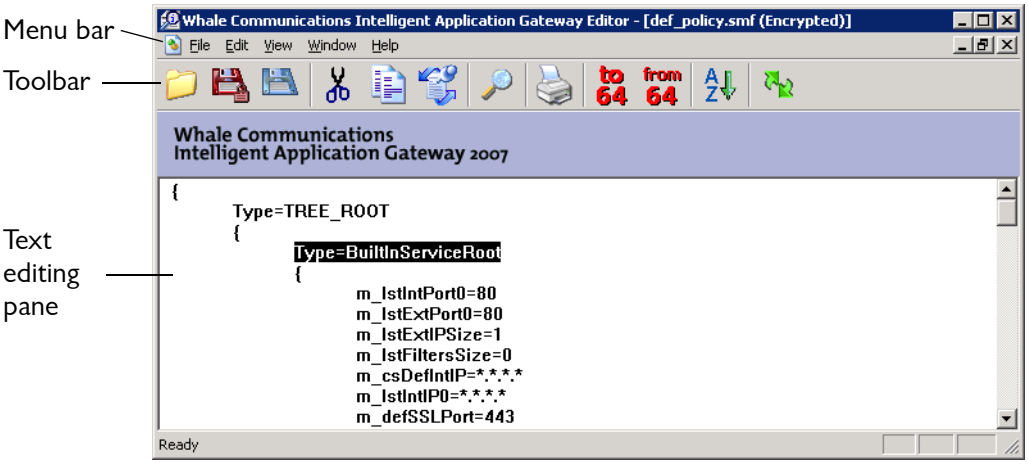


Table 3. Editor—Toolbar Buttons

Button		Description
	Open	Opens an existing text file
	Save As Encrypted	Saves the file with encryption
	Save As Decrypted	Saves the file as a non-encrypted ascii file
	Cut	Cuts the selected text
	Copy	Saves a copy of the selected text in the clipboard
	Paste	Paste a copy of the text in the clipboard to the cursor location
	Find	Finds text

Table 3. Editor—Toolbar Buttons (Cont'd)

Button		Description
	Print	Print the file
	Convert to Base64	Convert the selection from ascii format to Base64 format
	Convert from Base64	Convert the selection from Base64 format to ascii format
	Open Sorted	Open a file sorted in alphabetical order
	Reload	Reload the current file without saving the current changes

Editing a File with the Editor

To edit files with the Editor:

1. In the Windows desktop of the IAG, click Start, then point to **Programs > Whale Communications IAG**, select **Additional Tools**, and click **Editor**.
2. In the toolbar of the Editor, click , select the file you wish to open, and click Open. If the file you are opening is encrypted, enter your passphrase in the Passphrase field.

> **Tip**
> Once you open the file, the title bar indicates whether the file is encrypted or not.

3. Edit the file. See the following section, "Editing Tips", for information about sorting and finding text, and about editing Base64 format text.
4. Save the file in one of these ways:
   - Select **File > Save** from the menu bar to save the file in its original state (encrypted or without encryption).
   - Click to save the file as an encrypted file.
   - Click to save the file without encryption.

Editing Tips

Opening files

- You can drag and drop files onto the Editor interface.
- You can open more than one file at a time.

Editing Base64 format text

- Highlight the selection you wish to edit.
- Click to convert the text from Base64 format to ascii format.
- When you finish editing the file, highlight the selection again, and click to convert the text back to Base64 format before you save the file.

Tip
If you are editing text within a tag, such as an HTML of XML tag, make sure to select only the text within the tag; tag delimiters should not be selected.

Locating text in files

- To search for a word or phrase click .
- To sort the content of an open file alphabetically:
  - If you made any changes, save the file.
  - Click .
  - Click to reload the file.

Note
Do not edit the sorted file, as it cannot be saved. After you have located the information you are looking for, turn the sort button off, reload the file by clicking , and edit the file.

- To open a new file already sorted alphabetically, click before you open the file.

Finding the Application-Type

Some of the IAG advanced options, such as URL Inspection rulesets, or some of the Application Customizers and AAP configuration tags, can be applied per application-type, that is, for all the applications of the specified type. When you configure an option for an application-type, you have to enter the type of the application. This section describes how you can use the Editor to find the definition of the application-type in the system, as should be used during the configuration.

Note

In Basic trunks, the application-type is OtherWeb.

To find the application-type of an application:

1. In the Application Properties dialog box, from the General tab, note the Application ID number.
2. Use the Editor to open the file `HTTPS_Filter.ini` or `HTTP_Filter.ini`, according to the type of trunk you are configuring, in the following location:

   `…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf`
3. Search for a section where the section title is the application ID, as you noted in step 1.

   For example:

   Information regarding the application whose ID is `ABC` is found in a section titled: `[ABC]`
4. Under this section, you can find the name of the application-type in the parameter `ApplicationType`.

Configuration Messages

You can view configuration messages in the following locations:

- Messages generated by the Service Policy Manager are displayed at the bottom of the Service Policy Manager main window, in the Message pane.
- Messages generated by the Configuration program are displayed at the bottom of the main window of the Configuration program, in the Message pane.
- In both the Service Policy Manager and the Configuration program, you can set the Messages Filter so that the messages are also displayed in the Windows Event Viewer, in the Application Log view.

The IAG issues three types of configuration messages: Information, Warning, and Error, as described in Table on page 45. You can use the Messages Filter to select what types of messages to view, and where to view the messages, as described in “Messages Filter” on page 45.

Tip

- You can select whether to show or hide the Message pane in the **View** menu.
- To clear the messages from the Message pane, select **Messages > Clear Messages**.
- If a message is not fully visible in the Message pane, position the mouse over the right-most edge of the Message column. When the mouse becomes a pointer , double-click to expand the column. If necessary, use the scroll bar.

Table 4. Configuration Messages—Icons

Icon		Description
	Information	Information regarding normal IAG operation, such as saving or activating the configuration.
	Warning	Warning messages. For example: if you create a trunk without a filter, the IAG issues a warning. You can continue the operation if you decide to do so in spite of the Warning message.
	Error	Error messages, issued when an error causes an operation to fail. For example: if you try to create a trunk with an IP address and port number already used by an existing trunk, an error message is displayed, and the trunks is not created.

Messages Filter

Use the Messages Filter to select the types of configuration messages you wish to view, as well as where the messages will be displayed: in the application’s Message pane, in the Windows Event Viewer, or both.

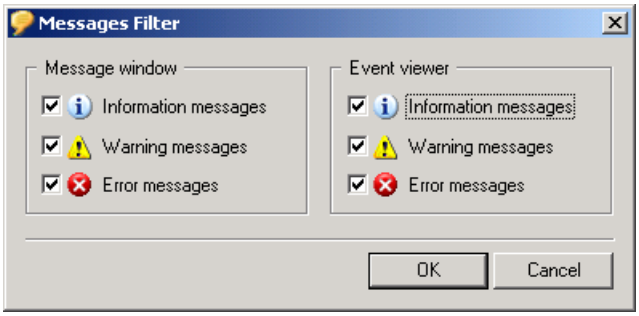
Note

The Messages Filter is identical on both the Configuration program and the Service Policy Manager interfaces.

To configure the Messages Filter:

1. From the menu bar, select **Messages > Filter...**

   The Messages Filter dialog box is displayed.



2. Select the types of configuration messages to display in each of the viewers, and click OK.

   The Messages Filter dialog box closes. Configuration messages will displayed according to your selection.

Chapter 3

Customizing Web Pages

During the day-to-day interaction with the Intelligent Application Gateway (IAG), users encounter various web pages, including:

- Information and interface pages that end-users encounter during login and logout.
- Portal homepage information and interface pages.
- Certified Endpoint Enrollment pages.
- Pages of the File Access application.
- The interface of the Web Monitor application.

This chapter describes how you can customize some aspects of the pages, including instructions on how to:

- Prepare files for customization, in "Before You Begin" on page 48.
- Customize the general user interaction pages, such as login and logout pages, in "Customizing General User Interaction Pages" on page 49.

Tip

If you wish to use you own, custom authentication pages, refer to "Using Your Own Authentication Pages" on page 103.

- Configure the portal homepage, in "Portal Homepage Configuration" on page 54, including how to:
  - Make changes in the default portal homepage, as described in "Customizing the Default Portal Homepage" on page 54.
  - Configure your own custom portal page, which is not based on the default portal homepage supplied with the IAG, as described in "Using a Custom Portal Homepage" on page 61.
- Make changes in the Certified Endpoint Enrollment pages, as described in "Customizing Certified Endpoint Enrollment Pages" on page 67.
- Change the language definitions of the File Access application, in "Changing File Access Language Definitions" on page 71.
- Customize the windows of the Web Monitor, in "Customizing the Web Monitor Windows" on page 72.

- Change the default text of the languages that are supported out-of-the-box, or add support for additional languages, in "Multi-Language Support" on page 77. This option is not relevant for the Web Monitor.

Note

This chapter describes only the customization aspects mentioned above. Other changes, such as using your own Login pages or other scripts, are described in the relevant sections of the IAG documentation. Do not make any changes other then those described in this chapter without first consulting the relevant documentation.

Before You Begin

This section describes how you prepare the files you wish to customize in order to change the default content or style of the web pages. The actual adjustments you can make are described in the relevant sections of this chapter.

Note

- Always use the method described here, else your customizations will be over-written when the system is next upgraded.
- File and folder names are case insensitive.

To prepare files for customization:

- Copy the file you wish to customize from the folder where it is installed by default to the `CustomUpdate` subfolder that is nested under it, as follows:
  - If you want the changes you make to affect all trunks, copy the file without renaming it. This is always the method you should apply for Web Monitor changes; since the Web Monitor monitors all the trunks in the IAG, it cannot be customized per-trunk.
  - If you want the changes you make to be applied to a single trunk, rename the file:
    `<Trunk_Name><Secure(0=no/1=yes)><OrigFile.extension>`

    Where:
    - `<Trunk_Name>` is the name of the trunk where you want the changes to take effect
    - `<Secure(0=no/1=yes)>`: enter `1` for an HTTPS trunk or `0` for an HTTP trunk
    - `<OrigFile.extension>` is the name of the original file, including file extension

> **Note**
> If you make changes per-trunk, you need to define a separate set of files for each trunk.

You can now make changes in the file you copied. The changes you make in the file will not be over-written when the IAG is upgraded.

Customizing General User Interaction Pages

The general user interaction pages that you can customize include pages that end-users encounter when using the IAG, such as login and logout pages. For Portal trunks, the changes you make here apply also to the System Information window, which, if enabled, is accessed via the System Information button on the portal homepage.

This section describes how you can customize visual and standard text attributes in these pages. The changes you make affect all of the relevant parameters in the pages; it is not possible to assign a different color scheme, logo, or standard text to each of the pages in this group.

For example: if you decide to change the logo graphic and text, these will automatically be changed uniformly on all pages. Pages where there was previously only a logo, and no logo text, will now include all the new definitions (graphics and text).

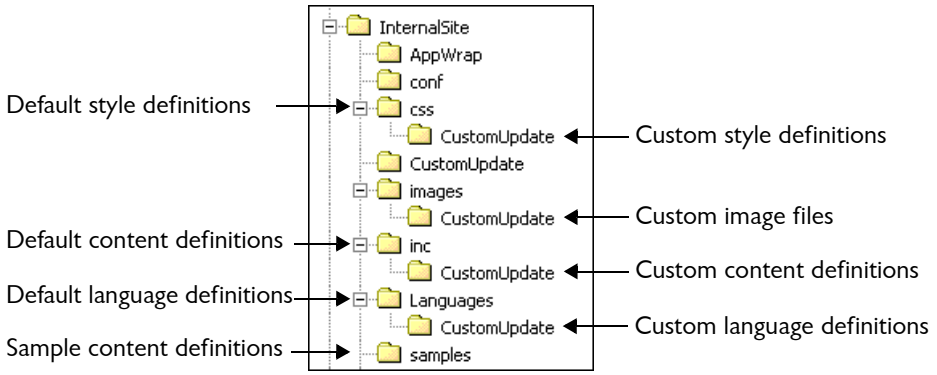
> **Note**
> Before you start customizing the pages, make sure you read the general instructions provided in "Before You Begin" on page 48.

The changes you can make in general user interaction pages are affected by the files located under this folder:

`…\Whale-Com\e-Gap\Von\InternalSite`

Figure 5 illustrates the structure of the `InternalSite` folder, including the locations where you should place the files that you wish to customize:

Figure 5. Definitions of General User Interaction Pages



You can customize the following aspects of the general user interaction pages:

- Customize **content** by changing parameters in `.inc` files that you place in this folder:

  `…\Whale-Com\e-Gap\Von\InternalSite\inc\CustomUpdate`

  For a detailed description of the changes you can make, see "Content Changes in General User Interaction Pages" on page 51.

> **Note**
> You can either use the sample `.inc` files supplied with the IAG as a basis for your changes, or create your own files. Sample files are located under this folder:
>
> `…\Whale-Com\e-Gap\Von\InternalSite\samples`
>
> If you create your own files, make sure you use the same parameter names as those that are used in the supplied `.inc` files. If you do not use identical names, your definitions might not be effective, since the IAG relates to these specific names.

- **Graphic** files you can change include:
  - Submit, Reset, and Go buttons
  - Logo

  Place the custom graphic files under this folder:

  `…\Whale-Com\e-Gap\Von\InternalSite\Images\CustomUpdate`

Note

When you use a custom graphic file, note the following:

- The name of the graphic file must correlate with the name in the relevant content file:
  - Submit, Reset, and Go buttons in the `customDefault.inc` file.
  - Logo in the `logo.inc` file.
- Change the path of the graphic file in the content file to the `Images\CustomUpdate` folder; be sure to make changes to the content file in the `inc\CustomUpdate` folder.

- Change **style** definitions, such as fonts and colors, in style sheets that you place in this folder:

  `…\Whale-Com\e-Gap\Von\InternalSite\css\CustomUpdate`

  - For PCs: `template.css`
  - For handheld devices: `PDA_template.css`

Note

- Changing the default style definitions requires a working knowledge of cascading style sheets.
- You can use the default style sheets as a basis for your changes; those are located under:

  `…\Whale-Com\e-Gap\Von\InternalSite\css`

- **Language definitions** are described in "Multi-Language Support" on page 77.

Content Changes in General User Interaction Pages

This section describes the content changes you can make in the general user interaction pages, as described in "Customizing General User Interaction Pages" on page 49:

- The following files control the positioning of the visual elements on the page:
  - For PCs: `logo.inc`, `waves.inc`, and `bottomText.inc`, as illustrated in Figure 6 on page 52.
  - For handheld devices: `PDA_logo.inc`, `PDA_waves.inc`, and `PDA_bottomText.inc`, as illustrated in Figure 7 on page 53. Note that `PDA_bottomText.inc` is empty by default.

- The file `customDefault.inc` controls the actual display of some of the elements on the page, such as whether to display the "waves" graphic or not, and some of the page parameters, such as the contact email address. Those are listed in Table 5 on page 53. Note that, by default, those parameters do not affect the display on handheld devices.

For example: in the file `customDefault.inc`, you define the image of the Submit button; in the file `waves.inc`, you determine where on the page the Submit button is displayed.

Figure 6. Sample User Interaction Page on a PC

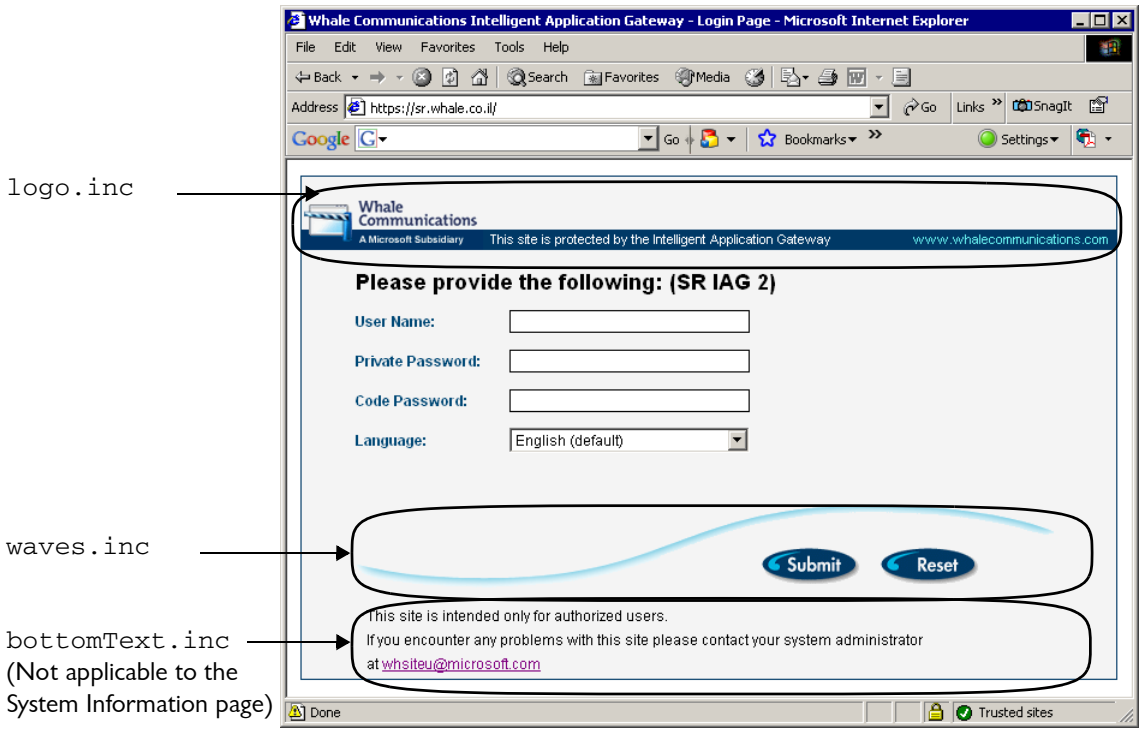


Figure 7. Sample User Interaction Page on a Handheld Device

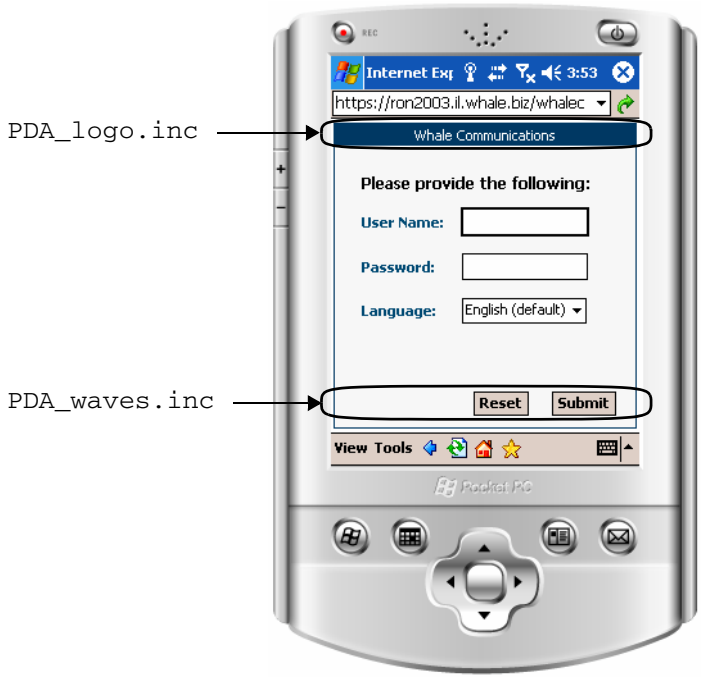


Table 5. General User Page Parameters Controlled by customDefault.inc

ID/Parameter Name	Description
1.* `logoLink`	URL of the link users access when clicking the `whaleLink` address, defined in `Logo.inc`
2. `administrator MailAddress`	Contact email address
3. `viewWaves`	Display of “waves” graphic
4. `submitButton` `submitButtonMouseOver` `resetButton` `resetButtonMouseOver`	Images of the Submit button

Table 5. General User Page Parameters Controlled by customDefault.inc

ID/Parameter Name	Description
5. `goButton` `goButtonMouseOver`	Images of buttons in "Browse additional internal application" page
6. `viewPopupMessage`	Display of message when site is opened as a popup (new window)
7. `MSHotfixURL`	URL of the link where users running Windows XP SP2 can download the Microsoft® SP2 hotfix, if the hotfix is not installed on their computer.

* The numerals correlate with those that are used in the sample `customDefault.inc` file.

Portal Homepage Configuration

You can make various changes in the standard content items and in the look-and-feel of the default portal homepage, or define your custom portal homepage.

This section provides guidelines and instructions for:

- Customizing the default portal homepage that is supplied with the IAG, as described in "Customizing the Default Portal Homepage" on page 54.
- Using your own, custom portal homepage, as described in "Using a Custom Portal Homepage" on page 61.

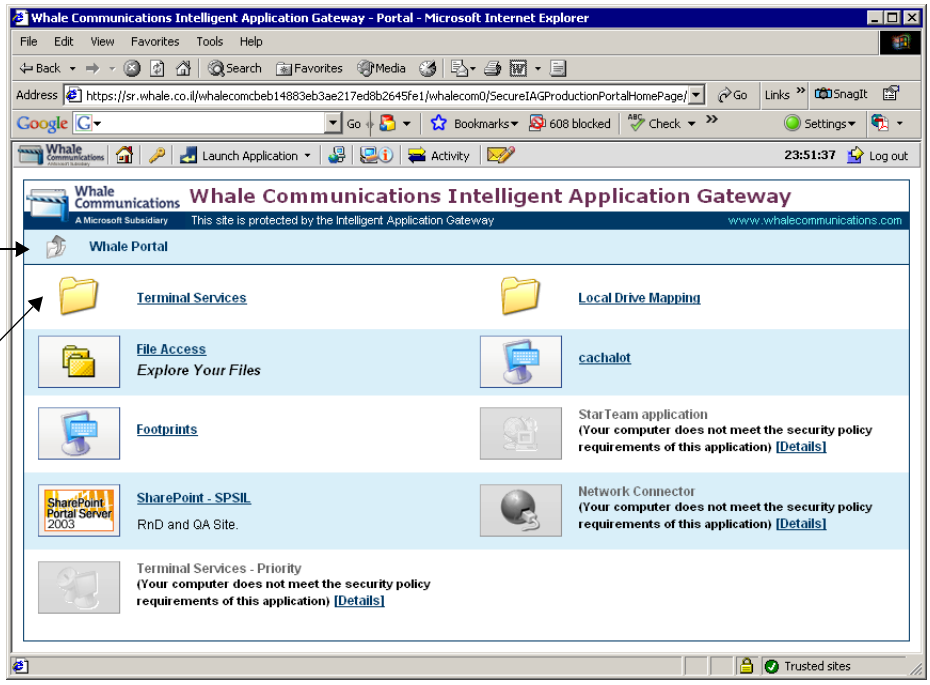
Customizing the Default Portal Homepage

The IAG is supplied with a ready-made portal homepage, illustrated in Figure 8. This section describes how you can customize the default portal homepage, to match your corporate look-and-feel.

Figure 8. Sample Portal Homepage with Links

Tip: Name of the "root" folder is the name of the Whale Portal application, as defined in the Portal Link tab of the Application Properties dialog box, in the "Portal Application Name" field

Tip: "Folder" option is configured in the Portal Link tab of Add Application Wizard or Application Properties dialog box



Tip

You can personalize the default portal homepage per-user, by configuring authorization for the applications enabled through the trunk. For a full description, refer to the *Intelligent Application Gateway User Guide*, to "Users Setup" on page 32.

Note

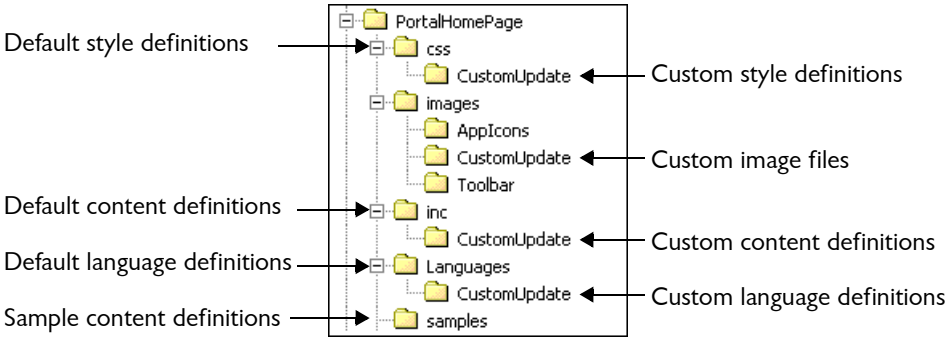
Before you start customizing the portal homepage, make sure you read the general instructions provided in "Before You Begin" on page 48.

The changes you can make in the default portal homepage are affected by changes in files located under this folder:

`…\Whale-Com\e-Gap\Von\PortalHomePage`

Figure 9 illustrates the structure of the `PortalHomePage` folder, including the locations where you should place the files that you wish to customize:

Figure 9. Portal Homepage Definitions



You can customize the following aspects of the default portal homepage:

- Customize **content** by changing `.inc` files that you place in this folder:

  `...\Whale-Com\e-Gap\Von\PortalHomePage\inc\CustomUpdate`

  You can customize the toolbar and logo of the page, as described in "Content Changes in the Default Portal Homepage" on page 57.

> **Note**
> You can either use the sample `.inc` files supplied with the IAG as a basis for your changes, or create your own files. Sample files are located under this folder:
>
> `...\Whale-Com\e-Gap\Von\PortalHomePage\samples`
>
> If you create your own files, make sure you use the same parameter names as those that are used in the supplied `.inc` files. If you do not use identical names, your definitions might not be effective, since the IAG relates to these specific names.

- Place custom **graphic** files in this folder:

  `...\Whale-Com\e-Gap\Von\PortalHomePage\Images\CustomUpdate`

> **Note**
> The names of the graphic files must correlate with the names in the relevant `.inc` file.

Note

When you use a custom graphic file, note the following:

- The name of the graphic file must correlate with the name in the relevant `.inc` file.
- Change the path of the graphic file in the `.inc` file to the `Images\CustomUpdate` folder; be sure to make changes to the `.inc` file in the `inc\CustomUpdate` folder.

- Change **fonts and colors** in style sheets that you place in this folder:

  `…\Whale-Com\e-Gap\Von\PortalHomePage\css\CustomUpdate`

  - For PCs: `portal.css`
  - For handheld devices: `PDA_portal.css`

Note

- Changing the default style definitions requires a working knowledge of cascading style sheets.
- You can use the default style sheets as a basis for your changes; those are located under:

  `…\Whale-Com\e-Gap\Von\PortalHomePage\css`

- **Language definitions** are described in "Multi-Language Support" on page 77.

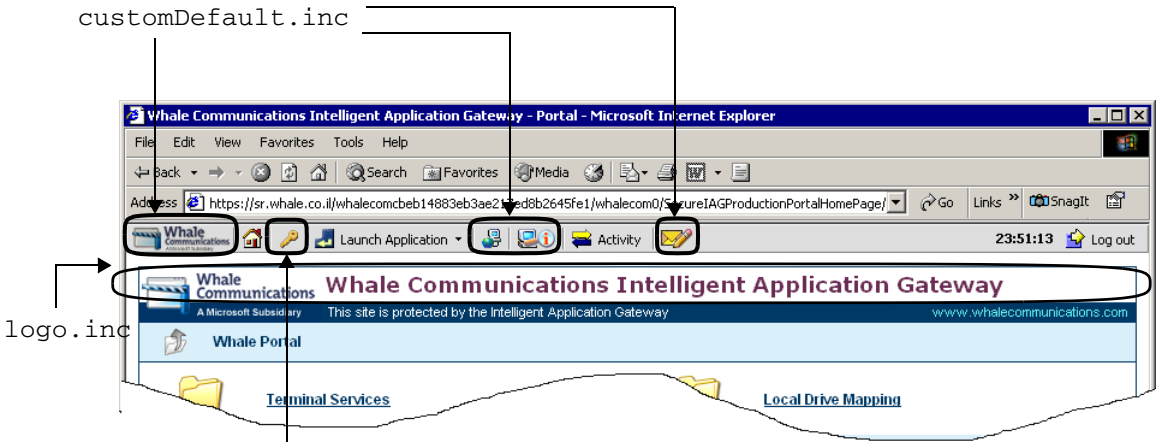
Content Changes in the Default Portal Homepage

This section describes the content changes you can make in the default portal homepage, as described in "Customizing the Default Portal Homepage" on page 54, including visuals and functionality:

- Figure 10 on page 58 illustrates the visual items that can be customized on a portal homepage on a PC.
- Figure 11 on page 58 illustrates the visual items that can be customized on a portal homepage on a handheld device.
- Table 6 on page 59 lists the items that are controlled by the `customDefault.inc` file. The numerals in the table correlate with those that are used in the sample `customDefault.inc` file.

These changes do not include the portal name, defined in the Create New Trunk Wizard, and application properties defined in the Add Application Wizard, including the application icon, name, and short and long descriptions.

Figure 10. Sample Portal Homepage on a PC



Controlled by the option "Enable Users to Manage Their Credentials", configured in the Authentication tab of the Advanced Trunk Configuration window

Figure 11. Sample Portal Homepage on a Handheld Device

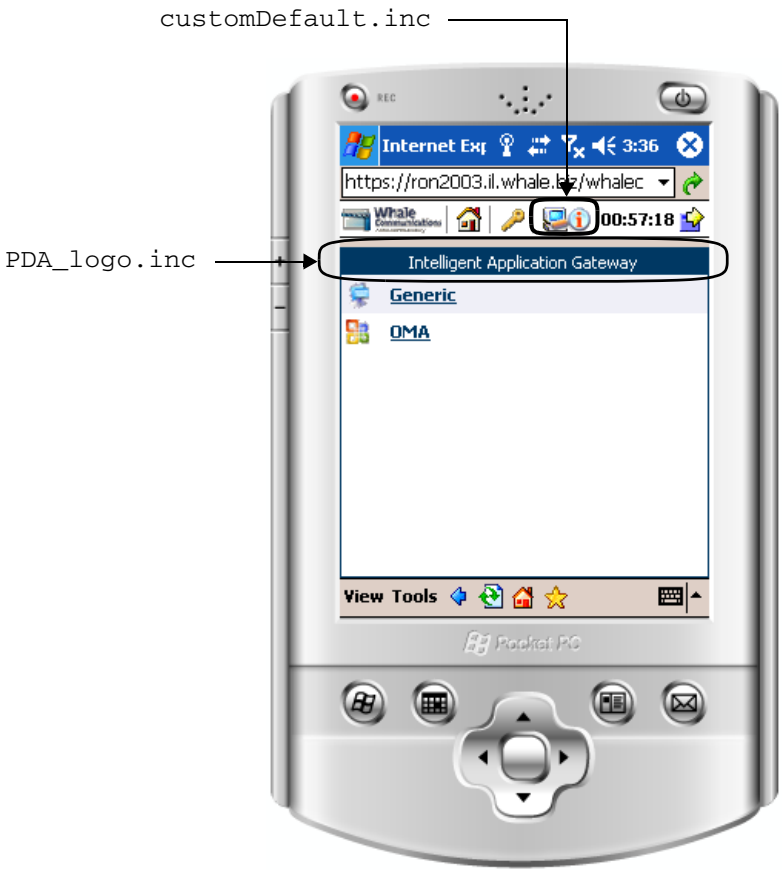


Table 6. Items Controlled by customDefault.inc

ID/Parameter Name	Description
1. `logoLink **`	Defines the URL of the link users access when clicking the `whaleLink` address, defined in `Logo.inc`.
2. `toolbarBgColor **`	Toolbar background color.
3. `toolbarImageSrc **` `toolbarImageLink **`	Toolbar logo image parameters.
4. `enableSystemInformation **`	Enable/disable "System information" button.
5. `informationPage *`	Enable/disable "Information page" button. To enable this button, place the information page in the `CustomUpdate` folder and enter the URL here.
6. `enableBrowseAdditional InternalLocation *`	Enable/disable "Browse additional internal applications" button.
7. `administratorMailAddress *`	Enable/disable "E-mail system administrator" button and define email address. If no address is defined here, the button is not displayed in the toolbar.

Table 6. Items Controlled by customDefault.inc (Cont'd)

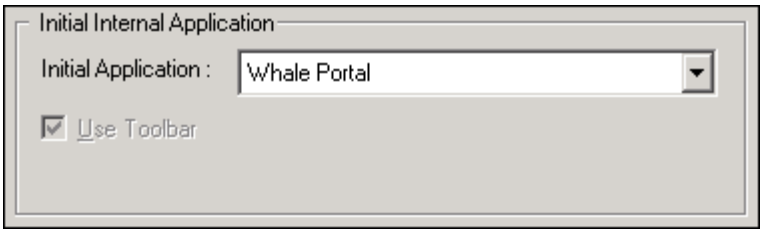
ID/Parameter Name	Description
8. `enableOffLineInstallation *` `offlineInstallationURL *`	Controls the appearance and functionality of the Whale Client Component Installer. For more details, refer to the *Intelligent Application Gateway User Guide*, to "Whale Client Components Installer" on page 154. • `enableOffLineInstallation`: enable/disable "Whale Client Components Installer" button. • `offlineInstallationURL`: define which installation file is downloaded onto the endpoint computer. For a list of files you can use, refer to the *Intelligent Application Gateway User Guide*, to Table 19 on page 155. **Note:** The button is only visible on endpoint computers running a Windows operating system.
9. `embeddedPortal *`	Enable/disable the use of the Whale portal and toolbar from within another application, such as SAP® Enterprise Portal. Enabling this option results in the following changes at the client-side: • Logout button is not displayed on the portal. • Session Timeout and Scheduled Logoff features are disabled. In order to enable them, refer to "Defining Scheduled Logoff and Session Timeout on a Custom Homepage" on page 65.

* Applicable for PCs only.

** Applicable for both PCs and handheld devices.

Using a Custom Portal Homepage

The definition of which application is displayed to users when they access the site (portal homepage), as well as whether the Whale toolbar is used with this application, is determined in the Configuration program in the "Initial Internal Application" area of the Configuration pane. When you create a Portal trunk, by default the Whale Portal application is selected as the Initial Application, and the Whale toolbar is selected to be used with the portal:



If you wish, you can use a different application as the portal's homepage, as described in this section. You can use one of the following types of applications:

- Any of the applications that are supported by the IAG out-of-the-box.
- A generic application or HTML page. You can find a sample portal page in the following location:
  `…\Whale-Com\e-Gap\Von\PortalHomePage\samples`

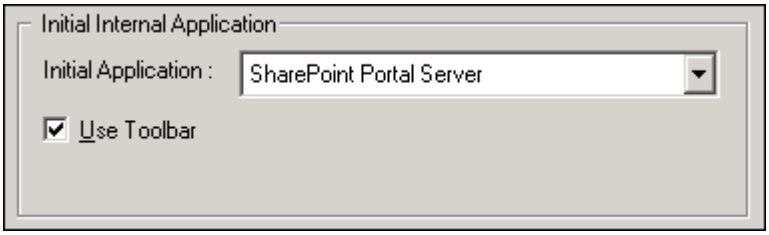
Note that when you use an application other than Whale Portal as your homepage, the following functionality is lost:

- Personalization is not possible. All users are able to view all the links that you configure in your custom portal homepage, regardless of their access permissions. Authorization, however, does function, that is, users are only able to access the applications for which they are authorized; access to unauthorized applications is denied.
- The Folder option, which in the default portal homepage enables you to group together a number of applications, is not applicable.
- The option to control the appearance of an application's link on clients that don't comply with the application's Access policy is not applicable. The policies, however, are enforced.

To use a custom portal homepage:

1. Use the Add Application Wizard to add the application you wish to use as the portal homepage to the trunk. If you are using a generic application or page, and not one of the applications that are supported out-of-the-box, add a generic application that will point to it. For example, to use an HTML page as the homepage, add a Generic Web Application.

2. If the application used as the portal homepage is a generic application, create a ruleset that will enable access to the application. For details, refer to Chapter 6: “Content Inspection”.
3. In the Configuration program, in the “Initial Internal Application” area of the Configuration pane, select the application you wish to use as the portal homepage as the Initial Application. In the following example, the selected application is SharePoint® Portal Server 2003:



4. In order to enable access from the portal homepage to the applications that are configured for this trunk, you can apply one or both of the following methods:
   - Enable access to the applications via the Whale toolbar (recommended): in the “Initial Internal Application” area, make sure the option “Use Whale Toolbar” is activated, as shown in the sample screen above. For each application you add to the portal, make sure the option “Add Link on Whale Portal and Toolbar” is activated. Users will then be able to access the application via the Launch Application toolbar button and list.
   - If you select not to use the Whale toolbar, or if you do use the Whale toolbar and wish links to applications to also be available from the homepage itself and not only from the toolbar, manually add application links on the custom portal homepage, as described in “Adding Application Links on a Custom Portal Homepage” on page 63. Users will then be able to access the application via the portal homepage.
5. If you wish to use the Scheduled Logoff or Session Timeout features in the portal, refer to “Defining Scheduled Logoff and Session Timeout on a Custom Homepage” on page 65.
6. If you do not use the Whale toolbar, you can manually enable the toolbar features on your page, as described in “Adding Links to IAG Features on a Custom Homepage” on page 66.

Note

If you do not use the Whale toolbar, make sure you enable users to return to the portal homepage from the applications and other features that you add to the portal, by doing one of the following:

- Add your own "home" button or link.
- Make sure all applications and features are opened in a new browser window, using the `window.open( )` or `target="_blank"` HTML options. You can see a sample configuration, including the size of the popup windows, in the sample file provided in the following location:

  `...\Whale-Com\e-Gap\Von\PortalHomePage\samples`

Adding Application Links on a Custom Portal Homepage

This section describes how you add links to portal applications on a custom portal homepage. The link format for each of the application types is provided in Table 7 on page 63.

Note

Make sure to add the applications to the trunk, using the Add Application Wizard.

Table 7. Adding Links to Applications on a Custom Homepage

Application Type	Link Details
Web Applications	Use a simple HTML link. **For example:** `http://whale.company/sales`
Client/Server and Legacy Applications	Use a dummy URL with this format: `http://localhost/WhaleSSLVpn/<Application_ID>` Where `<Application_ID>` is the ID of the application. You can view and copy the ID in the Application Properties dialog box, in the General tab.
Browser-Embedded Applications	Use a simple HTML link. **For example:** to create a link to the Terminal Services Web Client, add the following: `TSWeb http://whale.company/TSweb/Default.htm`

Table 7. Adding Links to Applications on a Custom Homepage (Cont'd)

Application Type	Link Details
Certified Endpoint/ Client Certificate Installation *	Add the following links: • Page displayed prior to installation of certificate: `http://localhost:6001/WhaleSep/certrqbi.asp?type=0&Site=<Trunk_Name>` • Certificate status examination and installation pages: `http://localhost:6001/WhaleSep/certckpn.asp?Site=<Trunk_Name>` **Note**: Make sure to add the Certified Endpoint Enrollment application to the trunk, using the Add Application Wizard.

File Access *	Use a simple HTML link, for example:
	`http://localhost:6001/whalefilesharing/`
	Tip: If you wish to restrict users' access to a specific folder, add the following parameters at the end of the link:
	`?Path=<path>&ShowFolders=False`
	Where `<path>` is the full path of the folder users will access.

Web Monitor	Use the following link: `http://localhost:50002`

* Port 6001 is used in this example since it is the default HTTP port of the IIS Default Web Site. If you changed this port on the IAG, make sure you use the correct port number in the link, as well. The definition of this port number is described in "Changing the Port Numbers of the IIS Default Web Site" on page 39.

Tip

If the application is configured with one or more prerequisite applications, add the following parameter to the link:

`WhaleFirstActivation=true`

For example:

```
TSWeb http://whale.company/TSweb/
Default.htm?WhaleFirstActivation=true
```

Defining Scheduled Logoff and Session Timeout on a Custom Homepage

This section describes how you enable the Scheduled Logoff and Inactive Session Timeout options on a custom portal homepage. These options are configured in the Session tab of the Advanced Trunk Configuration dialog box, as described in "Default and Privileged Session Settings" on page 137.

Note

In order to use the Scheduled Logoff option, the option "Automatic Scheduled Logoff After ... Minutes" has to be activated in the Session tab.

To enable Scheduled Logoff and Session Timeout on a custom portal homepage:

1. Using the Editor, open the Application Customization file used with the trunk; if you have not made any changes to the default template, you need to create a custom template.
   - For a description of how you create the Application Customization file, refer to "Creating a Custom Application Customization File" on page 220.
   - For a description of the Editor, refer to "Editor" on page 40.
2. Using the Editor again, access the following folder:

   `…\Whale-Com\e-Gap\von\samples\appwrap`

   Open the file: `TimeoutManipulationAppWrap.xml`
3. From the file you opened in step 2, copy the <DATA_CHANGE> element, and paste it into the Application Customization template you opened in step 1.

Tip

For a detailed description of this element and its sub-elements, see "<DATA_CHANGE>" on page 226.

4. In the Application Customization template, edit the <DATA_CHANGE> element you have copied in step 3, as follows:
   a) In the <URL> element, enter the primary URL of the portal, usually the URL of the trunk's Initial Application:
      - The Initial Application is defined in Configuration program, in the trunk's main Configuration pane.
      - The application's primary URL is defined in the Portal Link tab of the Application Properties dialog box, in the "Application URL" field.
      - Use regular expressions to define the URL.

b) In the <SEARCH> element, enter a string where the Logoff code, which is defined in the <REPLACE> element, will be added. Note that the string you enter here will be replaced by the Logoff code; if you wish to retain the original scrip, add it to the <REPLACE> element, in addition to the Logoff code.

Note

The text within both <SEARCH> and <REPLACE> elements is encoded in Base64 format. In order to add text to the <SEARCH> element or edit text in the <REPLACE> element, you need to convert the text to and from Base64. For details, refer to "Editing Tips" on page 43.

c) Save the Application Customization template as Decrypted.

5. In order to display the Scheduled Logoff counter on the custom portal homepage, take the following steps:

   a) Add an HTML object such as "label", "span", or "td" to the page, with `id="timer"`.

   b) In the Application Customization template, in the <DATA_CHANGE> element you have copied in step 3, change the line:

   ```
   var trunkType = "USERDEFINED"
   ```

   to

   ```
   var trunkType = "PORTAL"
   ```

   Save the Application Customization template as Decrypted.

Adding Links to IAG Features on a Custom Homepage

This section describes how you add links to IAG features, such as the Whale Client Components Installer, on a custom portal homepage where the Whale toolbar is not used. The format you use to create a link to each of the supported IAG features is provided in Table 8 below.

Table 8. Adding Links to IAG Features on a Custom Homepage

Link to:	Details
System Information	Add the following link: `http://localhost:6001/InternalSite/systeminformation.asp?=site_name <Trunk_Name>&secure=<0=no/1=yes>`
Browse Additional Location	Add the following link: `http://localhost:6001/InternalSite/browseAdditional.asp?site_name= <Trunk_Name>&secure=<0=no/1=yes>`

Table 8. Adding Links to IAG Features on a Custom Homepage (Cont'd)

Link to:	Details
Credentials Management	Add the following link: `http://localhost:6001/InternalSite/CredentialsSettings.asp?site_name=<Trunk_Name>&secure=<0=no/1=yes>&orig_url=<Portal_Homepage_URL>`
Whale Client Components Installer	Add the following link: `http://localhost:6001/Secure<Trunk_Name>PortalHomePage/<installer_file_name>` Where: • "Secure" is only required for HTTPS trunks. • `<installer_file_name>` is the name of the file that will be downloaded onto the endpoint computer. For details, refer to the *Intelligent Application Gateway User Guide*, to "Whale Client Components Installer" on page 154. **Note:** Configure the link so that it is visible only on computers running a Windows operating system.

Note

In the above table, port 6001 is used in all the links since it is the default HTTP port of the IIS Default Web Site. If you changed this port on the IAG, make sure you use the correct port number in the links, as well. The definition of this port number is described in "Changing the Port Numbers of the IIS Default Web Site" on page 39.

Customizing Certified Endpoint Enrollment Pages

Certified Endpoint Enrollment pages are the pages that are presented to users when the "Make this computer certified" link is pressed in the portal homepage. For more information, refer to the *Intelligent Application Gateway User Guide*, to the chapter "Endpoint Security", to the section "End-User Interaction (Local CA Only)" on page 140.

This section describes how you can adjust the visual aspects, standard text attributes, and status of the fields in these pages. The changes you make affect all of the relevant parameters in these pages; it is not possible to assign a different color scheme, logo, or standard text to each of the pages in this group.

For example: if you decide to change the logo graphic and text, these will automatically be changed uniformly on all pages. Pages where there was previously only a logo and no logo text will now include all of the new definitions.

Note

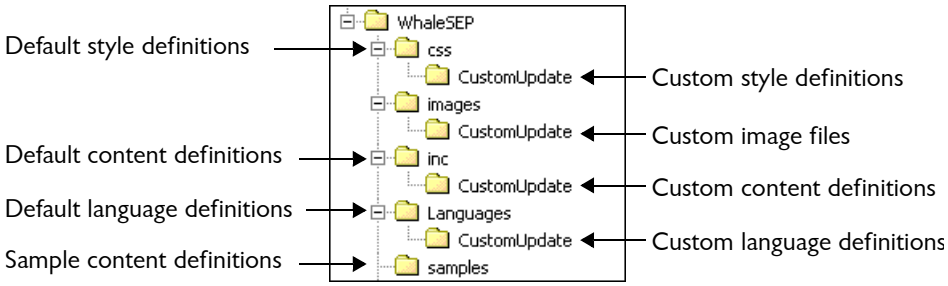
Before you start customizing the pages, make sure you read the general instructions provided in "Before You Begin" on page 48.

The changes you can make in the Certified Endpoint Enrollment pages are affected by changes in files located under this folder:

`…\Whale-Com\e-Gap\Von\WhaleSEP`

Figure 12 illustrates the structure of the `WhaleSEP` folder, including the location where you should place the files that you wish to customize:

Figure 12. Definitions of Certified Endpoint Enrollment Pages



You can customize the following aspects of all Certified Endpoint Enrollment pages:

- Customize **content** by changing parameters in `.inc` files that you place in this folder:

  `…\Whale-Com\e-Gap\Von\WhaleSEP\inc\CustomUpdate`

  For a detailed description of the changes you can make, see "Content Changes in Certified Endpoint Enrollment Pages" on page 70.

Note

You can either use the sample `.inc` files supplied with the IAG as a basis for your changes, or create your own files. Sample files are located under this folder:

`…\Whale-Com\e-Gap\Von\WhaleSEP\samples`

If you create your own files, make sure you use the same parameter names as those that are used in the supplied `.inc` files. If you do not use identical names, your definitions might not be effective, since the IAG relates to these specific names.

- Place custom **graphic** files under this folder:

  `…\Whale-Com\e-Gap\Von\WhaleSEP\Images\CustomUpdate`

Note

When you use a custom graphic file, note the following:

- The name of the graphic file must correlate with the name in the relevant `.inc` file.
- Change the path of the graphic file in the `.inc` file to the `Images\CustomUpdate` folder; be sure to make changes to the `.inc` file in the `inc\CustomUpdate` folder.

- Change **font and colors** in the `cert.css` file, that you place in this folder:

  `…\Whale-Com\e-Gap\Von\WhaleSEP\css\CustomUpdate`

Note

- Changing the default definitions requires a working knowledge of cascading style sheets.
- You can use the following file as a basis for your changes:

  `…\Whale-Com\e-Gap\Von\WhaleSEP\css\cert.css`

- **Language definitions** are described in "Multi-Language Support" on page 77.

Content Changes in Certified Endpoint Enrollment Pages

This section describes the content changes you can make in the default Certified Endpoint Enrollment pages, as described in "Customizing Certified Endpoint Enrollment Pages" on page 67, including visuals and functionality:

- Figure 13 on page 70 shows a sample user interaction page, and indicates the items that can be customized.
- Table 9 on page 71 lists the items that are controlled by the `info.inc` file.

Figure 13. Content Items You Can Customize in Certified Endpoint Enrollment Pages

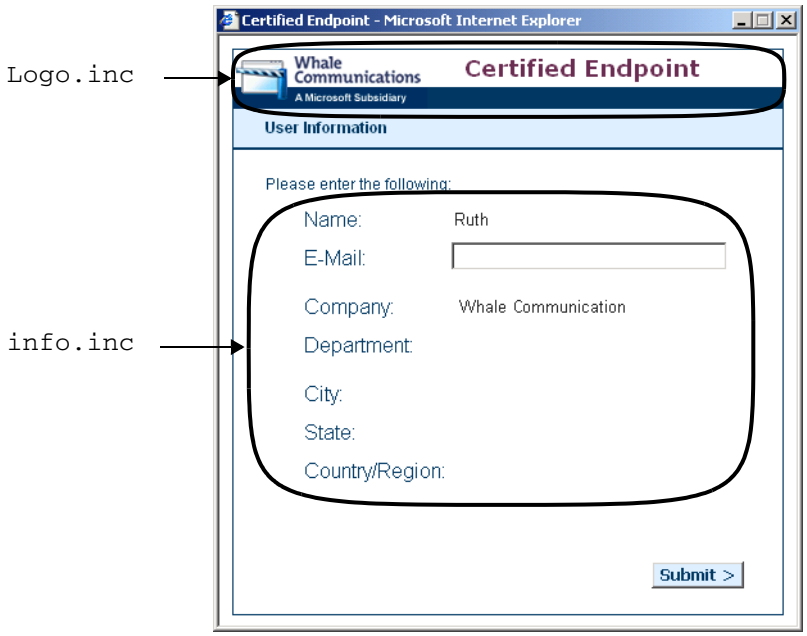


Table 9. Customizing Certified Endpoint Enrollment Pages in info.inc

ID/Parameter Name	Description

`editCommonName`
`editEmail`
`editCompany`
`editDepartment`
`editLocalCity`
`editState`
`editCountry`
	Determines the status for each of the fields. You can select one of the following options:
	• FIELD_READONLY
	• FIELD_EDITABLE
	• FIELD_HIDDEN
	Note: The content of the CommonName, Company, Department, LocalCity, State, and Country/Region fields are automatically filled in, based on the certificate, therefore it is recommended that these fields retain their default READONLY status.

Changing File Access Language Definitions

For the **File Access application**, you can change the language definitions of the end-user pages, in the following location:

`…\Whale-Com\e-Gap\Von\FileAccess\Languages`

Language definitions are described in “Multi-Language Support” on page 77.

Note

Before you start customizing the File Access language definitions, make sure you read the general instructions provided in “Before You Begin” on page 48.

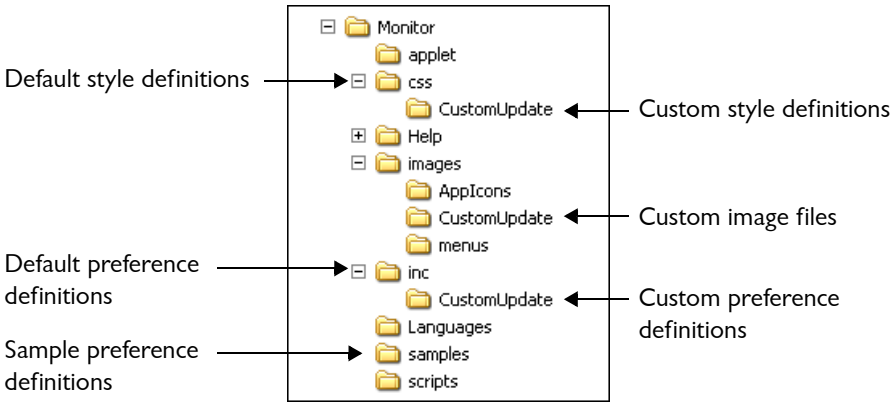
Customizing the Web Monitor Windows

This section describes customizations that you can make to the Web Monitor windows. The changes you can make are affected by the files located under this folder:

`…\Whale-Com\e-Gap\Von\Monitor`

Figure 14 illustrates the structure of the `Monitor` folder, including the locations where you should place the files you wish to customize:

Figure 14. Definitions of Web Monitor Windows



Note

Before you start customizing the Web Monitor, make sure you read the general instructions provided in "Before You Begin" on page 48.

You can customize the following aspects of the Web Monitor:

- **Preference** definitions, including:
  - **Web Monitor preferences**, in the file `customDefault.inc`, described in "Customizing Web Monitor Preferences" on page 74.
  - The appearance of **charts,** in the file `chartParams.inc`, described in "Customizing Web Monitor Charts" on page 75.
  - The appearance of the **strip at the top of the monitor windows**, in the file `LogoMonitor.inc`.

  To edit the default definitions, make your changes under the following custom folder:

  `…\Whale-Com\e-Gap\von\Monitor\inc\CustomUpdate`

Note

You can either use the sample `.inc` files supplied with the IAG as a basis for your changes, or create your own files. Sample files are located under this folder:

`…\Whale-Com\e-Gap\Von\Monitor\samples`

If you create your own files, make sure you use the same parameter names as those that are used in the sample `.inc` files. If you do not use identical names, your definitions might not be effective, since the IAG relates to these specific names.

- Replace the default **graphic** files used in the strip at the top of the monitor windows, or add graphics to the display. Place the custom graphic files under this folder:

  `…\Whale-Com\e-Gap\Von\Monitor\Images\CustomUpdate`

Note

When you use a custom graphic file, note the following:

- The name of the graphic file must correlate with the name in the `LogoMonitor.inc` file.
- Change the path of the graphic file in the `LogoMonitor.inc` file to the `Images\CustomUpdate` folder; be sure to make changes to the `LogoMonitor.inc` file in the `inc\CustomUpdate` folder.

- Change **style** definitions, such as fonts and colors, in template files that you place in this folder:

  `…\Whale-Com\e-Gap\Von\Monitor\css\CustomUpdate`

  - The file `monitor.css` holds definitions that apply to all browsers.
  - The file `monitor_ie.css` is holds definitions that apply to Internet Explorer browsers only.

Note

- Changing the style definitions requires a working knowledge of cascading style sheets.
- You can use the files under this folder as a basis for your changes:

  `…\Whale-Com\e-Gap\Von\Monitor\css`

Customizing Web Monitor Preferences

The following table lists the parameters you can customize in the file `customDefault.inc`, as described in "Customizing the Web Monitor Windows" on page 72.

Table 10. Web Monitor Preferences—customDefault.inc

ID/Parameter Name	Description
`1.* weekStartDay`	First day of the week: • `0`: Sunday • `1`: Monday Affects the Session Monitor, Application Monitor, and User Monitor Statistics windows, and the Event Query windows
`2. dateFormat`	Date format: • `0`: dd/MM/yyyy • `1`: MM/dd/yyyy
`3. monitorChartsRefreshRate`	Refresh rate of the Session Monitor, Application Monitor, and User Monitor Current Status windows, in seconds.
`4. monitorListsRefreshRate`	Refresh rate of the Session Monitor, Application Monitor, and User Monitor Active Sessions windows, in minutes.
`5. lineGraphRefreshRate`	Refresh rate of the line charts generated from within the Session Monitor, Application Monitor, and User Monitor Current Status windows, in seconds.
`6. lineGraphPoints`	Number of intervals that are displayed on a single page in the default view of the following windows: • Query results, in the Session Monitor and Application Monitor Statistics windows. • Monitor Over Time windows, accessed from the Session Monitor, Application Monitor, and User Monitor Current Status windows.

Table 10. Web Monitor Preferences—customDefault.inc (Cont'd)

ID/Parameter Name	Description
7. `statisticsLimit`	Maximal number of intervals that can be queried in the Session Monitor and Application Monitor Statistics windows. **Note:** A value of over 1,500 intervals is not recommended and might slow down the monitor's performance considerably.
8. `eventViewRefreshRate`	Refresh rate of the Event Viewer window, in seconds.
9. `chartLimit`	Maximal number of items that can be viewed in column charts generated from within the Application Monitor Current Status and Statistics windows.
10. `showAppID`	Determines whether the application ID is displayed in the Application Monitor - Current Status window.

* The numerals correlate with those that are used in the `customDefault.inc` file.

Customizing Web Monitor Charts

This section describes the parameters you can customize in the file `chartParams.inc`, as described in "Customizing the Web Monitor Windows" on page 72:

- Figure 15 on page 76 illustrates the aspects of the charts which are customizable.
- Table 11 on page 76 lists the corresponding parameters in the file.

Figure 15. Customizable Aspects of Web Monitor Charts

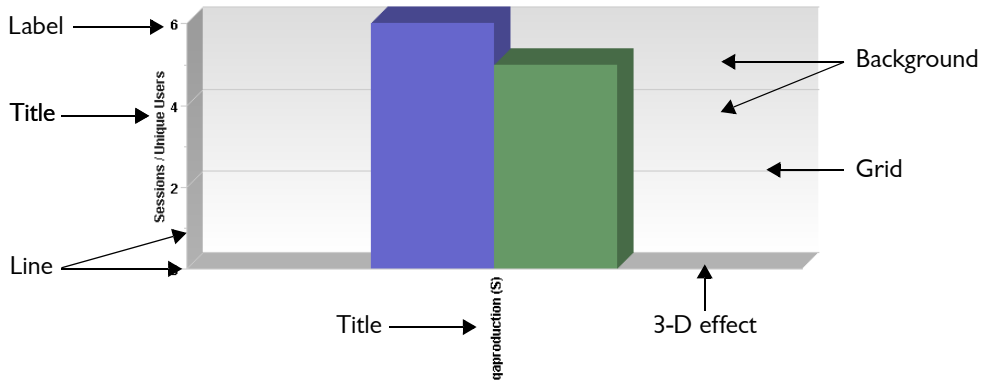


Table 11. Chart Parameters—chartParams.inc

ID/Parameter Name	Description
1.* `threeD`	3-D effects.

2. `plotAreaColor`
`plotAreaGradient`
`plotAreaSecondaryColor`
	Background color.
	Gradient of the background color is defined by `plotAreaGradient`, as follows:
	• `0`: left/right mirrored
	• `1`: top/bottom mirrored
	• `2`: top to bottom
	• `3`: left to right

3. `gridColor`	Grid color.
4. `labelColor`	Label color.
5. `lineColor`	Line color.
6. `titleColor`	Title color.
7. `font`	Font family, size, and style.

* The numerals correlate with those that are used in the `chartParams.inc` file.

Multi-Language Support

The IAG supports multiple languages for text that appears in the windows and messages of the IAG pages with which users interact, including:

- General user interaction pages, described in “Customizing General User Interaction Pages” on page 49.
- Default portal homepage, as described in “Customizing the Default Portal Homepage” on page 54.
- Certified Endpoint Enrollment pages, as described in “Customizing Certified Endpoint Enrollment Pages” on page 67.
- File Access application, as described in “Changing File Access Language Definitions” on page 71.

Out-of-the-box support includes English (default), German, French, and Spanish. You can change the default text of the languages that are supported out-of-the-box, or add support for additional languages, as required.

Tip

The language used on the IAG pages is selected when users access the site. You define the method that is used to select the language in the Authentication tab of the Advanced Trunk Configuration window, in the option “Enable Users to Select Language”. For details, refer to “Configuration in the Authentication Tab” on page 82”.

This section describes the following:

- How you can change text strings in the languages that are supported out-of-the-box, such as changing the company name in the title of the browser window, in “Editing Out-Of-The-Box Languages” on page 78.
- How you can configure support for additional languages, in “Configuring Support for Additional Languages” on page 78.
- Guidelines for editing and writing strings that include special characters and for changing specific application strings, in “General Guidelines” on page 79.

Editing Out-Of-The-Box Languages

This section describes how you can edit the text of the languages that are supported out-of-the-box: English, German, French, and Spanish.

To edit the text of an out-of-the-box language:

1. Copy the relevant file from the relevant `Languages` folder to the relevant `Languages\CustomUpdate` folder:
   - English: `sample_default.xml`
   - German: `de.xml`
   - French: `fr.xml`
   - Spanish: `es.xml`
2. If you are editing strings in the English language, rename the file `default.xml`. No rename is required for the other languages.
3. In the file you copied in step 1, edit the required strings, using a text editor that supports UTF-8, such as Notepad.

> **Tip**
> This file can be partial, including only the strings you wish to edit. Other strings will be taken from the default system pages.

Configuring Support for Additional Languages

This section describes how you can configure support for languages that are not supported out-of-the-box.

To configure a custom language:

1. Copy the file `sample_default.xml` from the relevant `Languages` folder to the relevant `Languages\CustomUpdate` folder.
2. In the file you copied in step 1, edit the required strings, using a text editor that supports UTF-8, such as Notepad.

> **Tip**
> This file can be partial, including only the strings you wish to edit. Other strings will be taken from the default English pages.

3. Rename the file `<Language_ID>.xml`, where `Language_ID` is defined as follows:
   - If the user selects the language, that is, the option “Enable Users to Select Language” is activated in the Authentication tab, `<Language_ID>` can be any string.

     For example: for Russian, name the file `Russian.xml`

- If the language is automatically selected based on the browser definitions, that is, the option “Enable Users to Select Language” is disabled in the Authentication tab, `Language_ID` must use the same language ID used by the browser.

  For example: for Russian, the file **must** be named `ru.xml`

> **Tip**
> To see the browser’s definition of languages, in Internet Explorer, on the **Tools** menu, select **Internet Options**, then click **Languages**.

4. If required, repeat steps 1–3 for each additional language you wish to enable.
5. Copy the `Languages.xml` file from the `InternalSite\Languages` folder to the `InternalSite\Languages\CustomUpdate` folder.
6. Edit the file you copied in step 5 to include all the languages you defined in steps 1–4, by creating a `<Language>` tag for each language, using the following convention:

   ```
   <Language id="Language_ID">Language_Name</Language>
   ```

   Where:

   - `Language_ID` is the same `Language_ID` you used when naming the file containing the strings, in step 3.
   - `Language_Name` is the name that will be displayed to users in the Language selection box, in the Login page, if the user is prompted to select a language.

   For example: for Russian, add the following line:

   ```
   <Language id="ru">Russian</Language>
   ```

General Guidelines

Follow the guidelines provided in this section when editing text for both out-of-the-box and custom languages.

Strings That Include Special Characters

Use CDATA if your string includes any of the following special characters:
`< > " ' &`

For example:

```
<String id="13"><![CDATA[If this prompt appears, <br>click
"Yes" to install the components.]]></String>
```

Note

Some of the strings contain variables, defined by `%<num>` or `{<num>}`.

For example:

```
<String id="139">Failed to log on user %1</String>
```

Where `%1` is the user name.

Application-Specific Strings

In order to change specific application strings, that is, the application's name, description, and long description, as displayed on the portal homepage, add the following lines for each application you wish to change:

- `<String id="Name_ApplicationID">Name</String>`
- `<String id="ShortDesc_ApplicationID">Description</String>`
- `<String id="LongDesc_ApplicationID">Long Description</String>`

You can copy the Application ID from the General tab of the Application Properties dialog box.

Chapter 4

Access Control

This chapter describes the following topics:

- Authentication: the Intelligent Application Gateway (IAG) supports strong two-factor authentication methods, and can leverage existing authentication infrastructures, such as RSA® SecurID®, VASCO and LDAP, without needing to touch the application servers. Authentication is performed by the IAG at the network perimeter, so non-authenticated users' traffic never reaches the intranet. For details, refer to "Authentication" on page 81.
- Server Name Translation and Initial Host Selection: in Basic trunks, where you do not apply Host Address Translation (HAT), you can use the Server Name Translation option to prevent exposure of the real names of your web servers to users outside the organization. You can also use this feature for Initial Host Selection, whereby the IAG transparently redirect requests to alternative servers within your organization, based on users authentication credentials.
    - For a description of Server Name Translation, refer to "Server Name Translation" on page 125.
    - For a description of Initial Host Selection, refer to "Initial Host Selection" on page 128.

Authentication

The IAG enables you to control access to your internal network by checking users against an authentication database, and opening a connection only for users who authenticate successfully, whereas unknown parties do not gain inappropriate access. Access is granted per user, and each authentication instance is only valid for one connection. The IAG can seamlessly integrate with numerous authentication schemes, even if the application being protected has no inherent support for the method you choose to implement, where the IAG serves as a client of the third-party authentication server.

The IAG also enables non-intrusive, forced periodic re-authentication, by applying a Logoff Scheme. After a pre-determined time, users must re-enter credentials to continue working. If they do, they resume right where they left off; if they do not, their sessions are terminated.

The IAG supports leading third-party authentication products such as: RSA SecurID, RADIUS, various LDAP products, LDAP Client Certificate, TACACS, WINHTTP and NT Domain. In addition to the supported third-party schemes, you can apply your own authentication scheme.

When you create an SSL VPN portal or a Webmail or Basic trunk, you configure authentication as follows:

- Session authentication is configured in the Create New Trunk Wizard, in the step "Authentication". You can later edit session authentication settings, including the Logoff Scheme, in the Authentication tab of the Advanced Trunk Configuration window, as described in "Configuration in the Authentication Tab" on page 82.
- Application authentication is configured as follows:
  - In Portal trunks, for Built-In Services, Web Applications, and Browser-Embedded Applications that require users to authenticate, in the Add Application Wizard, in the step "Authentication".
  - In Webmail and Basic trunks, for applications that require users to authenticate, in the Create New Trunk Wizard, in the step "Application Login".

  You can later edit each application's authentication settings in the Web Settings tab of the Application Properties dialog box, as described in "Application Authentication" on page 74.

Additional options are described in:

- "Configuration of the Authentication Servers" on page 89
- "Defining a New Server Type" on page 90
- "Credentials Management by Users" on page 91
- "Change Password Requirements" on page 93
- "Configuring Cross-Site Single Sign On" on page 94

Advanced authentication topics are discussed in:

- "The Authentication Process" on page 95
- "Authentication Pages" on page 96
- "Authentication Schemes" on page 104

Configuration in the Authentication Tab

You can use the Authentication tab to:

- Configure session authentication settings.
- Configure the session's Logoff Scheme.

Figure 16. Authentication Tab

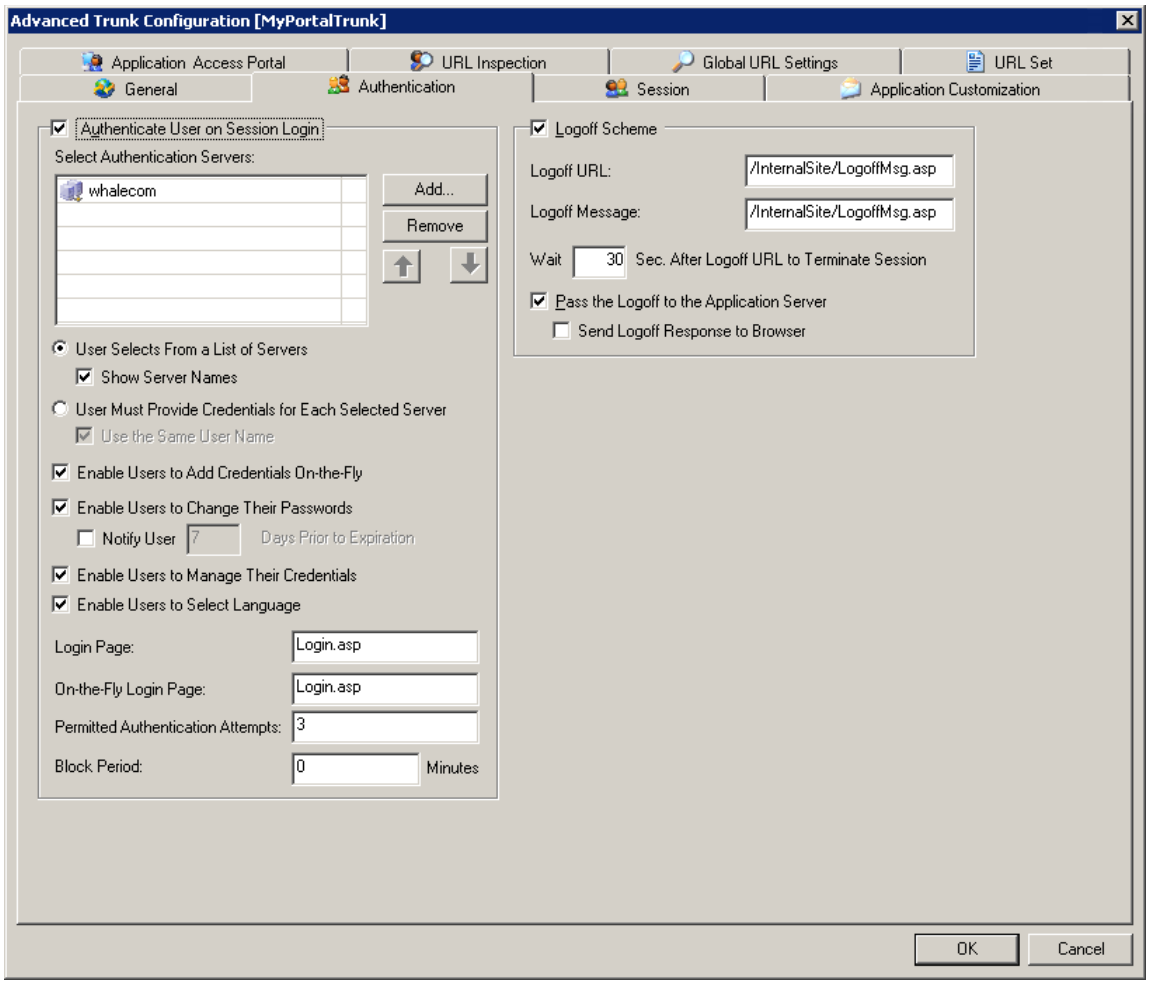


Table 12. Authentication Tab Parameters

Parameter	Description
Authenticate User on Session Login	Activates session authentication and requires users to authenticate when accessing the SSL VPN Portal or Webmail/Basic trunk.
Select Authentication Servers	Select a server or number of servers, against which users will authenticate when accessing the SSL VPN Portal or a Webmail/Basic trunk. To select a server: • Double-click anywhere in the table, or click Add... • In the Authentication and User/Group Servers dialog box, click Help for instructions.

Table 12. Authentication Tab Parameters (Cont'd)

Parameter	Description
User Selects From a List of Servers	When you select this option, users are prompted during login to select the server against which they will authenticate. Users can select to authenticate against any of the servers you select in "Select Authentication Servers", above. If you only select one server, no list is displayed, and users authenticate against the selected server.
Show Server Names	When you activate this option, users are prompted to select a server from a drop-down list, containing all selected servers. When this option is not activated, users are prompted to enter the name of the server against which they wish to authenticate.
User Must Provide Credentials for Each Selected Server	When you select this option, users are prompted during login to authenticate against all the servers you select in "Select Authentication Servers", above. **Note:** If during the initial login users authenticate against a number of servers, the first server on the list is used as the lead server. **For example:** • After a Scheduled Logoff, users are only requested to authenticate against the lead server. • If you use Endpoint Certification, and the option "Verify User Name Against Certificate" is activated, the user name of the lead server is verified against the certificate.
Use the Same User Name	When you activate this option, users are prompted to enter one user name that is used for authentication against all selected servers. When this option is not activated, users are prompted to enter a user name for each of the selected servers.

Table 12. Authentication Tab Parameters (Cont'd)

Parameter	Description
Enable Users to Add Credentials On-the-Fly	Enables users to add authentication credentials in cases where access to the application with the user's current credentials is denied. **For example:** when this option is not activated, the portal homepage displays only applications for which the user has "Allow" authorization permissions. Applications with "View" permissions are not displayed, since the user has no means by which to enter the credentials that are required in order to access the application. When this option is activated, the portal displays links for "View" applications also, and users are prompted to add the required credentials when they attempt to access the application.
Enable Users to Change Their Passwords*	Enables users to change passwords, either when their password expires, or a user-initiated change. **For example:** if a user cannot login because the password has expired prior to the login attempt, they are prompted to change the password. Once they do, login is enabled. This option is applicable only for the following authentication servers: NT Domain, Active Directory®, Netscape® LDAP, Notes® Directory, and Novell® Directory. For a description of the settings required in order to enable users to change their password, refer to "Change Password Requirements" on page 93.
Notify User ... Days Prior to Expiration	Applicable only when the option "Enable Users to Change Their Passwords", above, is activated. Notifies users when their password is about to expire and enables them to change the password prior to its expiration.
Enable Users to Manage Their Credentials	Applicable for Portal trunks only. Adds a "Credentials management" button on the portal homepage and enables end-users to initiate the following options any time during a session: • Add authentication credentials. • Change their password. This option is applicable only when the option "Enable Users to Change Their Passwords", above is activated.* For details, refer to "Credentials Management by Users" on page 91.

Table 12. Authentication Tab Parameters (Cont'd)

Parameter	Description

Enable Users to Select Language	Defines the method that is used to select the language of the IAG pages with which users interact, including text that appears in windows and in messages. Language is selected when users access the site:
	• When this option is activated, a "Language" drop-down box is displayed on the Login page, enabling users to select the language from the list of the languages enabled for the site. The selected language is used for all the computer's sessions with the IAG, until such a time when a user chooses to change the selection in the Login page.
	• When this option is disabled, the endpoint browser automatically selects the language for each user, based on the language that is defined in the browser. In this scenario, the language is selected as follows:
	  • If the browser's selected language is one of the languages that are enabled for the site, it is used in the IAG pages.
	  • For all other languages, the browser uses the default, English language.
	• If you disable this option after the user has already selected a language via the "Language" drop-down box, the user's selected language is still used, regardless of the language that is defined in the browser. Once a language is selected by the user, the language can only be changed by the user selecting a different language in the Login page.
	Tip: For more details on the support of multiple languages, refer to "Multi-Language Support" on page 77.

Login Page	URL of the Login page, that is, the reply script that the IAG sends to users attempting access. Prompts the user for login information. If you wish to use your own Login page instead of the page that is configured here by default, refer to "Using Your Own Authentication Pages" on page 103.

Table 12. Authentication Tab Parameters (Cont'd)

Parameter	Description
On-the-Fly Login Page	URL of the Login page that will be presented to users when they are required to login after the initial login stage, such as when adding authentication credentials. By default, the same page as the initial login page. If you wish to use your own On-the-Fly Login page instead of the page that is configured here by default, refer to "Using Your Own Authentication Pages" on page 103.
Permitted Authentication Attempts	Number of consecutive times users can: • Attempt to login into the site, before the login failure is final. • Attempt to change their password, when the option "Enable Users to Change Their Passwords", above, is enabled. Note that when the attempt to change a password fails because the new password does not comply with the corporate password policy, the attempt is not counted here.
Block Period	Period, in minutes, during which users are blocked from accessing the site after they failed to login or change password for the number of times defined in "Permitted Authentication Attempts", above. **Tip:** Activating this feature helps to prevent "brute force" attacks on the site.
Logoff Scheme	Activating the Logoff Scheme and operates a full and secure logoff and clean-up process for the internal applications. **Tip:** • When the Application Customization option is activated, activating the Logoff Scheme adds a Log Off button to the interface of applications where the button is not an integral part of the interface. For details, refer to Chapter 7: **"Application Customizers"**. • You can also configure an "Automatic Scheduled Logoff", where the IAG automatically triggers the Logoff Scheme after a pre-defined period of time. For details, refer to "Default and Privileged Session Settings" on page 137.

Table 12. Authentication Tab Parameters (Cont'd)

Parameter	Description
Logoff URL	URL of the Logoff page that serves as a trigger for the termination of the session. You can define the Logoff URL on any internal application with a Logoff mechanism.
Logoff Message	Page containing the message that will be sent to the browser when the Logoff Scheme is activated. **Note:** If you activate the option "Send Logoff Response to Browser", below, the application server's response to the logoff request is sent to the browser, and not the message you define here.
Wait ... Sec. after Logoff URL to terminate session	Once Logoff is initiated, the IAG waits for the period of time specified here before closing down the IAG session.
Pass the Logoff to the Application Server	This option is only applicable if you define the "Logoff URL", above, on an internal application with a Logoff mechanism. Selecting this option indicates to the IAG that the Logoff page is hosted on the application server that is defined in "Logoff URL", above, and therefore the logoff request should be forwarded to the application server. In this case, the application will also be terminated. If this option is not selected, once logoff is initiated, and until the IAG session is closed down, no requests will be forwarded to the application server.
Send Logoff Response to Browser	Applicable when the option "Pass the Logoff to the Application Server", above, is activated. When the option "Send Logoff Response to Browser" is activated, the application server's response to the logoff request is forwarded to the requesting browser, and not the message defined in "Logoff Message", above.

* The IAG supports authentication following password changes originating from within an application. This option is pre-defined for out-of-the-box applications, and can be added for custom applications, as part of the custom form definition process. For details, refer to Appendix C: "Form Authentication Engine".

Configuration of the Authentication Servers

This section describes how you define authentication servers in the IAG Configuration program. If you need to define a type of server that is not supported by the IAG by default, refer to "Defining a New Server Type" on page 90.

Note

For all server-types, if a firewall is positioned between the IAG and the authentication server, you need to open the relevant ports on the firewall.

For example: if you are configuring an Active Directory server, which uses port 389, open port 389 on the firewall.

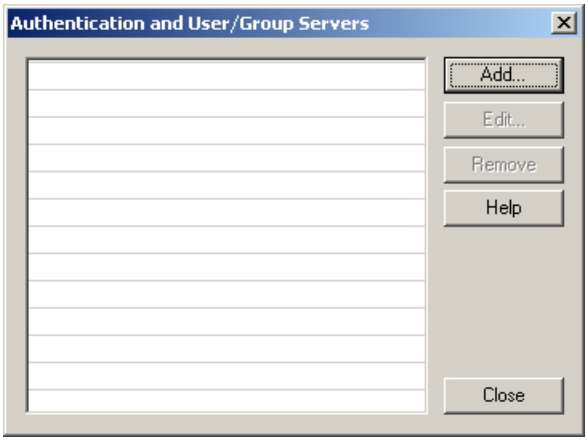
Tip

You can define the local computer's Windows Local Users and Groups Manager as an authentication server, by selecting the NT Domain server in the Add Server dialog box, as described below.

To define authentication servers:

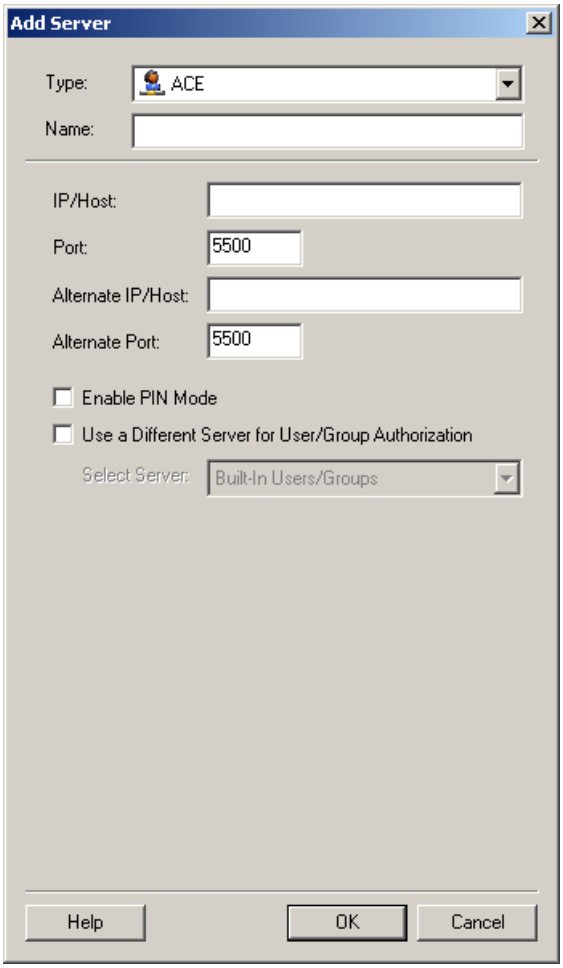
1. In the Configuration program, on the **Admin** menu, click **Authentication and User/Group Servers...**

   The Authentication and User/Group Servers dialog box is displayed.



2. In the Authentication and User/Group Servers dialog box, click Add...

   The Add Server dialog box is displayed.



3. Use the Add Server dialog box to define the server. For details, click Help.
4. Repeat steps 2–3 to define all the required authentication servers.

Defining a New Server Type

This section describes how you define a type of authentication server that is not supported by the IAG by default.

For example:

The Notes LDAP server is configured to recognize users by their Common Name (CN) or Unique ID (UID). If you wish to use a server that recognizes users only by their Unique ID (UID), you need to define a new server type.

To define a new server type:

1. Open the file:

   `…\Whale-Com\e-Gap\von\conf\RepositoryType.xml`

   This file defines the default authentication server types, each in a `<RepositoryType>` element. Find an element that defines a server which is similar to the server you wish to create, and copy it.
2. Access the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\conf\CustomUpdate`
3. In the custom folder, create the file `RepositoryType.xml`; if such a file already exists, use the existing file. Paste the `<RepositoryType>` element you copied in step 1 into the file under the custom folder. Make sure to place it within a `<RepositoryTypes>` element.
4. Change the `RepositoryType` element you copied in step 3, to define the new server type.
5. If the Configuration program is open, close it. From the Windows services, stop and then re-start the `Whale UserMgr` service.

   When you next access the Configuration program, the new server type is added to the list of types you can select in the Add Server dialog box, and can be used for authentication and authorization.

Credentials Management by Users

Credentials management enables users to do the following while working with the portal:

- Add authentication credentials, as described in "Adding Authentication Credentials" on page 92.
- Change their password, as described in "Changing the Password" on page 93.

You enable users' credentials management in the Authentication tab, in the option "Enable Users to Manage Their Credentials".

- When this option is activated, a "Credentials management" button is automatically added on the default portal homepage supplied with the IAG: .
- If you use a custom homepage, you can manually add this button to your page, as described in "Adding Links to IAG Features on a Custom Homepage" on page 66.

Clicking this button opens the Credentials Management window, where users can access one or both of the Credentials Management options via this window.

Note

If the option "Enable Users to Change Their Passwords" is disabled in the Authentication tab, users can only use credentials management to add authentication credentials, and not to change their password. In this setup, when users click the "Credentials management" button, the Credentials Management window is not displayed, and users are directly prompted to enter the new credentials.

Adding Authentication Credentials

Note

Adding authentication credentials is only relevant for applications that are configured with the option "Automatically Reply to Authentication-Specific Authentication Requests". For details, refer to the *Intelligent Application Gateway User Guide*, to "Application Authentication" on page 74.

While users are accessing multiple applications via a single portal, they might need to use more than one set of authentication credentials.

For example: user Sara might need to authenticate against the Telnet application with the user name `sara` and the password `sara_m`, while the File Access application requires her to authenticate with the user name `qa_sara`, and the password `sara`. If Sara is logged in as `sara`, when she tries to access the File Access application, she is requested to enter the File Access credentials, `qa_sara`, in order to access the application.

The portal homepage enables users to supply the required credentials in advance, for all relevant applications, thus eliminating the need to supply those credentials when they actually try to access the application. In the example above, Sara can add her `qa_sara` credentials before she attempts to access the File Access application.

Once users enter authentication credentials for an authentication server, the credentials are used for all the applications that are configured for authentication against this server.

Tip

Adding credentials is accumulative. For example: when you enter two sets of authentication credentials, both are applicable, and each is used with the relevant applications.

Users can add credentials via the “Credentials management” portal button. Depending on the configuration of the IAG, one of the following is displayed:

- The Credentials Management window. Users then click the link **Add Credentials** and enter the required credentials in the Login Page.
- If adding credentials is the only option available via credentials management, users are prompted for credentials as soon as they click the “Credentials management” button.

The Login Page that is displayed to users here is determined by the option “On-the-Fly Login Page”, in the Authentication tab, described in “Configuration in the Authentication Tab” on page 82.

Changing the Password

In the Credentials Management feature, the Change Password option enables users to initiate the changing of their current password, even before it expires. It is applicable for the following authentication schemes: NT Domain, Active Directory, Netscape LDAP, Notes Directory, and Novell Directory.

Users can initiate the changing of their password via the “Credentials management” portal button. In the Credentials Management window, users click the link **Change Password** and enter the required credentials at the prompt.

- In order for this option to be available via credentials management, the option “Enable Users to Change Their Passwords” must be enabled in the Authentication tab.
- For a description of the settings required in order to enable users to change their password, refer to “Change Password Requirements” on page 93.

Change Password Requirements

This section describes the configuration and server settings that are required in order to enable users to change their password, when the option “Enable Users to Change Their Passwords” is enabled in the Authentication tab.

Enabling users to change their password is applicable for the following authentication servers: NT Domain, Active Directory, Netscape LDAP, Notes Directory, and Novell Directory.

Note

This option is not related to application Change Password forms, which can be handled by the Form Authentication Engine.

When configuring any of the supported servers to enable the "change password" capability, follow these guidelines:

- The Server Access Credentials you assign when you configure the authentication server in the IAG Configuration program should enable the changing of passwords.
- The capability to change passwords should also be supported by the authentication server, that is, the server should be configured so that users can change their password. For example, in a Notes Directory server, you should configure the Notes server to allow password changes, in other words, allow LDAP users write access.
- For NT Domain and Active Directory authentication servers only: in order to enable the "change password" functionality, at the IAG, run the following utility:

  `…\Whale-Com\e-Gap\common\bin\fa-allow-nt4shares.bat`

Configuring Cross-Site Single Sign On

This section describes how you configure the IAG to support Single Sign On (SSO) across sites, so that once users log into one site, they can access additional sites without having to re-authenticate.

In order for cross-site SSO to operate, the following is required for all the sites that will be accessed using SSO:

- All trunks must be configured with the same authentication settings, that is, the same authentication server is selected for session login. For details, refer to "Configuration in the Authentication Tab" on page 82.
- The public hostname of all trunks must belong to the same domain.

Tip
The public hostname of a trunk is displayed in the main window of the Configuration program.

To configure cross-site SSO:

1. Access the `CustomUpdate` folder under the `Conf` folder of one of the trunks which will be part of the SSO; if it does not exist, create it:

   `…\Whale-Com\e-Gap\Von\Conf\WebSites\<Trunk_Name>\Conf\CustomUpdate`
2. Copy the file `WFEList.xml` from the trunk's `Conf` folder to the `CustomUpdate` folder; if such a file already exists, use the existing file.
3. At the end of the `WFEList.xml` in the `CustomUpdate` folder, add the following line:

   `<DLL active="1" dll_name="WhlFiltSSO.dll"/>`

4. Copy this file:

   `…\Whale-Com\e-Gap\Von\Conf\WhlFiltSSO.ini`

   into the trunk's `Conf\CustomUpdate` folder (the folder you accessed in step 1).
5. Edit the file you copied in step 4, changing the `Domain` and `Key` fields:
   - `Domain`: domain to which the trunk belongs
   - `Key`: free-text, used for encryption
6. Repeat steps 1–5 for each of the trunks that are part of the SSO.
7. Access the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`

   Copy the following file into the custom folder:

   `…\Whale-Com\e-Gap\von\InternalSite\samples\site_sso.inc`
8. Edit the file you copied in step 7, as follows:
   - `WHL_KEY`: key you entered in step 5
   - `SSO_COOKIE_DOMAIN`: domain you entered in step 5
9. Rename the file you edited in step 8 as follows:

   `<Trunk_Name>sso.inc`

   Where `<Trunk_Name>sso.inc` is the name of one of the trunks that are part of the SSO.

> **Note**
> Regardless of the trunk-type of the SSO sites (HTTP Connections or HTTPS Connections), the file is named `<Trunk_Name>sso.inc`.

10. Repeat step 9 for each of the trunks that are part of the SSO, by copying and renaming the file as many times as required, so that each SSO trunk has one `<Trunk_Name>sso.inc` file under the custom folder `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`

The Authentication Process

The authentication process takes place when the following occurs:

- At the session level: during the initial login attempt, or after a Scheduled Logoff.
- At the application level: when an application is configured to reply to the application's authentication requests, and either the required credentials were not entered by the user, or the user has only View authorization permissions on that application.

Authentication is carried out as follows:

- The User Manager, on the IAG, sends the client (the requesting browser) a Login page, or, if the LDAP Client Certificate scheme is used, the user is prompted to select a client certificate.
- The user submits the authentication data, according to the scheme selected for the trunk or application.
- The User Manager validates the authentication data against the authentication server. In case of failure, the user is prompted to retry to authenticate again, until the number of "Permitted Authentication Attempts", configured in the Authentication tab, is reached.
- If the user needs to continue to authenticate, for example in order to enter a new PIN, they are prompted to continue, until all required credentials are entered and validated.
- If the validation is successful, and providing that all the relevant URL Inspection rules are passed successfully, the User Manager returns the requested page to the user.
- If, on the other hand, validation fails for the number of "Permitted Authentication Attempts", login failure is final. In order to be able to connect from this browser (in case of a session authentication failure), or to access the application (in case of an application authentication failure), the user has to log off the site and attempt to log in again. A new authentication process is then started.

Note

You can configure an Inactive Session Timeout period for the trunk. When the client is inactive for the period of time defined as Inactive Session Timeout, the connection is closed. If the user then retries to connect, it is considered a new connection request, and the user has to go through the authentication process again.
Inactive Session Timeout is defined in the Session tab of the Advanced Trunk Configuration window, as described in "Default and Privileged Session Settings" on page 137.

Authentication Pages

The IAG uses the following Active Server Pages (ASPs) to implement the authentication process:

- `Login.asp`: the first page that is invoked when the user attempts to access the site or the application. Prompts the user for authentication credentials, and passes the credentials to `Validate.asp`.

- `Validate.asp`: receives the user's login credentials from `Login.asp` and validates the credentials against the relevant authentication server or servers.
  - If this stage of the verification is successful, and the authentication process prompts users to change their password, `Validate.asp` activates `LoginChangePassword.asp`.
  - If this stage of the verification is successful, and the authentication process requires additional verification, such as the RSA SecurID New PIN mode, `Validate.asp` activates `LoginContinue.asp`.
  - If this stage of the verification is successful, and the authentication process does not require additional verification, `Validate.asp` activates `PostValidate.asp`.
  - If the verification fails, `Validate.asp` passes the failure to `Login.asp`.
- `LoginChangePassword.asp` (optional): activated in schemes that enable users to change their password. For details, refer to Table 12, "Authentication Tab Parameters", on page 83, to the description of the option "Enable Users to Change Their Password".
  - In cases where the IAG initiates the change, and in Webmail and Basic trunks, this script is activated by `Validate.asp`.
  - In cases where the change is user-initiated, this script is initiated by "Change Password" link in the Credentials Management window. For details, refer to "Credentials Management by Users" on page 91.
- `ValidateChangePassword.asp` (optional): receives the "change password" credentials from `LoginChangePassword.asp` and validates the credentials against the authentication server.
  - If the password change is successful, `ValidateChangePassword.asp` activates `PostValidate.asp`.
  - If the password change fails, `ValidateChangePassword.asp` passes the failure to `LoginChangePassword.asp`.
- `LoginContinue.asp` (optional): activated by `Validate.asp`, in schemes where the user is required to undergo a second stage of authentication. Prompts the user for the "continue" authentication credentials, and passes the credentials to `ValidateContinue.asp`.
- `ValidateContinue.asp` (optional): receives the "continue" authentication credentials from `LoginContinue.asp` and validates the credentials against the authentication server.
  - If the verification is successful, `ValidateContinue.asp` activates `PostValidate.asp`.

    - If the verification fails, `ValidateContinue.asp` passes the failure to `Login.asp`.

- `PostValidate.asp`: activated by `Validate.asp`, `ValidateChangePassword.asp`, or `ValidateContinue.asp`, as applicable, when the user completes the authentication process successfully, and:
    - Adds the user or users to the session.
    - Passes the user's request to the original URL.
    - Activates the Attachment Wiper™, where applicable.
- `LoginChooseUser.asp`: used only for applications where the option "Automatically Reply to Application-Specific Authentication Requests" is activated. Invoked when the user that attempts to access the application has more than one set of credentials that can be used by the application.
- `ValidateChooseUser.asp`: receives the user's login credentials from `LoginChooseUser.asp`, sets the user in the Session Manager and redirects the request to the application.
- `Notes.asp`: applicable only for Domino® iNotes™ applications that are configured with the option "Create Redirect URL". Invoked when the user attempts to access the application.

The ASPs implement several "include" (`.inc`) files, which carry out the required functions. **The default ASPs and include files should not be edited or changed in any way.** If you wish to add functionality to the default pages, or use your own authentication pages, refer to one of the following options:

- Create "hooks", which will be loaded by the default ASPs, as described in "Adding Functionality to the Default Pages" on page 99.
- Retrieve the source IP of the request from the authentication pages, described in "Retrieving the Source IP From the Authentication Pages" on page 101.
- Use custom validation "include" pages, which will be loaded by the default ASPs, as described in "Using Custom Include Pages" on page 101.
- Use your own authentication pages for the authentication process, as described in "Using Your Own Authentication Pages" on page 103.

Tip

When you use the default pages, you can change some aspects of the pages' look-and-feel. For details, refer to "Customizing General User Interaction Pages" on page 49.

Adding Functionality to the Default Pages

This section describes how you can add functionality to the default authentication pages supplied with the IAG. In order to do so, you have to create additional pages, which will be loaded by the default pages.

Caution

You must not change the default authenticating pages supplied with the IAG.

To create a hook that will be loaded by the default authentication pages:

1. Access the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`

2. Under the `CustomUpdate` folder, create an `inc` "hook", naming the file as follows:

   `<Trunk_Name><Secure(0=no/1=yes)><Hook_Name>`

   Where the names of the hooks you can use are listed in Table 13 on page 99. All the files use free text.

   For example:

   For an HTTPS trunk named "WhalePortal", to create a "PrePostValidate" hook, create the file:

   `WhalePortal1PrePostValidate.inc`

Table 13. Authentication Hooks

File/Hook Name	Called By
Login.inc	Login.asp
LoginForm.inc	Login.asp
PreValidate.inc	Validate.asp
ValidateSuccess.inc	Validate.asp
ValidateFailed.inc	Validate.asp
PostValidate.inc	Validate.asp
LoginChangePassword.inc	LoginChangePassword.asp
LoginChangePasswordForm.inc	LoginChangePassword.asp

Table 13. Authentication Hooks (Cont'd)

File/Hook Name	Called By
PreValidateChangePassword.inc	ValidateChangePassword.asp
ValidateChangePassword.inc	ValidateChangePassword.asp
ValidateChangePasswordSuccess.inc	ValidateChangePassword.asp
ValidateChangePasswordFailed.inc	ValidateChangePassword.asp
PostValidateChangePassword.inc	ValidateChangePassword.asp
LoginContinue.inc	LoginContinue.asp
LoginContinueForm.inc	LoginContinue.asp
PreValidateContinue.inc	ValidateContinue.asp
ValidateContinueSuccess.inc	ValidateContinue.asp
ValidateContinueFailed.inc	ValidateContinue.asp
PostValidateContinue.inc	ValidateContinue.asp
PrePostValidate.inc	PostValidate.asp
PostPostValidate.inc	PostValidate.asp
LoginChooseUser.inc	LoginChooseUser.asp
LoginChooseUserForm.inc	LoginChooseUser.asp
Notes.inc	Notes.asp

Tip
The authentication ASPs, which load the hooks listed here, are described in "Authentication Pages" on page 96.

Retrieving the Source IP From the Authentication Pages

The IAG authentication process gets the request's source IP, that is, the IP address of the client from where the request originated. You can retrieve this information during the user authentication process, in order to use it for your own needs.

For example: use the source IP to deny access to pre-defined IP addresses, or send different messages to users accessing the site from different clients.

Note

You can only utilize this functionality if you use the default authentication pages supplied with the IAG.

To retrieve the source IP address:

1. Create an authentication hook, or use an existing hook, as described in "Adding Functionality to the Default Pages" on page 99. You can use any of the listed hooks, according to the stage of the authentication process where you wish to use the source IP address.
2. In the hook, add the following lines:

```
<%
<Source_IP>=GetSessionParam(g_cookie,"SourceIP")
%>
```

Using Custom Include Pages

If you wish, you can use your own validation "include" pages, instead of some of the default pages that are loaded by the authentication ASPs.

Caution

You must not change the default authenticating pages supplied with the IAG.

To use custom "include" pages:

1. Access the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`
2. Under the `CustomUpdate` folder, place your `inc` file, naming the file as follows:

   `<Trunk_Name><Secure(0=no/1=yes)><File_Name>`

   Where the files you can use, including the parameters each file must set, are described in the sections that follow.

For example:

For an HTTPS trunk named "WhalePortal", to use your own `Validate.inc`, create the file:

`WhalePortal1Validate.inc`

Tip

The authentication ASPs, which load the files described in the following sections, are described in "Authentication Pages" on page 96.

Validate.inc

This file is loaded by `Validate.asp`. It must set the following parameters:

- `Session("CredentialsNum")`: number of credentials that were entered during login.
- `Session("CurrentCredentialsNum")`: current credentials index.
- `Session("user_name"&num)`: all users.
- `Session("password"&num)`: all passwords.
- `Session("repository"&num)`: all repositories.
- `user_name`: current user.
- `password`: current password.
- `repository`: current authentication server or user/group repository.
- `num`: current credentials index.

ValidateChangePassword.inc

This file is loaded by `ValidateChangePassword.asp`. It must set the following parameters:

- `user_name`: current user.
- `password`: current password.
- `new_password`: new password.
- `repository`: current authentication server or user/group repository.

ValidateContinue.inc

This file is loaded by `ValidateContinue.asp`. It must set the following parameters:

- `challenge`
- `handle`
- `error_code`
- `repository`

ValidateChooseUser.inc

This file is loaded by `ValidateChooseUser.asp`. It must set the following parameters:

- `user`
- `repository`

<repository>.inc

This file is loaded by the `WhaleRepository.inc`, which, in turn, is loaded by the `Validate.asp` and `ValidateContinue.asp`. It is used when you define an authentication scheme other than those that are supported by the IAG by default, where `repository` is the name you assigned the authentication server. For details, refer to "User-Defined Authentication Scheme" on page 121.

Using Your Own Authentication Pages

This section describes how you use your own authentication pages, instead of the default IAG pages. It applies to the following options:

- The page that initiates the authentication process, defined in the "Login Page" field of the Authentication tab.
- The page that is displayed to users whenever they need to authenticate after the initial authentication, defined in the "On-the-Fly Login Page" field of the Authentication tab.

Caution

You must not change the default authenticating pages supplied with the IAG.

To use your own authentication pages:

1. Place your own login page in one of the following custom folders:
   - `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`
   - `…\Whale-Com\e-Gap\von\InternalSite\CustomUpdate`

Tip

You can base the custom login page on the default page supplied with the IAG, using the sample page provided in the following location:

`…\Whale-Com\e-Gap\von\InternalSite\samples\Login.asp`

The links from this file to the "include" files are relative to the `CustomUpdate` folder, where you place the file.

Caution

If you place the file in a location other than the `CustomUpdate` folders, it may be deleted when you next upgrade the IAG.

2. In the Authentication tab, in the “Login Page” or “On-the-Fly Login Page” field, as applicable, enter the location of the page; make sure the page name is preceded by the location of the file under the `InternalSite` folder. The following example shows an entry for a Login page that is located in the folder …`\InternalSite\CustomUpdate`:

Note

We recommend that you use the default name of the login page, `Login.asp`, as the name of your own login page, as well. If your login page or on-the-fly login page are named differently, or if a page uses different URL parameters than the parameters used in the default login page, you need to configure a corresponding URL Inspection rule for the page, so that it passed the IAG URL inspection. For details, refer to “Configuring URL Inspection Rules” on page 143.

Authentication Schemes

The following sections describe the authentication schemes you can use for session and application authentication and authorization:

- “LDAP Authentication Schemes” on page 105, including:
  - Netscape Directory Server (V. 4.1)
  - Notes Directory Server
  - Novell Directory Server
  - Windows 2000/2003 Active Directory
- “LDAP Client Certificate Authentication Scheme” on page 107
- “NT Domain Authentication Scheme” on page 111
- “RADIUS Authentication Scheme” on page 112
- “RSA SecurID Authentication Scheme” on page 114
- “TACACS Authentication Scheme” on page 118
- “WINHTTP Authentication Scheme” on page 120
- “User-Defined Authentication Scheme” on page 121

Tip

For a description of how you configure the authentication servers, refer to “Configuration of the Authentication Servers” on page 89.

LDAP Authentication Schemes

Overview

Lightweight Directory Access Protocol (LDAP) is a protocol for accessing online directory services. The LDAP authentication server keeps information about users, including authentication information such as user properties and authentication scripts, in Directories (special-purpose databases.) When a connection request arrives at the IAG, user name and password are authenticated against the LDAP Directory.

The IAG implements the following LDAP authentication schemes:

- Netscape Directory Server (V. 4.1)
- Notes Directory Server
- Novell Directory Server
- Windows 2000/2003 Active Directory

The supported LDAP authentication schemes are capable of:

- Operating with two LDAP authentication servers. If the primary LDAP server fails, the IAG User Manager accesses the alternate LDAP server.
- Supporting a secure port. If the authentication server uses a secure port, the IAG uses a secure connection, even if this was not configured when the scheme was defined.
- In the Novell Directory Server: unique users do not need to enter their context when entering the user name. A unique user appears only in one context in the tree, or, if a “Base” is defined, the user appears only in one context under the Base.

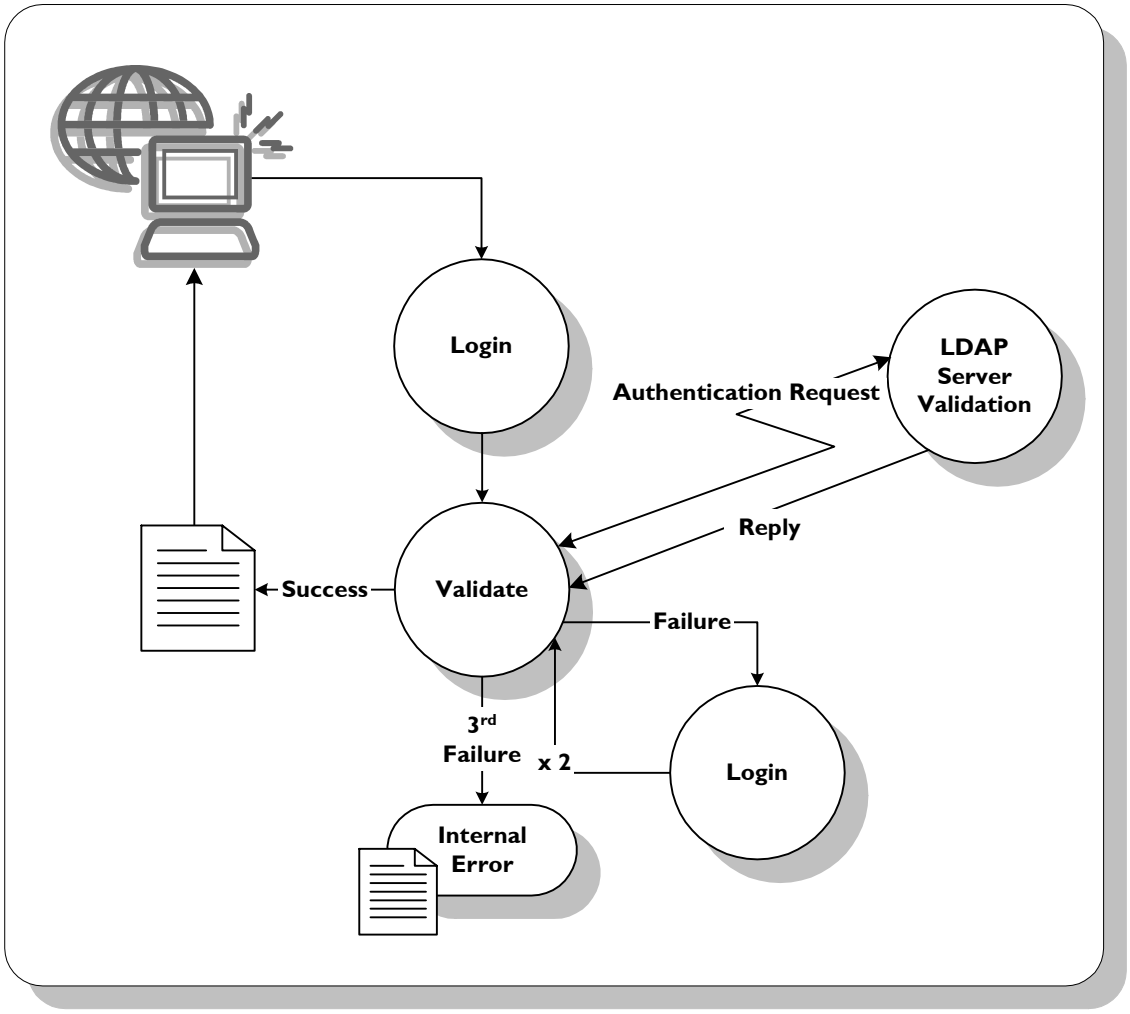
LDAP Authentication Flow

The following figure illustrates the authentication process users go through when the LDAP authentication scheme is implemented with one authentication server.

Note

The flow allows for three login attempts, after which login failure is final. The actual number of login attempts users are allowed is determined in the Authentication tab, in “Permitted Authentication Attempts”.

Figure 17. LDAP Authentication Flow



Configuring the LDAP Authentication Server or Servers

The system administrator has to configure the LDAP server or servers used in the scheme to determine how the User Name text box will appear in the Login page:

- If the User Name box appears empty, the user has to enter the complete distinguished name, including the hierarchical address.
- It is possible to configure the Login page so that the hierarchical address is automatically displayed in the text box, and the user has to enter only the user name. The system administrator has to see to it that during the authentication process the complete DN is passed on to the Validation page.

LDAP Client Certificate Authentication Scheme

Note

- This scheme is functional in HTTPS Connections trunks, where the Secure Sockets Layer (SSL) protocol is used. You cannot operate this scheme from an HTTP Connections trunk.
- This scheme is not supported on Camino® browsers on Mac® OS X.

Overview

Client Certificate authentication schemes require users to authenticate by supplying a client certificate, which is installed on their local disks. No login information—user name and password—is required for the authentication process.

The LDAP Client Certificate authentication scheme supported by the IAG operates with one or two LDAP authentication servers. LDAP authentication servers keep information about users in directories, including authentication and authorization information such as user properties and access rights. When the trunk is configured to apply the LDAP Client Certificate authentication scheme, and a connection request arrives at the IAG, the authentication scheme goes through the following stages:

- Authenticate the user: a user requesting to connect is prompted by the browser to select a client certificate. When the user selects a certificate, the IAG verifies the validity of the certificate and the identity of the user.
- Authorize the user: once the certificate is validated and the user is recognized, the IAG User Manager checks with the LDAP authentication server to verify that the user is authorized to access the Application server.

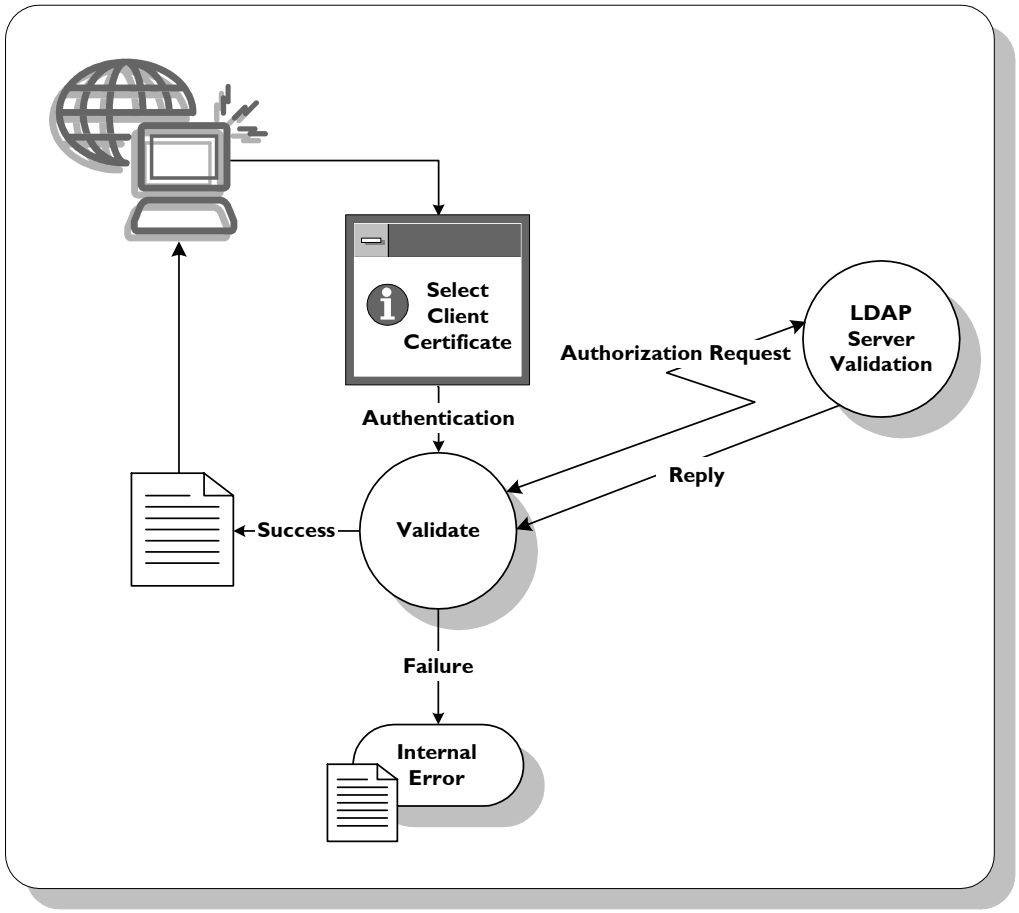
Each registered user in the LDAP server is assigned a Distinguished Name (DN), which includes a hierarchical address. For example: `organization\organizational_unit\username`.

When the LDAP Client Certificate scheme operates with two LDAP authentication servers, if the primary LDAP server fails, the User Manager accesses the alternate LDAP server.

LDAP Client Certificate Authentication Flow

The following figure illustrates the authentication process users go through when the LDAP Client Certificate authentication scheme is implemented with one authentication server.

Figure 18. LDAP Client Certificate Authentication Flow



Configuration at the IAG

To configure the LDAP Client Certificate authentication scheme:

1. In the IAG Configuration program, access the Authentication and User/Group Servers dialog box, where you define an LDAP server that will be used for this scheme.
   - For details on defining authentication servers, refer to "Configuration of the Authentication Servers" on page 89.
   - LDAP servers include the following: Active Directory; Netscape LDAP Server; Notes Directory; Novell Directory.

2. Copy the file `site_secure_cert.inc` from:

   `…\Whale-Com\e-Gap\von\InternalSite\samples`

   To the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`

3. Rename the file as follows:

   `<Trunk_Name>1cert.inc`

   For example:

   For a trunk named "WhalePortal", name the file:

   `WhalePortal1cert.inc`

> **Tip**
> The digit `1`, which is part of the file name, indicates that this is an HTTPS trunk.

By default, this file checks the user's email address in order to verify the certificate. You can edit the file to change this functionality or add other functions, if required.

> **Caution**
> `cert.inc` must set the number of parameters that are checked.
>
> **For example**: in the default settings, where one parameter (email) is checked, `cert.inc` sets the following:
>
> `Dim subject_array(0)`
>
> If you edit the file, make sure you change this function accordingly.

4. From the `samples` folder you accessed in step 2, copy the file `site_secure_login_for_cert.inc` to the `CustomUpdate` folder. Rename the file as follows:

   `<Trunk_Name>1login.inc`

5. From the `samples` folder, copy the file `site_secure_validate_for_cert.inc` to the `CustomUpdate` folder. Rename the file as follows:

   `<Trunk_Name>1validate.inc`

6. In the `validate.inc` file you copied in step 5, enter the name of the authentication server that you defined in step 1., in the line:

   `Session("repository1") = ""`

   For example:

   If you named the server "LDAPCert", this line should read:

   `Session("repository1") = "LDAPCert"`

7. From the `samples` folder, copy the file `repository_for_cert.inc` to the `CustomUpdate` folder. Rename the file as follows:

   `<Server_Name>.inc`

   Where `<Server_Name>` is the name of the authentication server you defined in step 1, in the "Name" field of the Add Server dialog box.

   For example:

   If you named the server "LDAPCert", name the file `LDAPCert.inc`
8. If you have changed the default functionality of the `cert.inc` file, in step 3, make the corresponding changes in the file you copied in step 7, as well. Otherwise, you do not need to edit this file.
   - The file includes lines that are commented out, which can be used to implement one function. Search for `param_x`, un-comment the line, and edit the text as required.
   - You need to repeat the above for each function you add.
   - To remove the function that checks the email address, search for `param_email` and comment out the relevant lines.

Configuring the LDAP Authentication Server or Servers

The system administrator has to register all the users in the LDAP server, and assign each user a Distinguished Name (DN), which includes a hierarchical address.

For example: `organization\organizational_unit\username`

The LDAP server or servers used in the scheme have to be configured to allow anonymous search access to the folder or folders where users are registered.

Preparing the Endpoint Browser

In order to use the Client Certificate authentication scheme, end-users have to install a client certificate on the browser that is used to access the site. The Certification Authority (CA) that signs the client certificate must be trusted by both the endpoint browser and the IIS on the IAG.

Tip

For a description of how you add a CA to the list of trusted Root Certificate Authorities, refer to the *Intelligent Application Gateway User Guide*, to "Adding the CA to the Certificate Trust List (All CAs)" on page 136.

NT Domain Authentication Scheme

Overview

The NT Domain authentication scheme checks user credentials against an NT server. You can also use it to define the local computer's Windows Local Users and Groups Manager as an authentication server.

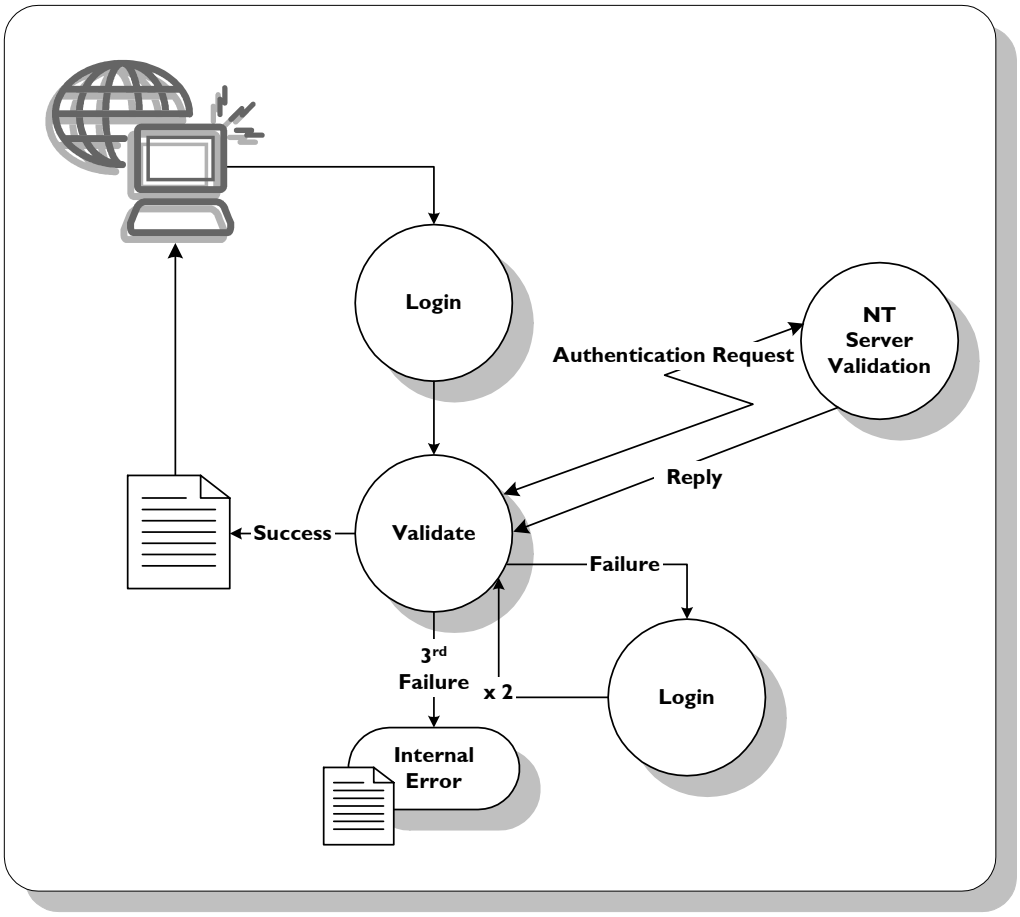
NT Domain Authentication Flow

The following figure illustrates the authentication process users go through when the NT Domain autentication scheme is implemented.

Note

The flow allows for three login attempts, after which login failure is final. The actual number of login attempts users are allowed is determined in the Authentication tab, in "Permitted Authentication Attempts".

Figure 19. NT Domain Authentication Flow



RADIUS Authentication Scheme

Overview

The RADIUS scheme applies the Remote Authentication Dial-In User Service (RADIUS) protocol to manage the exchange of authentication information in the internal network. The IAG and the RADIUS authentication server operate in a client-server mode, where the IAG has to be configured as a client of the RADIUS server.

Secret Key

The RADIUS protocol utilizes a secret key to encrypt the credentials that the user enters in the Login Script. The authentication server then decrypts the data and compares it to its database.

Challenge-Response Modes

The RADIUS authentication scheme supports all the challenge-response authentication modes available on the RADIUS server.

For example: allowing the user to create a new PIN (Private Identification Number), requiring the user to create a new PIN, requiring the user to enter the token that is displayed on the authenticator, and more.

RADIUS Groups

You can configure the RADIUS authentication scheme to extract users' group membership from a RADIUS attribute.

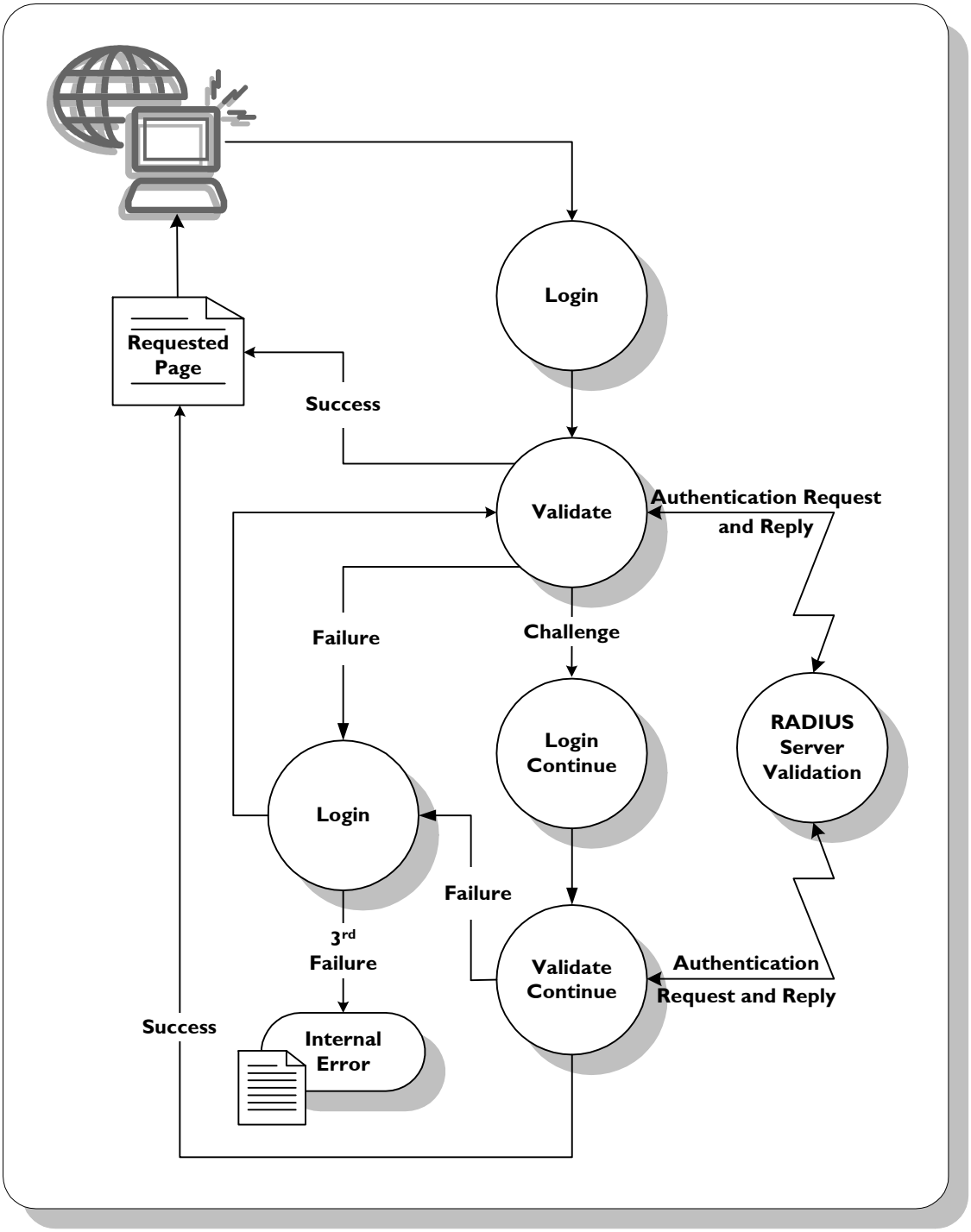
RADIUS Authentication Flow

The following figure illustrates a sample authentication process, through which users go when the RADIUS authentication scheme is implemented in a challenge-response mode. In this mode, the user can be challenged a number of times before the request is accepted, depending on the configuration of the RADIUS server.

Note

The flow allows for three login attempts, after which login failure is final. The actual number of login attempts users are allowed is determined in the Authentication tab, in "Permitted Authentication Attempts".

Figure 20. RADIUS Authentication Scheme—Sample Flow



Configuring the RADIUS Authentication Server

It is the responsibility of the system administrator to configure the RADIUS server to operate in conjunction with the IAG client. In order to do so, take the following steps:

- Add the IAG to the client list of the RADIUS server.
- Define the users that will be connecting through this client, to authenticate in a User-Login mode.
- Configure the challenge-response modes that will be used in the authentication process.
- Define the secret key that will be used to encrypt and decrypt the user password. This key must be identical to the secret key assigned to the authentication scheme when you define the RADIUS server in the IAG Configuration program, as described in "Configuration of the Authentication Servers" on page 89.

RSA SecurID Authentication Scheme

Note

In order to use the RSA SecurID authentication scheme, an RSA agent for Windows must be installed on the IAG. The following agents are supported:

- RSA Authentication Agent 6.0 for Microsoft Windows
- RSA ACE/Agent 5.6 for Windows
- RSA ACE/Agent 5.5 for Windows

Overview

The SecurID authentication scheme authenticates the user on a RSA ACE/Server. When challenged, the user has to enter a password that is a combination of two numbers: a Private Identification Number (PIN), supplied by RSA, combined with a token code, which is the number displayed on the RSA SecurID authenticator.

The SecurID scheme also supports additional challenge-response modes: Next Token and New PIN. Those are described in the following sections.

Next Token Mode

Applied in cases where the authentication process requires additional verification of the token code. The user is then challenged to enter the next token code, that is, to wait for the number that is displayed on the authenticator to change and enter the new number (without the PIN).

New PIN Mode

Applied in cases where the authentication process requires additional verification of the PIN. In this case, the user is required to use a new PIN. The new PIN is derived from one of two sources, depending on the configuration of the ACE/Server:

- The user is prompted to select and enter a new PIN.
- The server supplies the user with a new PIN.

The user is then required to re-authenticate with the new PIN.

The use of the New PIN mode is optional, and can be enabled or disabled in both the authentication server and the IAG Configuration program, whereby the IAG Configuration program takes precedence over the authentication server settings. That is, if the IAG is configured to disable the New PIN mode, the option will be disabled even if it is enabled in the server.

Note

For security considerations, we recommend that you do not enable the New PIN mode.

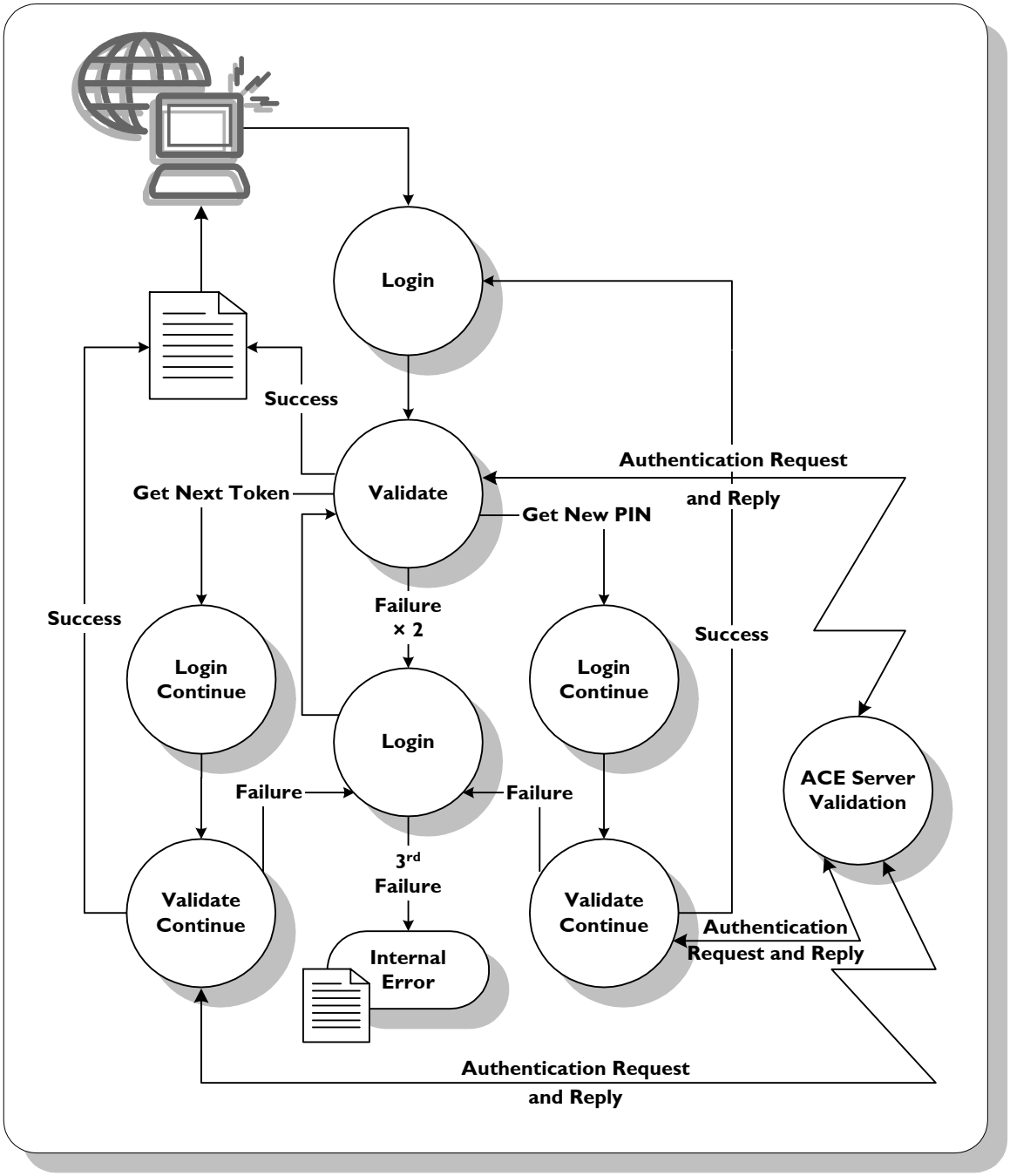
RSA SecurID Authentication Flow

The following figure illustrates the authentication process users go through when the RSA SecurID scheme is implemented.

Note

- The flow includes both Next Token and New PIN modes, which are only applicable under the conditions described in this section's overview.
- The flow allows for three login attempts, after which login failure is final. The actual number of login attempts users are allowed is determined in the Authentication tab, in "Permitted Authentication Attempts".

Figure 2I. RSA SecurID Authentication Flow



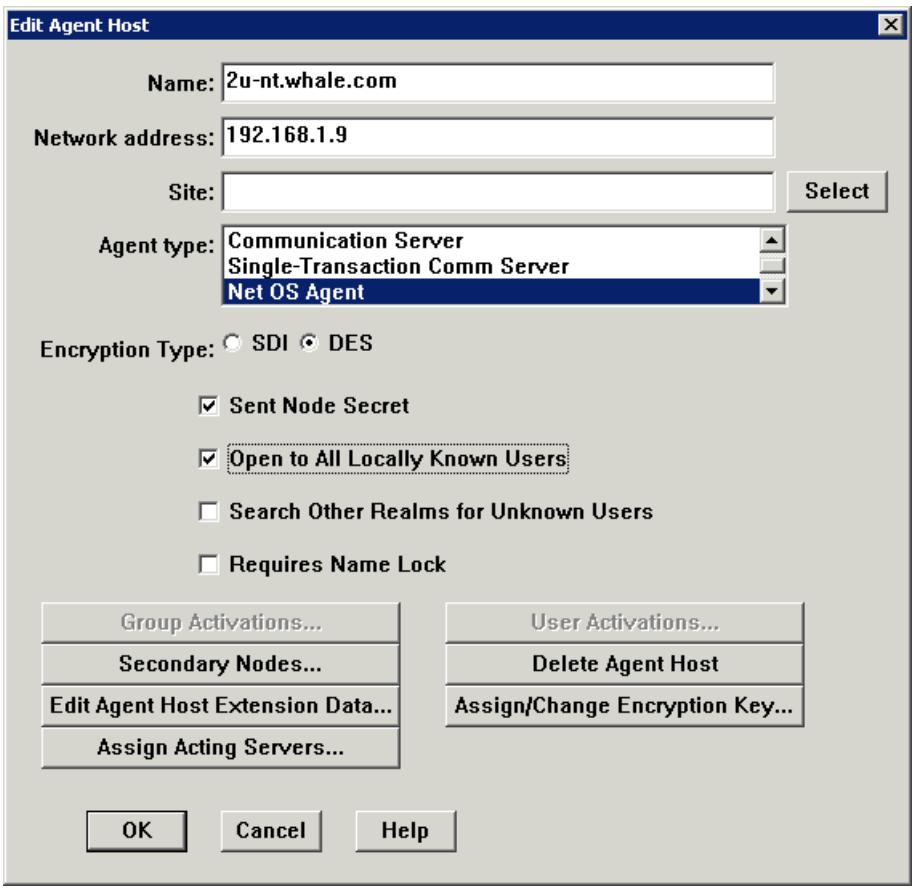
Configuring the RSA ACE Authentication Server

It is the responsibility of the system administrator to define the IAG as an agent host (new server versions) or a client host (older server versions) of the RSA ACE/Server. In the dialog box where you define the IAG, make sure the following is configured:

- Agent type: Net OS Agent
- The option “Open to All Locally Known Users” is selected

Following is a sample Edit Agent Host dialog box:

Figure 22. Defining the IAG as an Agent



If the New PIN mode is used, the system administrator must define whether the new PIN will be system generated, or requested from the user. These settings will affect the appearance and functionality of the Login Continue page. The IAG supports 4–8 digit PINs.

TACACS Authentication Scheme

Overview

The Terminal Access Controller Access Control System (TACACS) protocol allows a Network Access Server (NAS) to offload the user administration to a central server. Where the TACACS authentication scheme is applied, user connection requests are directed by the NAS to the TACACS authentication server, where user identity is compared against the server's user database, and users are granted or denied access accordingly.

The IAG and the TACACS authentication server operate in a client-server mode, where the IAG has to be configured as a client of the TACACS server.

The TACACS authentication scheme utilizes a secret key to encrypt the authentication request. This key must be identically configured in both the IAG Configuration program and the TACACS authentication server, as described in the following sections.

The TACACS authentication scheme has been tested against the NTTacPlus authentication server.

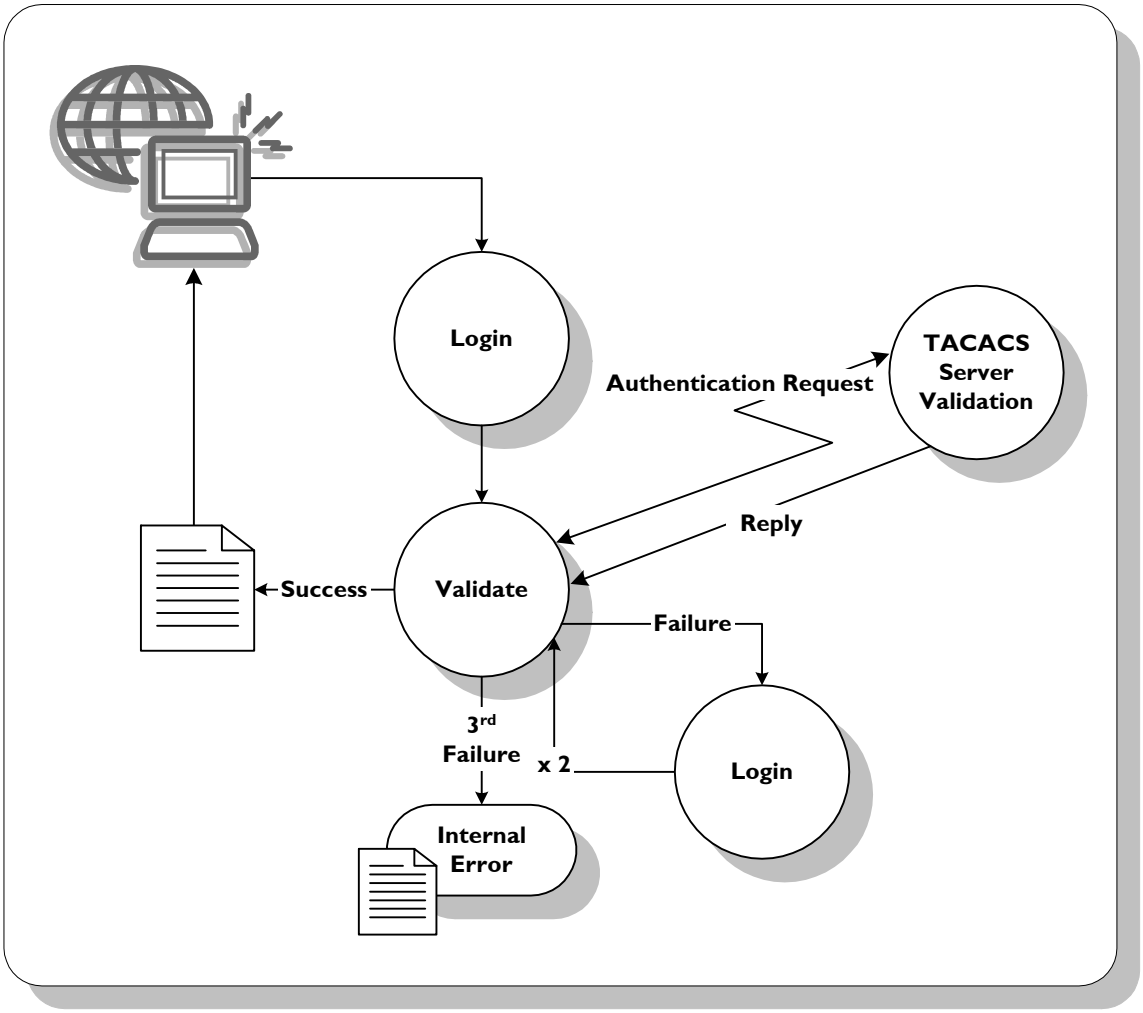
TACACS Authentication Flow

The following figure illustrates the authentication process users goes through when the TACACS authentication scheme is implemented.

Note

The flow allows for three login attempts, after which login failure is final. The actual number of login attempts users are allowed is determined in the Authentication tab, in "Permitted Authentication Attempts".

Figure 23. TACACS Authentication Flow



Configuring the TACACS Authentication Server

It is the responsibility of the system administrator to configure the TACACS server to operate in conjunction with the IAG client. In order to do so, take the following steps:

- Add the IAG to the client list of the TACACS server.
- Define the users that will be connecting through this client, to authenticate in a User-Login mode.
- Define the secret key that will be used to encrypt and decrypt the user request. This key must be identical to the secret key assigned to the authentication scheme when you define the TACACS server in the IAG Configuration program, as described in “Configuration of the Authentication Servers” on page 89.

WINHTTP Authentication Scheme

Overview

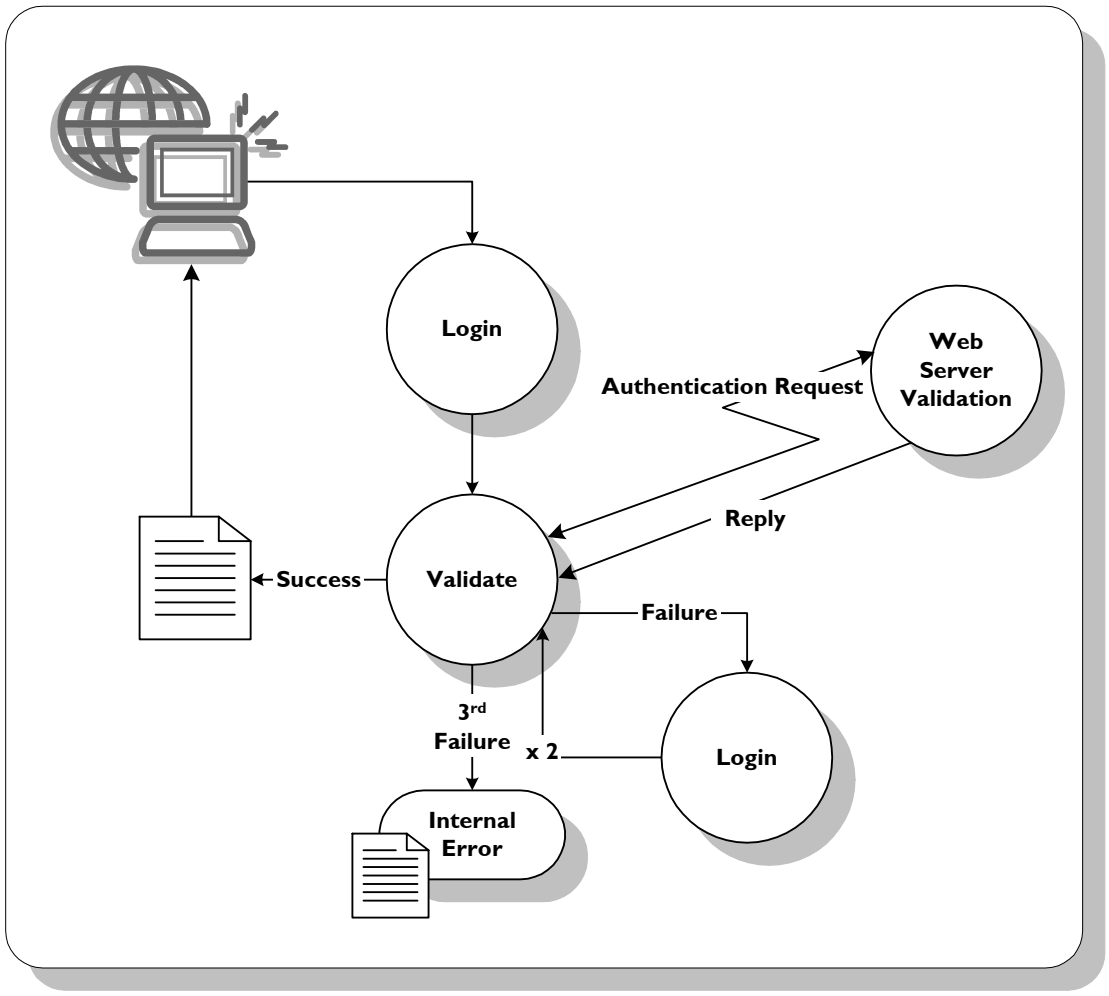
The WINHTTP authentication scheme checks users credentials as follows:

- You assign a URL of a page that requests users to authenticate (using an HTTP 401 request).
- The web server you define checks whether the user is authorized to access the requested URL. Only users that are authorized to access the URL are deemed authenticated.

WINHTTP Authentication Flow

The following figure illustrates the authentication process users go through when the WINHTTP authentication scheme is implemented.

Figure 24. WINHTTP Authentication Flow



Note

The flow allows for three login attempts, after which login failure is final. The actual number of login attempts users are allowed is determined in the Authentication tab, in "Permitted Authentication Attempts".

User-Defined Authentication Scheme

This section describes how you can implement a custom authentication scheme, which is not one of the schemes that are supported by the IAG out-of-the-box.

To implement a user-defined authentication scheme:

1. In the Configuration program, use the Authentication and User/Group Servers dialog box to define an authentication server of type "Other". For details on defining authentication servers, refer to "Configuration of the Authentication Servers" on page 89.
2. Copy the file `repository.inc` from:

   `…\Whale-Com\e-Gap\von\InternalSite\samples`

   To the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`
3. Rename the file as follows:

   `<Server_Name>.inc`

   Where `<Server_Name>` is the name of the authentication server you defined in step 1.

   For example:

   If you named the server "CustomAuth", name the file `CustomAuth.inc`
4. The file includes five functions. You can edit any of the functions, to define your custom implementation; delete any functions where you wish to apply the default implementation.
   - The functions are described in Table 14, "Authentication Scheme Definition File: Functions", on page 122.
   - For functions that return an object, the fields that can be set by the object are described in "Authentication Scheme Object Fields" on page 122.

Tip

You can see a sample implementation under the `samples` folder, in the file `repository_for_cert.inc`

Authentication Scheme Functions

This section describes the functions that are used to define the authentication scheme.

Table 14. Authentication Scheme Definition File: Functions

Function	Return Value
CheckCredentials(user_name, password)	Boolean
CheckChallenge(challenge)	Boolean
AuthenticateRepositoryUser (repository,user_name,password) And, ContinueAuthenticateRepositoryUser (repository,handle,challenge, challenge_code)	Object "UserMgrComLayer.AuthenticateUserOut" The function sets the following fields: • Success • ErrorCode • Handle • Message • FullUserName For a description of the fields refer to "Authentication Scheme Object Fields" on page 122.
ChangeRepositoryUserPassword (repository,user_name,password, new_password)	Object "UserMgrComLayer.ChangePasswordOut" The function sets the following fields: • Success • ErrorCode • Handle • Message For a description of the fields refer to "Authentication Scheme Object Fields" on page 122.

Authentication Scheme Object Fields

This section describes the fields that are used by the objects that are returned by some of the authentication scheme functions. The functions and the objects they can return are described in "Authentication Scheme Functions" on page 122.

Success

The value that is returned by the "Success" field determines the next step of the authentication. It can be one of the following:

- AUTHENTICATE_SUCCESS: authentication succeeded, users are redirect to `PostValidate.asp`, or, if further authentication with different credentials is required, to `Validate.asp`.
- AUTHENTICATE_FAIL: authentication failed, users are redirected to the error page `InternalError.asp`.
- AUTHENTICATE_CONTINUE: users need to continue the authentication process, and are redirected to `LoginContinue.asp`. **For example:** when the authentication server operates in a "Next Token" challenge-response mode, and users are required to enter a tokencode.
- AUTHENTICATE_AGAIN: users need to re-authenticate, and are redirected to the `Login.asp` again.
- CHANGE_PASSWORD: users are redirected to `LoginChangePassword.asp`, where they are required to change their password.
- PASSWORD_EXPIRATION_THRESHOLD: the password expiration threshold was reached. Users are redirected to `PasswordExpired.asp`, where they are notified that the password is about to expire within a pre-defined number of days, and are given the option to change their password.

ErrorCode

The value that is returned by the "ErrorCode" field provides additional details on the value that is retuned by the "Success" field, above. It can be one of the following:

- ERROR_CODE_FAILED_TO_AUTHENTICATE: user failed to authenticate.
- ERROR_CODE_SUCCEEDED_TO_AUTHENTICATE: user succeeded to authenticate.
- ERROR_CODE_CHANGE_PASSWORD: user is required to change the password.
- ERROR_CODE_FAILED_TO_CHANGE_PASSWORD: user failed to change the password.
- ERROR_CODE_PASSWORD_POLICY_NOT_OK: user failed to change the password; the new password does not comply with the password policy set by the administrator.

- ERROR_CODE_USER_IS_BLOCKED: authentication failed since the user's account is blocked.
- ERROR_CODE_USER_DEFINED: the message that the user receives in the next page contains the content of the "Message" field, below.

When the value of the "Success" field is AUTHENTICATE_CONTINUE, the authentication needs to be continued. The "ErrorCode" determines what type of "continue" is required, and what message is displayed to the user. In this case, the value of "ErrorCode" can be one of the following:

- ERROR_CODE_CONTINUE_AUTHENTICATE_TYPE_1: used for "New PIN" challenge-response mode, when the authentication server supplies the user with the new PIN. The following message is displayed to the user: "Your current PIN is invalid. Your new PIN is: <new PIN, as appears in the "Message" field>".
- ERROR_CODE_CONTINUE_AUTHENTICATE_TYPE_2: used for "Next Token" challenge-response mode. The following message is displayed to the user: "Your token needs to be validated. Please wait for the current tokencode to change, and enter the next tokencode:"
- ERROR_CODE_CONTINUE_AUTHENTICATE_TYPE_3: used for "New PIN" challenge-response mode, when the user is prompted to enter the new PIN. The following message is displayed to the user: "Your current PIN is invalid. Please enter a new PIN (4–8 digits):"
- ERROR_CODE_CONTINUE_AUTHENTICATE_TYPE_4: the message contains the text that is received in the "Message" field.

Handle

The "Handle" field is used when the authentication process needs to continue. Contains a value that is used by the authentication server to identify the user.

Message

The "Message" field contains text that is received from the authentication server. Can be one of the following:

- User's new PIN, generated by the server.
- Server-generated message.
- Number of remaining days until the password expires.
- User-defined text in the custom pages.

FullUserName

The "FullUserName" field contains the user name.

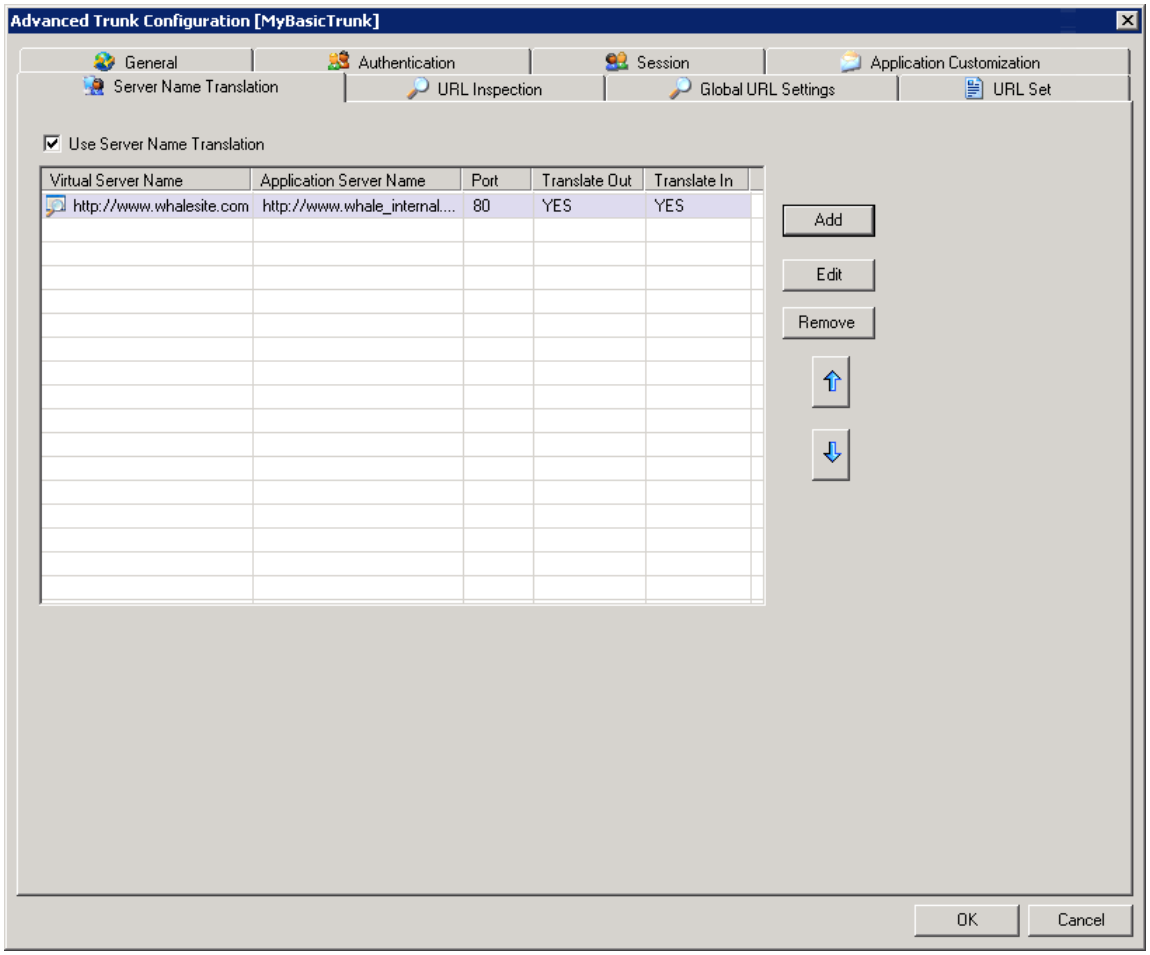
Server Name Translation

Note

The Server Name Translation option is only applicable in Webmail and Basic trunks. In Portal trunks, the inherent Host Address Translation (HAT) eliminates the need to use this option.

You configure Server Name Translation in the Advanced Trunk Configuration window, in the Server Name Translation tab. You can use this option to replace the name of the application server with the name of a virtual server in all web pages that are sent out of your organization. You can also use this option to direct incoming data directly to alternative servers within your organization. The types of data, in which server names are searched and replaced, are determined by the global Content-Type list, in the Application Customization tab, as described in "Global Content-Type List" on page 210.

Figure 25. Server Name Translation Tab



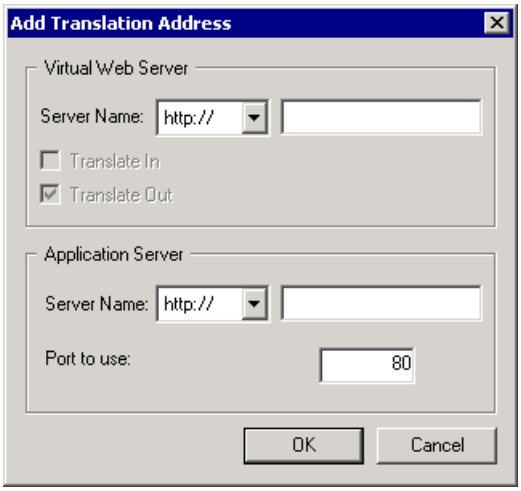
Creating and Editing Translation Rules

The following procedures describe how you create, edit, and delete translation rules in the Server Name Translation tab.

To create a translation rule:

1. In the Server Name Translation tab, check the "Use Server Name Translation" box to activate the option ☑ Use Server Name Translation, and click Add...

   The Add Translation Address dialog box is displayed.



2. Enter the required details in the Virtual Web Server and Application Server boxes, as described in Table 15 on page 127, and click OK.

   The translation rule is added to the list of rules in the Server Name Translation tab.

To edit a translation rule:

1. Select a translation rule in the list of rules in the Server Name Translation tab, and click Edit...
2. Edit the rule in the Edit Translation Address dialog box, and click OK.

   The translation rule is updated in the list of rules.

To remove a translation rule from the list of rules:

- Select a translation rule in the list of rules in the Server Name Translation tab and click Remove.

  The rule is removed from the list of rules.

To move a translation rule up or down in the list of rules:

- Select a translation rule in the list of rules in the Server Name Translation tab and click ⇧ or ⇩.

The rule moves up or down, respectively.

Table 15. Server Name Translation Tab Parameters

Parameter	Description
Virtual Web Server	Identification and translation settings of the virtual web server: • Server Name Select the service type—HTTP or HTTPS—in the combo box, and add the virtual web server's name in the text box. Once you add the translation rule to the list, this is the name that will be seen in the browser's URL Address box, and in the web page's HTML source file. • Translate In When this option is selected, translation is applied to all incoming requests. That is, clients accessing the virtual web server will be referred to the specific application server that is defined here. This option overrides the Application Server IP Address parameter that has been defined for the trunk, and enables data transfer to more than one server per trunk. • Translate Out This option is always selected. Translation is applied to outgoing HTML pages. That is, the name of the application server will be replaced by the name of the virtual web server as the pages are sent.
Application Server	Identification of the application server: • Server Name Select the service type—HTTP or HTTPS—in the combo box, and add the application server's name in the text box. • Port to use When "Translate In" is selected, incoming requests will be sent to the application server using this port.

Initial Host Selection

Note

Initial Host Selection is only applicable in Webmail and Basic trunks. In Portal trunks, the inherent Host Address Translation (HAT) mechanism eliminates the need to use this option.

This option enables you to configure a one-to-many access control scheme, where one trunk controls access to multiple servers, while throughout a session each user is allowed access to only one server.

You configure the Initial Host Selection option in two stages:

- Configure host selection in the authentication pages, described in "Configuring Host Selection in the Authentication Pages" on page 128.
- In the Configuration program, in the Advanced Trunk Configuration window, in the Server Name Translation tab, configure a translation rule, as described in "Configuring Initial Host Selection in the Server Name Translation Tab" on page 130.

Configuring Host Selection in the Authentication Pages

Tip

This procedure involves the customization of authentication pages. For a full description of the pages and the customization options available to you, refer to "Authentication Pages" on page 96.

To configure host selection in the authentication pages:

1. Access the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`

2. Under the `customUpdate` folder, create an `inc` "hook", which will be activated before the `PostValidate.asp` reaches the client side:

   `PrePostValidate.inc`

   Or,

   `PostPostValidate.inc`

   Name the file as follows:

   `<Trunk_Name><Secure(0=no/1=yes)><Hook_Name>`

   For example:

   For an HTTPS trunk named "WhaleSite", to create a "PrePostValidate" hook, create the file:

   `WhaleSite1PrePostValidate.inc`

Tip

If a file by this name already exists, you can use the existing file; you do not need to create a new file in this case.

3. In the hook file you defined in step 2, implement the required host selection, using any one of the following server variables, or any combination of the variables, describing the server to where the request is directed:
   - IP address:
     `SetSessionParam g_cookie, "WhlRWSIP", "<Application_Server_IP_Address>"`
   - Server name:
     `SetSessionParam g_cookie, "WhlRWSName", "<Application_Server_Name>"`
   - Port number:
     You can add a port number at the end of either of the server variables. For example, add a port number to the `WhlRWSIP` variable as follows:
     `SetSessionParam g_cookie, "WhlRWSIP", "<Application_Server_IP_Address>:<Port_Number>"`

   For example:

   To configure a rule whereby users who are authenticated against the "whale" server are routed to the application server 192.168.1.136, to port 81, and all other users are routed to 192.168.1.23, enter the following:

```
<%
     Repository = session("repository1")
     if Repository = "whale" then
          SetSessionParam g_cookie, "WhlRWSIP", "192.168.1.136:81"
     else
          SetSessionParam g_cookie, "WhlRWSIP", "192.168.1.23"
     end if
%>
```

   Where `repository1` is the authentication server used for session authentication, as retrieved from the Session Manager.

Note

Make sure that, for every server you define in the authentication pages, you configure corresponding access rules in the Microsoft© Internet Security and Acceleration (ISA) Server firewall. For more details, consult the ISA help system.

Configuring Initial Host Selection in the Server Name Translation Tab

Note
For a description of how you use the Server Name Translation tab refer to "Creating and Editing Translation Rules" on page 126.

To configure Initial Host Selection in the Server Name Translation tab:

1. In the Advanced Trunk Configuration window, select the Server Name Translation tab.
2. Activate the Server Name translation option by checking the box **Use Server Name Translation**: ☑ Use Server Name Translation.
3. Create a translation rule as follows:

Table 16. Initial Host Selection—Server Name Translation Tab Parameters

Parameter	Description
Virtual Web Server	• Server Name: name or IP address of the virtual web server. **Note**: Select the service type—HTTP or HTTPS—in the combo box • Translate In: must be activated for incoming connections.
Application Server	• Server Name: name of the server variable you entered in the authentication hook, either `WhlRWSIP` or `WhlRWSName`, as described in "Configuring Host Selection in the Authentication Pages" on page 128. If you used a port number to define the server, enter a `WhlRWSPort` variable here, as well. **For Example:** If you entered a port number after the `WhlRWSIP` variable, define the server here as follows: `WhlRWSIP:WhlRWSPort` In this case, the port number you define in the authentication hook overrides the port number defined in "Port to use", below. **Note**: Select the service type—HTTP or HTTPS—in the combo box. • Port to use: port number.

For the example provided on page 129, you need to configure the Server Name Translation rule as follows:

Virtual Server Name	Application Server Name	Port	Translate Out	Translate In
https://www.whalesite.com	http://WhlRWSIP:WhlRWSPort	80	YES	YES

Tip

- If users access the server by the server name, create a rule with the server name, not the IP address.
- You can create two rules for the same virtual web server, one with the server's name and one with the server's IP address. You cannot, however, configure two rules translating the same virtual web server into two different application servers.

Configuring Initial Host Selection for OWA 2000/2003 Applications

This section describes additional steps you need to take if you configure Initial Host Selection for Outlook® Web Access 2000 or Outlook Web Access 2003, for an HTTPS trunk. In this setup, when the IAG sends requests to the Exchange server using HTTP, it adds the header `Front-End-Https: On` to each request. This header indicates to the Exchange server that the original request was sent over HTTPS, and the server sends the replies in HTTPS, accordingly.

In order for the reply to be routed to the requesting server, you have to configure an additional Server Name Translation rule, which will translate the HTTPS reply. For the example provided on page 129, you need to configure the following rules:

Virtual Server Name	Application Server Name	Port	Translate Out	Translate In
https://www.whalesite.com	http://WhlRWSIP:WhlRWSPort	80	YES	YES
https://www.whalesite.com	https://WhlRWSIP:WhlRWSPort	80	YES	NO

Chapter 5
Session Settings

The Intelligent Application Gateway (IAG) allows for a pre-defined session control, where you can determine the maximum number of concurrently open sessions, define an inactive session timeout, and configure a logoff scheme.

This chapter describes the settings you can configure in the Session tab of the Advanced Trunk Configuration window, including:

- Configuration of session parameters in "Session Configuration" on page 133.
- Session settings for both "regular" and "privileged" users are described in "Default and Privileged Session Settings" on page 137.

You can also use the Session tab to:

- Change the selection of endpoint security policies for the trunk, in the "Endpoint Policies" area. For details, refer to the *Intelligent Application Gateway User Guide*, to "Endpoint Policies" on page 93.
- Configure endpoint settings, in the "Endpoint Settings" area. For details, refer to the *Intelligent Application Gateway User Guide*, to "Endpoint Settings" on page 108.

Session Configuration

This section describes the Session Configuration settings:

Figure 26. Session Configuration Area, in the Session Tab

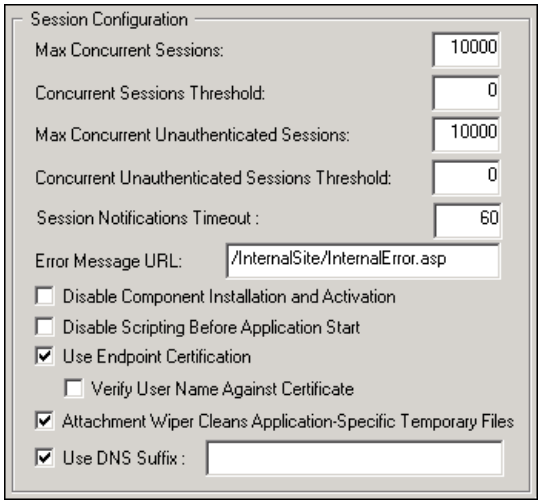


Table 17. Session Configuration Parameters

Parameter	Description
Max Concurrent Sessions	The maximal number of sessions that can be open through the trunk at the same time. **Note:** The default number of concurrent sessions is 10,000; however, the number of actual concurrent sessions may differ, depending on the IAG hardware.
Concurrent Sessions Threshold	Once the number of sessions reaches the threshold defined here, whenever a new session is opened, the IAG reports it by sending a message to the following: • To the Web Monitor. Messages can be viewed in the Event Viewer window. • To the log file that is defined in Built-In reporter of the Event Logging, when this reporter is enabled. For details, refer to the *Intelligent Application Gateway User Guide*, to Chapter 9: "Monitoring and Control".
Max Concurrent Unauthenticated Sessions	The maximal number of unauthenticated sessions that can be open through the trunk at the same time. **Note:** The default number of concurrent sessions is 10,000; however, the number of actual concurrent sessions may differ, depending on the IAG hardware.
Concurrent Unauthenticated Sessions Threshold	Once the number of unauthenticated sessions reaches the threshold defined here, whenever a new session is opened, the IAG reports it by sending a message to the following: • To the Web Monitor. Messages can be viewed in the Event Viewer window. • To the log file that is defined in the built-in Event Logging reporter, when this reporter is enabled. For details, refer to the *Intelligent Application Gateway User Guide*, to "Event Logging Reporters" on page 238.

Table 17. Session Configuration Parameters (Cont'd)

Parameter	Description
Session Notifications Timeout	Length of time (in seconds) for which session notifications are displayed to the user, including the following messages: Inactive Session Timeout; Scheduled Logoff; SSL Wrapper channel disconnection. **For example:** If the Session Notifications Timeout is 60, users are notified 60 seconds prior to a Scheduled Logoff that they are about to be logged off from the site. **Note:** In order to be able to receive session notifications, the browser must allow pop-ups from the site. If the endpoint browser features a pop-up blocker, it is recommended to add the site to the pop-up blocker's allowed sites list. This can be done automatically for Internet Explorer browsers running on Windows, which feature a pop-up blocker, such as Internet Explorer on Windows XP SP2, by activating the option "Add Site to Pop-Up Blocker's Allowed Sites", in the "Endpoint Settings" area of the Session tab. For details, refer to the *Intelligent Application Gateway User Guide*, to "Endpoint Settings" on page 108.
Error Message URL	URL containing error messages, displayed to end-users.
Disable Component Installation and Activation	Disables the installation of the Whale Client Components on endpoint computers, including the installation of the SSL Wrapper Java™ applet. For details refer to the *Intelligent Application Gateway User Guide*, to "Whale Client Components" on page 147. **Caution:** Disabling the installation of the Whale Client Components disables some of the IAG functionality, such as Endpoint Detection, SSL Wrapper, Attachment Wiper, and Certified Endpoints, and affects all endpoint computers, including computers where the components are currently installed. Make sure you are familiar with the Client Components before you activate this option. **Note:** When this option is activated, the selection of endpoint policies is disabled, since endpoint computers' compliance to the policies cannot be detected. This is reflected in the following locations: • "Endpoint Policies" area of the Session tab. • Add Application Wizard, in the "Application Setup" step. • Application Properties dialog box, in the General tab.

Table 17. Session Configuration Parameters (Cont'd)

Parameter	Description
Disable Scripting Before Application Start	Applicable only for Portal trunks. Disables advanced scripting options that are activated when an application is accessed, which are defined in the file: `…\Whale-Com\e-Gap\von\InternalSite\StartApp.asp` The options defined in this file include: • Activation of the application's prerequisite applications, if defined in the General tab of the Application Properties dialog box. • Application's Startup page, if defined in the Portal Link tab of the Application Properties dialog box. In addition, activating this option disables all the applications that are part of the "Client/Server and Legacy Applications" group. **Note:** Activate this option for troubleshooting purposes only.
Use Endpoint Certification	Activates the Certified Endpoint option. For a full description of this option, refer to the *Intelligent Application Gateway User Guide*, to "Certified Endpoints" on page 118. **Note:** This option is enabled in HTTPS trunks only.
Verify User Name Against Certificate	Applicable only if the option "Use Endpoint Certification", above, is activated. Activate this option to compare the user's login name with the user to which the certificate was issued. When this option is activated, Certified Endpoint status is determined per user and not per computer.
Attachment Wiper Cleans Application-Specific Temporary Files	Activate this option if you wish the Attachment Wiper to delete application-specific files. For details refer to the following: • The Attachment Wiper is described in the *Intelligent Application Gateway User Guide*, in "Attachment Wiper" on page 110. • Deletion of application-specific files is described in the *Intelligent Application Gateway Application Aware Settings* guide. **Tip:** You activate the Attachment Wiper option for default and privileged sessions in the right side of the Session tab, in the "Default Session Settings" and "Privileged Session Settings" areas, respectively.

Table 17. Session Configuration Parameters (Cont'd)

Parameter	Description
Use DNS Suffix	This option is applicable for: • Portal trunks, for Client/Server and Legacy Applications and Browser-Embedded Applications. • Webmail trunks, for trunks where the application being accessed through the trunk is Domino iNotes, either Single Server or Multiple Servers Activating this option ensures that when you define the application's servers, in the Server Settings tab, the system automatically completes the entry so that two types of entries are defined for each server: • NetBIOS hostname, for example: appserver • Fully Qualified Domain Name (FQDN), for example: appserver.whale.com The value you enter in the "Use DNS Suffix" field determines the extension that the system adds to NetBIOS hostnames, in order to complete the server's FQDN name: • If this field is left empty, the system adds the suffix of the local host, that is, the IAG. • If you enter a suffix in this field, the suffix you enter here is added to the server name. Likewise, when you enter an FQDN server name, the system derives the NetBIOS hostname from this name.

Default and Privileged Session Settings

For each trunk you create, you select a Session Access Policy and a Privileged Session Policy, as described in the *Intelligent Application Gateway User Guide*, in "Session Endpoint Policies" on page 95. Those policies define the conditions under which an endpoint establishes a "default" session with the IAG, with no special privileges, and the conditions under which an endpoint establishes a "privileged" session.

You can configure different session settings for default and privileged sessions, in the Session tab of the Advanced Trunk Configuration window, in the "Default Session Settings" and "Privileged Session Settings" areas, respectively.

- Default and privileged session settings are shown on Figure 27 on page 138. The parameters for both types of sessions are identical.
- For a description of the parameters, refer to Table 18 on page 138.

Figure 27. Default and Privileged Session Settings

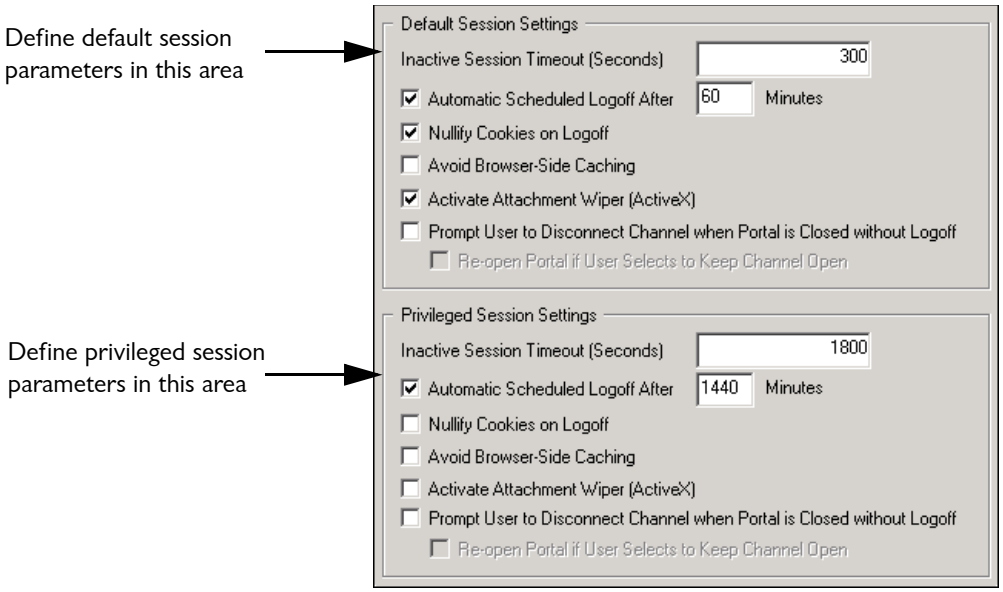


Table 18. Default and Privileged Session Parameters

Parameter	Description
Inactive Session Timeout (seconds)	The maximum time a session can be inactive before it expires. 30 seconds before session timeout, the user is prompted to renew the session. If the session is not renewed, it is deemed invalid and is closed; when the browser next sends a request, a new session is opened, and, if authentication is enabled, the user is requested to re-authenticate. In this manner, the filter invalidates abandoned sessions, while preventing data loss in cases where the user wishes to continue the session. For SSL Wrapper applications, that is, Client/Server and Legacy Applications and Browser-Embedded Applications in Portal trunks, and Domino iNotes Webmail trunks, inactivity timeout is counted from the last mouse movement or keyboard stroke. **Note:** In Portal trunks that use a custom homepage, you need to take additional steps in order to enable this feature. For details refer to "Defining Scheduled Logoff and Session Timeout on a Custom Homepage" on page 65.

Table 18. Default and Privileged Session Parameters (Cont'd)

Parameter	Description
Automatic Scheduled Logoff After ... Minutes	This option is only applicable if a Logoff Scheme is configured in the Authentication tab, as described in "Configuration in the Authentication Tab" on page 82. When the this option is activated, the IAG automatically triggers the configured Logoff Scheme after the time period defined here. The user is then requested to re-authenticate. Once users re-authenticate, they can continue working; if a user fails to authenticate, the session closes. **Note:** In Portal trunks that use a custom homepage, you need to take additional steps in order to enable this feature. For details refer to "Defining Scheduled Logoff and Session Timeout on a Custom Homepage" on page 65.
Nullify Cookies on Logoff	When this option is activated all the session cookies are set to "Expired" when the session is terminated and are deleted from the endpoint computer.
Avoid Browser-side Caching	Activating this option adds a header requesting the browser not to cache, so that no pages are kept in the browser's cache during a session. **Note:** This is achieved by adding the header "vary: *" to the response. If this header causes the application to malfunction, do not select this option. You can configure the Application Customization mechanism to add a different header to the response by adding an <ADD_HEADER> element to the Application Customization file: • For a description of how you create the file, refer to "Creating a Custom Application Customization File" on page 220. • For a sample <ADD_HEADER> code refer to page 223.
Activate Attachment Wiper (ActiveX®)	Activate the Attachment Wiper option. For details, refer to "Attachment Wiper" on page 110. **Tip:** You determine whether the Attachment Wiper—when activate—cleans application-specific files, for both default and privileged sessions, on the left side of the Session tab, in the "Session Configuration" area.

Table 18. Default and Privileged Session Parameters (Cont'd)

Parameter	Description
Prompt User to Disconnect Channel when Portal is Closed without Logoff	This option is applicable for: • Portal trunks that use SSL Wrapper applications (Client/Server and Legacy Applications and Browser-Embedded Applications). • Webmail Domino iNotes trunks (both Single Server and Multiple Servers). When the portal closes without the user having logged off the site, such as when the browser crashes, or when the user accesses a non-portal page from within the portal, the channel that was opened between the client and the portal, in order to enable connection to applications, remains open. Activating this option ensures that, when the portal window closes without logoff, users are prompted to disconnect the channel that was opened by this trunk.
Re-open Portal if User Selects to Keep Channel Open	Applicable only if the option "Prompt User to Disconnect Channel when Portal is Closed without Logoff", above, is activated. Activating this option ensures that when the portal window closes without logoff, if a user selects not to close the open SSL Wrapper channel, the portal window is re-opened. This prevents a situation where SSL Wrapper applications can run outside of the browser environment.

Chapter 6

Content Inspection

The Intelligent Application Gateway (IAG) includes an application-level control engine that stops application-level HTTP-based attacks and enforces application data validation, thus preventing web server exploits such as URL manipulation, buffer overflows, and more. HTTP filtering uses positive logic: only explicitly allowed traffic is permitted to pass through, while non-conforming traffic is automatically rejected. This protects against vulnerabilities not yet discovered and prevents unknown future attacks, since it relies on the positive logic of the application.

The IAG thoroughly inspects not only basic URLs, but parameters and any other incoming data as well. Application-level information can be stringently inspected to the degree of exact lengths and types of URLs, parameters, methods, and their combinations, which are permitted and expected by the application server. For example, attempts to crash or compromise the server by sending very long URLs, unexpected parameters, or unexpected methods will fail.

The IAG supplies you with pre-defined, Application Aware rulesets, which are designed to protect the portal and the internal website, and to meet the specific needs of each of the Web and Browser-Embedded applications you enable through the trunk.

Caution

Application Aware rulesets, which are supplied for the out-of-the-box applications, are based on the usage of the English language. If an application uses a different language, do one of the following:

- Change the application's ruleset to accommodate the required language. In order to do so, contact technical support.
- Set the enforcement level of the URL set to Medium, so that parameters are not checked. For details, refer to "URL Inspection Tab—URL Set Level" on page 146.

You can create additional rules, for example for proprietary applications, as well as edit existing rules, as required. Application rules are created and applied per application-type. You can also create a set of general rules, which are applied globally, for the entire trunk, and are checked prior to the application rules.

- If a request is not accepted by one of the general rules, it is rejected, regardless of its conformity to the application rules.
- If a request is accepted by the general rules, it is checked against the relevant application rules, and is only allowed if it is accepted by the application rules, as well.

Tip

Use general rules to protect your site against known web server vulnerabilities. You can see a sample of a ruleset that protects against IIS vulnerabilities in:

`…\Whale-Com\e-Gap\von\samples\RuleSet\WhaleIISBasic.rul`

This chapter describes the following topics:

- Configuration of rulesets in the Advanced Trunk Configuration window is described in "Configuring URL Inspection Rules" on page 143.
- "Using Variables in URLs" on page 173 describes how you can use user-dependant variables, such as a user name, in URLs.
- To assist in the deployment of an application-level rule base for your applications, the IAG includes a utility that enables you to record sample user sessions, in both production and non-production environments. Rules are generated to allow all the legitimate accesses performed during the sample sessions, and to deny access to all other attempts. The utility is described in "Rule Creating Utility" on page 177.
- "Exporting and Importing Rulesets" on page 187 describes how you can export and import rulesets from and to a trunk as well as across trunks.
- "Checking HTTP Headers" on page 190 shows you how to configure the filtering mechanism to check headers of incoming requests, so that it can reject unwanted headers, cookies, and parameters.
- "Support of Non-Compliant HTTP Applications" on page 201 deals with specific HTTP Request for Comments (RFC) irregularities.

Configuring URL Inspection Rules

You configure inspection rules in the Advanced Trunk Configuration window, which you access from the Configuration program. You use the following three tabs to edit the pre-defined inspection rules, and to create additional rules:

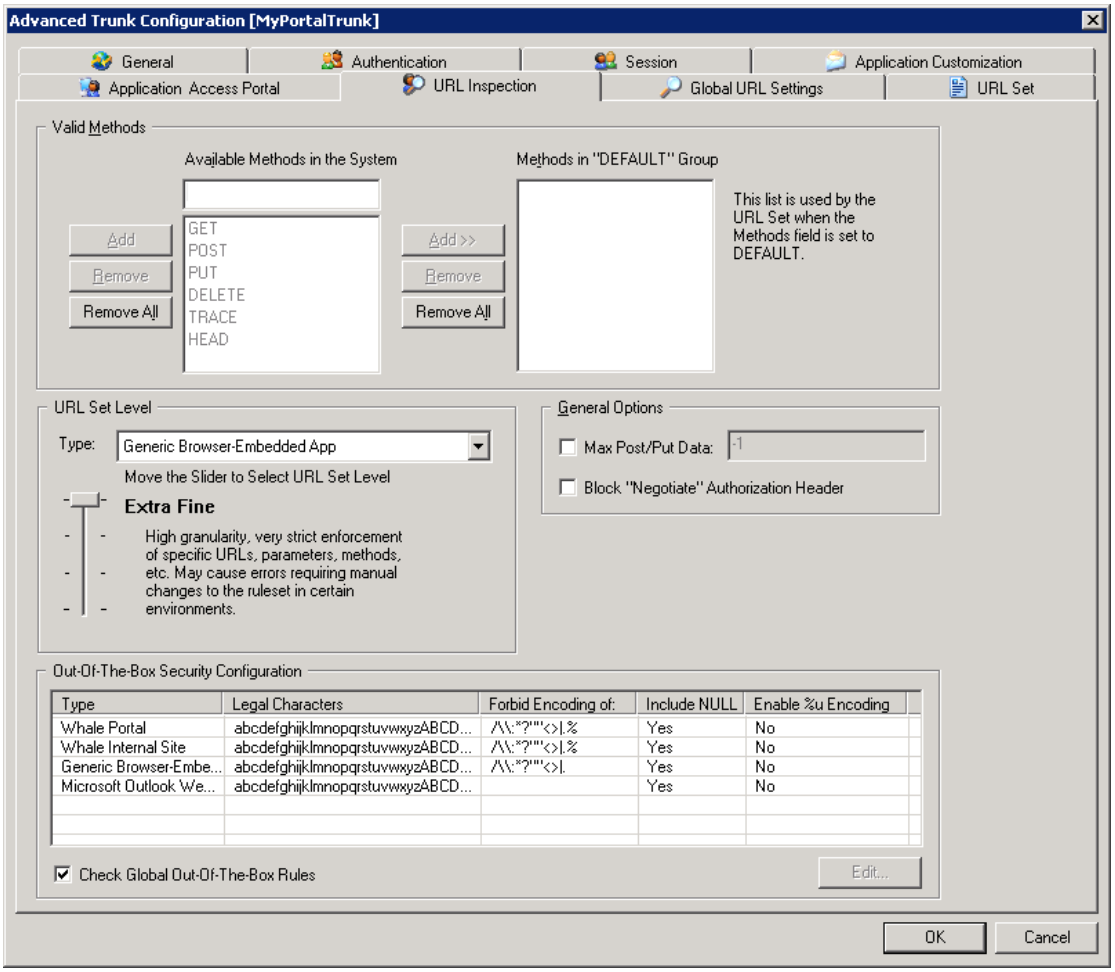
- The **URL Inspection** tab enables you to define valid methods, URL Set level, general options, and out-of-the-box security settings, as described in "Configuration in the URL Inspection Tab" on page 143.
- In the **Global URL Settings** tab you can configure global parameter rules, a global list of rejected parameter values, and global URL settings, including Download and Upload URLs and the requests that are ignored in session timeout calculations. For details, refer to "Configuration in the Global URL Settings Tab" on page 148.
- In the **URL Set** tab you can configure the URL Inspection rules, including URLs and parameters, as described in "Configuring a Ruleset in the URL Set Tab" on page 164.

Configuration in the URL Inspection Tab

Options you can configure in the URL Inspection tab include:

- Methods that clients can use when accessing the applications that are enabled through the trunk, including a set of default methods, which can be applied as a group. The Method options are described in "URL Inspection Tab—Valid Methods" on page 144.
- For Web and Browser- Embedded applications: the level of enforcement of the URL set, including URLs, parameters, and methods. Available URL Set levels are described in "URL Inspection Tab—URL Set Level" on page 146.
- General options, including the maximal size of pages sent using the POST and PUT methods and the option "Block "Negotiate" Authorization Header", described in "URL Inspection Tab—General Options" on page 146.
- Out-of-the-box security configuration, including application-specific character definitions, described in "URL Inspection Tab—Out-Of-The-Box Security Configuration" on page 147.

Figure 28. URL Inspection Tab



URL Inspection Tab—Valid Methods

This section lists the methods that can be used to access the URLs, as described in the following table. Only the methods available here will be available for selection in the URL Set tab, when you create a URL inspection rule.

Table 19. URL Inspection Tab—Valid Methods Parameters

Parameter	Description
Available Methods in the System	Pre-defined methods are grayed out; those cannot be removed from the list. You can add and remove user-defined methods to and from the list, as described in "Editing Valid Methods" on page 145.

Table 19. URL Inspection Tab—Valid Methods Parameters (Cont'd)

Parameter	Description
Methods in "DEFAULT" Group	Groups a list of methods, which can be used as a default group in the URL Set tab. You can add and remove methods to and from the group, as described in "Editing Valid Methods" on page 145.

Editing Valid Methods

Adding a user-defined method to the available methods list:

- Enter the method name in the text box above the list of available methods in the system, and click Add on the left side of the list.

  The method is added to the list of available methods.

Removing a user-defined method from the available methods list:

- Select a method and click Remove on the left side of the list of available methods.

  The method is removed from the list of available methods.

Removing all user-defined methods from the available methods list:

- Click Remove All on the left side of the list of available methods.

  All user-defined methods are removed from the list of available methods.

Adding a method to the DEFAULT group:

- Select a method from the list of available methods and click Add >>.

  The method is added to the DEFAULT group.

Removing a method from the DEFAULT group:

- Select a method in the DEFAULT methods list and click Remove on the left side of the list.

  The method is removed from the DEFAULT group.

Removing all methods from the DEFAULT group:

- Click Remove All on the left side of the DEFAULT methods list.

  All methods are removed from the DEFAULT group.

URL Inspection Tab—URL Set Level

For Web and Browser-Embedded applications, the URL Set Level determines the level of rule enforcement for each application-type, including URLs, parameters, methods, etc.

Table 20. URL Inspection Tab—URL Set Level Parameters

Parameter	Description
Type	Application-type to which the enforcement level applies. Applied individually for each of the Web and Browser-Embedded applications enabled through the trunk.
URL Set Level Slider	Move the slider to select one of the following enforcement levels: • **Extra Fine**—High granularity, very strict enforcement of specific URLs, parameters, methods, etc. May cause errors requiring manual changes to the ruleset in certain environments. • **Fine**—Specific, strong enforcement of URLs, parameters, methods, etc. • **Medium**—Flexible, not as bound to highly specific parameters. • **Rough**—Basic rule enforcement, very minimal risk of ruleset violations.

URL Inspection Tab—General Options

General option are described in the following table.

Table 2I. URL Inspection Tab—General Options

Parameter	Description
Max Post/Put Data	Check this parameter to set the maximal size of pages sent using the POST or PUT methods, in bytes. Default: -1, where size of data is not checked (size is unlimited). If you enter a positive value, when the size of a page exceeds this value, the request is denied and the Error Messages URL is sent in reply.
Block “Negotiate” Authorization Header	Blocks all “Negotiate” authorization headers. For details, refer to “Blocking the “Negotiate” Authorization Header” on page 193.

URL Inspection Tab—Out-Of-The-Box Security Configuration

Defines which ASCII characters may appear in URLs, and in what form. Can be applied individually for the portal, internal site, and each Web and Browser-Embedded application enabled through the trunk, or applied globally.

- Legal characters may appear in the URL as is (in both encoded and unencoded forms).
- Characters that do not appear in the list of Legal characters are only allowed to appear in the URL in an encoded form.
- Characters in the "Forbid Encoding of" list may not appear in the URL in an encoded form.

Tip
If a character does not appear in the list of legal characters, and does appear in the list of characters that can not appear in an encoded form, it is not allowed in the URL in any form.

The default definitions vary according to the application-type.

Table 22. URL Inspection Tab—Out-of-the-Box Security Parameters

Parameter	Description
Type	Type of application to which the out-of-the-box parameters apply.
Legal Characters	Characters that are allowed in the URL as is. **Note:** If you wish the filter to inspect encoded characters, do not include the character "%" in the list of Legal characters, since "%" is used as a prefix to encoded characters.
Forbid Encoding of	Characters that are not allowed in the URL in an encoded form.
Include NULL	Since the NULL character cannot be entered in the "Forbid Encoding of" field, use this parameter to indicate that NULL cannot appear in an encoded form.

Table 22. URL Inspection Tab—Out-of-the-Box Security Parameters (Cont'd)

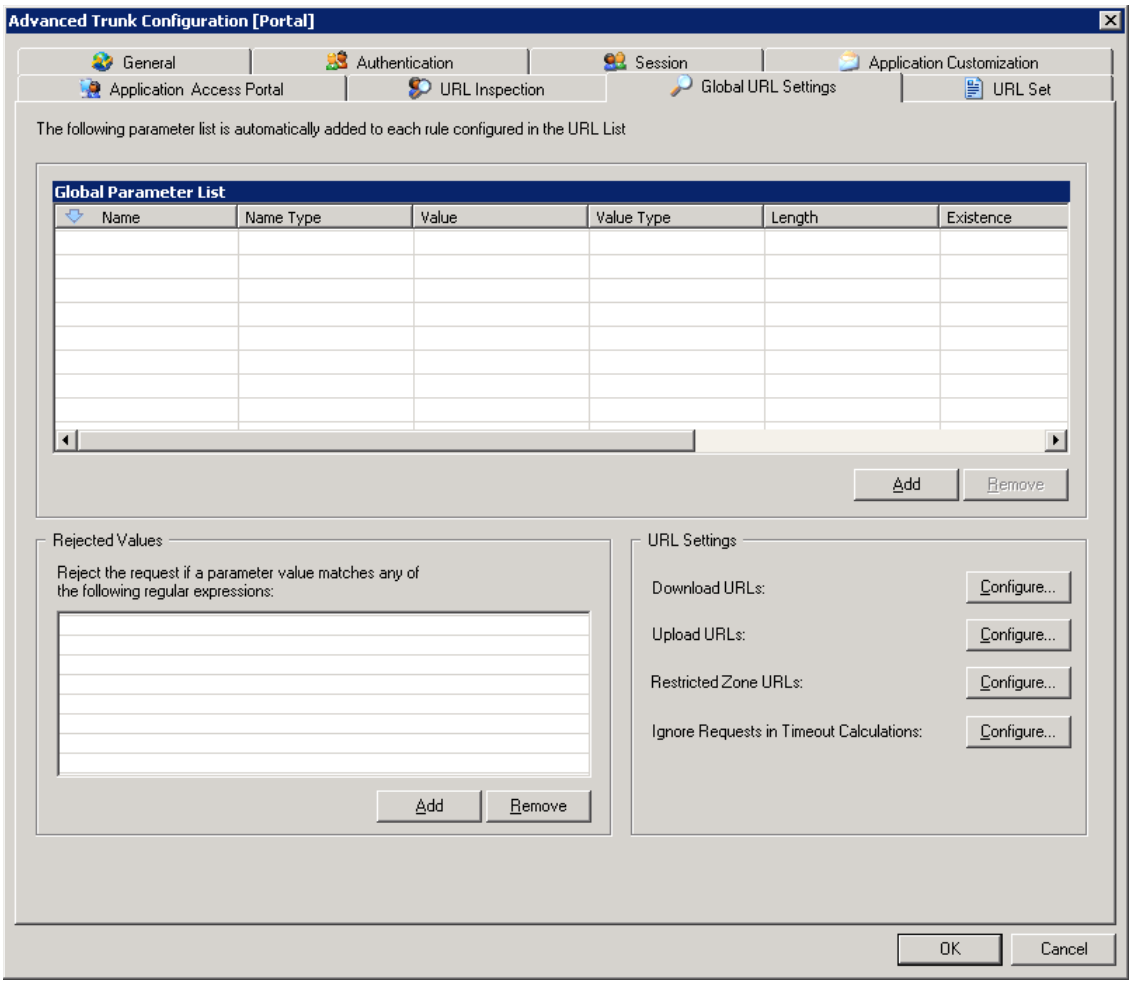
Parameter	Description
Enable %u Encoding	Enables the use of Microsoft %u encoding in URLs and parameters. That is, the filter will decode and inspect characters that are encoded using the %u encoding method in requests, in URLs and parameters. Note that, in order to support encoding, the character "%" can not appear in "Legal characters", above. **Tip:** Escaped encoding ("%" hex hex) is enabled by default.
Check Global Out-Of-The-Box Rules	When this option is activated, out-of-the-box security rules are enforced globally, that is, all the rules that are defined in the "Out-Of-The-Box Security Configuration" area are enforced on all the relevant applications in the trunk. **For example:** if encoding of the character ? is forbidden for the Whale Internal Site, it will automatically be forbidden for all the other relevant applications, regardless of the individual configuration of the option "Forbid Encoding of" for those applications.

Configuration in the Global URL Settings Tab

You use Global URL Settings tab to define the following:

- Global parameter rules, that will automatically be added to each of the parameter rules you define in the URL Set tab, described in "Global URL Settings Tab—Global Parameter List" on page 149.
- A global list of rejected parameter values, described in "Global URL Settings Tab—Rejected Values" on page 152.
- Global URL settings, including Download and Upload URLs and the requests that are ignored in session timeout calculations, described in "Global URL Settings Tab—URL Settings" on page 152.

Figure 29. Global URL Settings Tab



Global URL Settings Tab—Global Parameter List

Tip

Acquaint yourself with the URL Set tab, and read the descriptions provided in "Configuring a Ruleset in the URL Set Tab" on page 164, before you configure the parameters in the Global Parameter List.

Parameter rules are rules that the IAG applies to a URL when its URL Inspection rule is set to "handle" parameters. The global parameter rules you configure in the Global Parameter List are automatically added to each of the URL Inspection rules, which you define in the URL Set tab. The global rules are added to the individual parameter rules you define in the URL Set tab, in the Parameter List. When the request is checked against the rule, the individual parameter rules are applied first, then the global parameter rules.

If the Global Parameter List contains parameters that are also configured in one or more of the individual parameter rules, the IAG alerts you of the duplication when you activate the configuration. If two parameters by the same name exist in both the individual and global lists, the individual parameter rule overrides the global parameter rule.

Note

Parameter names are **not** case sensitive, that is, you can **not** use parameters with the same name but with different capitalization.

Table 23. Global URL Settings Tab—Global Parameter List Parameters

Parameter	Description
Name	Parameter name. Must match name sent by the browser.
Name Type	Type of parameter name: String or Regular Expression. For a description of the regular expressions used with rulesets, refer to Appendix B: “Regex++, Regular Expression Syntax”.
Value	Parameter value. Depends on the type of value, as defined in the Value Type column. • For strings, enter a regular expression that defines the acceptable values. For a description of the regular expressions used with rulesets, refer to Appendix B: “Regex++, Regular Expression Syntax”. • For Integer and Real parameters, a comma divides between values; a colon represents a range of values. **Note:** Parameter values must be listed according to their length, in a descending order, from the longest parameter to the shortest parameter. For example: Value editattach|contents|dialog
Value Type	Type of parameter value: Integer, Real, or String.
Length	Length of the value.

Table 23. Global URL Settings Tab—Global Parameter List Parameters (Cont'd)

Parameter	Description
Existence	• Mandatory: URL will only be considered valid if this parameter is present. • Optional: this parameter is optional. • Reject: if this parameter appears in the request, the request will be judged invalid.
Occurrences	Define whether the parameter can appear in the URL once, or multiple times.
Max total length	Total length of parameter values of all occurrences of this parameter.
Rejected values checking	Select whether to check this parameter against the Rejected Values list: • On: check against the list • Off: do not check against the list Configuration of rejected values is described in "Global URL Settings Tab—Rejected Values" on page 152.

To add a global parameter rule:

1. Under the Global Parameter List, click Add.

   The global parameter rule is added at the end of the list of the existing parameter rules.
2. Configure the rule, as described in Table 23 on page 150.

To remove a global parameter rule:

- Select the rule and click Remove.

  Once you confirm that you wish to delete the selected rule, it is removed from the Global Parameter List.

Global URL Settings Tab—Rejected Values

This section describes how you use the Global URL Settings tab to configure a Rejected Values list. If a requested URL, or a parameter in a URL that's configured to "Handle" parameters, is configured to be checked against this list, and one of its parameter values is matched with a "rejected" value, the request is rejected. You can select whether to check parameter values against the Rejected Values list when you configure any of the following:

- Global parameter rules, described in "Global URL Settings Tab—Global Parameter List" on page 149.
- Parameters that you configure for the individual URL Inspection rules, in the URL Set tab, in the Parameter List, as described in "URL Set Tab—Parameter List Parameters" on page 168.
- Unlisted Parameters, in the URL Set tab, as described in "URL Set Tab—Unlisted Parameters" on page 169.

If a request's parameter value is checked against this list, and it matches any of the values you configure here, the request is denied.

To configure rejected values:

- In the "Rejected Values" area, click Add, then enter the value, using regular expressions. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax".

Global URL Settings Tab—URL Settings

You use the URL Settings area to configure the following:

- Download URLs and Upload URLs rules. These rules are applicable in Portal trunks for Built-In Services, Web Applications, and Browser-Embedded Applications, and in Webmail and Basic trunks. They are used by the IAG when the method by which an application identifies downloads or uploads, in order to enforce its Download or Upload policy, is set to "Identify by URLs".
  - The method by which an application identifies URLs is determined in the Application Properties dialog box, in the Download/Upload tab. For details, refer to the *Intelligent Application Gateway User Guide*, to "Download/Upload Tab" on page 82.
  - For a description of how you configure the Download URLs and Upload URLs rules, refer to "Download URLs" on page 153 and "Upload URLs" on page 155, respectively.

- Restricted Zone URLs rules. These rules are applicable in Portal trunks for Built-In Services, Web Applications, and Browser-Embedded Applications, and in Webmail and Basic trunks. They are used by the IAG when the Restricted Zone option is activated for an application.
  - You activate the Restricted Zone option in the Application Properties dialog box, in the Web Settings tab. For details, refer to the *Intelligent Application Gateway User Guide*, to "Web Settings Tab" on page 73.
  - For a description of how you configure Restricted Zone URLs rules, refer to "Restricted Zone URLs" on page 158.
- The option to ignore pre-defined URLs, such as a URLs requested by KeepAlive mechanisms, in system timeout calculations, described in "Ignoring URL Requests in Inactive Session Timeout Calculations" on page 162.

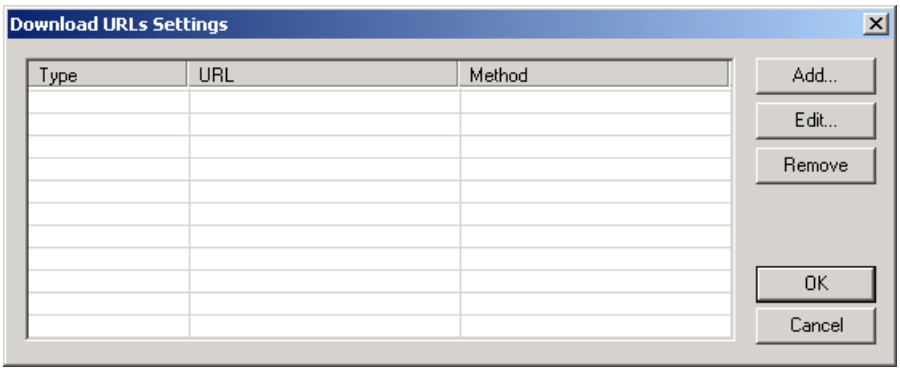
Download URLs

This section describes how you configure the rules that the IAG uses in order to identify downloads, when the application's method for identifying downloads is set to "Identify by URLs". For some applications, the IAG supplies you with pre-defined, Application Aware rules, as applicable.

To configure Download URLs rules:

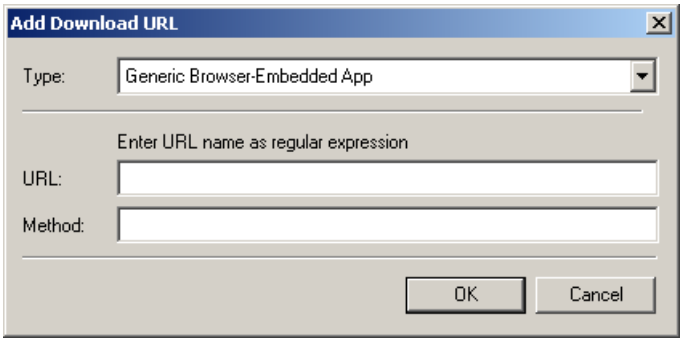
1. In the Global URL Settings tab, in the "URL Settings" area, click Configure... next to "Download URLs".

   The Download URLs Settings dialog box is displayed.



2. Click Add...

   The Add Download URLs dialog box is displayed.



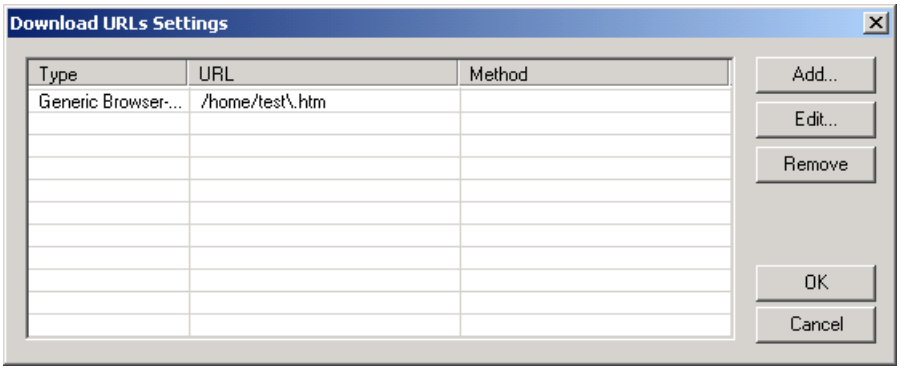
Tip

To edit an existing rule, select it and click Edit... in the Download URLs Settings dialog box.

3. Enter the following:
   - Type:
     - Portal trunks: select the application type from the list of relevant applications.
     - Webmail and Basic trunks: select whether to apply this rule to the application server or to the internal website.
   - URL: enter the URL, using regular expressions. For a description of regular expressions, refer to Appendix B: **"Regex++, Regular Expression Syntax"**.
   - Method (optional): HTTP method used to access the URL. Multiple methods are comma-separated.

   Click OK.

   The Add Download URLs dialog box closes. The rule is added in the Download URLs Settings dialog box.



4. Repeat steps 2–3 to add Download URLs rules, for all the relevant applications, as required.

   After you next activate the configuration, the rules you define here are used to identify downloads.

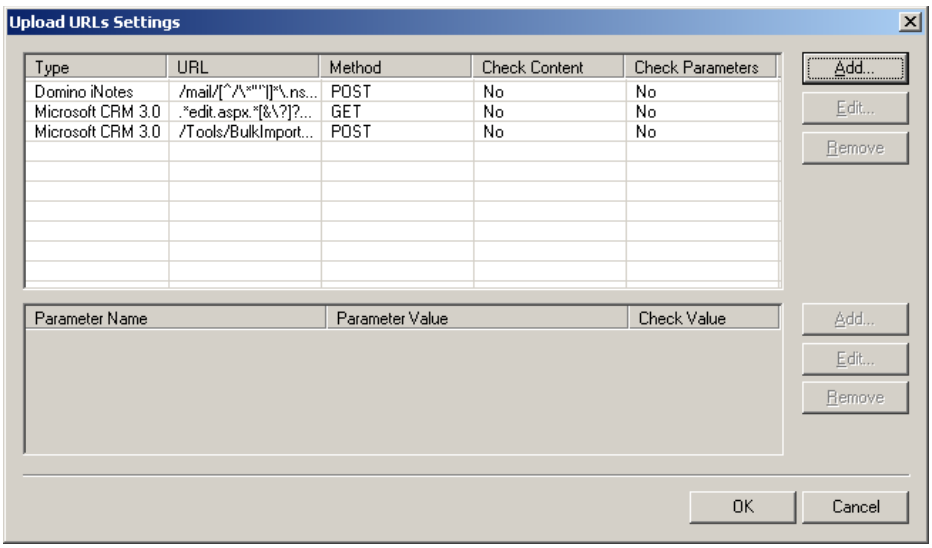
Upload URLs

This section describes how you configure the rules that the IAG uses in order to identify uploads, when the application's method for identifying uploads is set to "Identify by URLs". For some applications, the IAG supplies you with pre-defined, Application Aware rules, as applicable.

To configure Upload URLs rules:

1. In the Global URL Settings tab, in the "URL Settings" area, click Configure... next to "Upload URLs".

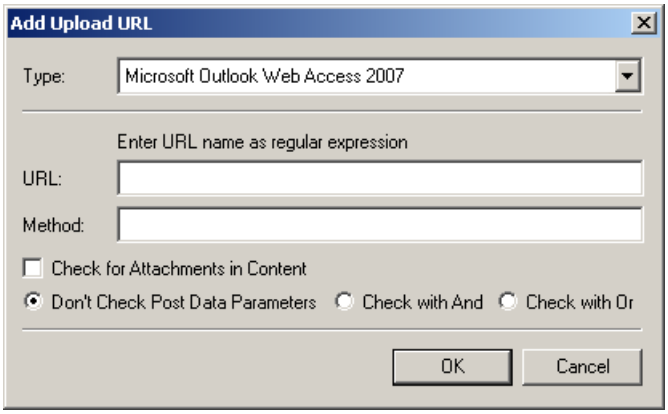
   The Upload URLs Settings dialog box is displayed.



2. In the upper area of the dialog box, click Add...

> **Tip**
> - To edit an existing rule, select it and click Edit...
> - If POST data parameters are configured for the rule you select, the parameters are listed in the lower area of the dialog box.

The Add Upload URL dialog box is displayed.



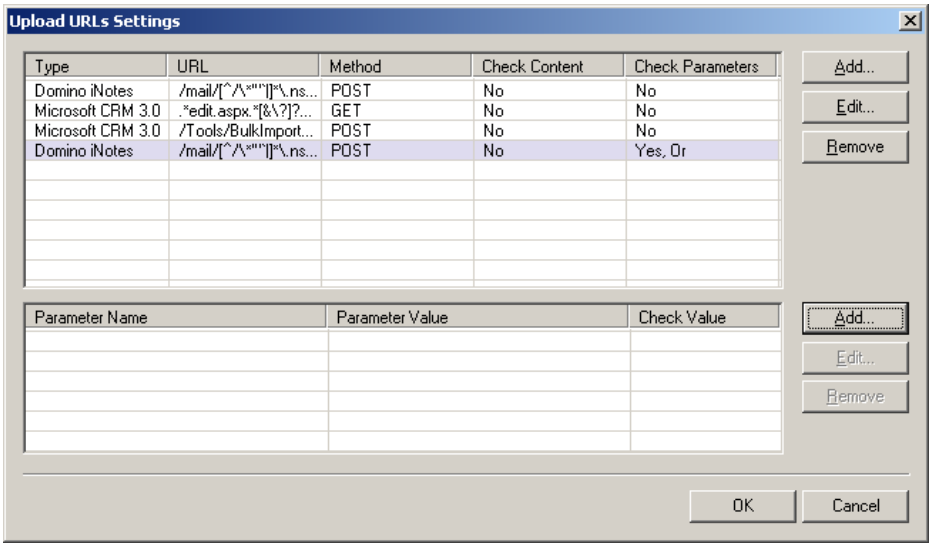
3. Configure the following:
   - Type:
     - Portal trunks: select the application type from the list of relevant applications.
     - Webmail and Basic trunks: select whether to apply this rule to the application server or to the internal website.
   - URL: enter the URL, as follows:
     - Use regular expressions, as described in Appendix B: “Regex++, Regular Expression Syntax”.
     - If you wish to check query string parameters, make sure the URL includes the query string.
   - Method (optional): HTTP method used to access the URL. Multiple methods are comma-separated.
   - Check for Attachments in Content: activate this option if you wish to check the content of the URL for attachments. When this option is activated, only URLs containing attachments are considered uploads.
   - Select whether to check parameters in POST data, and, if required, the method by which to check them:
     - Check with And: the data must contain all the rule’s parameters, as you define in the parameter list, in the lower area of the Rest URLs Settings dialog box.
     - Check with Or: the data must contain one or more of the defined parameters.

Note

If you configure the rule to check POST data parameters, make sure you define the parameters in the parameter list, as described in step 4 of the procedure.

Click OK.

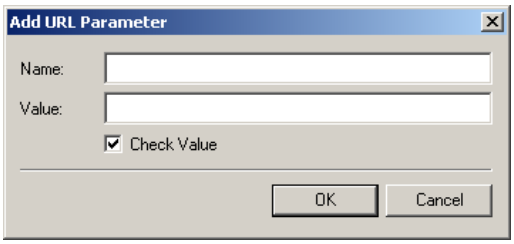
The Add Upload URL dialog box closes. The rule is added in the Upload URLs Settings dialog box.



4. If you configured the rule to check POST data parameters, configure the parameters:

   a) In the lower area of the dialog box, click Add...

   The Add URL Parameter dialog box is displayed

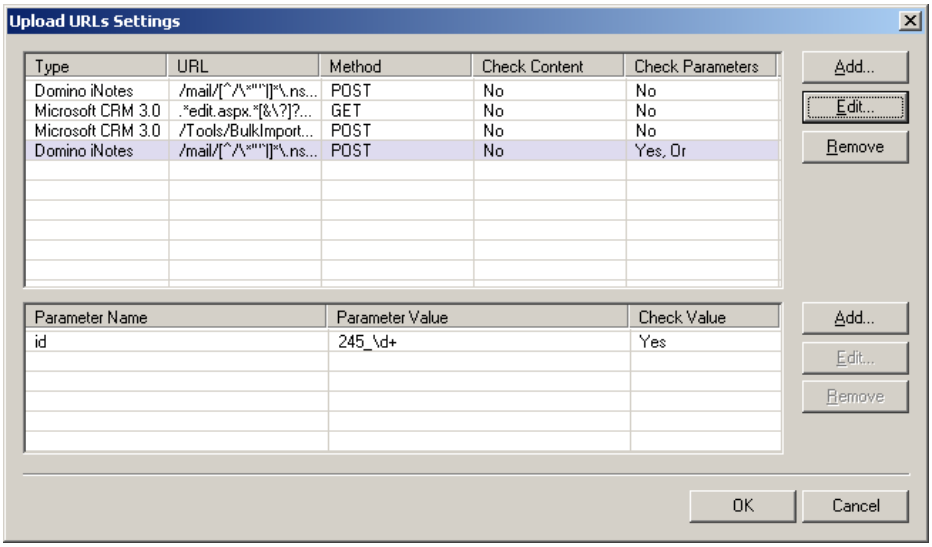


   b) Configure the following:

   - Name: parameter name.
   - Value (optional): parameter value.
   - Check Value: activate this option if you wish to check the value of the parameter. In this case, make sure you enter the parameter value in the appropriate field.

   Click OK.

   The Add URL Parameter dialog box closes. The parameter is added in the Upload URLs Settings dialog box.



c) Repeat steps a–b to add as many parameters as required for the rule.

5. Repeat steps 2–4 to add Upload URLs rules, for all the relevant applications, as required.

   After you next activate the configuration, the rules you define here are used to identify uploads.

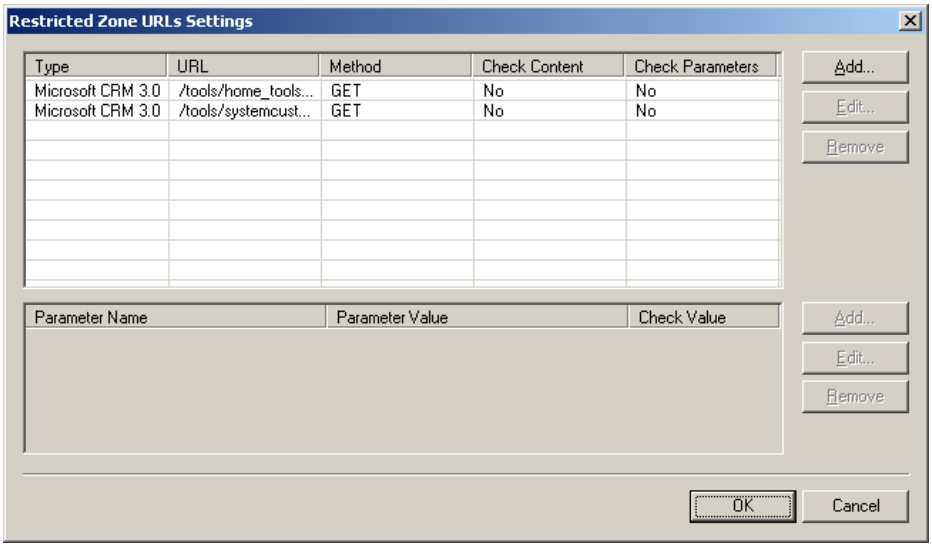
Restricted Zone URLs

This section describes how you configure the rules that the IAG uses in order to identify the application's restricted zone, when the option "Activate Restricted Zone" is activated. For some applications, the IAG supplies you with pre-defined, Application Aware rules, as applicable.

To configure Restricted Zone URLs rules:

1. In the Global URL Settings tab, in the "URL Settings" area, click Configure... next to "Restricted Zone URLs".

   The Restricted Zone URLs Settings dialog box is displayed.

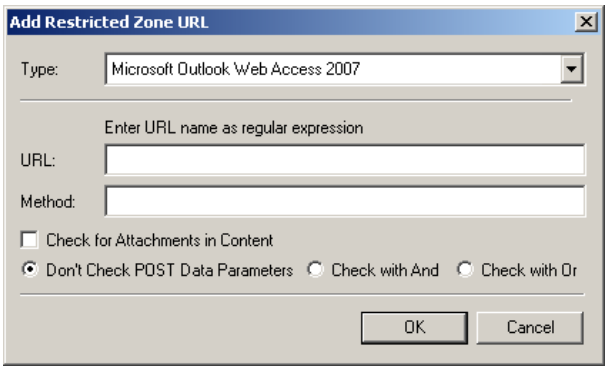


2. In the upper area of the dialog box, click Add...

Tip

- To edit an existing rule, select it and click Edit...
- If POST data parameters are configured for the rule you select, the parameters are listed in the lower area of the dialog box.

The Add Restricted Zone URL dialog box is displayed.



3. Configure the following:
   - Type:
     - Portal trunks: select the application type from the list of relevant applications.
     - Webmail and Basic trunks: select whether to apply this rule to the application server or to the internal website.
   - URL: enter the URL, as follows:

- Use regular expressions, as described in Appendix B: “Regex++, Regular Expression Syntax”.
- If you wish to check query string parameters, make sure the URL includes the query string.

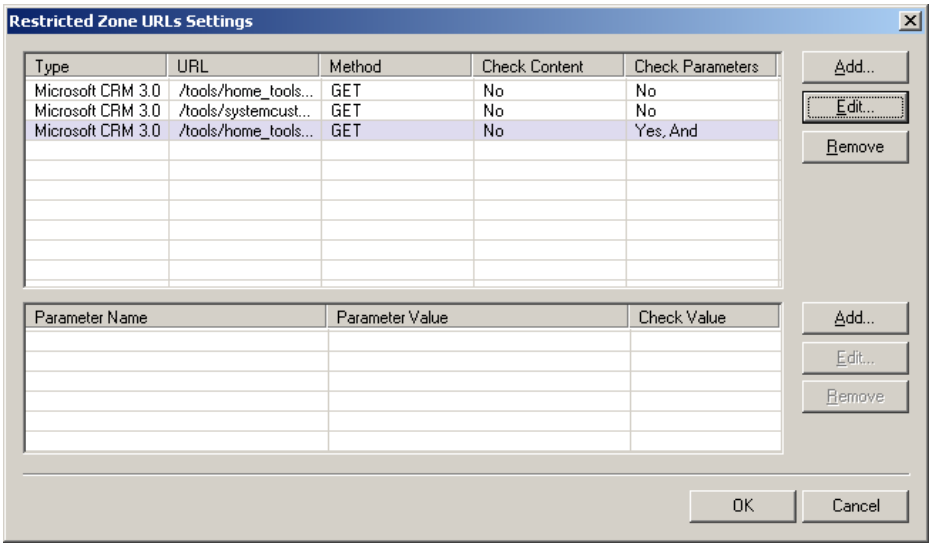
- Method (optional): HTTP method used to access the URL. Multiple methods are comma-separated.
- Check for Attachments in Content: activate this option if you wish to check the content of the URL for attachments. When this option is activated, only URLs containing attachments are considered uploads.
- Select whether to check parameters in POST data, and, if required, the method by which to check them:
  - Check with And: the data must contain all the rule’s parameters, as you define in the parameter list, in the lower area of the Restricted Zone URLs Settings dialog box.
  - Check with Or: the data must contain one or more of the defined parameters.

Note

If you configure the rule to check POST data parameters, make sure you define the parameters in the parameter list, as described in step 4 of the procedure.

Click OK.

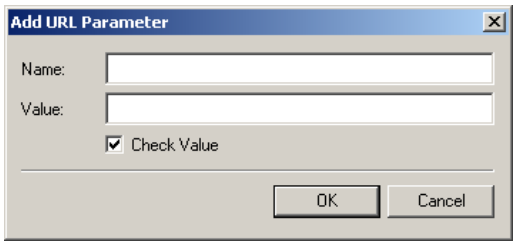
The Add Restricted Zone URL dialog box closes. The rule is added in the Restricted Zone URLs Settings dialog box.



4. If you configured the rule to check parameters, configure the parameters:

   a) In the lower area of the dialog box, click Add...

   The Add URL Parameter dialog box is displayed

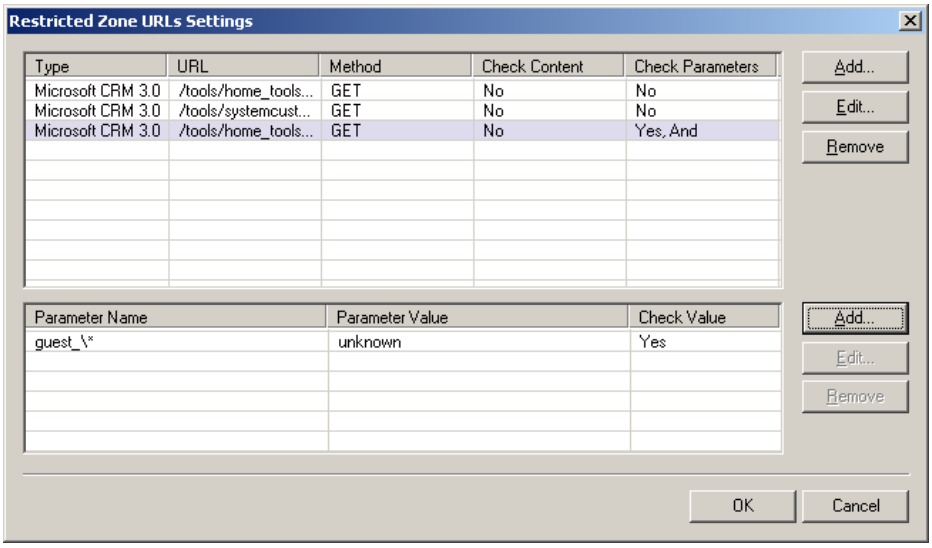


   b) Configure the following:

   - Name: parameter name.
   - Value (optional): parameter value.
   - Check Value: activate this option if you wish to check the value of the parameter. In this case, make sure you enter the parameter value in the appropriate field.

   Click OK.

   The Add URL Parameter dialog box closes. The parameter is added in the Restricted Zone URLs Settings dialog box.



   c) Repeat steps a–b to add as many parameters as required for the rule.

5. Repeat steps 2–4 to add Restricted Zone URLs rules, for all the relevant applications, as required.

   After you next activate the configuration, the rules you define here are used to identify the Restricted Zone.

Ignoring URL Requests in Inactive Session Timeout Calculations

Note

This feature is applicable for Built-In Services, Web Applications, and Browser-Embedded Applications.

In the Session tab, you can define an "Inactive Session Timeout", whereby when a session is inactive for the defined timeout, it expires. Some applications, however, send regular requests for specific URLs, as does, for example, a KeepAlive mechanism. When the IAG calculates the Inactive Session Timeout, those requests are included in the calculation.

For example: if the configured Inactive Session Timeout is 300 seconds, and the application sends a KeepAlive request every 200 seconds, the Inactive Session Timeout is reset every 200 seconds, when the application requests the KeepAlive URL. In this example, the session is never deemed inactive, even if no requests are received from the browser for more than 300 seconds.

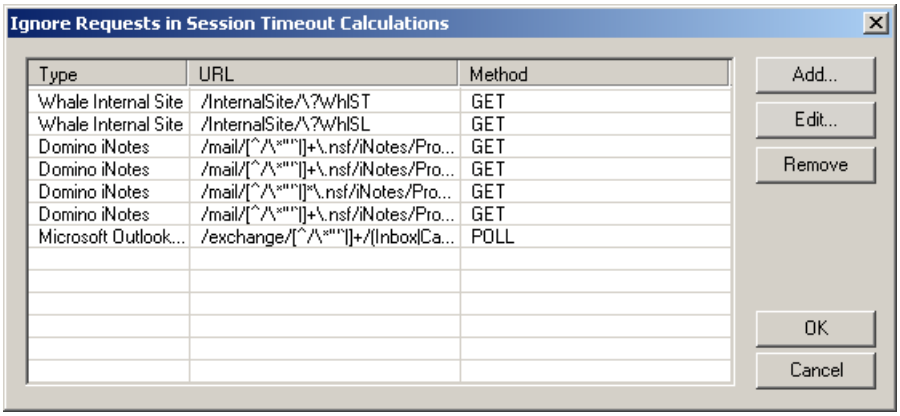
You can configure a list of URLs that will be ignored in the calculation of the Inactive Session Timeout. In the example above, if you configure the KeepAlive URL in the "ignore" list, the request is excluded from Inactive Session Timeout calculations, and the Inactive Session Timeout is not reset when the application requests the KeepAlive URL. If no requests are received from the browser for 300 seconds, the filter deems the session invalid and closes it.

- Activation of the option for an application is determined in the Application Properties dialog box, in the Web Settings tab, by the option "Ignore Requests in Timeout Calculations".
- Internal website URLs that should be ignored during timeout calculations are configured by default. The IAG also applies out-of-the-box, Application Aware rules for supported applications, as applicable.
- This section describes how you configure additional URLs that will be ignored in session timeout calculation, when the option "Ignore Requests in Timeout Calculations" is activated. The list is configured per application-type, and is applied for all applications of this type.

To configure the list of URLs that are ignored in Inactive Session Timeout Calculations:

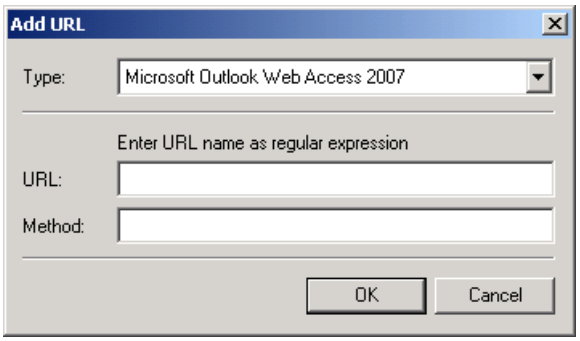
1. In the Global URL Settings tab, in the "URL Settings" area, click Configure... next to "Ignore Requests in Timeout Calculations".

   The Ignore Requests in Session Timeout Calculations dialog box is displayed.



2. Click Add...

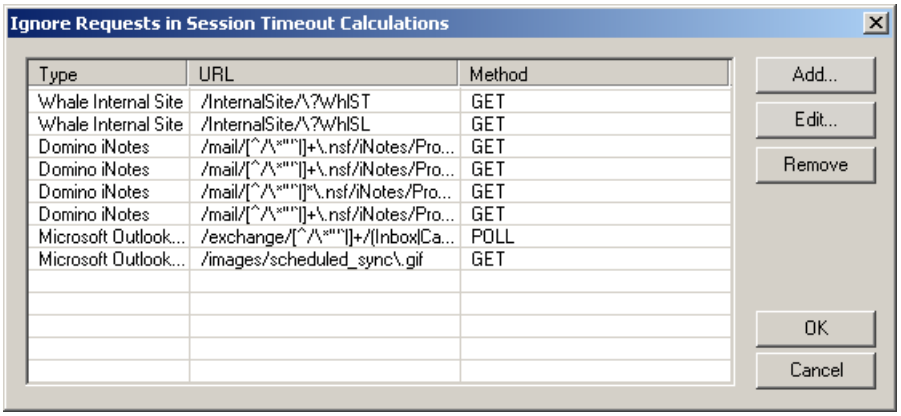
   The Add URL dialog box is displayed.



3. Enter the following:
   - Type:
     - Portal trunks: select the application type from the list of relevant applications.
     - Webmail and Basic trunks: select whether to apply this rule to the application server or to the internal website.
   - URL: URL of the request, using regular expressions. For a description of regular expressions, refer to Appendix B: **"Regex++, Regular Expression Syntax"**.
   - Method (optional): HTTP method used to access the URL.

   Click OK.

   The Add URL dialog box closes. The URL is added in the Ignore Requests in Session Timeout Calculations dialog box.



4. Repeat steps 2–3 to add all the URLs that you wish to ignore in Inactive Session Timeout calculations to the list.

   When the filter calculates the Inactive Session Timeout, it will ignore requests for the URLs listed here.

Configuring a Ruleset in the URL Set Tab

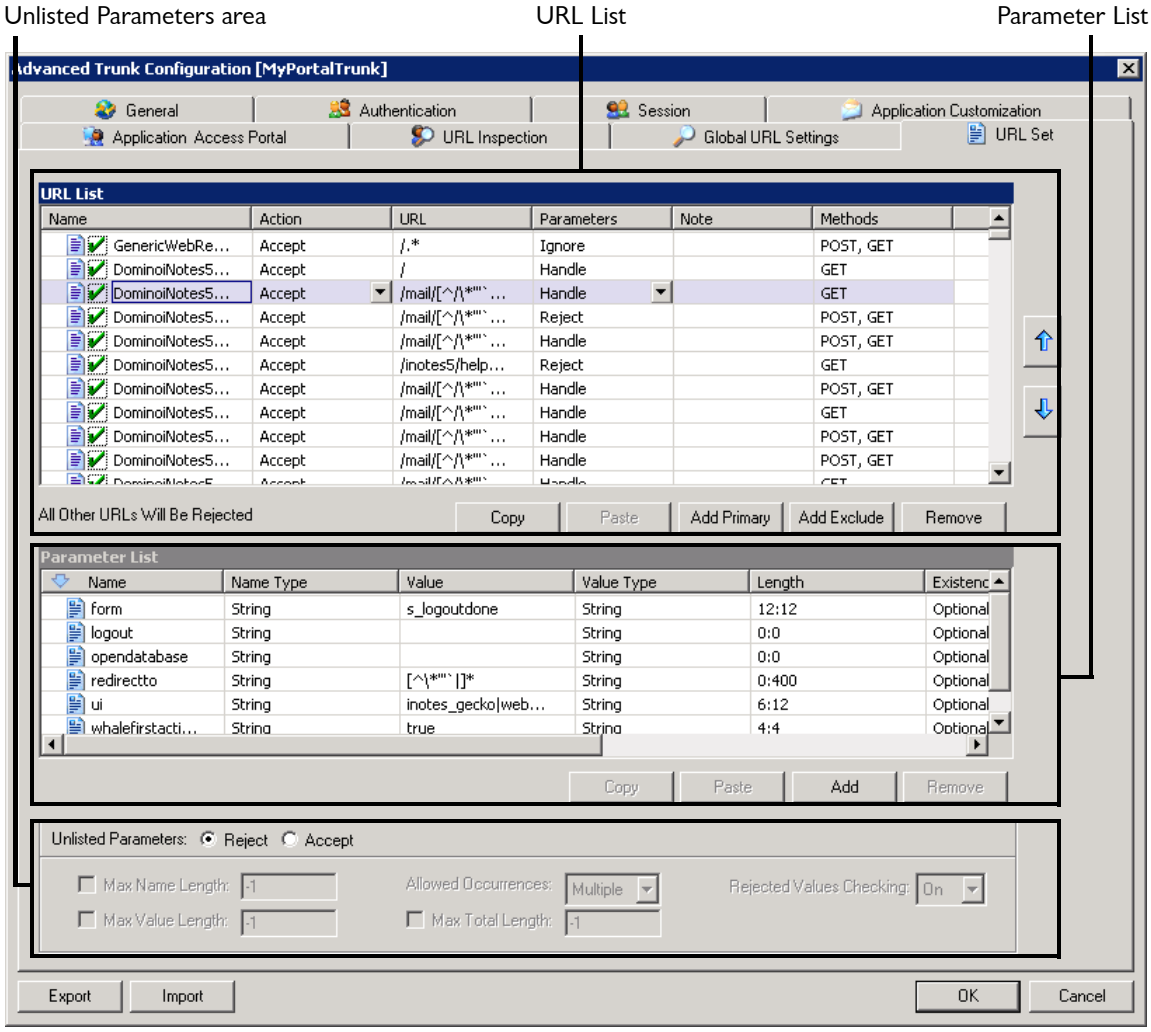
The URL Set tab is divided into three areas:

- URL List, described in "URL Set Tab—URL List" on page 165.
- Parameter List, described in "URL Set Tab—Parameter List" on page 168.
- "Unlisted Parameters" area, described in "URL Set Tab—Unlisted Parameters Area" on page 169.

For instructions on how to use the URL Set tab to create, edit, and delete rules, refer to "Creating, Editing, and Removing Rules" on page 170.

At the bottom part of the tab, the Import and Export buttons enable you to import a pre-defined set of rules, and to export the current ruleset so that it is saved for future use or for use in other trunks or other IAG servers. For further information, refer to "Rule Creating Utility" on page 177.

Figure 30. URL Set Tab



URL Set Tab—URL List

The URL List defines the URL Inspection rules, including system and application-specific pre-defined rules, and the custom rules that you define. URLs that are not listed here will be denied access. The rules are configured and applied per application-type.

For each primary rule in the URL List, you can define exclusionary rules. Exclusions define exceptions to the primary rule. For example, you may wish to reject all graphic `.jpg` files, with the exception of the file: `logo.jpg`. In this case, you create a primary rule to reject all the `.jpg` files, then add an exclusionary rule to accept the file `logo.jpg`.

> **Note**
> When you disable a primary rule, its exclusionary rules are also disabled, accordingly. Once you enable the primary rule again, the exclusionary rules are not automatically enabled. You have to enable each exclusionary rule individually.

URL List parameters are described in Table 24 on page 167.

Note the following:

- Do not change, add, or delete any of the rules protecting the IAG Built-In Services, including: Internal Site, Web Monitor, File Access, Whale Portal, and Certified Endpoint Enrollment. Any changes you make to those rules will be over-written when the IAG software is next upgraded or a patch is applied.
- If you change, add, or delete any of the rules protecting the applications that are supported out-of-the-box, such as SharePoint, Sametime, or Domino iNotes, note that, when the IAG software is next upgraded, or when a patch is applied, the changes you make will only be supported for backward compatibility:
  - For applications that were added to the trunk prior to the upgrade, the ruleset will retain the customization.
  - For new applications that are added after the upgrade or patch application, the customization will not be retained, and the ruleset will be created with the default definitions.

  If you wish to apply the changes to new applications, as well, make sure to back up the changes you made in an external file.
- Disabling the option “Verify URLs” in the Web Settings tab of an application’s Application Properties dialog box disables application-specific URL inspection for that application. For details, refer to the *Intelligent Application Gateway User Guide*, to “Web Settings Tab” on page 73.
- Activating the option “Debug Mode” in the General tab of the Advanced Trunk Configuration window disables URL inspection altogether. For details, refer to the *Intelligent Application Gateway User Guide*, to “Editing in the General Tab” on page 61.

Table 24. URL Set Tab—URL List Parameters

Parameter	Description
Name	Rule name. A rule name must be preceded by the following prefix: • General rules: GENERAL For information about how you define general rules, refer to "Creating a Set of General Rules" on page 171. • Portal rules: Portal • Internal website rules: InternalSite • Application rules: application-type. For Portal and Webmail trunks, you must enter the application-type as defined in the IAG. For details, refer to "Finding the Application-Type" on page 44. For Basic trunks, the application-type is always OtherWeb. Rule names are separated from the prefix by an underscore. For example: GENERAL_IISRule1. **Note:** Exclusionary rules do not require a prefix.
Action	Select what action is taken when this rule is triggered: • Accept: the IAG checks the URL against the rule. If the URL passes all checks, the IAG returns the requested contents to the browser. • Reject: the request is denies. The IAG sends the error message defined in the URL Inspection tab to the browser.
URL	A URL or group of URLs to which the rule is applied. Use regular expressions to describe URLs. For a description of the regular expressions used with rulesets, refer to Appendix B: "Regex++, Regular Expression Syntax".
Parameters	Action to take when the request contains parameters: • Reject: reject the request. • Handle: check the parameters. Define the list of parameters to check for this URL rule in the "Parameters List" area of the URL Set tab, or in the Global URL Settings tab. • Ignore: do not check the parameters.
Note	Optional: a note describing the rule.
Methods	Defines acceptable request method or methods for this URL (multiple selection is possible). If the request uses a method other than those defined here, it is rejected. **Tip:** Once you select the method or methods from the list, click outside the list to apply.

URL Set Tab—Parameter List

The Parameter List defines the rules that the IAG will apply to URL parameters when a URL rule is set to "handle" parameters. This list displays the parameter rules for the selected URL rule. Parameter List parameters are described in Table 25 on page 168.

Note

In addition to the parameters you define in the Parameter List, you can define global parameter rules that will apply to all the URLs defined in the URL List. Refer to "Configuration in the Global URL Settings Tab" on page 148.

Table 25. URL Set Tab—Parameter List Parameters

Parameter	Description
Name	Parameter name. Must match name sent by the browser.
Name Type	Type of parameter name: String or Regular Expression. For a description of the regular expressions used with rulesets, refer to Appendix B: "Regex++, Regular Expression Syntax".
Value	Parameter value. Depends on the type of value, as defined in the Value Type column. • For strings, enter a regular expression that defines the acceptable values. For a description of the regular expressions used with rulesets, refer to Appendix B: "Regex++, Regular Expression Syntax". • For Integer and Real parameters, a comma divides between values; a colon represents a range of values. **Note:** Parameter values must be listed according to their length, in a descending order, from the longest parameter to the shortest parameter. For example: Value editattach|contents|dialog
Value Type	Type of parameter value: Integer, Real, or String.
Length	Length of the value.

Table 25. URL Set Tab—Parameter List Parameters (Cont'd)

Parameter	Description
Existence	• Mandatory: URL will only be considered valid if this parameter is present. • Optional: this parameter is optional. • Reject: if this parameter appears in the request, the request will be judged invalid.
Occurrences	Define whether the parameter can appear in the URL once, or multiple times.
Max total length	Total length of parameter values of all occurrences of this parameter.
Rejected values checking	Select whether to check parameters against the "Rejected Values" list, defined in the Global URL Settings tab. • On: check against the list • Off: do not check against the list

URL Set Tab—Unlisted Parameters Area

In the "Unlisted Parameters" area you define how to handle parameters where the URL rule is set to "handle" parameters, and no rule is defined for this parameter in the Parameter List:

- **Reject**: URL is rejected.
- **Accept**: URL is accepted if it matches the rules configured in this area, as listed in Table 26 on page 169.

Table 26. URL Set Tab—Unlisted Parameters

Parameter	Description
Max Name Length	Maximal length of a parameter name. Default: -1, where length of the name is not checked (length is unlimited).
Max Value Length	Maximal length of a parameter value. Default: -1, where length of the value is not checked (length is unlimited).

Table 26. URL Set Tab—Unlisted Parameters (Cont'd)

Parameter	Description
Allowed Occurrences	Define whether the same parameter can appear in the URL multiple times.
Max Total Length	Available only when "Allowed occurrences", above, is set to "Multiple". Total length of parameter values of all occurrences of this parameter. Default: -1, where the total length of the values is not checked (total length is unlimited).
Rejected Values Checking	Select whether to check parameters against the "Rejected Values" list, defined in the Global URL Settings tab. • On: check against the list • Off: do not check against the list

Creating, Editing, and Removing Rules

This section describes how you use the URL Set tab to:

- Define primary and exclusionary rules in the URL List, and add rules to the Parameter list, as described in "Adding New Rules" on page 170.
- Create a set of general rules that are applied globally, for the entire trunk, as described in "Creating a Set of General Rules" on page 171.
- Copy, edit, or remove rules, in both the URL and the Parameter lists, as described in "Copying, Editing, and Removing Rules" on page 172.

Adding New Rules

The following procedures describe how you add new rules to the URL and Parameters lists.

To add a primary rule in the URL List:

1. Under the URL List, click Add Primary.
   The rule is added at the end of the list of existing rules.
2. Add rule parameters, as described in Table 16.

To add an exclusionary rule to a primary rule:

1. Select the primary rule to which you wish to add the exclusionary rule; under the URL List, click Add Exclude.

   An Exclusionary rule is added under the primary rule.
2. Add rule parameters, as described in Table 24.

To add a rule in the Parameter List:

1. Under the Parameter List, click Add.

   The parameter rule is added at the end of the list of existing parameter rules.
2. Define the parameter rule, as described in Table 25.

Creating a Set of General Rules

You can create a set of general rules, that are applied globally, for the entire trunk, and are checked prior to the application rules. You create the set of general rules in the URL List, including:

- Any number of general rules with a Reject action
- A general rule with the Accept action, which must be the last rule in the list of general rules.

The general rules are run before application rules, as follows:

- The general rules with a Reject action are run. If the request is rejected at this stage, it is not subjected to examination by the application rules.
- The general rule with the Accept action is run. Its function is to indicate to the IAG that the set of general rules has been completed, and that the request should now be submitted for examination by the relevant application rules. Note that if you do not configure this rule, the request will not be submitted for further examination, and will be rejected.

The following procedure describes how you create a set of general rules.

To create a set of general rules:

1. Create the required general rules, with the Reject action. For more information about creating and editing rules, see "Creating, Editing, and Removing Rules" on page 170.
2. Create a general rule, with these values:
   - **Name:** `GENERAL_<Rule_Name>`
   - **Action:** `Accept`
   - **URL:** `/.*`
   - **Parameters**: `Ignore`

- **Methods:** make sure you include all of the methods that might be used by any of the applications in this trunk.

  For example:

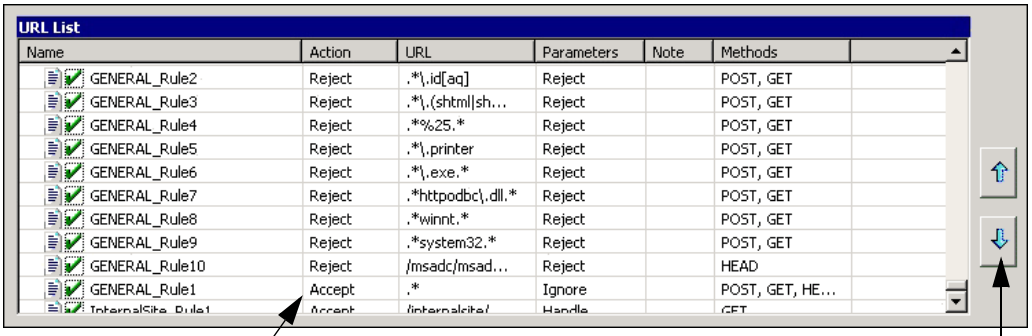
  - For the SharePoint Portal Server 2003 application, include all of these methods: `GET`, `POST`, `HEAD`, `MOVE`, `COPY`, `PUT`, `DELETE`, `PROPFIND`, `OPTIONS`, `LOCK`, `UNLOCK`, `MKCOL`, `PROPPATCH`, `GETLIB`
  - For the Microsoft Outlook Web Access 2003 application, include all of these methods: `GET`, `POST`, `HEAD`, `COPY`, `TRACE`, `SEARCH`, `PUT`, `PROPPATCH`, `MOVE`, `PROPFIND`, `BDELETE`, `POLL`, `BMOVE`, `BCOPY`, `BPROPPATCH`, `SUBSCRIBE`, `MKCOL`, `DELETE`.

Caution

Make sure that all of the values defined for this general Accept rule are correct, as described above. If not, the request will be rejected, even if the ruleset includes an application rule that accepts this request.

Make sure that the general rule with the Accept action is the last rule in the set of general rules. You can use the arrows at the right of the URL List to move a selected rule up or down, to arrange the order as required.

Following is an example of a set of general rules:



URL List

Name	Action	URL	Parameters	Note	Methods
GENERAL_Rule2	Reject	.*\.id[aq]	Reject		POST, GET
GENERAL_Rule3	Reject	.*\.(shtml|sh...	Reject		POST, GET
GENERAL_Rule4	Reject	.*%25.*	Reject		POST, GET
GENERAL_Rule5	Reject	.*\.printer	Reject		POST, GET
GENERAL_Rule6	Reject	.*\.exe.*	Reject		POST, GET
GENERAL_Rule7	Reject	.*httpodbc\.dll.*	Reject		POST, GET
GENERAL_Rule8	Reject	.*winnt.*	Reject		POST, GET
GENERAL_Rule9	Reject	.*system32.*	Reject		POST, GET
GENERAL_Rule10	Reject	/msadc/msad...	Reject		HEAD
GENERAL_Rule1	Accept	.*	Ignore		POST, GET, HE...
InternalSite_Rule1	Accept	/internalsite/	Handle		GET

The "Accept" general rule is the last rule in the set of general rules

Use these buttons to move a selected rule up or down in the list

The IAG runs the set of general rules before proceeding to the application-specific rules.

Copying, Editing, and Removing Rules

The following procedures describe how you use the URL Set tab to copy and edit existing rules, or to remove them.

To copy an existing rule:

1. Select the rule and click Copy below the applicable list:
   - URL List for URL rules, or,
   - Parameter List for Parameter rules
2. Click Paste below the applicable list.

Note

If you are copying an exclusionary rule, before pasting it, select the primary rule under which you wish to add the copied rule.

The rule, including its exclusionary rules where applicable, is added at the bottom of the rule list, or, if it is an exclusionary URL rule, under the selected primary rule. For primary URL rules and parameter rules, a temporary name is assign to the rule.

3. Where applicable, name the copied rule.

To edit rule parameters:

1. Place the cursor over the field of the parameter you wish to edit, and click inside it.

   If you have to enter parameter values as text, the color of the text field changes to light blue. If you have to select from a list of options, a combo box list drops down.
2. Click inside the text box, enter parameter value, and press <Enter>,

   Or,

   Select the parameter from the list.

To remove rules from the URL or Parameter lists:

1. Select the rule and click Remove below the applicable list:
   - URL List for URL rules, or,
   - Parameter List for Parameter rules

   The rule is removed, together with all its Exclusions where applicable.

Using Variables in URLs

This section describes how you can configure URL inspection rules, where the path of the application URL includes a user-dependent variable. In this manner you can ensure users are only granted access to the portion of the application for which they are authorized, and the same user is logged in throughout the session. For example: organizations where users are allowed access to their department's database only, or implementations where users need only access a specific URL set, and the login user name is part of the URLs they are trying to access.

In order to configure a user variable as part of the URL, you have to take the following steps:

- For each application where you wish to use variables, create a ruleset that contains the variable, as described in "Creating a Ruleset" on page 174.
- For each trunk where you wish to use variables, configure the authentication pages, as described in "Configuring the Use of URL Variables in the Authentication Pages" on page 175.
- For each application where you wish to use variables, activate the option "Use Variables in URLs" in the Web Settings tab of the Application Properties dialog box. For details, refer to the *Intelligent Application Gateway User Guide*, to "General Web Settings" on page 75.

Note

Before you start configuring this option, read the information provided in "Important Notes" on page 176.

Creating a Ruleset

Note

For a full description of how you create rulesets, refer to "Configuring a Ruleset in the URL Set Tab" on page 164. Pay special attention to the rule naming convention, as described in "URL Set Tab—URL List" on page 165.

In the URL Set tab of the Advanced Trunk Configuration window, under "URL List", create a rule with an Accept action. Under "URL", enter a URL where the user variable is represented by a "symbol" parameter; note that the "symbol" parameter has to be enclosed by angle brackets, as demonstrated in the examples that follow.

For example:

A URL where a portion of the path is a user name variable changes according to the user that logs in:

- For the user Mark, the URL is `/exchange/Mark/myhome`
- For Benjamin, the URL is `/exchange/Benjamin/myhome`
- And for Sean, the URL is `/exchange/Sean/myhome`

If you wish to represent the user name with the "symbol" `<UserName>`, for example, create a rule with the following URL:

```
/exchange/<UserName>/myhome
```

For Portal trunks, the configuration is per-application; if you use URL variables in multiple applications, repeat this procedure for each application.

For Webmail or Basic trunks, the configuration is per-trunk; if you use URL variables in multiple trunks, repeat this procedure for each trunk.

Configuring the Use of URL Variables in the Authentication Pages

This section describes how you configure the authentication pages to include both the "symbol" parameter and the parameter value. The configuration is per-trunk; if you use URL variables in multiple trunks, repeat the procedure described here for each trunk.

To configure user variables in the authentication pages:

1. Access the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\von\InternalSite\inc\CustomUpdate`

2. Under the `customUpdate` folder, create an `inc` "hook", which will be activated before the `PostValidate.asp` reaches the client side:

   `PostPostValidate.inc`

   Name the file as follows:

   `<Trunk_Name><Secure(0=no/1=yes)>PostPostValidate.inc`

   For example:

   For an HTTPS trunk named "WhaleSite", create the file:

   `WhaleSite1PostPostValidate.inc`

> **Tip**
> If a file by this name already exists, you can use the existing file; you do not need to create a new file in this case.

3. In the hook file you created in step 2, add the following lines:

```
<%
SetSessionResourceParam g_cookie, <Application_ID>, "ParamName",
"Parameter_Symbol"
SetSessionResourceParam g_cookie, <Application_ID>, "ParamVal",
"Parameter_Value"
%>
```

For example:

```
<%
SetSessionResourceParam g_cookie, <Application_ID>, “ParamName”,
"<UserName>"
SetSessionResourceParam g_cookie, <Application_ID>, “ParamVal”,
Session(“user_name1”)
%>
```

Where in this example:

- `<UserName>` is the “symbol” parameter you entered in the URL List, in the URL Set tab.
- `Session(“user_name1”)` is a dynamic representation of the lead user name entered by the user during login. The value of `user_name1` changes according to the user that is logged in, for example: Mark, Benjamin, or Sean.

Important Notes

Before you configure this option, take note of the following.

Limitations

You can only configure one user parameter per application; each URL can contain one instance of this parameter. You **cannot** configure the following examples:

- `/exchange/mark/inbox/mark`, where the dynamic user name “mark” appears twice in the URL path,

or,

- `/exchange/benjamin/sean/calendar`, where two dynamic user names, “benjamin” and “sean”, are part of the URL path.

Repetitions

If a value of a dynamic, user-dependent URL parameter also appears as a portion of the URL path **prior** to the dynamic parameter, you have to take extra measures when configuring the feature.

Example 1: Outlook Web Access 2000

Take a look at the example used in “Creating a Ruleset” on page 174. Let’s say one of the values of the dynamic user parameter is the user name “`chang`”. Note that “`chang`” also appears in the URL path, as part of the word `“exchange”`. In this case, you have to take the following steps:

1. In the URL Set tab, edit the rule’s URL and take out the slashes before and after the dynamic parameter. In our example:

   `/exchange<UserName>myhome`

2. In the authentication page, edit the `ParamVal` line as follows:

   ```
   SetSessionResourceParam g_cookie, <Application_ID>,
   "ParamVal", "/"+"Parameter_Value"+"/"
   ```

   For example:

   ```
   SetSessionResourceParam g_cookie, <Application_ID>,
   "ParamVal", "/"+Session("user_name1")+"/"
   ```

Example 2: Domino Webmail

In this example, we are creating a rule to accept the URL:

```
/ToDo/User_Name.nsf/Weekly
```

Where the rule's URL looks like this:

```
/ToDo/<UserName>\.nsf/Weekly
```

However, if one of the users in this example is "`tod`", note that "tod" also appears in the URL path, as part of the word `"ToDo"`. In this case, you have to take the following steps:

1. In the URL Set tab, edit the rule's URL and take out the file extension (`.nsf`) after the dynamic parameter. In our example:

   ```
   /ToDo/<UserName>/Weekly
   ```

2. In the authentication page, edit the `ParamVal` line as follows:

   ```
   SetSessionResourceParam g_cookie, <Application_ID>,
   "ParamVal", "Parameter_Value"+".nsf"
   ```

   For example:

   ```
   SetSessionResourceParam g_cookie, <Application_ID>,
   "ParamVal", Session("user_name1")+".nsf"
   ```

Rule Creating Utility

The IAG rule creating utility automates the process of generating positive-logic rulesets: it records HTTP or HTTPS requests, converts them into rules, and processes the list of rules into an optimized ruleset. The ruleset can then be imported into other trunks.

- The rule creating utility, including detailed instructions on how to operate it, is described in "Creating Rulesets with the Utility" below.
- For instructions on how to export and import rulesets, refer to "Exporting and Importing Rulesets" on page 187.

Creating Rulesets with the Utility

The following sections describe how you create rulesets using the IAG rule creating utility:

- First, you use the online Recorder utility to record the requests you wish to enable in the trunk.
- The Optimizer utility then takes the list of recorded requests and optimizes it by eliminating duplications and combining requests that hold similar parameters. The optimized ruleset is then ready to be imported into the trunk.

Note

You can create rulesets defining which requests will be accepted, thus rejecting any other requests; you can not create rulesets listing requests that should be rejected.

Recording Rules

The online Recorder utility enables recording of both HTTP and HTTPS requests. The requests can be recorded while the IAG is in production.

By default, each instance of the Recorder records all the requests that arrive at the trunk; optionally, you can configure it so that it only records requests that arrive at a specific application. You can also activate a number of online Recorders, to record a number of trunks.

In order to activate the online Recorder utility for a trunk, you take the following steps:

- Configure the Recorder utility, as described in “Configuring the Online Recorder Utility” on page 179.
- Use a browser to generate the requests you wish to enable in the ruleset, as described in “Recording the Ruleset” on page 180.

Once you activate the online Recorder utility for a trunk, it records all the requests that arrive at the IAG. You can then optimize the recorded rules and create a ruleset, as described in “Optimizing the Ruleset” on page 181.

Tip

Repeat the steps described in the following sections for all the trunks through which you wish to record rulesets.

Configuring the Online Recorder Utility

To configure the online Recorder for a trunk:

1. Copy this file:

   `…\Whale-Com\e-Gap\Von\Conf\WhlFiltRecorder.ini`

   into one of the following folders:

   - If you are using the default Recorder settings, copy the file into this folder:

     `…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf`

     Proceed to "Recording the Ruleset" on page 180.

   - If you wish to make changes to the default settings, copy the file into the `CustomUpdate` folder under the trunk's `Conf` folder; if the `CustomUpdate` folder does not exist, create it:

     `…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf\CustomUpdate`

     Proceed to step 2.

> **Note**
> If the `CustomUpdate` folder already exists and contains the `WhlFiltRecorder.ini` file, this file may include previously defined custom changes. In this case, make your changes in the existing file and do not overwrite it.

2. Configure the file, as shown in the table below and in the following sample.

Parameter	Description	Values/Notes
Type	Functionality of the output file.	**Must** be 1.
OutputFilesPath	Path where the output files are created.	Must be the same folder where the Optimizer utility files are located. Folder must be created before you run the Recorder application.
HttpOutputFileName	Name of the output file for HTTP rules.	
HttpsOutputFileName	Name of the output file for HTTPS rules.	

Parameter	Description	Values/Notes
ApplicationID	Optional. When this option is applied, the Recorder records only requests that arrive at the application specified here.	You can view and copy the application ID in the Application Properties dialog box, in the General tab.
AppendToOutput	In cases where a rule list already exists, select whether to append the output to the existing file, or whether to create a new file.	0=do not append. Other value=append.

Note
Make sure to use different names for the HTTP and HTTPS output files.

A sample `WhlFiltRecorder.ini` file:

```
[GENERAL]
# 1 - rules file
Type                        = 1
OutputFilesPath             = C:\temp\
HttpOutputFileName          = OutHttpRecord.ini
HttpsOutputFileName         = OutHttpsRecord.ini
# 0 - new file other - append
AppendToOutput              = 0
```

Recording the Ruleset

To record the ruleset:

- Use a web browser to generate requests to the application server, which you wish to enable in the ruleset.

  The requests are recorded by the Recorder utility and a rule list is created in the output file defined in `WhlFiltRecorder.ini`.

Tip

To de-activate the Recorder utility, rename the `WhlFiltRecorder.ini` file, in one of the following locations:

- If you are using the default online Recorder settings:

  `...\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf`

- If you have made changes to the default settings:

  `...\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf\CustomUpdate`

Once you finish recording all the requests you wish to enable in the ruleset, take the following steps:

1. At the browser, stop generating requests to the application server.
2. At the Configuration program, click to activate the configuration.
3. Optimize the rule list, as described in "Optimizing the Ruleset" on page 181.

Optimizing the Ruleset

Once you record the rules, using the online Recorder utility, you optimize the ruleset using the Optimizer utility, as follows:

- Optionally configure the file that holds the optimization settings, `Optimizer.ini`.
- Optionally configure the file that holds the settings of the additional, user defined optimizing rules, `userRules.ini`.
- Run the Optimizer utility. The application optimizes the rule base that was generated by the Recorder utility in the following stages:
  - Merges all identical requests.
  - Combines all the requests with the same URL into one rule, and sets the statistics for all the URL parameters. Each combined rule will contain all the parameters that appeared in any request for the URL of the rule.
  - Reads the user-defined optimizing rules, and, according to this input, further combines matching rules.

The ruleset is then ready to be imported into the trunk.

Note

The Optimizer utility files are located in the following folder:
`...\Whale-Com\e-Gap\Utils\HttpRecorder\`

Optional Optimizer.ini Configuration

This section describes how you configure optional customized Optimizer utility settings.

Note

If you customize the `Optimizer.ini` file, it is recommended that the `Optimizer.ini` and `userRules.ini` files are both located in the `CustomUpdate` folder. If the `userRules.ini` file is located in a different folder than the `Optimizer.ini` file, you must enter the absolute path for the `userRules.ini` file in `user_rules_file_name` parameter of the `Optimizer.ini` file.

To customize the Optimizer.ini file:

1. Copy the file `Optimizer.ini` from this folder:
   `...\Whale-Com\e-Gap\Utils\HttpRecorder`
2. Access the following `CustomUpdate` folder; if it does not exist, create it:
   `...\Whale-Com\e-Gap\Utils\HttpRecorder\CustomUpdate`
3. Place the file `Optimizer.ini` in the `CustomUpdate` folder.

Note

If the `CustomUpdate` folder already contains the `Optimizer.ini` file, this file may include previously defined custom changes. In this case, make your changes in the existing file and do not overwrite it.

4. Set the parameters in the `Optimizer.ini` file according to the descriptions and notes in the following table, and to the sample file provided thereafter.

Key	Description	Parameters/Notes
recorder_ file_name	Name of the file generated by the Recorder utility.	The file name is defined in the Recorder configuration file, `WhlFiltRecorder.ini`, in one of the following parameters: `HttpOutputFileName` Or, `HttpsOutputFileName`
user_rules_ file_name	Name of the file that holds the additional, user defined optimizing parameters.	By default: `userRules.ini`.*

Key	Description	Parameters/Notes
optimized_rules_file_name	Name of the output file, that is, the optimized ruleset that will be imported into the trunk.	In order to import the file into the trunk, file name **must** be `HTTP_RuleSet.ini`.
log_level	Select whether to create a log file, and, if a log is created, the level of severity of the events recorded in the file.	0=no log is created. 1=record high severity events. 2=record high and medium severity events. 3=record all events.
create_first_level_rules_file	Select whether to create the first-level rule file, that is, the rule list that the Optimizer utility generates before it runs the additional, user defined optimizing parameters.	0=no. 1=yes.

* If the `Optimizer.ini` and `userRules.ini` files are not located in the same folder, that is, the `userRules.ini` file is not located in the `CustomUpdate` folder, you must enter the absolute path for the `userRules.ini` file.

A sample `Optimizer.ini` file:

```
[GENERAL]

# The recorded HTTP requests file
#-------------------------------
recorder_file_name             = outRecord.ini

# The user rules file
#---------------------
user_rules_file_name           = userRules.ini

# The output file name
#---------------------
optimized_rules_file_name      = HTTP_RuleSet.ini

# log_level can take the values:
#
# 0 - Log file is not active at all.
# 1 - Only high severity events will be written to the log.
# 2 - High & medium severity events will be written to the log.
# 3 - All events will be written to the log.
#-------------------------------------------------------------
log_level                      = 0

# Indicate whether to create the first level rule file
# Can take the values: 1 = true or 0 = false
#---------------------------------------------
create_first_level_rules_file  = 0
```

Optional userRules.ini Configuration

This section describes how you configure additional, optional, user-defined optimizing rules.

Note

If you customize the `userRules.ini` file, it is recommended that the `Optimizer.ini` and `userRules.ini` files are both located in the `CustomUpdate` folder. If the `userRules.ini` file is located in a different folder than the `Optimizer.ini` file, you must enter the absolute path for the `userRules.ini` file in `user_rules_file_name` parameter of the `Optimizer.ini` file.

To customize the userRules.ini file:

1. Access the following `CustomUpdate` folder; if it does not exist, create it:
   `…\Whale-Com\e-Gap\Utils\HttpRecorder\CustomUpdate`
2. Copy the file `userRules.ini` from this folder:
   `…\Whale-Com\e-Gap\Utils\HttpRecorder`
   and place it in the `CustomUpdate` folder you accessed in step 1.

Note

If the `CustomUpdate` folder already contains the `userRules.ini` file, this file may include previously defined custom changes. In this case, make your changes in the existing file and do not overwrite it.

3. The `userRules.ini` file is divided into two parts. At the top, the "General user parameters" rule holds parameters that apply to all the consecutive rules, as described in the following table:

Key	Description	Parameters/Notes
is_url_ casesensitive	Defines whether URLs are case sensitive.	0=case insensitive 1=case sensitive
is_param_name _casesensitive	Defines whether parameter names are case sensitive.	0=case insensitive 1=case sensitive
is_param_val_ casesensitive	Defines whether parameter values are case sensitive.	0=case insensitive 1=case sensitive

The "User defined rules" area is divided into sections, each section holding one rule. The following table describes the parameters of one rule; a sample file is provided thereafter. Create as many rules as you wish.

Key	Description	Parameters/Notes
match_name	A regular expression to be matched.	For example: `.*\.asp`
method	Method or methods allowed in the rule.	Only the GET and POST methods are applicable **Note:** Where both methods are allowed, separate them with a comma, without a space: GET,POST

Note

- Section/rule names must be unique.
- The rules are handled in a lexicographic order, as in a dictionary, and not according to the order by which they appear in the file. We recommend that you name the user-defined rules using a numeric method. For example:
  `Rule_1`
  `Rule_2`
  `Rule_n`

A sample `userRules.ini` file:

```
# General user parameters
#------------------------

[RULE_0]
is_url_casesensitive            = 1
is_param_name_casesensitive     = 0
is_param_val_casesensitive      = 1

# User defined rules
#-------------------

[RULE_1]
method                          = GET,POST
match_name                      = .*\.asp

[RULE_2]
method                          = GET
match_name                      = .*\.gif
```

Running the Optimizer Utility

This section describes how you run the Optimizer utility, in order to optimize the ruleset generated by the Recorder utility. It is identical regardless of whether you have created custom Optimizer utility files in a `CustomUpdate` folder, or are using the default settings.

The Optimizer utility checks requests against the rule parameters in the following order:

❶ `match_name`, including `casesensitive` parameters

❷ `method`

To run the Optimizer utility:

- Open a Command Prompt and run the following file:

  `…\Whale-Com\e-Gap\utils\HttpRecorder\rulesoptimizer.exe`.

  The application runs on the input ruleset, and optimizes it. Once the optimization is finished, a message is displayed in the Command Prompt.

  The Optimizer utility generates the output ruleset in the file `HTTP_RuleSet.ini`*, which it creates under the same folder where it had run. You can import the file into any HTTP trunk, as described in "Importing Rulesets" on page 189.*

Exporting and Importing Rulesets

This section describes the following:

- How you export the set of rules that is currently defined in the trunk, in "Exporting Rulesets" on page 188.
- Once you create or export a ruleset, you can import it into other trunks. Refer to "Importing Rulesets" on page 189.

Note

If you wish to import the rules into other IAG servers, make sure all servers are installed with the same encryption keys. For details, refer to "Creating Encryption Keys" on page 20.

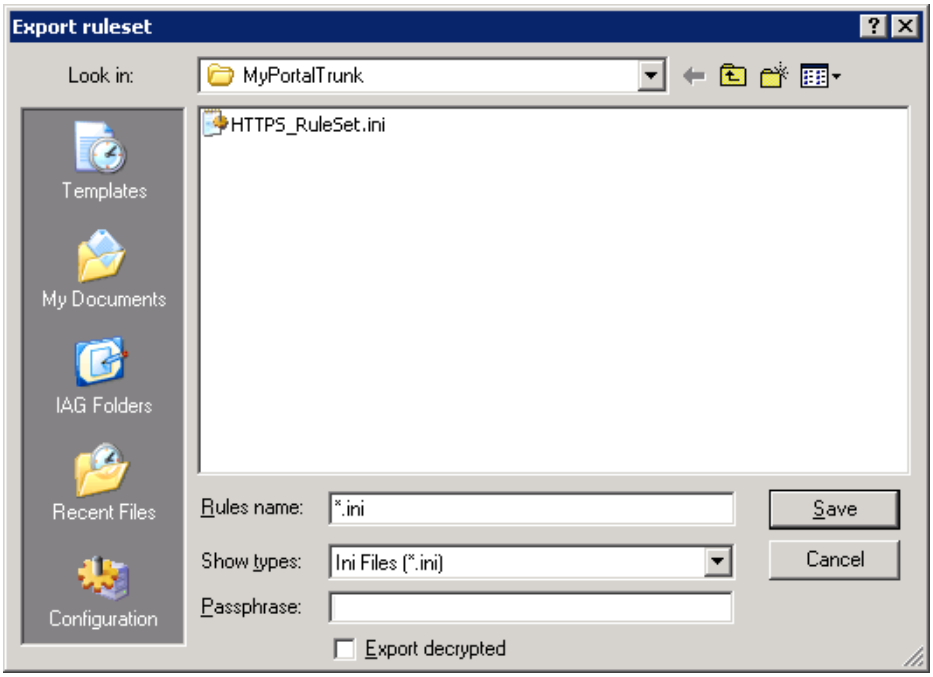
Exporting Rulesets

You can export any ruleset that is defined in an HTTP or an HTTPS trunk. You can then import the ruleset into other HTTP and HTTPS trunks, as described in "Importing Rulesets" on page 189.

To export a ruleset:

1. In the Advanced Trunk Configuration window, in the URL Set tab, click Export.

   The Export Ruleset dialog box is displayed.



2. Enter the following:
   - Select the location to where you wish to export the ruleset.
   - Enter your passphrase.
   - By default, the file is encrypted when it is exported. If you wish the file not to be encrypted, select the option **Export decrypted**:

     ☑ Export decrypted

   Click Export.

   The Export Ruleset dialog box closes. The ruleset is exported to the selected location.

Importing Rulesets

You can import rulesets as follows:

- Rulesets that are created by the rule creating utility, as described in "Creating Rulesets with the Utility" on page 178, or any pre-defined rulesets, can be imported into both HTTP and HTTPS trunks.
- Rules sets that you export from one trunk, as described in "Exporting Rulesets" on page 188, can be imported into other trunks of the same type, either HTTP or HTTPS.
- You can find rulesets that support spell checkers by AutoSpell (www.spellcecker.com) in the following location on the IAG:

  `…\Whale-Com\e-Gap\Von\Samples\RuleSet`

  - For Microsoft Outlook Web Access 2000 SP3: `Ruleset_SpellCheck_ForOWA2000SP3.rul`
  - For Microsoft Outlook Web Access 5.5: `RuleSet_SpellCheck_ForOWA55.rul`

This section describes the import procedure.

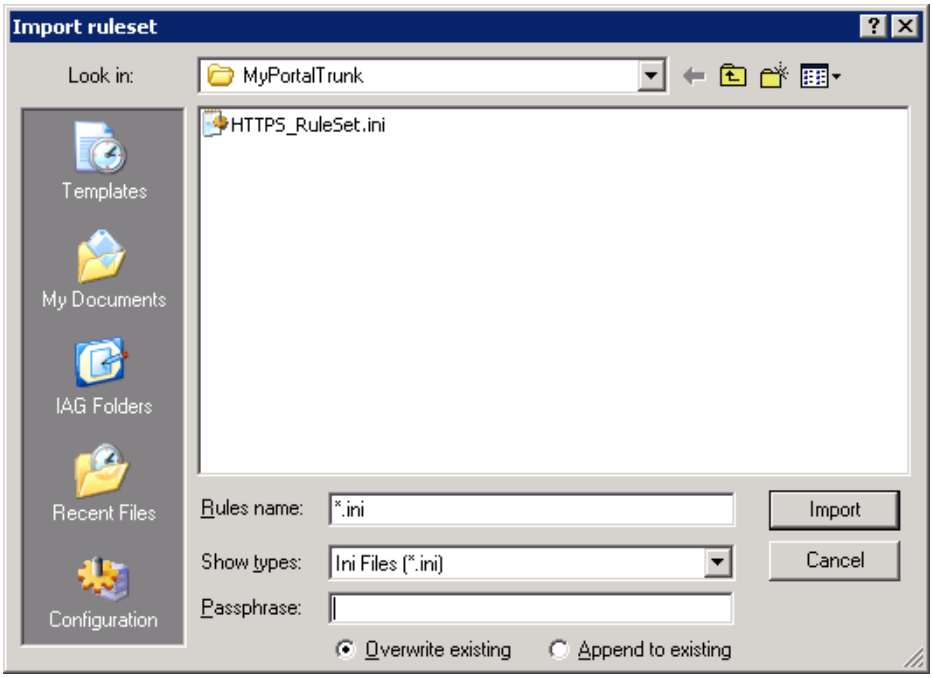
Note

If you wish to import rulesets you create on one IAG server into another IAG server, make sure both servers are installed with the same encryption keys. Refer to "Creating Encryption Keys" on page 20.

To import a ruleset:

1. In the Advanced Trunk Configuration window, in the URL Set tab, click Import.

   The Import Ruleset dialog box is displayed.



2. Enter the following:
   - Select the folder where the ruleset is located.
   - Enter your passphrase.
   - Select **Overwrite existing** to overwrite the rules that are already defined for the trunk, or **Append to existing** to append the imported rules to the existing rules.

   Click Import.

   The Import Ruleset dialog box closes. The ruleset is loaded and the rules are displayed in the Advanced Trunk Configuration window, in the URL Set tab.
3. In the URL Set tab, in the URL List, in the Name column, edit the rules you imported to include the application-type prefix. For details, refer to "URL Set Tab—URL List" on page 165.

Checking HTTP Headers

The IAG can check the headers of incoming HTTP requests, and reject unwanted headers, cookies, and parameters. You can configure the header checks to do the following:

- Enforce character ranges
- Check size of header names and values
- Enforce length, type, and values of specific headers

- Check size of cookies
- Enforce length, type, and values of specific cookies
- Enforce length, type, and values of cookies and headers that are not specifically configured

This section describes:

- "The Flow of HTTP Header Checks" on page 191
- "Blocking the "Negotiate" Authorization Header" on page 193
- "Configuring and Activating Additional Header Checks" on page 193

The Flow of HTTP Header Checks

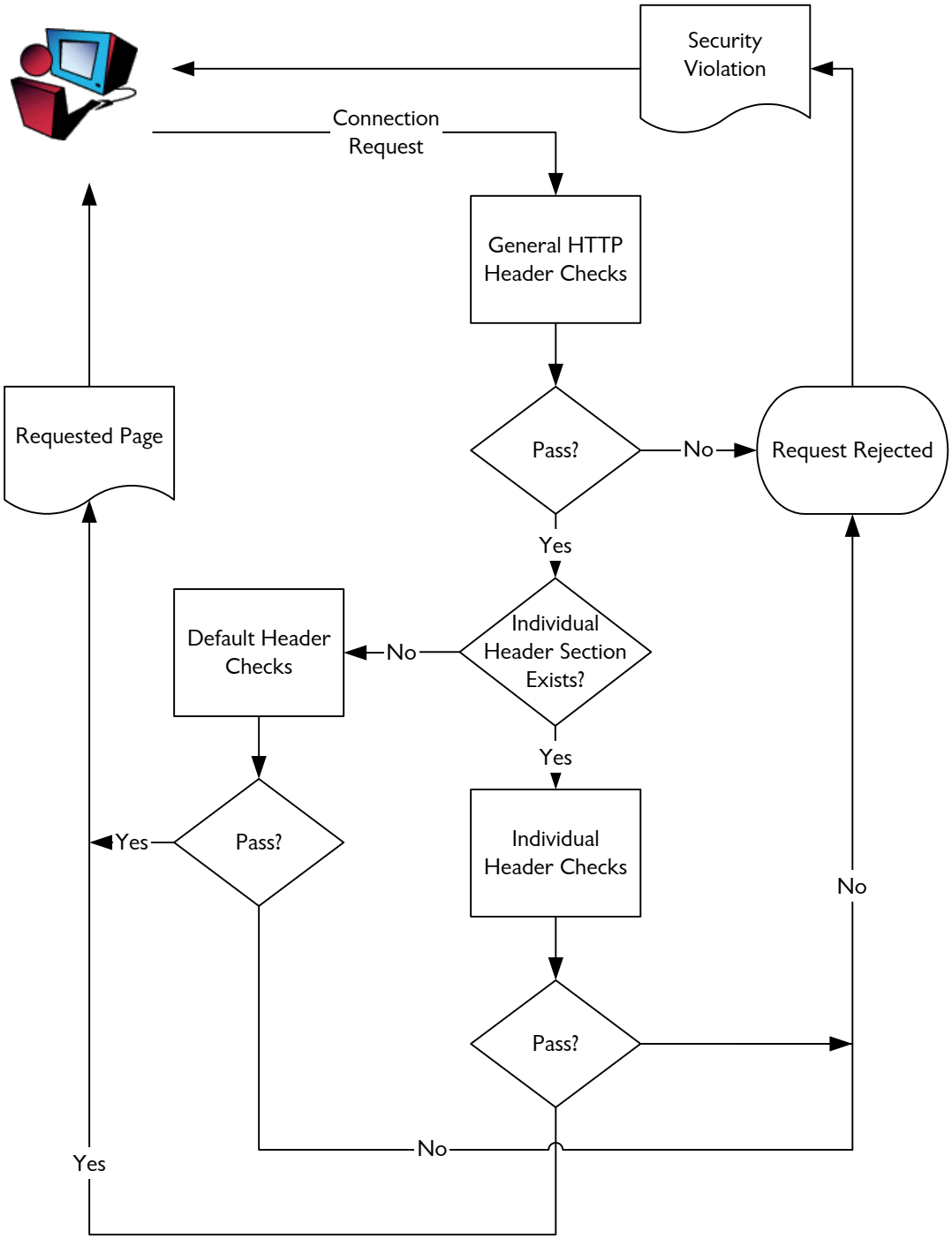
The header checks configuration file defines the checks that the headers and cookies in the HTTP request's header go through. All headers and cookies first go through general checks that are defined in the file in one, general section. Once a header or cookie passes the general checks, the IAG checks the header or cookie against the applicable header or cookie section:

- If an individual section is configured for the header or cookie in the file, the header or cookie is checked against that section.
- If no individual section is configured for the header or cookie, it is checked against the default header or cookie section, respectively.

If any of the headers or cookies in the HTTP header fail either the general or the individual checks, the request is rejected and the user receives a Security Violation message.

Figure 31 on page 192 illustrates the flow of checking one header; the flow is identical for cookie checks. "Configuring and Activating Additional Header Checks" on page 193 describes the steps you take in order to configure and activate the option for a trunk, including a detailed description of the configuration file and its various sections.

Figure 3I. HTTP Header Checks—Flow



Blocking the "Negotiate" Authorization Header

A `negotiate` authorization header sent by clients may contain malformed code, which could cause denial of service and browser crashes. This vulnerability was announced in Microsoft's Security Bulletin MS04-011, as ASN.1 "Double Free" Vulnerability - CAN-2004-0123.

The IAG can overcome this vulnerability by blocking all headers starting with:

```
authorization: negotiate
```

You configure this option in the URL Inspection tab of the Advanced Trunk Configuration window, in the "General Options" area:

Configuring and Activating Additional Header Checks

You can configure the HTTP header checks in the configuration file `HdrRuleSet.ini`.

To activate the Header Checks option for a trunk, take the following steps:

1. Copy the configuration file, `HdrRuleSet.ini`, from:

   `…\Whale-Com\e-Gap\Von\Samples\RuleSet`

   To the following custom folder; if it does not exist, create it:

   `…\Whale-Com\e-Gap\Von\Conf\WebSites\<Trunk_Name>\Conf\CustomUpdate`
2. Configure the option for the trunk, as described in "HTTP Header Checks Configuration File" on page 193.
3. Repeat the steps above for all the trunks for which you wish the implement the Header Checks option.
4. At the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

HTTP Header Checks Configuration File

The file `HdrRuleSet.ini` is divided into a number of sections, which define check parameters for both headers and cookies in the HTTP header. The number of sections in the file depends on the number of headers and cookies that are configured in the file, as described in the following sections.

General Parameters

The General section defines the parameters for the general checks, against which all headers and cookies are checked.

Table 27. HdrRuleSet.ini—General Parameters

Parameter	Description
ParseCookies	Yes=parse and check cookies. No=do not check cookies.
AllowedInName	Lists the characters that are allowed in the header or cookie name.
AllowedInValue	Lists the characters that are allowed in the values of the header or cookie.
MaxHeaderLen	Maximum number of characters allowed in a header name.
MaxValueLen	Maximum number of characters allowed in a header value. The length which is defined in the individual header sections cannot exceed the value defined here.
MaxCookieLen	Maximum number of characters allowed in a cookie name.
MaxCookieValue	Maximum number of characters allowed in a cookie value. The length which is defined in the individual cookie sections cannot exceed the value defined here.
NumberOfCookies	Maximum number of cookies allowed in the HTTP header.

A sample General section:

```
[General]
ParseCookies=Yes
AllowedInName=  "!#$%&'*+-.0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
                ^_`abcdefghijklmnopqrstuvwxyz~|"
AllowedInValue= "!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
                PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~"
MaxHeaderLen=20
MaxValueLen=200
MaxCookieLen=20
MaxCookieValue=200
NumberOfCookies=20
```

Header Parameters

Header parameters are defined in three types of sections:

- The Headers section defines the number and names of the headers that are subsequently defined in individual header sections. It is described in "Headers Section" on page 195.
- For each header that is defined in the Headers section, the `HdrRuleSet.ini` file holds an individual header section, where the section name is the header name as defined in the Headers section. The header is checked against the parameters in that individual section.
  For example: if in the Headers section Header1=Host, the first individual header section in the file is named Host, and it defines the parameters against which the Host header is checked.
- Headers that are not defined in the Headers section, and for which no individual section is configured, are checked against the default header section—Default:Header. Both the individual and the default sections are described in "Individual and Default Header Sections" on page 196.

Headers Section

Table 28 describes the parameters in the Headers section; be sure to read the note that follows before you configure this section. A sample section follows.

Table 28. HdrRuleSet.ini—Headers Section Parameters

Parameter	Description
Count	Number of headers for which individual header sections are configured in the file.
Header#	Header definitions: each line defines the sequential number and name of the header for which individual check parameters are defined in the file.

Important Note:

The IAG only checks headers that are defined in the Headers section against the individual header sections, first according to the number of headers defined in the Count parameter, then according to the header definitions.

- If the number of individual header sections that are defined in the file exceeds the number defined in the Headers section, either in the Count parameter or in the header definitions, the IAG ignores the excessive header sections and checks the headers against the default header parameters.
- If the number of headers defined in this section—either in the Count parameter or in the header definitions—exceeds the number of header sections, the IAG will not check any headers.

For example:

- If Count = 2 and Header1 and Header2 are defined in this section, only the headers that are configured in the first two individual header sections will be checked against their sections. Additional headers will be checked against the default section, even if an individual header section is configured in the file for them.
- If Count = 1 and Header1 and Header2 are defined in this section, only the header that is configured in the first individual header section will be checked the individual section.

If Header1 and Header2 are defined in the Headers section, but an individual header section is defined for Header1 only, no headers will be checked.

A sample Headers section:

```
[Headers]
Count=2
Header1=Host
Header2=User-Agent
```

Individual and Default Header Sections

Table 29 on page 197 lists the parameters against which headers are checked once they pass the checks defined in the General section; sample sections follow. All the parameters that are described in the table have to appear in each header section; they also appear in the Default section.

By default, the file `HdrRuleSet.ini` holds the following header sections:

- Host—name of the host requesting to connect.
- User-Agent—type of browser.
- Default:Header—default header section.

For each specific header section you add to the file you have to configure the Headers section, accordingly.

Table 29. HdrRuleSet.ini—Individual and Default Header Parameters

Parameter	Description
Length	Length of the header value, in characters. The value defined here cannot exceed the value defined in the General section.
Type	Type of header value: • Integer • Real • String
Value	Depends on the header value type, as defined in Type, above. • For strings, enter a regular expression that defines the acceptable values. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax". • For Integer and Real parameters, a comma divides between values; a colon represents a range of values.
AllowedAppearances	N/A

Sample header sections, including the Default header section:

```
[Host]
Length=200
Type=String
Value=([a-zA-Z0-9_-])+(\.[a-zA-Z0-9_-])+/)
AllowedAppearances=1

[User-Agent]
Length=100
Type=String
Value=.*
AllowedAppearances=1

[Default:Header]
Length=200
Type=String
Value=.*
AllowedAppearances=1
```

Cookie Parameters

Cookie parameters are only applicable if, in the General section, ParseCookies=Yes (as described in Table 27 on page 194). They are defined in three types of sections:

- The Cookies section defines the number and names of cookies that are subsequently defined in individual cookies sections. It is described in "Cookies Section", below
- For each cookie defined in the Cookies section, the `HdrRuleSet.ini` file holds a section, where the section name is the cookie name as defined in the Cookies section.
  For example: if in the Cookies section Cookie1=UID, the next section in the file is named UID, and it defines the parameters of the UID cookie.
- Cookies that are not defined in the Cookies section, and for which no individual section is configured, are checked against the default cookie section: Default:Cookie. Both the individual and the default sections are described in "Individual and Default Cookie Parameters" on page 199.

Cookies Section

Table 30 describes the parameters in the Cookies section; be sure to read the note that follows before you configure this section. A sample section follows.

Table 30. HdrRuleSet.ini—Cookies Section Parameters

Parameter	Description
Count	Number of cookies for which individual cookie sections are configured in the file.
Cookie#	Cookie definitions: each line defines the sequential number and name of the cookie for which individual check parameters are defined in the file.

Important Note:

The IAG only checks cookies that are defined in the Cookies section against the individual cookie sections, first according to the number of cookies defined in the Count parameter, then according to the cookie definitions.

- If the number of individual cookie sections that are defined in the file exceeds the number defined in the Cookies section, either in the Count parameter or in the cookie definitions, the IAG ignores the excessive cookie sections and checks the cookies against the default cookie parameters.
- If the number of cookies defined in this section—either in the Count parameter or in the cookie definitions—exceeds the number of cookie sections, the IAG will not check any cookies.

For Example:

- If Count = 2 and Cookie1 and Cookie2 are defined in this section, only the cookies that are configured in the first two individual cookie sections will be checked against their sections. Additional cookies will be checked against the default section, even if an individual cookie section is configured in the file for them.
- If Count = 1 and Cookie1 and Cookie2 are defined in this section, only the cookie that is configured in the first individual cookie section will be checked the individual section.
- If Cookie1 and Cookie2 are defined in the Cookies section, but an individual cookie section is defined for Cookie1 only, no cookies will be checked.

A sample Cookies section:

```
[Cookies]
Count=1
Cookie1=UID
```

Individual and Default Cookie Parameters

Table 31 lists the parameters against which cookies are checked once they pass the checks defined in the General section; sample sections follow. All the parameters that are described in the table have to appear in each cookie section; they also appear in the Default section.

By default, the file `HdrRuleSet.ini` holds the following cookie section:

- UID—configuration of the UID cookie.

For each specific cookie section you add to the file you have to configure the Cookies section, accordingly.

Table 3I. HdrRuleSet.ini—Individual and Default Cookie Parameters

Parameter	Description
Length	Length of the cookie value, in characters. The value defined here cannot exceed the value defined in the General section.
Type	Type of cookie value: • Integer • Real • String
Value	Depends on the cookie value type, as defined in Type, above. • For strings, enter a regular expression that defines the acceptable values. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax". • For Integer and Real parameters, a comma divides between values; a colon represents a range of values.
AllowedAppearances	N/A

Sample cookie sections, including the Default cookie section:

```
[UID]
Length=200
Type=String
Value=.*
AllowedAppearances=1

[Default:Cookie]
Length=200
Type=String
Value=.*
AllowedAppearances=1
```

Support of Non-Compliant HTTP Applications

Various Registry keys can be added to the HTTP filter in order to enable the IAG to communicate with applications that do not comply with the HTTP Request for Comments (RFC) regarding the following:

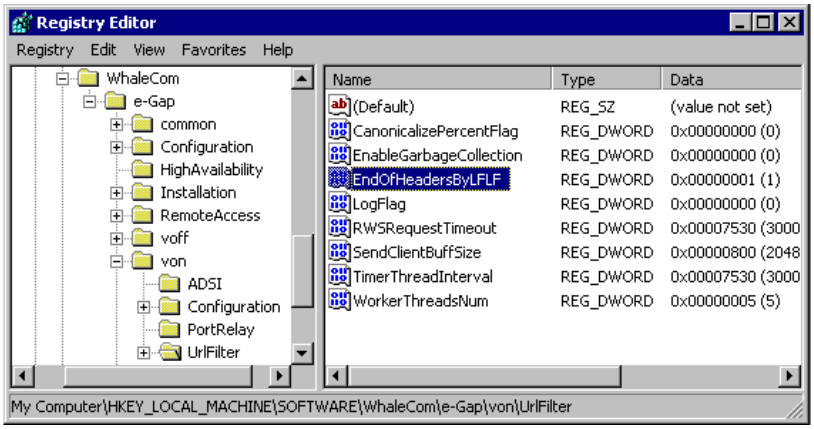
- Separation of header and data, described in “Irregular Separation of Header and Data” on page 201.
- Data chunking, described in “Irregular HTTP Data Chunking” on page 202.
- Header Content-Type, described in “Irregular HTTP Header Content-Type” on page 202.
- Enabling characters to appear after escaped characters in URL parameters, as described in “Allowing Characters After Escaped Characters in URL Parameters” on page 203.

Irregular Separation of Header and Data

According to the HTTP RFC, an HTTP header block has to be separated from the HTML data with the string: CRLF CRLF. If a non-compliant application uses the string LF LF instead, you can enable the IAG filters to communicate with it, as described in the following procedure.

To enable irregular separation of header and data:

1. At the IAG, access the following Registry key:
   `…\WhaleCom\e-Gap\Von\UrlFilter`
2. Create a new DWORD value named `EndOfHeadersByLFLF`, and set the value to 1.



3. At the Configuration program, click to activate the configuration, select the option “Apply changes made to external configuration settings”, and click Activate >.

   Once the configuration is activated, the filter can accept headers ending with LF LF.

Irregular HTTP Data Chunking

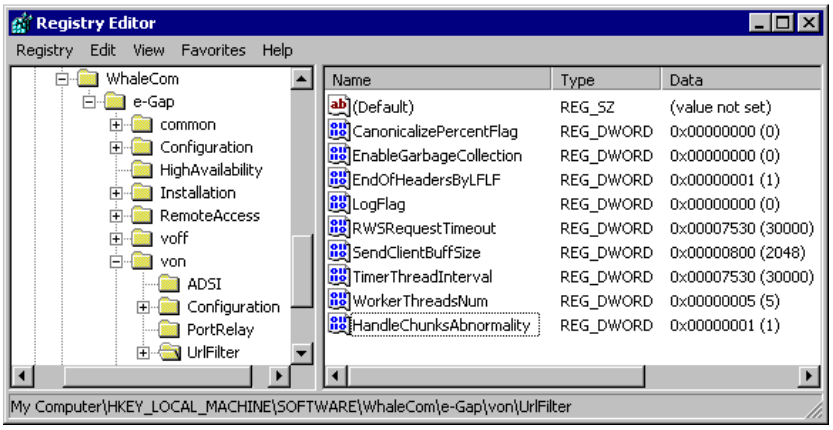
This section describes how you enable the HTTP filter to handle data chunking where four extra null bytes are added to the end of the chunk. Such abnormal header chunks are sent by some application servers, even though they do not comply with the HTTP RFC.

To enable irregular HTTP data chunking:

1. At the IAG, access the following Registry key:

   `…\WhaleCom\e-Gap\Von\UrlFilter`

2. Create a new DWORD value named `HandleChunksAbnormality`, and set the value to 1.



3. At Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

   Once the configuration is activated, the filter can handle the irregular data chunking described above.

Irregular HTTP Header Content-Type

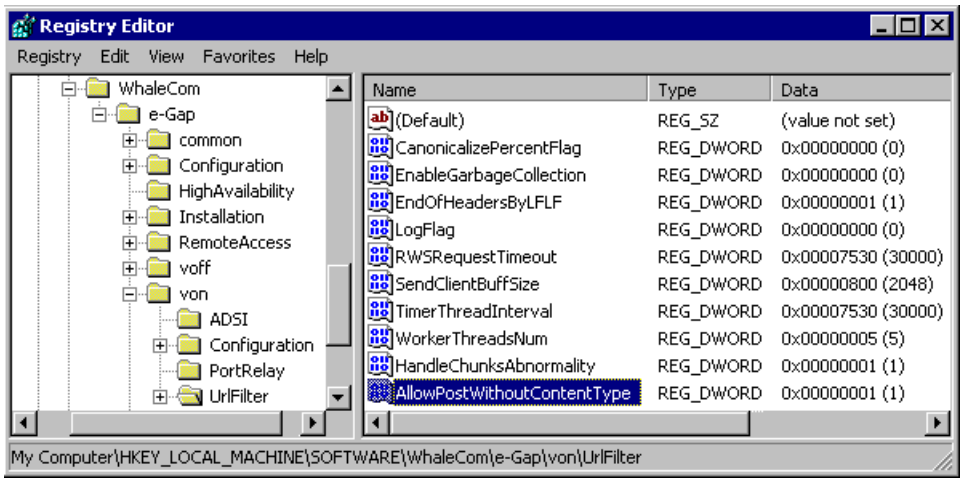
This section describes how you enable the HTTP filter to handle headers in which the "Content-Type" field is empty.

To enable filtering of a header without a content-type:

1. At the IAG, access the following Registry key:

   `…\WhaleCom\e-Gap\Von\UrlFilter`

2. Create a new DWORD value named `AllowPostWithoutContentType`, and set the value to 1.



3. At the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

   Once the configuration is activated, the filter can handle HTTP headers without a content-type, as described above.

Allowing Characters After Escaped Characters in URL Parameters

HTTP RFC 2396 allows the percent character "%" to appear only as part of an encoded character. For internal purposes, the IAG URL Inspection engine does allow the "%" character to appear as a regular character after escaped characters, in URL parameters only. You can use this capability to enable the URL Inspection engine to allow additional characters to appear after escaped characters in URL parameters, as described in this procedure.

To define which characters can appear after escaped characters:

1. At the IAG, access the following Registry key:

   `…\WhaleCom\e-Gap\Von\UrlFilter`

2. Create a new String value named `AllowedCharactersAfterEscapeChar`.

3. Modify the value data, entering the characters that will be allowed to appear after escaped characters in URL parameters, in the following format:

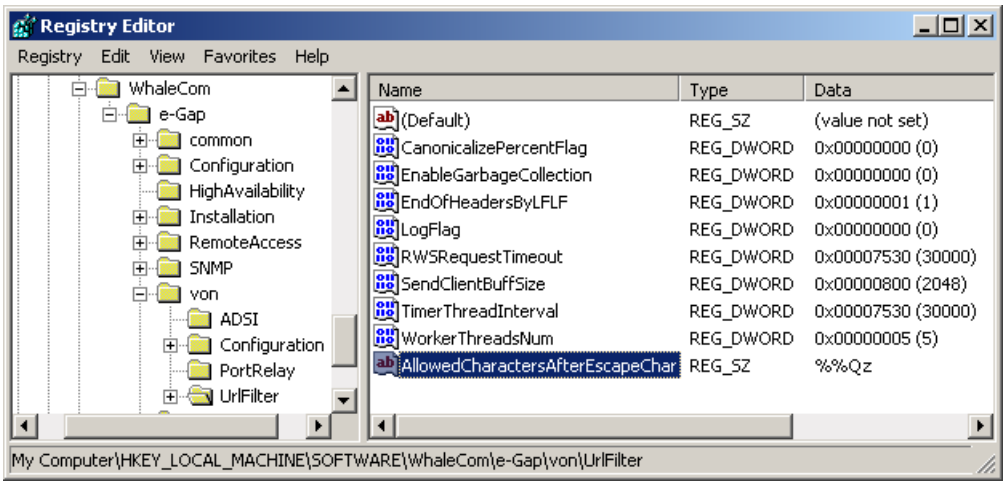
   `%<character><character>`

   Note the following:

   - Once you create this Registry key, you need to add the "%" character, which is otherwise allowed by default, to the list of allowed character, else it will be rejected by the filter.
   - You can add up to 256 characters to the list.
   - Values are case sensitive.

For example: to allow the characters "Q" and "z" to appear after escaped characters, in addition to the "%" character, enter the following:

```
%%Qz
```



4. At the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

   Once the configuration is activated, the URL Inspection engine will accept the characters you defined here when they appear after escaped characters in URL parameters.

Chapter 7

Application Customizers

Application customizers enable you to add functionality such as authentication, authorization, session timeout, or secured logoff to any web application, or change the appearance of the application by adding graphics, messages, buttons or banners to the interface. The customizers enable the Intelligent Application Gateway (IAG) filter to intercept data that is transmitted over the IAG, and manipulate both requests, on their way to the application server, and responses, before they are returned to the browser. You can set the filter to manipulate the data of the original content, or "wrap" any third-party web application in a frame.

Tip

For a description of when it is advisable to wrap the application in a frame, and when you are better off manipulating the data inside the application, refer to "Using Frames" on page 206.

The IAG automatically customizes the trunk by applying a default template to the trunk. This is implemented in different ways for the different trunk-types, as follows:

- Portal trunks—the default portal homepage supplied with the IAG is customized.
- Webmail trunks—the Webmail application users access through the trunk is customized.
- Basic trunks—the application users access through the trunk is customized.

You can further customize the default template, or select to use a custom template, as described in this chapter.

In addition to the application customizers, you can configure the following options in the application customization interface:

- Configure the types of content on which data is searched and replaced by the application customization, HAT, and server name translation processes. For details, refer to "Global Content-Type List" on page 210.
- Configure the handling of compresses data in responses, as described in "Handling Compression in Responses" on page 212.

Using Frames

Application customizers enable you to “wrap” the application in a frame. That is, you can add functionality to the application in an environment which you create and add to the original application, while the original application data stays unchanged. This is especially advisable in applications where users, while using the application, go through a large number of URLs, each requiring identical manipulation in order to add the required functionality. Under these conditions, if you manipulate the application data, you have to configure a large number of “data change” elements, one for each URL. Those will be difficult to maintain, and might affect system performance. However, we recommend you do not use frames in the following cases:

- If only a limited number of URLs in the application require identical manipulation.
- If the application breaks out of frames.
- If you wish the application to occupy the entire screen space.

Note

Frames must be used in conjunction with authentication. When you configure a trunk for an application that utilized frames, make sure the authentication option is enabled for the trunk.

Tip

- You configure frames using <FRAME_WRAP>, described on page 230.
- Data-change elements are described in <DATA_CHANGE> on page 226.

Sample Implementations

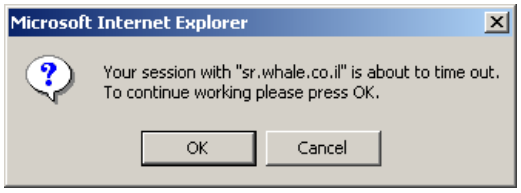
This section describes two of the application customizers implementations, which demonstrate how application customizers enhance the functionality of the IAG:

- “Session Timeout Warning” on page 207
- “Replacement of the Server Value in Response Headers” on page 207

Session Timeout Warning

The application customizers enable the IAG to warn users in advance when their session with the application is about to end, according to the Inactive Session Timeout configured for the trunk:

Figure 32. Session Timeout Popup



Users can click OK to renew the session and continue working without having to log into the application again.

Replacement of the Server Value in Response Headers

When a server sends a response to the client, one of the response headers it sends is a "Server" header. This header contains information about the software used by the server to handle the request, and can contain multiple product tokens and comments identifying the server and any significant subproducts.

For example:

An Apache™ server returns the following Server response header:

```
Server: Apache/2.0.47.(Unix).mod_ssl/2.0.47.OpenSSL/0.9.7b.DAV/2
```

In order not the expose information regarding the application servers, the application customizers replace the value of the Server header with the value that represents the IAG (Windows 2003 Server):

```
Server: Microsoft-IIS/6.0
```

This replacement ensures that, to the outside world, the application server always appears to be the IAG, while the real application server remains hidden.

Customizing Your Web Applications

You can customize your applications as follows:

- In the Advanced Trunk Configuration window, in the Application Customization tab, you can select whether to use the default IAG template, or a customized template. This is also where you activate the option for Basic trunks, where application customization is not applied by default. The options available in this tab are described in "Application Customization Tab" on page 208.

- Optionally, edit the application customization templates, described in "Advanced Configuration of the Templates" on page 215.

Note

The application customization templates are XML files. In order to edit them, you need to have a working knowledge of XML technology.

Application Customization Tab

The Application Customization tab is part of the Advanced Trunk Configuration window, which you access from the main window of the Configuration program. You can use it to:

- Select whether to use the default application customization template that is supplied with the IAG, or use a custom template.
- Enable and configure the application customization option in Basic trunks, where application customization is not applied by default.
- Configure the global Content-Type and URL Extension lists.

Figure 33. Application Customization Tab

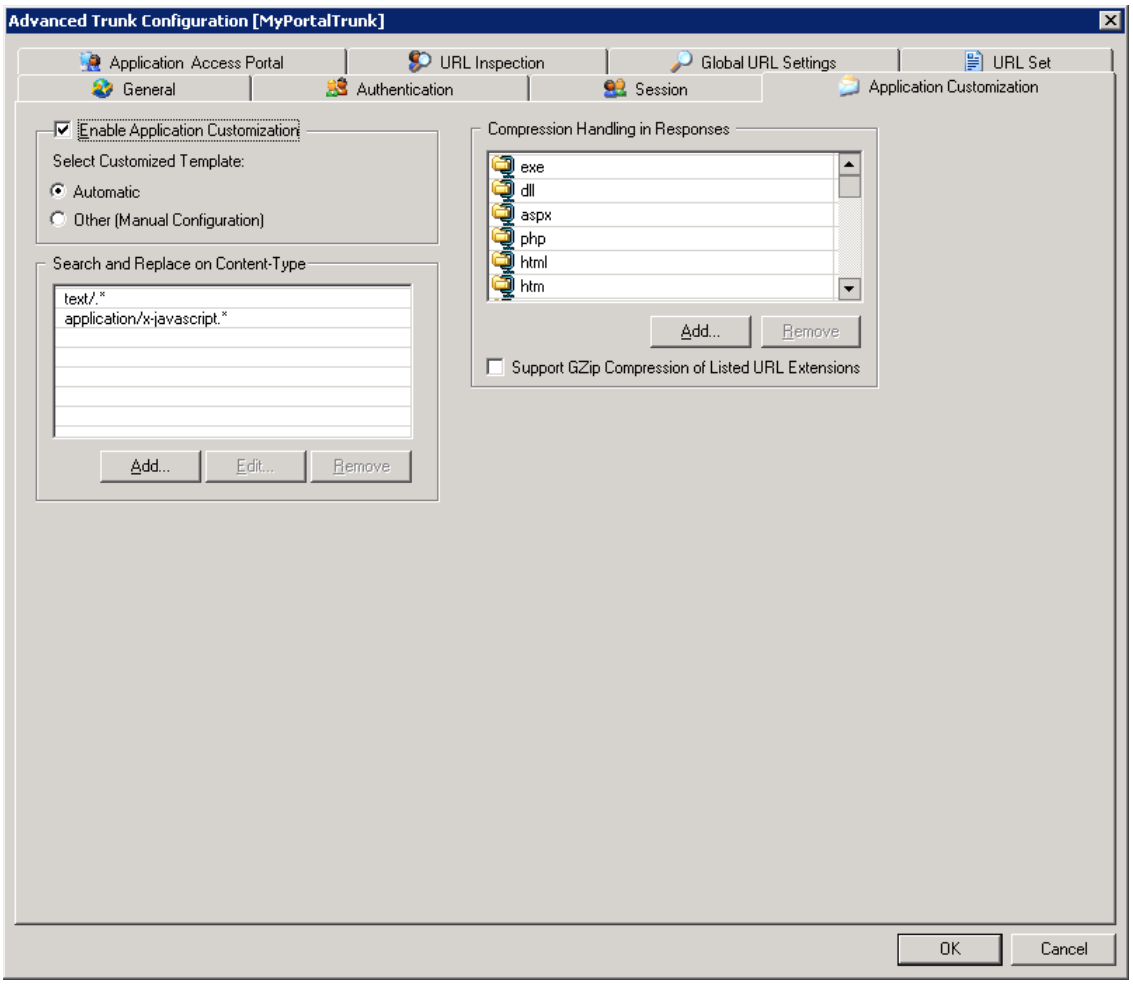


Table 32. Application Customization Tab Parameters

Parameter	Description
Enable Application Customization	Enables the application customization option, and applies the settings configured in the application customization templates, according to your selection in "Select Customized Template", below.

Table 32. Application Customization Tab Parameters (Cont'd)

Parameter	Description
Select Customized Template	Select application customization template: • Automatic • Other (Manual Configuration) For a description of the application customization templates, refer to "Advanced Configuration of the Templates" on page 215.
Search and Replace on Content-Type	Defines the global list of content-types on which the filter search and replace data. For details, refer to "Global Content-Type List" on page 210.
Compression Handling in Responses	Defines how to handle compression in responses, as described in "Handling Compression in Responses" on page 212.

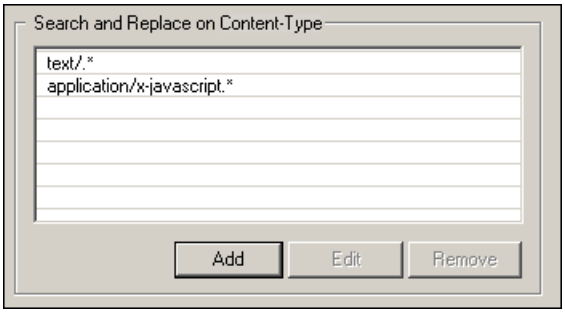
Global Content-Type List

The content-type list defines the types of content on which data is manipulated by the application customization, HAT, and server name translation processes. Content-type is defined in the value of the HTTP header "Content-Type".

By default, the list includes the following content-types:

- `text/.*`
  Any type of text, for example:
  `text/html`, `text/css`, `text/plain`.
- `application/x-javascript.*`
  Text within JavaScript™.

If any of your applications require the manipulation of other content-types, you can add the required content-type to the list; you can also remove existing content-types from the list. You configure the list in the Application Customization tab, in the "Search and Replace on Content-Type" area:



Note

For content that requires manipulation by the HAT, whenever you configure an additional content-type in the global list, you must also configure the same content-type in the AAP configuration file, in <PARSERS_CONFIGURATION>. If you fail to configure the new content-type in the AAP configuration file, it will not be manipulated by the HAT even if you do configure it in the global list.

For a description of the configuration of content-types in the AAP configuration file, refer to "Adding Content-Types to the Content Parsers" on page 343.

To configure the list of manipulated content-types:

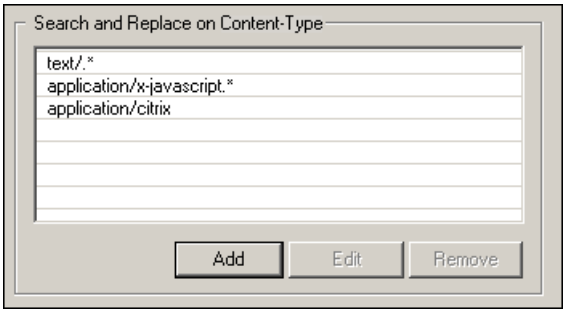
1. In the Application Customization tab, in the "Search and Replace on Content-Type" area, click Add...

   The Add Content-Type dialog box is displayed.

2. Enter the content-type you wish to add to the list, using regular expressions. For a description of regular expressions, refer to Appendix B: **"Regex++, Regular Expression Syntax"**.

   Click OK.

   In the "Search and Replace on Content-Type" area, the content-type you entered is added to the list:



3. Repeat steps 1–2 to add all the content-types that require manipulation to the list.

   The application customization, HAT, and server name translation processes will search and replace data in all the content-types listed here.

Handling Compression in Responses

When the browser requests a URL, it can declare that it is willing to accept the content of the URL in a compressed form. This is specified in the HTTP header `accept-encoding`, where the header value defines the encoding type.

For example: `accept-encoding: gzip,deflate`

When the server receives this header, it can send the requested content in an encoded form, according to the encoding type or types specified by the client.

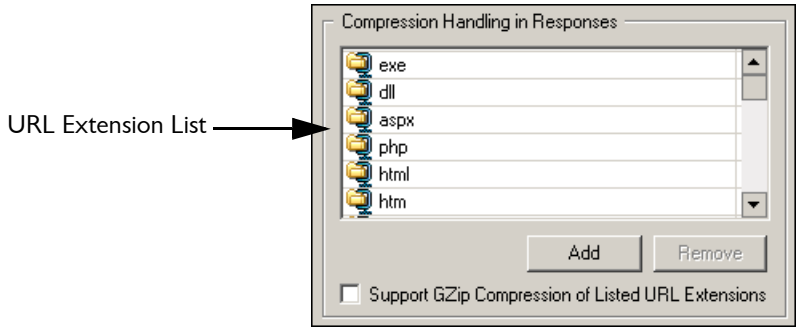
In order to be able to apply the application customization, HAT, and server name translation capabilities, the IAG filter might need to manipulate the content that the server sends to the browser.

The option “Compression Handling in Responses”, which is defined in the Application Customization tab, reduces latency in delivery time, without losing those capabilities, by defining URL extensions for the types of content that:

- The application servers might send in a compressed form.
- Might contain textual data that needs to be manipulated by the IAG.

The filter will pass all other types of content that it receives from the application servers, and which do not require manipulation, to the client without intervention.

Figure 34. Compression Handling in Responses (Application Customization Tab)



You can add URL extensions to the default extensions listed in this area, as described in "Configuring the URL Extension List" on page 213. For a description of the option "Support GZip Compression of Listed URL Extensions", refer to "GZip Compression" on page 214.

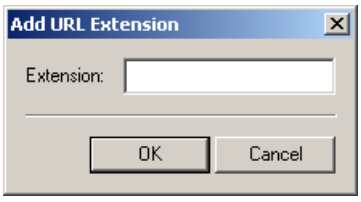
Configuring the URL Extension List

This procedure describes how you add extensions to the default list of URL extensions where compression handling in responses is applied.

To configure the list of URL extensions:

1. In the Application Customization tab, in the "Compression Handling in Responses" area, click Add...

   The Add URL Extension dialog box is displayed.



2. Enter the URL extension you wish to add to the list and click OK.

   In the Application Customization tab, the extension you entered is added to the URL Extension list, in the "Compression Handling in Responses" area.

3. If required, check the option "Support GZip Compression of Listed URL Extensions". For details, refer to "GZip Compression" on page 214. This option applies to all the extensions in the list.

GZip Compression

For the extension types listed in the URL Extension list, you can do one of the following:

- Activate the option “Support GZip Compression of Listed URL Extensions”:

  ☑ Support GZip Compression of Listed URL Extensions

  In this setup, when the filter receives a request for content that is listed here, when it passes the request to the application server, it does one of the following:

  - If the requesting browser supports GZip encoding, the filter informs the application server that the browser supports this type of encoding. The server can then send the content GZip-encoded, in which case the filter will decompress it, manipulate the links as required, compress it again, and send it to the requesting browser.
    Note that even if the browser supports additional encoding forms, the filter informs the application server that the browser supports only GZip encoding.
  - If the requesting browser does not support GZip encoding, the filter informs the application server that the browser does not support encoding, even if the browser supports other types of encoding. In this case, the server sends unencoded content to the filter. The filter then manipulates the links as required, and sends the content to the requesting browser in an unencoded form.

- If you do not activate the option “Support GZip Compression”, when the filter receives a request for content that is listed in the URL Extension list, when it passes the request to the application server, it informs the server that the browser does not support encoding. The server then sends unencoded content to the filter. The filter manipulates the links as required, and sends the content to the requesting browser in an unencoded form.

Advanced Configuration of the Templates

The IAG supplies you with pre-defined application customization templates. If required, you can add functionality to your system by configuring additional options, removing options that are not applicable for your environment, or changing the configuration of existing options.

The following sections describe the functionality of the default application customization templates, and how you create a custom configuration file where you can modify and enhance the default settings. Before you create a custom configuration file, be sure to read the section "Before You Begin" on page 219; it holds important information you should note.

Functionality of the Templates

This section describes the functionality of each of the Application Customization templates supplied with the IAG.

Note

This section refers to HTTPS templates. Unless otherwise noted, the functionality of the HTTP templates is identical to that of the HTTPS templates.

Microsoft Outlook Web Access 5.5

`HTTPS_WhlFiltAppWrap_ForOWA55.xml`

- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Add client-side Upload URL blocking
- Add the header `Server: Microsoft-IIS/6.0`

Microsoft Outlook Web Access 2000 SP2/SP3

`HTTPS_WhlFiltAppWrap_ForOWA2000SP2.xml`

- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Change Logoff Button URL to Logoff URL defined in Authentication tab
- Add the header `Front-End-Https: On` (HTTPS template only)
- Add the header `Server: Microsoft-IIS/6.0`

Microsoft Outlook Web Access 2000 POST SP3

`HTTPS_WhlFiltAppWrap_ForOWA2000POSTSP3.xml`

- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Change Logoff Button URL to Logoff URL defined in Authentication tab
- Add the header `Front-End-Https: On` (HTTPS template only)
- Add the header `Server: Microsoft-IIS/6.0`

Microsoft Outlook Web Access 2003

`HTTPS_WhlFiltAppWrap_ForOWA2003.xml`

- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Change Logoff Button URL to Logoff URL defined in Authentication tab
- Add the header `Front-End-Https: On` (HTTPS template only)
- Add the header `Server: Microsoft-IIS/6.0`

Microsoft Outlook Web Access 2003 SP1

`HTTPS_WhlFiltAppWrap_ForOWA2003SP1.xml`

- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Change Logoff Button URL to Logoff URL defined in Authentication tab
- Add the header `Front-End-Https: On` (HTTPS template only)
- Add the header `Server: Microsoft-IIS/6.0`

Domino Webmail 4.6 Interface

`HTTPS_WhlFiltAppWrap_ForDominoWebMail46.xml`

- Wrap the application in a frame and add a frame with a Logoff button above the application frame
- Add the header `Server: Microsoft-IIS/6.0`

Domino Webmail 5.0/6.0 Interface

`HTTPS_WhlFiltAppWrap_ForDominoWebMail.xml`

- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff

- Plant a Logoff button in the Webmail 5.0x template with the following updates:
  - Button is located at top of navigator, on left hand side
  - The word Logoff is written in upper case, to make it more visible
- Fix Logout option from upper menu in Webmail 6.0
- Add the header `Server: Microsoft-IIS/6.0`

Domino iNotes (Single Server)

`HTTPS_WhlFiltAppWrap_ForDolsInotesSingle.xml`

- Eliminate the popup message "Do you want to display nonsecure items?"
- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Add and install an SSL VPN object for iNotes 5.0 and iNotes 6.0 & 6.02
- DOLS 5.x, 6, 6.02: change the installation URL
- Add the header `Server: Microsoft-IIS/6.0`

Domino iNotes (Multi Servers)

`HTTPS_WhlFiltAppWrap_ForDolsInotesCluster.xml`

- Eliminate the popup message "Do you want to display nonsecure items"
- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Add and install an SSL VPN object for iNotes 5.0, 6.0, 6.02, and 6.5
- DOLS 5.x, 6, 6.02: change the installation URL
- Add the header `Server: Microsoft-IIS/6.0`

All Domino (Webmail 5.x/6.x/7.x and iNotes) Interfaces

`HTTPS_WhlFiltAppWrap_ForAllDomino.xml`

- Eliminate the popup message "Do you want to display nonsecure items"
- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Plant a Logoff button in the Webmail 5.0x template with the following updates:
  - Button is located at top of navigator, on left hand side

    - The word Logoff is written in upper case, to make it more visible
- Enable attachment blocking (forward email attachments or reply with an attachment) in Domino iNotes version 6.5 and higher
- Fix Logout option from upper menu in Webmail 6.0
- Add the header `Server: Microsoft-IIS/6.0`

Portal Template

`HTTPS_WhlFiltAppWrap_ForPortal.xml`

- Change any instance of `logout&redirectto` to the Logoff URL defined in the Authentication tab
- In Domino iNotes 5.x, 6.0, 6.02, 7.0 and in OWA 2000 and OWA 2003: change the Logoff button URL to the Logoff URL defined in the Authentication tab, and prevent the Logoff from deleting the IAG session cookie
- DOLS 5.x, 6, 6.02, 7: change URL for installation
- Prevent a situation where a logoff from Citrix® and OWA 5.5 kills the upper frame
- Activate the Attachment Wiper on logout
- Embed JavaScript calls for popup warnings for Session Timeout and Scheduled Logoff
- Add the header `Server: Microsoft-IIS/6.0`

"OtherWeb" Template

Note

This template is used for generic web applications, in both Portal and Basic trunks.

`HTTPS_WhlFiltAppWrap_ForOtherWeb.xml`

- Sample Data Change element (described in <DATA_CHANGE> on page 226)
- Sample URL Change element (described in <URL_CHANGE> on page 251)
- Add the header `Server: Microsoft-IIS/6.0`

Before You Begin

The templates that control the application customization are located in this folder:

…\Whale-Com\e-Gap\Von\Conf\WizardDefaults\AppWrapTemplates

> **Tip**
> For a detailed description of the templates, refer to "Functionality of the Templates" on page 215.

When a new trunk is created, the relevant template is copied to the following location:

…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf

The file is renamed as follows:

- In HTTP trunks:

  WhlFiltAppWrap_HTTP.xml

- In HTTPS trunks:

  WhlFiltAppWrap_HTTPS.xml

In order to change the default application customization definitions, create a custom file, as described in "Creating a Custom Application Customization File" on page 220, and implement the changes in the custom file.

> **Caution**
> Do not make any changes to the templates that are supplied with the system. Implement your changes only in the custom file.

Conventions Used by the Application Customization Templates

The application customization templates follow these conventions:

- Tag names are uppercase:

  <TAG>...**</TAG>**

- Attribute names are lowercase:

  <TAG **attribute**=value>...</TAG>

- Attribute values are case insensitive:

  <TAG attribute=**value**>...</TAG>

  is identical to:

  <TAG attribute=**VALUE**>...</TAG>

- Some XML elements take regular expressions in the data. When regular expressions are used, this is indicated in the description of the element usage. When using regular expressions, note the following:
  - Use the syntax described in Appendix B: **"Regex++, Regular Expression Syntax"**
  - In order to use a character as is, make sure it is preceded by a slash. For example, to use the asterisk (`*`) wildcard in data that takes regular expressions, enter `\*`
- The supported encoding is **Base64**. In order to edit encoded text, use the Editor to convert from and to Base64, as described in "Editor" on page 40.

Creating a Custom Application Customization File

This section describes how you create a custom application customization file. The changes you make in the custom file are merged with the definitions of the template that is supplied with the system.

Caution

Do not make any changes to the templates that are supplied with the system. Implement your changes only in a custom file, as described in this section.

To create a custom application customization file:

1. Access the following location on the IAG:

   `…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf`
2. Under the `Conf` folder, create the following subfolder:

   `CustomUpdate`. If such a folder already exists, use the existing folder.
3. From the `Conf` folder, copy the following file to the `CustomUpdate` subfolder:

   `WhlFiltAppWrap_HTTP.xml` or `WhlFiltAppWrap_HTTPS.xml`, depending on the trunk-type.
4. For Basic trunks, enable and configure the application customization option, as described in "Application Customization Tab" on page 208.
5. At the Configuration program, click to activate the configuration. Select the option "Apply changes made to external configuration settings", and click Activate >.

Syntax of the Application Customization Template

An application customization template contains the following elements:

- The first element nested under the root element must be `<MANIPULATION>`
- The child elements that are nested under `<MANIPULATION>` manipulate the application data. You can use any of the elements described in the following sections, and include as many child elements as you wish under `<MANIPULATION>`.

Application Customization Template—Sample Code

```
<APP_WRAP ver="3.0" id="AppWrapSample#1.xml">

 <MANIPULATION>

   <URL_CHANGE>
    <FROM_URL case_sensitive="false">/webload/
    test_[0-9.]*k.html</FROM_URL>
    <TO_URL>/webload/test_1k.html</TO_URL>
   </URL_CHANGE>

   <DATA_CHANGE>
    <URL case_sensitive="false">/webload/test_1k.html</URL>
    <SAR>
     <SEARCH>start</SEARCH>
     <REPLACE>Why</REPLACE>
    </SAR>
   </DATA_CHANGE>

   <FRAME_WRAP>
    <PARENT encoding="base64">PEhUTUw+</PARENT>
    <MY_FRAME encoding="base64">PEhUTUw+0KPC9IVE1MPg==</MY_FRAME>
   </FRAME_WRAP>

   <ADD_HEADER add_to="request" filter_to_rws="http">
    <NAME>Front-End-Https:</NAME>
    <VALUE>On</VALUE>
   </ADD_HEADER>
```

Application Customization Template—Sample Code (Cont'd)

```
    <HEADER_CHANGE>
      <RESPONSE>
       <APPLICATION>
         <SERVER_NAME mask=""192\.168\.1\.1</SERVER_NAME>
         <PORT>80</PORT>
         <URL>
           <URL_NAME>/test\.html</URL_NAME>
           <EDIT>
            <HEADER>
              <NAME>Set-Cookie:AppCookie</NAME>
              <SAR>
                <SEARCH encoding=""">125</SEARCH>
                <REPLACE encoding=""">126</REPLACE>
              </SAR>
            </HEADER>
           </EDIT>
         </URL>
       </APPLICATION>
      </RESPONSE>
    </HEADER_CHANGE>

  </MANIPULATION>

</APP_WRAP>
```

Manipulation Elements

The following elements can be nested under <MANIPULATION>:

- <ADD_HEADER>, described on page 223.
- <DATA_CHANGE>, described on page 226.
- <FRAME_WRAP>, described on page 230.
- <HEADER_CHANGE>, described on page 234.
- <URL_CHANGE>, described on page 251.

The following sections describe the elements and their usage, their child elements, and the relationships between them.

<ADD_HEADER>

> **Note**
> The customization you configure in the <ADD_HEADER> element applies to all the applications that you enable through the trunk. In order to add headers to specific servers or applications, use the <HEADER_CHANGE> element, described on page 234.

Description

Adds a header to either requests or responses.

> **Note**
> Whenever the conditions set in the <ADD_HEADER> attributes are met, the header is added.

Usage

- <ADD_HEADER> is optional.
- An unlimited number of <ADD_HEADER> elements can be nested under <MANIPULATION>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
add_to	request, response	Mandatory
filter_to_rws	http, https	Optional. If this attribute is used, the header is added to the request only if the IAG filter and the application server communicate using the protocol defined in the attribute value.

Child Elements

<ADD_HEADER> must contain one of each of the following elements:

- <NAME>, described on page 224.
- <VALUE>, described on page 224.

<NAME>

Description

Child element of <ADD_HEADER>. Defines the header name.

Note

Header names must always end with a colon (:). The colon is part of the header.

Usage

- One <NAME> element must be nested under each <ADD_HEADER> element.
- Only one <NAME> element can be nested under each <ADD_HEADER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<VALUE>

Description

Child element of <ADD_HEADER>. Value of the header defined in <NAME>.

Usage

- One <VALUE> element must be nested under each <ADD_HEADER> element.
- Only one <VALUE> element can be nested under each <ADD_HEADER> element.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <ADD_HEADER>

```
// Adds the header "Front-End-Https: On" to each request from the
// filter to the RWS (Depending the communication from the filter
// to the RWS is HTTP).
<ADD_HEADER add_to="request" filter_to_rws="http">
     <NAME>Front-End-Https:</NAME>
     <VALUE>On</VALUE>
</ADD_HEADER>

// Adds the header "Cache-Control: no-cache" to each response from
// the filter to the browser.
<ADD_HEADER add_to="response">
     <NAME>Cache-Control:</NAME>
     <VALUE>no-cache</VALUE>
</ADD_HEADER>
```

Tip

The header “Cache-Control: no-cache” is used in this example to request the browser to avoid browser-side caching, instead of the header “vary: *”, which is used when you activate the option “Avoid Browser-Side Caching” in the Session tab. Other headers you can use for the same purpose are:

- “pragma: no-cache”
- “Expires: -1”

Before you use any of those headers, verify that they are suitable for your application.

<DATA_CHANGE>

Description

Searches and replaces data string or strings in the data received from the application server before data is sent to the browser.

> **Tip**
> If you have to configure a large number of identical <DATA_CHANGE> elements for your application, you might want to consider the use of frames, instead. For more details refer to "Using Frames" on page 206.

Usage

- <DATA_CHANGE> is optional.
- An unlimited number of <DATA_CHANGE> elements can be nested under <MANIPULATION>.

Attributes & Attribute Values

None.

Child Elements

<DATA_CHANGE> must contain one of each of the following elements:

- <URL>, described on page 226.
- <SAR>, described on page 227.

<DATA_CHANGE> ➤ <URL>

<URL>

Description

Child element of <DATA_CHANGE>. Defines the URL in which data is searched and replaced.

Usage

- One <URL> element must be nested under each <DATA_CHANGE> element.
- Only one <URL> element can be nested under each <DATA_CHANGE> element.
- Takes regular expressions.

Attributes & Attribute Values

Attribute	Values	Default	Type/Comments
case_sensitive	true, false	false	Optional

Child Elements

None.

<DATA_CHANGE> ➤ <SAR>

<SAR>

Description

Child element of <DATA_CHANGE>. Defines a string to search and replace.

Usage

- One <SAR> element must be nested under each <DATA_CHANGE> element.
- An unlimited number of <SAR> elements can be nested under each <DATA_CHANGE> element.

Attributes & Attribute Values

None.

Child Elements

<SAR> must contain one of each of the following elements:

- <SEARCH>, described on page 228.
- <REPLACE>, described on page 228.

Note

<SEARCH> and <REPLACE> elements always come in pairs—for every <SEARCH> element there must be a <REPLACE> element.

<SEARCH>

Description

Child element of <SAR>. Defines the string that <SAR> searches.

Usage

- One <SEARCH> element must be nested under each <SAR> element.
- Only one <SEARCH> element can be nested under each <SAR> element.
- Two <SEARCH> elements (nested under different <SAR> elements), where one is a sub-string of the other, would cause an error.
- The search is case insensitive.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional

Child Elements

None.

<REPLACE>

Description

Child element of <SAR>. Defines the string with which <SAR> replaces the string defined in <SEARCH>.

Usage

- One <REPLACE> element must be nested under each <SAR> element.
- Only one <REPLACE> element can be nested under each <SAR> element.
- Case is maintained.
- The following variables can be used as placeholders in the data of <REPLACE>; note that variable names are case sensitive:

- `WhlSessionTimeout`: the maximum time a session can be inactive before it expires. Configured in the Session tab of the Advanced Trunk Configuration window, under "Inactive Session Timeout" (described in "Default and Privileged Session Settings" on page 137).
- `WhlLogoffURL`: URL to be recognized by the filter as the logoff URL. Configured in the Authentication tab of the Advanced Trunk Configuration window, under "Logoff URL" (described in "Configuration in the Authentication Tab" on page 82).

> **Tip**
> You can see an example of the use of the `WhlLogoffURL` variable in a <FRAME_WRAP> element in "Sample Code: <FRAME_WRAP>" on page 233.

- `WhlScheduledLogoffTimer`: after the time defined in this variable elapses, the user is required to re-authenticate. Configured in the Session tab of the Advanced Trunk Configuration window, under "Automatic Scheduled Logoff After …. Minutes" (described in "Default and Privileged Session Settings" on page 137).
- `WhlOwnURL`: URL where the graphic files used by the internal website are located: /`InternalSite/`

> **Tip**
> You can see an example of the use of the `WhlOwnURL` variable in <REPLACE> in "Sample Code: <DATA_CHANGE>" on page 230.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional

Child Elements

None.

Sample Code: <DATA_CHANGE>

```
// When a request to the URL "/webload/test_1k.html" is identified,
// the response received from the application server would be
// manipulated prior to sending it to the browser. In the first SAR
// sample, every occurrence of the word "start" (case insensitive)
// will be replaced with the word "Stop". The second SAR is a bit
// more complex. The replace entry contains the reserved
// placeholder WhlOwnURL. This placeholder will be replaced by
// its actual value. Since the value of WhlOwnURL is
// "/InternalSite/", SAR will search for "ImageFromOurServer.jpg"
// and replace it with the following:
// "/InternalSite/ImageFromOurServer.jpg"
<DATA_CHANGE>
     <URL case_sensitive="false">/webload/test_1k\.html</URL>
     <SAR>
          <SEARCH>start</SEARCH>
          <REPLACE>Stop</REPLACE>
     </SAR>
     <SAR>
          <SEARCH>src=ImageFromOurServer.jpg</SEARCH>
          <REPLACE>src=WhlOwnURLImageFromOurServer.jpg</REPLACE>
     </SAR>
</DATA_CHANGE>
```

<FRAME_WRAP>

Note

Frames must be used in conjunction with authentication. When you configure a trunk for an application that utilized frames, make sure the authentication option is enabled.

Description

Defines a frame that wraps the application, including the frame's size, position, and attributes, and the features that the frame adds to the application.

Tip

For a description of when it is advisable to wrap the application inside a frame refer to "Using Frames" on page 206.

Usage

- <FRAME_WRAP> is optional.
- Only one <FRAME_WRAP> element can be nested under <MANIPULATION>.

Attributes & Attribute Values

None.

Child Elements

<FRAME_WRAP> must contain one of each of the following elements:

- <PARENT>, described on page 231.
- <MY_FRAME>, described on page 232.

<FRAME_WRAP> ➤ <PARENT>

<PARENT>

Description

Child element of <FRAME_WRAP>. Defines the position, size, and attributes of the frame that wraps the application.

Usage

- One <PARENT> element must be nested under <FRAME_WRAP>.
- Only one <PARENT> element can be nested under <FRAME_WRAP>.
- The frame is a top frame; it must not be wrapped by an external frame.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. **Note:** We recommend that you encode the data under <PARENT>. For details, see "Content", below.

Content

<PARENT> contains an HTML file that describes the structure of the frame. The file is encoded, using Base64 encoding, so that XML syntax does not collide with the HTML content.

The HTML content **must** contain a <FRAMESET> element, which defines the original URL requested by the user and the frame wrapping of that URL, using <FRAME> child elements, as follows:

- `<FRAME src="/WhlFrame.html">`

  The frame that will be used, either the default frame supplied with the IAG, or a user-defined frame.

  Definition of the frame is described in <MY_FRAME>, on page 232

- `<FRAME src="/WhlOrigURL.html">`

  The URL originally requested by the user.

Caution

- <FRAMESET> must contain the two <FRAME> elements described above.
- **Do not** change the value of the attributes of these elements. Note that the values are case-sensitive.

[<FRAME_WRAP>] ➤ [<MY_FRAME>]

<MY_FRAME>

Description

Child element of <FRAME_WRAP>. Defines the features that the frame adds to the application.

Usage

- One <MY_FRAME> element must be nested under <FRAME_WRAP>.
- Only one <MY_FRAME> element can be nested under <FRAME_WRAP>.
- The following variables can be used as placeholders in the data of <MY_FRAME>; note that variable names are case sensitive:
  - `WhlSessionTimeout`: the maximum time a session can be inactive before it expires. Configured in the Session tab of the Advanced Trunk Configuration window, under "Inactive Session Timeout" (described in "Default and Privileged Session Settings" on page 137).
  - `WhlLogoffURL`: URL to be recognized by the filter as the logoff URL. Configured in the Authentication tab of the Advanced Trunk Configuration window, under "Logoff URL" (described in "Configuration in the Authentication Tab" on page 82).

Tip

You can see an example of the use of the `WhlLogoffURL` variable in <MY_FRAME> in "Sample Code: <FRAME_WRAP>" on page 233.

- `WhlScheduledLogoffTimer`: after the time defined in this variable elapses, the user is required to re-authenticate. Configured in the Session tab of the Advanced Trunk Configuration window, under "Automatic Scheduled Logoff After … Minutes" (described in "Default and Privileged Session Settings" on page 137).
- `WhlOwnURL`: URL where the graphic files used by the internal website are located under: …`\Whale-Com\e-Gap\von\InternalSite`.

Tip
You can see an example of the use of the `WhlOwnURL` variable in a <DATA_CHANGE> element in "Sample Code: <DATA_CHANGE>" on page 230.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. **Note:** We recommend that you encode the data under <MY_FRAME>.

Child Elements

None.

Sample Code: <FRAME_WRAP>

```
// A frame wrap is composed of a frameset and a specific frame.
<FRAME_WRAP>
   <PARENT encoding="base64">
     // Note: For the sake of clarity of this sample, the code
     // is not encoded to base64 (in real life it should be).
     // In this sample we define a frameset composed of two
     // frames (those two frames are the required minimum,
     // however you could add more frames). The first frame
     // holds a user's customized frame and the second one holds
     // the requested URL. The respective logical identifiers for
     // those frames are "/WhlFrame.html" and "/WhlOrigURL.html"
     // (case-sensitive).
     <HTML>
        <FRAMESET id=main rows='11%, *'>
           <FRAME src='/WhlFrame.html' scrolling='no'>
           <FRAME src='/WhlOrigURL.html'>
        </FRAMESET>
     </HTML>
   </PARENT>
```

Sample Code: <FRAME_WRAP> (Cont'd)

```
    // This defines the look and content of the user's customized
    // frame which is referred to by the logical name
    // "/WhlFrame.html".
    <MY_FRAME encoding="base64">

      // This is a definition of a simple html document containing
      // a button triggering a call to the logoff url.
      <HTML>

      <HEAD>
      </HEAD>

      <BODY topmargin=0 leftmargin=0 >
         <form action="/WhlLogoffURL.html" method=get>
            <input type=submit value=Logoff>
         </form>
      </BODY>

      </HTML>
    </MY_FRAME>
</FRAME_WRAP>

// Since both the frameset and the user's customized frame are
// controlled by this template, you can easily achieve a wide range
// of capabilities gained by using various Internet technologies
// (JavaScript, HTML, DHTML, etc.).
```

<HEADER_CHANGE>

Note

If a cookie is encrypted by the cookie encryption mechanism, it cannot be manipulated by the <HEADER_CHANGE> element. For details, refer to the *Intelligent Application Gateway User Guide*, to "Cookie Encryption Tab" on page 80.

Description

Adds and deletes headers, and replaces header values, in either requests or responses. Header manipulation can be configured for a single application, a range of applications, or an application-type.

Tip

For a sample of <HEADER_CHANGE>, refer to "Sample Code: <HEADER_CHANGE>" on page 249.

You can also find a sample header manipulation on the IAG, in the following location:

`...\Whale-Com\e-Gap\Von\Samples\AppWrap`

Usage

- <HEADER_CHANGE> is optional.
- Only one <HEADER_CHANGE> element can be nested under <MANIPULATION>.

Attributes & Attribute Values

None.

Child Elements

<HEADER_CHANGE> can contain one each of the following elements:

- <REQUEST>, described on page 235.
- <RESPONSE>, described on page 236.

<HEADER_CHANGE> ➤ <REQUEST>

<REQUEST>

Description

Child element of <HEADER_CHANGE>. Manipulates headers in requests.

Usage

- <REQUEST> is optional.
- Only one <REQUEST> element can be nested under <HEADER_CHANGE>.

Attributes & Attribute Values

None.

Child Elements

- <REQUEST> must contain at least one <APPLICATION> element, described on page 236.
- <REQUEST> can contain an unlimited number of <APPLICATION> elements.

<HEADER_CHANGE> ➤ <RESPONSE>

<RESPONSE>

Description

Child element of <HEADER_CHANGE>. Manipulates headers in responses.

Usage

- <RESPONSE> is optional.
- Only one <RESPONSE> element can be nested under <HEADER_CHANGE>.

Attributes & Attribute Values

None.

Child Elements

- <RESPONSE> must contain at least one <APPLICATION> element, described on page 236.
- <RESPONSE> can contain an unlimited number of <APPLICATION> elements.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION>

<APPLICATION>

Description

Child element of <REQUEST> and of <RESPONSE>. Defines header manipulations for an application or application-type.

Usage

An unlimited number of <APPLICATION> elements can be nested under <REQUEST> and under <RESPONSE>.

Attributes & Attribute Values

None.

Child Elements

<APPLICATION> must contain:

- One of the following:
  - One <SERVER_NAME> element, described on page 237;

  Or,

- One <APPLICATION_TYPE> element, described on page 239.

- One or more <URL> elements, described on page 239.

Optionally, for each <SERVER_NAME> element, <APPLICATION> can also contain one <PORT> element, described on page 238.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <SERVER_NAME>

<SERVER_NAME>

Description

Child element of <APPLICATION>. Defines the server or subnet where manipulated is required.

Usage

- One <SERVER_NAME> element can be nested under each <APPLICATION> element.
- Only one <SERVER_NAME> element can be nested under <APPLICATION>.
- For each <SERVER_NAME> element, one <PORT> element can also be nested under <APPLICATION>.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the “mask” attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

    Note: In this syntax, enter the actual IP address, not regular expressions.

Tip

For an example of the use of the both methods, refer to “Sample Code: <HEADER_CHANGE>” on page 249, to the <RESPONSE> section.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <PORT>

<PORT>

Description

Child element of <APPLICATION>. Defines the port number on which the server, defined in <SERVER_NAME>, listens to incoming requests.

Usage

- Optional.
- For each <SERVER_NAME> element, one <PORT> element can be nested under <APPLICATION>.
- Only one <PORT> element can be nested under <APPLICATION>.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Defines the type of application where manipulated is required.

Note

For instructions on where you can find the name of the application-type, as should be entered here, refer to "Finding the Application-Type" on page 44.
In Basic trunks, the application-type is OtherWeb.

Usage

- One <APPLICATION_TYPE> element can nested under each <APPLICATION> element.
- Only one <APPLICATION_TYPE> element can be nested under <APPLICATION>.

Attributes & Attribute Values

None.

Child Elements

None.

<URL>

Description

Child element of <APPLICATION>. Defines the URL or range of URLs for which header manipulation is configured, and the required manipulations.

Usage

- One <URL> element must be nested under each <APPLICATION> element.
- An unlimited number of <URL> elements can be nested under <APPLICATION>.

Attributes & Attribute Values

None.

Child Elements

<URL> must contain the following:

- One <URL_NAME> element, described on page 240.
- One of the following elements:
  - <ADD>, described on page 241.
  - <DELETE>, described on page 243.
  - <EDIT>, described on page 245.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <URL> ➤ <URL_NAME>

<URL_NAME>

Description

Child element of <URL>. Defines the URL or range of URLs for which header manipulation is configured.

Usage

- One <URL_NAME> must be nested under <URL>.
- Only one <URL_NAME> can be nested under <URL>.
- Takes regular expressions.

Attributes & Attribute Values

Attribute	Values	Type/Comments
InternalScheme	http, https	Optional
ExternalScheme	http, https	Optional

Child Elements

None.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <URL> ➤ <ADD>

<ADD>

Description

Child element of <URL>. Defines headers to add to either requests or responses, for the application defined in <APPLICATION>.

Usage

Only one <ADD> element can be nested under <URL>.

Attributes & Attribute Values

None.

Child Elements

<ADD> can contain an unlimited number of <HEADER> elements, described on page 241.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <URL> ➤ <ADD> ➤ <HEADER>

<HEADER>

Description

Child element of <ADD>. Defines the header that is added.

Usage

- At least one <HEADER> element must be nested under <ADD>.
- An unlimited number of <HEADER> elements can be nested under <ADD>.

Attributes & Attribute Values

None.

Child Elements

<HEADER> must contain one each of the following elements:

- <NAME>, described on page 242.
- <VALUE>, described on page 242.

Note

When you define an <ADD> element, under <HEADER>, <NAME> and <VALUE> elements always come in pairs—for every <NAME> element there must be a <VALUE> element, defining the header value.

<NAME>

Description

Child element of <HEADER>. Name of the header that is added.

Usage

- One <NAME> element must be nested under <HEADER>.
- Only one <NAME> element can be nested under <HEADER>.

Attributes & Attribute Values

None.

Child Elements

None.

<VALUE>

Description

Child element of <HEADER>. Value of the header that is added.

Usage

- One <VALUE> element must be nested under <HEADER>.
- Only one <VALUE> element can be nested under <HEADER>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional

Child Elements

None.

<DELETE>

Description

Child element of <URL>. Defines headers to delete from either requests or responses, for the application defined in <APPLICATION>.

Note

- In requests, you cannot delete "host" headers.
- You can delete all cookies—in either requests or responses—using one element; for details, refer to <NAME> on page 244.

Usage

Only one <DELETE> element can be nested under <URL>.

Attributes & Attribute Values

None.

Child Elements

- <DELETE> must contain at least one <HEADER> element, described on page 243.
- <DELETE> can contain an unlimited number of <HEADER> elements.

<HEADER>

Description

Child element of <DELETE>. Defines the header that is deleted.

Usage

- At least one <HEADER> element must be nested under <DELETE>.
- An unlimited number of <HEADER> elements can be nested under <DELETE>.

Attributes & Attribute Values

None.

Child Elements

- <HEADER> must contain one <NAME> element, described on page 244.
- <HEADER> can contain only one <NAME> element, described on page 244.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <URL> ➤ <DELETE> ➤ <HEADER> ➤ <NAME>

<NAME>

Description

Child element of <HEADER>. Defines the name of the header that is deleted.

Note

In order to delete a "Cookie" or "Set-Cookie" header, you must include the cookie name in the <NAME> element, in addition to the header name.

For example:

In order to delete a "Set-Cookie" header where the name of the cookie is "a", enter the following:

```
<NAME>Set-Cookie:a</NAME>
```

Note that the cookie name is separated from the header name by a colon.

Tip

You can delete all cookies by entering the following:

- In requests: `<NAME>Cookie</NAME>`
- In responses: `<NAME>Set-Cookie</NAME>`

Usage

- One <NAME> element must be nested under <HEADER>.
- Only one <NAME> element can be nested under <HEADER>.
- In "Cookie" and "Set-Cookie" headers only: takes regular expressions in the cookie name.

Attributes & Attribute Values

None.

Child Elements

None.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <URL> ➤ <EDIT>

<EDIT>

Description

Child element of <URL>. Defines the replacement of header values in either requests or responses, for the application defined in <APPLICATION>.

Usage

Only one <EDIT> element can be nested under <URL>.

Attributes & Attribute Values

None.

Child Elements

- <EDIT> must contain at least one <HEADER> element, described on page 245.
- <EDIT> can contain an unlimited number of <HEADER> elements, described on page 245.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <APPLICATION> ➤ <URL> ➤ <EDIT> ➤ <HEADER>

<HEADER>

Description

Child element of <EDIT>. Defines the header that is edited.

Usage

An unlimited number of <HEADER> elements can be nested under <EDIT>.

Attributes & Attribute Values

None.

Child Elements

<HEADER> must contain one each of the following elements:

- <NAME>, described on page 246.
- <SAR>, described on page 247.

<NAME>

Description

Child element of <HEADER>. Defines the name of the header whose value is replaced.

Note

In order to edit a "Cookie" or "Set-Cookie" header, you must include the cookie name in the <NAME> element, in addition to the header name.

For example:

In order to edit a "Set-Cookie" header where the name of the cookie is "a", enter the following:

```
<NAME>Set-Cookie:a</NAME>
```

Note that the cookie name is separated from the header name by a colon.

Usage

- One <NAME> element must be nested under <HEADER>.
- Only one <NAME> element can be nested under <HEADER>.
- In "Cookie" and "Set-Cookie" headers only: takes regular expressions in the cookie name.

Attributes & Attribute Values

Attribute	Values	Default	Type/Comments
method	Name, Value, All	Value	Optional; applicable for cookies only ("Cookie" and "Set-Cookie" headers). Defines where the replacement takes place: • Name: cookie name • Value: cookie value • All: both cookie name and value **Note:** Attribute values are case insensitive

Child Elements

None.

<SAR>

Description

Child element of <HEADER>. Defines header value that is replaced, and the replacement value.

Usage

An unlimited number of <SAR> elements can be nested under <HEADER>.

Attributes & Attribute Values

None.

Child Elements

<SAR> must contain one each of the following elements:

- <SEARCH>, described on page 248.
- <REPLACE>, described on page 248.

Note

<SEARCH> and <REPLACE> elements usually come in pairs—for every <SEARCH> element there must be a <REPLACE> element. An exception to this rule is when you replace a cookie name in “Cookie” and “Set-Cookie” headers (value of the “method” attribute is “Name”). For this type of replacement, there is no need for a <SEARCH> element.

For example:

```
<HEADER>
    <NAME method="Name">cookie:abc.*</NAME>
    <SAR>
     <REPLACE encoding="">ddd</REPLACE>
    </SAR>
</HEADER>
```

<SEARCH>

Description

Child element of <SAR>. Defines the header value that is replaced.

Usage

- One <SEARCH> element must be nested under <SAR>.
- Only one <SEARCH> element can be nested under <SAR>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional

Child Elements

None.

<REPLACE>

Description

Child element of <SAR>. Defines the replacement header value.

Usage

- One <REPLACE> element must be nested under <SAR>.
- Only one <REPLACE> element can be nested under <SAR>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional

Child Elements

None.

Sample Code: <HEADER_CHANGE>

```
<HEADER_CHANGE>
  <REQUEST>
    <!--
    In a request to server 192.168.1.1 and URL /test.html, the code
    in this sample does the following:
    1.In all "cookie" headers that start with "abc", search:
       xxx,replace with: yyy
    2.Add the header "xyz" with the value "zzz"
    3.Delete the "referer" header
    -->
    <APPLICATION>
     <SERVER_NAME mask="">192\.168\.1\.1</SERVER_NAME>
     <PORT>80</PORT>
     <URL>
        <URL_NAME>/test\.html</URL_NAME>
        <EDIT>
           <HEADER>
            <NAME>cookie:abc*</NAME>
            <SAR>
                <SEARCH encoding="">xxx</SEARCH>
                <REPLACE encoding="">yyy</REPLACE>
            </SAR>
           </HEADER>
        </EDIT>
        <ADD>
           <HEADER>
            <NAME>xyz</NAME>
            <VALUE encoding="">zzz</VALUE>
           </HEADER>
        </ADD>
        <DELETE>
           <!-- NOTE: You cannot delete a Host header from
           requests, you can only edit it
           -->
           <HEADER>
            <NAME>referer</NAME>
           </HEADER>
        </DELETE>
     </URL>
    </APPLICATION>
  </REQUEST>
```

Sample Code: <HEADER_CHANGE> (Cont'd)

```
  <RESPONSE>
    <!--
    In response from server 192.168.1.1 and URL /test.html, the
    code in this sample does the following:
    1.In a "set-cookie" header named "AppCookie", search: 125,
       replace with: 126
    2.Add the header "xyz" with the value "zzz"
    3.Delete all the 'Set-Cookie' headers with cookies that start
       with xxx
    In response from subnet 192.168.1.0 and URL /control.html, the
    code in this sample deletes the 'Cache-Control' header.
    -->
    <APPLICATION>
     <SERVER_NAME mask="">192\.168\.1\.1</SERVER_NAME>
     <PORT>80</PORT>
     <URL>
        <URL_NAME>/test\.html</URL_NAME>
        <EDIT>
          <HEADER>
           <NAME>Set-Cookie:AppCookie</NAME>
           <SAR>
               <SEARCH encoding="">125</SEARCH>
               <REPLACE encoding="">126</REPLACE>
           </SAR>
          </HEADER>
        </EDIT>
        <ADD>
          <HEADER>
           <NAME>xyz</NAME>
           <VALUE encoding="">zzz</VALUE>
          </HEADER>
        </ADD>
        <DELETE>
          <HEADER>
           <NAME>Set-Cookie:xxx.*</NAME>
          </HEADER>
        </DELETE>
     </URL>
     <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
     <URL>
        <URL_NAME>/control\.html</URL_NAME>
          <DELETE>
           <HEADER>
               <NAME>Cache-Control</NAME>
           </HEADER>
          </DELETE>
     </URL>
    </APPLICATION>
  </RESPONSE>
</HEADER_CHANGE>
```

<URL_CHANGE>

Description

Used for internal redirection. Redirects the URL before the request is sent to the application server. Can be used to direct users to alternative sites, display messages, and more.

> **Tip**
> Since URL inspection is performed prior to request manipulation by the application customization process, the inspection rules you configure for the trunk are not affected by the <URL_CHANGE> elements.

Usage

- <URL_CHANGE> is optional.
- An unlimited number of <URL_CHANGE> elements can be nested under <MANIPULATION>.
- <URL_CHANGE> is **not** a transitive redirection—a URL can only be redirected once.
  For example: if you redirect a request from `/default.jpg` to `/images/default.jpg`, you cannot then redirect it from `/images/default.jpg` to a different destination by configuring an additional <URL_CHANGE> element.
- <URL_CHANGE> elements are processed in the order by which they appear in the template; the first matching <URL_CHANGE> element is activated.

> **Tip**
> - <URL_CHANGE> elements are processed prior to <DATA_CHANGE> elements (described on page 226).
> - If you wish to combine a <DATA_CHANGE> element with a <URL_CHANGE> element, the URL must reside on the application server; such a combination is not possible when the URL resides on the local host.

Attributes & Attribute Values

None.

Child Elements

<URL_CHANGE> must contain one of each of the following elements:

- <FROM_URL>, described on page 252.
- <TO_URL>, described on page 252.

<FROM_URL>

Description

Child element of <URL_CHANGE>. Defines the URL to be redirected.

Usage

- One <FROM_URL> element must be nested under each <URL_CHANGE> element.
- Only one <FROM_URL> element can be nested under each <URL_CHANGE> element.
- Takes regular expressions.

Attributes & Attribute Values

Attribute	Values	Default	Type/Comments
case_sensitive	true, false	false	Optional

Child Elements

None.

<TO_URL>

Description

Child element of <URL_CHANGE>. Defines the location to where the URL, which is defined in <FROM_URL>, is redirected.

Usage

- One <TO_URL> element must be nested under each <URL_CHANGE> element.
- Only one <TO_URL> element can be nested under each <URL_CHANGE> element.
- Must contain a specific URL (does not take regular expressions).

Attributes & Attribute Values

Attribute	Values	Default	Type/Comments
local_host	true, false	false	Optional. • If local_host=false, the page resides on the application server. • If local_host=true, the page must reside under the IAG's root directory: …\von\InternalSite

Child Elements

None.

Sample Code: <URL_CHANGE>

```
// In this sample we intend to change every URL of the format
// /webload/test_[0-9]*k\.html (pay attention there is a use of
// a regular expression here, which defines a sequence of zero or
// more digits).
// The following URL would be considered a match:
// 1) /webload/test_105k.html
// 2) /webload/test_4k.html
// When identifying a matching URL we change it to the URL defined
// in the TO_URL entry. The TO_URL could be located under the RWS
// (default) or under the filter's root directory, as in this
// sample.

<URL_CHANGE>
     <FROM_URL case_sensitive="false">/webload/
     test_[0-9]*k\.html</FROM_URL>
     <TO_URL local_host="true">/webload/test_1k.html</TO_URL>
</URL_CHANGE>
```

Chapter 8

Optimizing Portal Performance

This chapter describes the configuration and troubleshooting options for Portal trunks, including:

- A description of how the Application Access Portal (AAP) enables access to multiple web applications by using Host Address Translation (HAT), in "Host Address Translation" on page 255
- Options available to you in the Application Access Portal tab of the Advanced Trunk Configuration window, in "Configuration in the Application Access Portal Tab" on page 263
- Advanced configuration of the AAP in the AAP configuration files, in "Advanced Configuration in the Configuration Files" on page 278
- AAP troubleshooting, in "Troubleshooting the Application Access Portal" on page 337

Host Address Translation

The AAP enables access to multiple web applications by using Host Address Translation (HAT). While users communicate with the external website to request, receive, and upload data to and from the applications they access via the IAG, the IAG transparently parses the requests and responses, using content-type parsers, and manipulates the URLs in those transactions on the fly. The parsers manipulate the data so that, to the user, all links to the applications that are enabled via the AAP point to one host—the public host.

- "What Types of Data are Manipulated?" on page 256 explains what types of data the filter manipulates, in both requests and responses.
- "Link Manipulation" on page 256 describes what links are manipulated and explains the manipulation process, including the inherent translation mechanism.
- "Processing Requests" on page 258 and "Processing Responses" on page 261 describe the flow of request and response processing by the filter, respectively.

What Types of Data are Manipulated?

The filter automatically manipulates all the headers in all requests and responses. The manipulation of the body of the transaction, however, depends on its content:

- In requests, body data is manipulated in the following content-types:
  - XML: `text/xml`
  - POST requests containing forms of the following type: `application/x-www-form-urlencoded`
- In responses, body data is manipulated in the following content-types:
  - `text/<suffix>`, where `<suffix>` can be any type of text, for example: `text/html`, `text/css`, `text/plain`
  - `application/x-javascript` (to a limited extent)

Note

If your applications require the manipulation of other types of content, you can configure the AAP configuration file to manipulate additional content-types. For details, refer to "Adding Content-Types to the Content Parsers" on page 343.

In order to streamline the manipulation of body content to conform to your applications, you can configure the types of tags each parser manipulates, or add token-to-token replacement, search and replace, and signature injection mechanisms for content that cannot be handled by the parser. You can also exclude parsing in irrelevant data, for performance tuning.

Tip

Since requests are always manipulated after they pass the filter's content inspection, the inspection rules you configure for the trunk are not affected by the manipulation described here.

Link Manipulation

The following types of URLs are manipulated by the filter in body data:

- Absolute URL: includes the scheme, the server name, and the full path to the page.

  For example:

  `http://www.whalecommunications.com/resources/system/egap.htm`
- Absolute path: representation of the location of the page, relative to the server on which it resides.

For example:

`/resources/system/egap.htm`

- Net path: the browser adds the path to the scheme used in the current URL.

  For example:

  `//www.whalecommunications.com/resources/system/egap.htm`

The filter manipulates URLs of the types described above on their way out of the organization, by adding a unique signature. When an HTML file is received from an application server, the filter signs all the relevant links in the file.

The signature is composed of the following:

- Unique Identifier. You can assign a unique identifier for the site during HAT configuration, as described in the *Intelligent Application Gateway User Guide*, in "Configuring Global Host Address Translation" on page 46. If you do not assign an identifier, the IAG automatically assigns an identifier to the site the first time you configure a Portal trunk.
- Encrypted Data, as determined by the filter.
- Unique Identifier with an additional suffix, which indicates whether the IAG uses SSL to communicate with the application server or not, in the following format:
  - `<Unique Identifier>0`
    IAG does not use SSL to communicate with the application server
  - `<Unique Identifier>1`
    IAG uses SSL to communicate with the application server

For example:

If the organization's unique identifier is `OurID`, the encrypted data created by the IAG filter is `xyz`, and the IAG communicated with the application server using SSL, the signature is:

`OurIDxyz/OurID1`

In this example, if the filter receives from the server `www.myserver.com`, using SSL, a file containing two links that require manipulation, it signs the links as follows:

Original Link	Signed Link
https://www.myserver.com/a.html	http://<Public Hostname>/OurIDxyz/OurID1/a.html
/a.html	/OurIDxyz/OurID1/a.html

> **Tip**
> Public Hostname is defined during portal configuration, as described in the *Intelligent Application Gateway User Guide*, in "Creating an SSL VPN Portal" on page 28.

In this example, if the user requests the second link, the browser sends a request for the URL:

```
/OurIDxyz/OurID1/a.html
```

The filter receives the request and translates the URL to:

```
/a.html
```

The filter then sends the translated request to the server:

```
www.myserver.com
```

Processing Requests

The filter processes incoming requests in the following order:

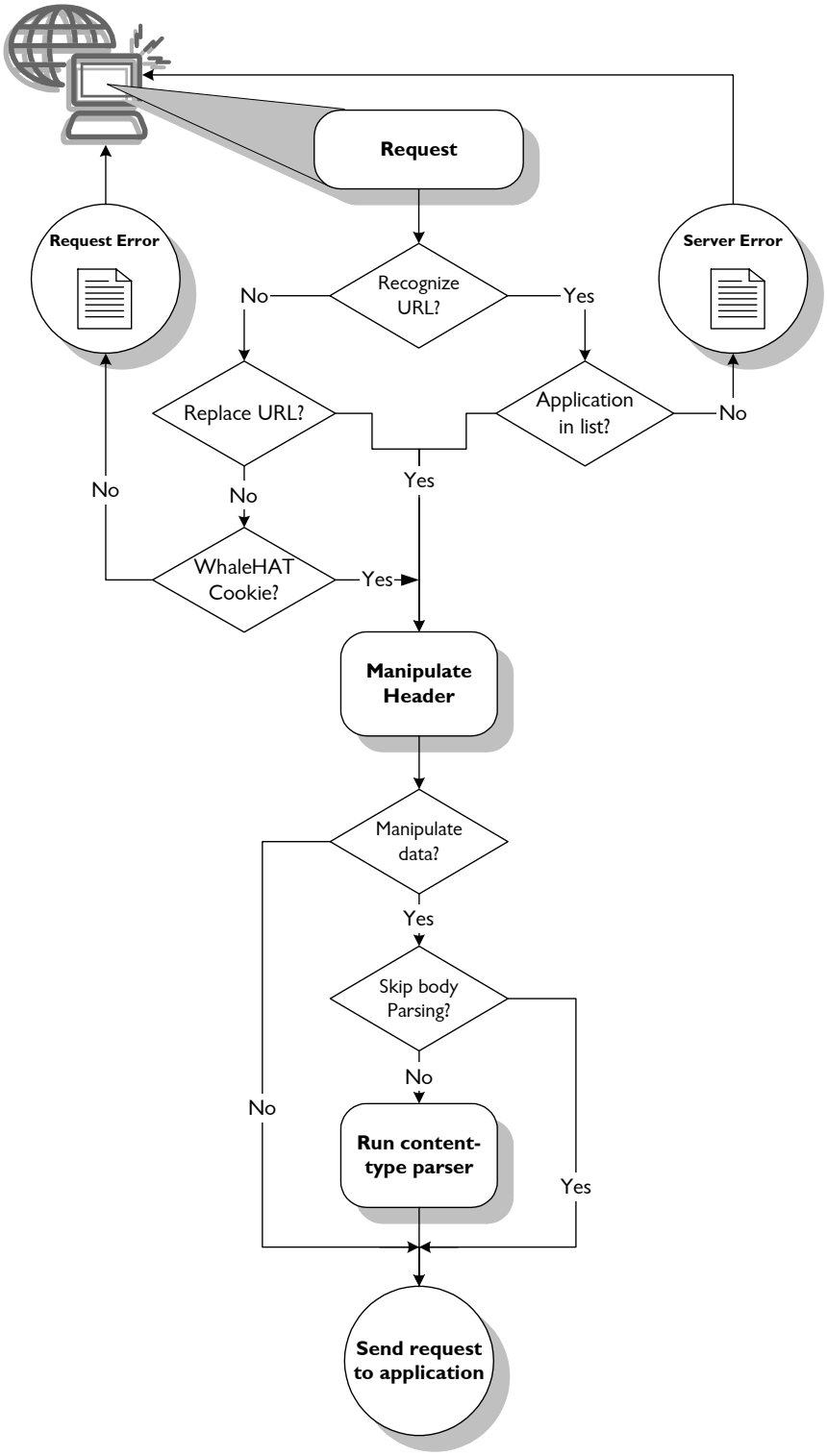
- Determines whether the requested URL is recognized. A recognized URL is either the portal homepage, or a URL that was processed by the filter when the link was sent to the client.
  - If the URL is recognized, the filter goes on to determine whether the application which it resolved in the URL is configured for access via the portal, in the Applications list of the Configuration program. Configuration of the list is described in the *Intelligent Application Gateway User Guide*, in "Creating an SSL VPN Portal" on page 28.
    - If the application is in the Applications list, the filter manipulates the request header, as described later on in this flow.
    - If the application is not in the Applications list, the filter returns a "Server Error" message to the client.
  - If the URL is unrecognized, the filter checks whether it appears in the URL Replacement list.

> **Tip**
> Refer to "Dealing with Unrecognized URLs in Responses" on page 339 for a description of why a URL might not be recognized, and the steps you can take in order to resolve unrecognized URLs.

    - If the URL appears in the URL Replacement list, the filter replaces the URL, then goes on to manipulate the request header, as described later on in this flow. Configuration of this option is described in "Manual URL Replacement" on page 272.
    - If the unrecognized URL is not on the URL Replacement list, the filter checks whether the HTTP request contains a `WhaleHATCookie` header. This cookie contains the server from which the last response was sent. If the `WhaleHATCookie` is found, the request is attributed to the server found in the cookie. The filter then goes on to manipulate the request header, as described in the next step.
    - If the HTTP request does not contain a `WhaleHATCookie`, the filter returns a "Request Error" message to the client.
- The filter manipulates the request header.
- The filter determines whether data manipulation is required in the body of the request, according to the content-type (as described in "What Types of Data are Manipulated?" on page 256).
    - If the request contains data that needs to be manipulated, the filter goes on to the next step.
    - If the request does not contain data that needs to be manipulated, the filter sends the request to the application.
- If the request contains data types that require manipulation, the filter checks whether the requested URL is configured so that body parsing is skipped, as described in "Skip Body Parsing per Application Server" on page 265.
    - If body parsing is required, the filter goes on to the next step.
    - If body parsing is skipped, the filter sends the request to the application.
- The filter runs the applicable parser on the data.
- The filter sends the request to the application.

Figure 35 describes the steps through which the filter goes as it processes requests:

Figure 35. Flow of Request-Processing by the Filter



Processing Responses

The filter processes responses in the following order:

- Manipulates the response header.
- Determines whether data manipulation in the body of the response is required, according to the content-type (as described in "What Types of Data are Manipulated?" on page 256). Content that never contains links, such as graphic files, is not manipulated, and the filter sends the response to the client.

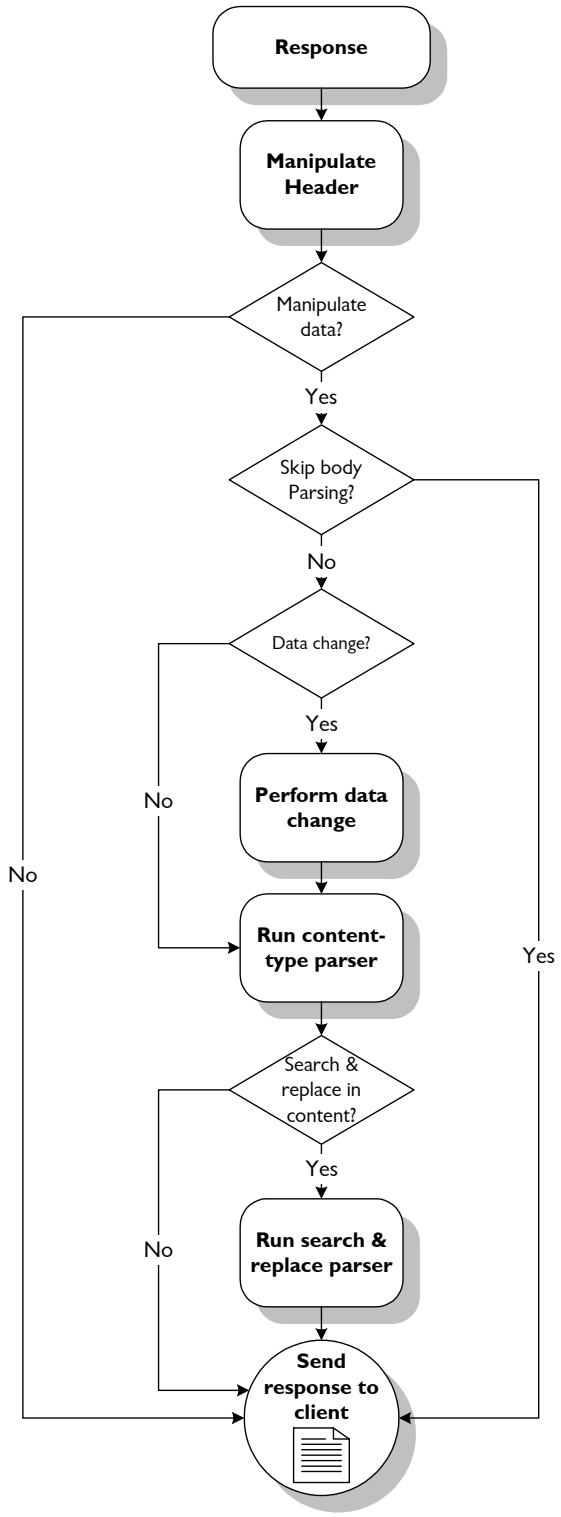
Note

Data that is received from the application server in chunk mode is transformed to content-length.

- Once the filter establishes that the response requires manipulation, it checks whether the response is configured so that body parsing is skipped, as described in "Skip Body Parsing per Application Server" on page 265.
  - If body parsing is required, the filter goes on to the next step.
  - If body parsing is skipped, the filter sends the response to the client.
- The filter checks whether data change is configured, and, if it is, performs the necessary changes. Data change is used for token-to-token replacement and for signature injection, as described in "Data Change" on page 289.
- The filter runs the applicable parser on the data, depending on the content-type (as described in "What Types of Data are Manipulated?" on page 256). The parsers manipulate the data so that, to the user, all links to the applications that are enabled via the AAP point to one site—the public host, that is, the external website. By default, all tags that can contain links are parsed. You can configure additional types of tags to parse, as described in "Configuring the HTML Parser" on page 280 and "Configuring the JavaScript Parser" on page 287, respectively.
- If "Search and Replace Response Content" is configured, the filter runs a search and replace parser on the entire content of the response, including tags that do not normally contain links. It is described in "Search and Replace in Content per Application Server" on page 269.
- The filter sends the response to the client.

Figure 36 describes the steps through which the filter goes while it processes responses:

Figure 36. Flow of Response-Processing by the Filter



Configuration in the Application Access Portal Tab

You can use the Application Access Portal tab, in the Advanced Trunk Configuration window, to configure the following AAP options:

- In the "Skip Body Parsing" area, you can define lists of URLs on which you do not wish to run the content-type parser in either the body of the request, the response, or both, in order to fine-tune system performance. This option is described in "Skip Body Parsing per Application Server" on page 265.

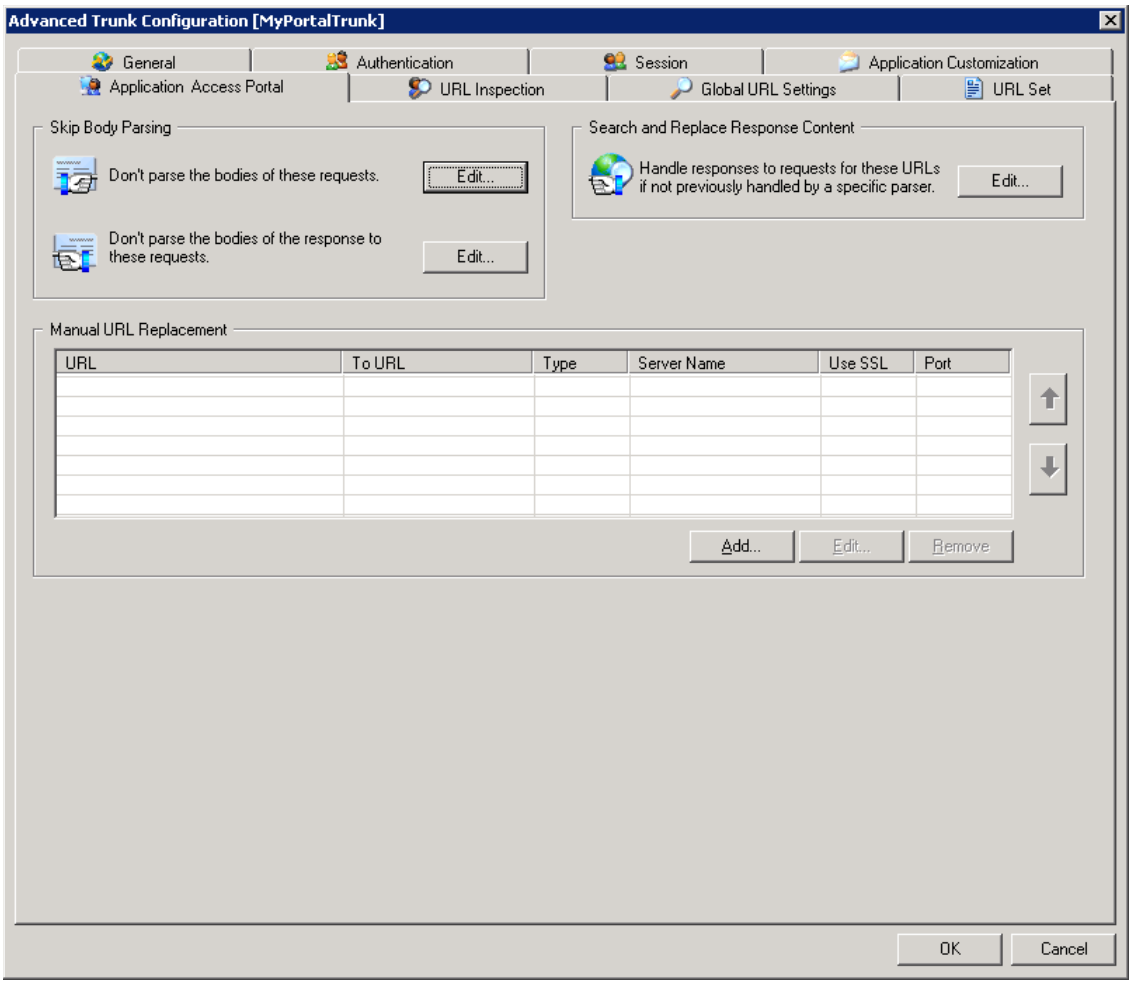
  If you wish to apply this option per application-type, you do this in the AAP configuration file. This option is described in "Application Aware Skip Body Parsing" on page 324.

- In the "Search and Replace Response Content" area, you can define a list of URLs where you wish to run a search and replace parser on the body of the response, as described in "Search and Replace in Content per Application Server" on page 269.

  If you wish to define this option per application-type, you do this in the AAP configuration file. This option is described in "Application Aware Search and Replace in Content" on page 331.

- In the "Manual URL Replacement" area, you can configure a list of URLs that, when requested, will be redirected or rerouted to the appropriate location, as described in "Manual URL Replacement" on page 272.

Figure 37. Application Access Portal Tab



Tip

- Additional options, which you can configure in the AAP configuration files, are described in "Advanced Configuration in the Configuration Files" on page 278.
- You can configure other tabs in the Advanced Trunk Configuration window, as indicated in the *Intelligent Application Gateway User Guide*, in "Where To Go From Here" on page 49.

To configure AAP options in the Application Access Portal tab:

1. In the Configuration program, next to "Advanced Trunk Configuration", click Configure...
   The Advanced Trunk Configuration window is displayed.

2. Select the Application Access Portal tab, and configure the required parameters, as described in the following sections.
3. When you have finished configuring all the required options for the filter, click OK.

   The Advanced Trunk Configuration window closes. When you next activate the trunk's configuration, the options you configured here will be applied to the trunk.

Skip Body Parsing per Application Server

In order to fine-tune IAG performance, you can configure URLs where body parsing is skipped. For example: pages that contain no links. You can also skip body parsing in pages where all the links are relative path URLs, where there is no need for link replacement, since the server name does not appear in the URL.

This section describes how you define the "Skip Body Parsing" option per application server, in the Configuration program. If you wish to implement this option per application-type, you can do this in the AAP configuration file, as described in "Application Aware Skip Body Parsing" on page 324.

The following procedure describes how you configure this option for responses. The configuration of this option for requests is identical.

Note

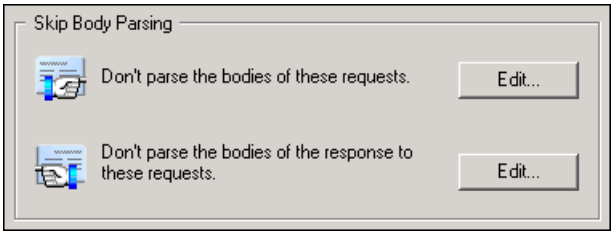
By default, body data is manipulated in requests in these content-types:

- XML: `text/xml`
- POST requests containing forms of the following type: `application/x-www-form-urlencoded`

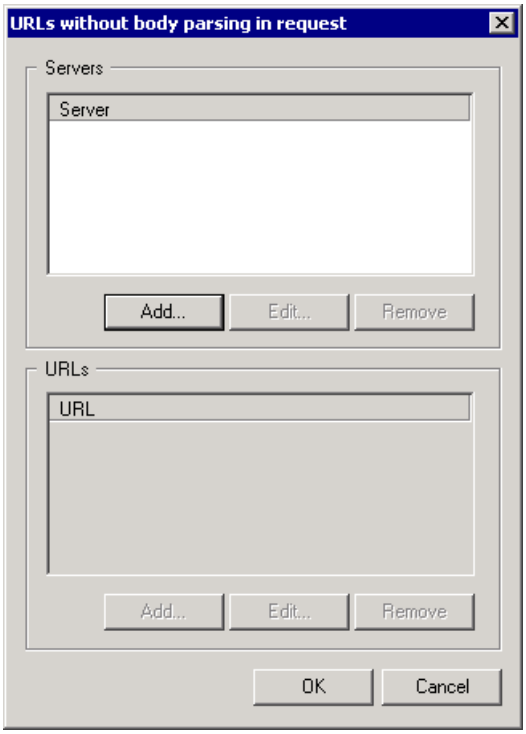
If necessary, you can configure additional content-types that will be manipulated in request data, as described in "Adding Content-Types to the Content Parsers" on page 343.

To configure responses where body parsing is skipped:

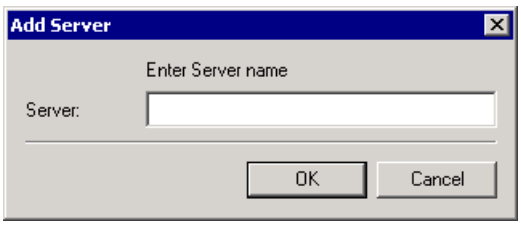
1. In the Application Access Portal tab, in the "Skip Body Parsing" area, click Edit... next to "Don't parse the bodies of the response to these requests".



The URL without body parsing in response dialog box is displayed.

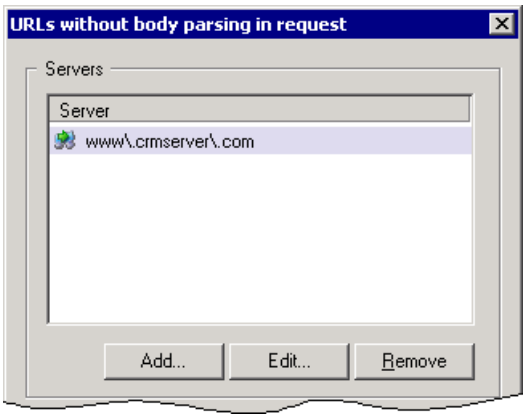


2. In the "Servers" area, click Add...
   The Add Server dialog box is displayed.



3. Enter the name of the site on which the page or pages reside, using regular expressions. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax".
   Click OK.

The Add Server dialog box closes. In the URL without body parsing in response dialog box, the site you configured is added in the "Servers" area.

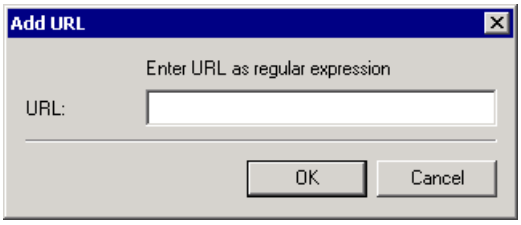


4. In the "URLs" area, click Add...

Note

For each site you configure, you must also configure at least one URL. You cannot configure the option for the entire site.

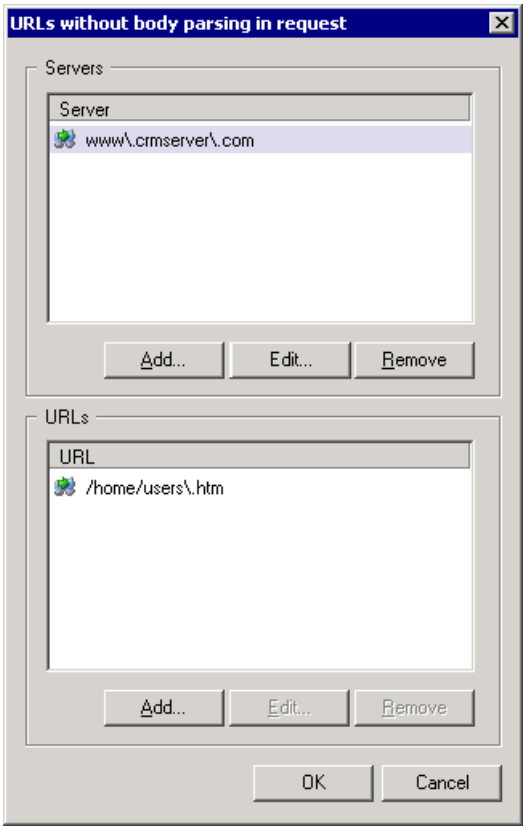
The Add URL dialog box is displayed.



5. Enter a URL or a range of URLs, using regular expressions. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax".

   Click OK.

   The Add URL dialog box closes. In the URL without body parsing in response dialog box, the URL or range of URLs you configured is added in the "URLs" area.



Tip
To display the URLs that are configured for a site, select the site in the Server list.

6. Repeat steps 2–5 to add all the sites and URLs for which you wish to skip body parsing.
7. In the URL without body parsing in response dialog box, click OK.
   The dialog box closes. Once the trunk is activated, the body of the requests you configured here will not be parsed.

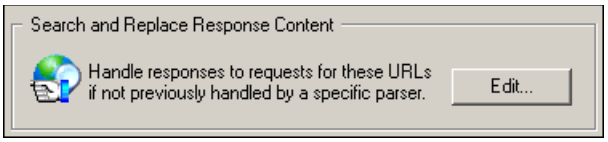
Search and Replace in Content per Application Server

This option is applicable for responses only. The search and replace engine runs on the entire HTTP data in the trunk, including all tags, and manipulates absolute URLs, in order to hide link names in body data that is not otherwise handled by the content-specific parser. For example: URLs that appear within HTML text tags, or Java comments.

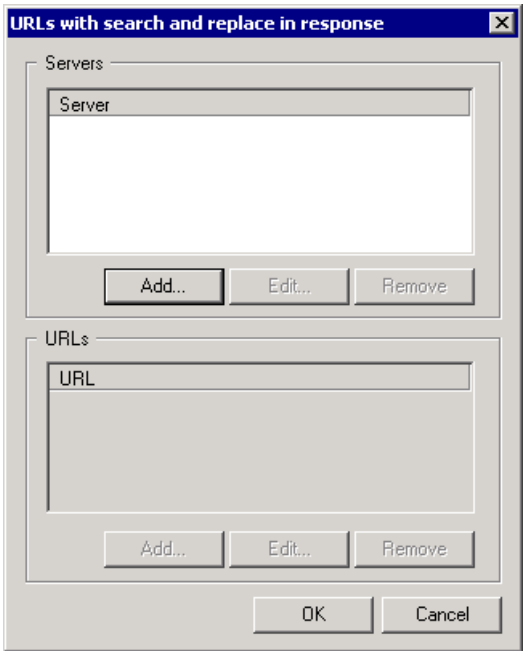
This section describes how you configure this option per application server, in the Configuration program. If you wish to implement this option per application-type, you can do this in the AAP configuration file, as described in "Application Aware Skip Body Parsing" on page 324.

To configure responses with content search and replace:

1. In the Application Access Portal tab, in the "Search and Replace Response Content" area, click Edit...



The URL with search and replace in response dialog box is displayed.



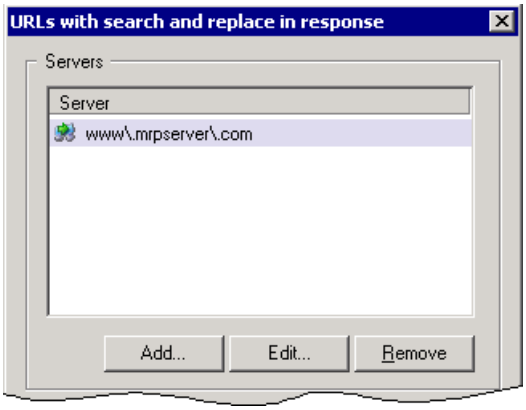
2. In the "Servers" area, click Add...

   The Add Server dialog box is displayed.

3. Enter the name of the site on which the page or pages reside, using regular expressions. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax".

   Click OK.

4. The Add Server dialog box closes. In the URL with search and replace in response dialog box, the site you configured is added in the "Servers" area.

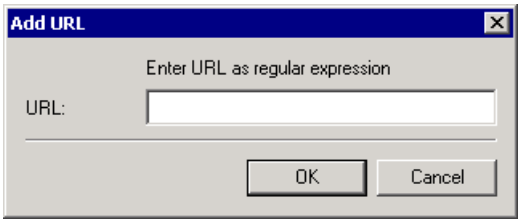


5. In the "URLs" area, click Add...

Note

For each site you configure, you must also configure at least one URL. You cannot configure the option for the entire site.

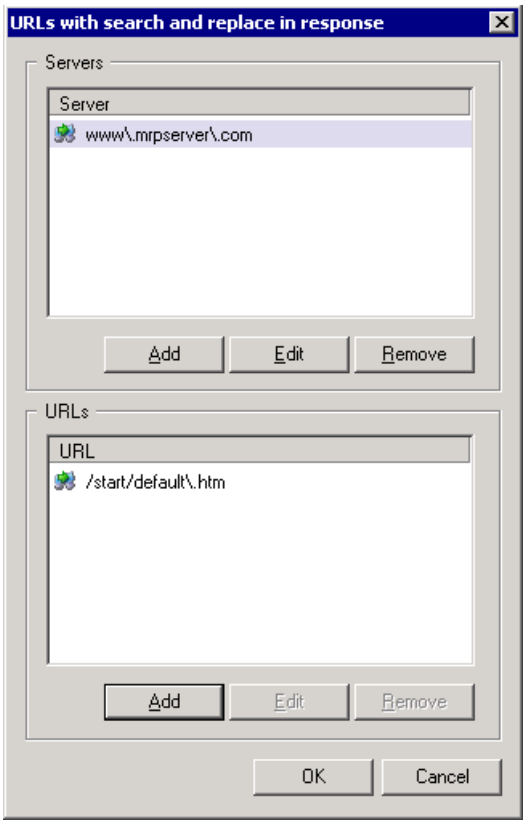
The Add URL dialog box is displayed.



6. Enter a URL or a range of URLs, using regular expressions. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax".

   Click OK.

The Add URL dialog box closes. In the URL with search and replace in response dialog box, the URL or range of URLs you configured is added in the "URLs" area.



Tip

To display the URLs that are configured for a site, select the site in the Server list.

7. Repeat steps 2–6 to add all the sites and URLs for which you wish to apply search and replace in the response content.
8. In the URL with search and replace in response dialog box, click OK.

   The dialog box closes. Once the trunk is activated, the entire body of responses to the requests you configured here will be parsed.

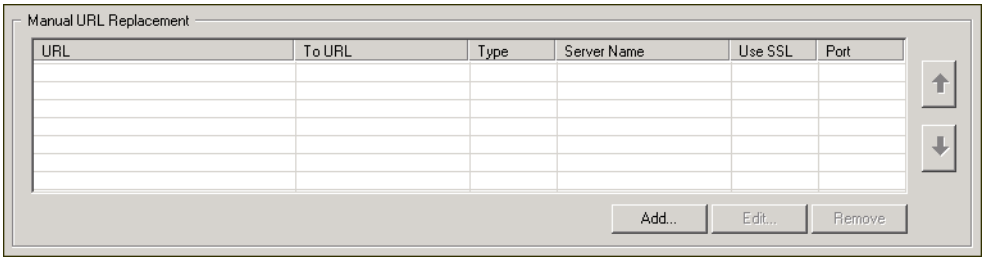
Manual URL Replacement

Use this option to replace unrecognized URLs in requests with a valid path, as described in “Processing Requests” on page 258.

Tip

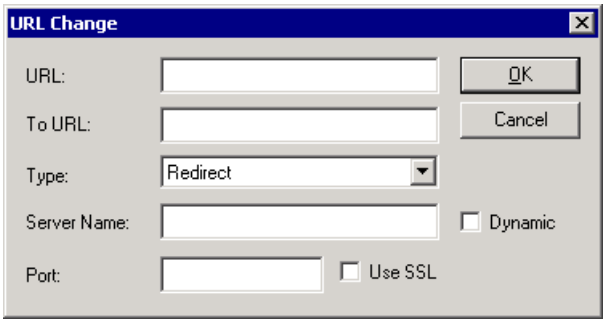
Make sure the URL to where you redirect or reroute the request is one of the sites that are on the Application List. For details, refer to the *Intelligent Application Gateway User Guide*, to “Creating an SSL VPN Portal” on page 28.

You configure the URL replacement rules in the Application Access Portal tab of the Advanced Trunk Configuration window, in the “Manual URL Replacement” area:



To configure Manual URL Replacement:

1. Under the Manual URL Replacement list, click Add...
   The URL Change dialog box is displayed.



2. Configure the URL replacement rule, as described in Table 33, “Manual URL Replacement Parameters”, on page 273.
3. Once you finish configuring the rule, in the URL Change dialog box, click OK.
   The URL Change dialog box closes. In the Application Access Portal tab, the replacement you configured is added in the “Manual URL Replacement” area.

Manual URL Replacement:

URL	To URL	Type	Server Name	Use SSL	Port
/exchange/.*		Redirect	192.181.1.138	No	80

Tip

If you wish to redirect the URL to a URL with a different name, enter the replacement URL in the "To URL" field:

Manual URL Replacement:

URL	To URL	Type	Server Name	Use SSL	Port
/exchange/.*	/exchange_internal/	Redirect	192.181.1.138	No	80

4. Repeat steps 2–3 to configure all the URLs you wish to replace.

   Once the trunk is activated, the absolute paths you configured here will be replaced with absolute URLs, and will then be processed by the filter.

Note

If the replacement type is "Rerouting", and you enter a value in "To URL", the replacement mechanism adds the value of the source URL as a parameter to the destination URL.

For example:

If "URL" is `/whale`, "To URL" is `/whale_internal`, and "Type" is Rerouting, the destination URL is:

`/whale_internal?WhlSourceURL=/whale`

Table 33. Manual URL Replacement Parameters

Parameter	Description
URL	URL path to be replaced. Must be a regular expression. For a description of regular expressions, refer to Appendix B: "Regex++, Regular Expression Syntax".

Table 33. Manual URL Replacement Parameters (Cont'd)

Parameter	Description
To URL	URL path that will replace the path defined in "URL", above. In most cases, this field is left empty, and the value of this field is derived from the URL path requested by the browser, on the site defined in "Server Name", below. **Note:** If you do fill in this field, enter an absolute path, using plain text; do not use regular expressions in this filed.
Type	Select the type of replacement: • Redirect: returns the HTTP 302 status, which redirects the browser to the URL path defined in "To URL", above. This option is selected by default, and is recommended for use with most applications, since all relative links within the HTML page are also directed to their true location when requested. • Rerouting: replaces the original URL with the URL defined in "To URL", above. Select this option only if your application uses a protocol that does not support the HTTP 302 status, such as WebDAV. In this case, you have to configure each of the relative path links that appear within the HTML page to be rerouted to their true location. **Note:** "Redirect" can only be applied to GET requests. All other requests are rerouted.
Server Name	IP address or domain name of the server to which "URL", above, will be redirected or rerouted (precedes the URL defined in "To URL", above). **Note:** If the rule you are creating is a dynamic forwarding rule, refer to "Dynamic Forwarding Rules" on page 275 for instructions on how to define this field.
Dynamic	Determines whether requests will be dynamically rerouted or redirected between different servers. A description of this feature and instructions on how to define it are provided in "Dynamic Forwarding Rules" on page 275.
Port	Port number of the server defined in "Server Name", above.
Use SSL	Check this option if the port is an SSL port.

Dynamic Forwarding Rules

When there is more than one server defined for an application, you can define a dynamic URL replacement rule that will automatically reroute or redirect requests to the appropriate application server for the current session. The dynamic forwarding rule must include two server-related definitions:

- A **dynamic parameter** that is used to determine the destination server to which the request is forwarded. You define the dynamic parameter in an authentication hook, and refer to it from the URL replacement rule.
- A **fallback server**, to which requests are forwarded in case the dynamic parameter cannot be resolved. The fallback server must be a server that is defined in the IAG, that is, it appears in the Web Servers tab of the Application Properties dialog box for one of the defined applications.

To create a dynamic forwarding rule:

1. Create an authentication hook, or use an existing hook, as described in "Adding Functionality to the Default Pages" on page 99. **You must use a hook that is executed before the request is processed, otherwise the fallback server will be used for the duration of the session.**
2. In the hook, add the required dynamic parameter value assignments. Each assignment must be in this format:

```
SetSessionParam g_cookie,
"<DynamicParamName>","<DynamicParamValue>"
```

Where:

- `DynamicParamName` is the session variable name that is referred to from the dynamic forwarding rule. You must use the same `DynamicParamName` in all the dynamic parameter value assignments that relate to the dynamic forwarding rule.
- `DynamicParamValue` is a dynamically assigned server hostname to be used at runtime when a request to the URL defined in the dynamic forwarding rule is handled.

For Example:

```
<%
If ConditionA Then
     SetSessionParam g_cookie, "DynamicServer",
     "app-serverONE.mycompany.com"
Else
     SetSessionParam g_cookie, "DynamicServer",
     "app-serverTWO.mycompany.com"
End If
%>
```

Where:

- `DynamicServer` is the session variable used for all assignments related to this rule.
- The first assignment determines that the server:
  `app-serverONE.mycompany.com`
  is used if the runtime check succeeds.
- The second assignment determines that the server:
  `app-serverTWO.mycompany.com`
  is used if the runtime check fails.

3. In the Application Access Portal tab of the Advanced Trunk Configuration window, in the "Manual URL Replacement" area, click Add...

   The URL Change dialog box is displayed.

4. Configure the URL replacement rule, as described in Table 33 on page 273, including the following:
   - In the Server Name field, enter this string:
     `*<DynamicParameterName>*<fallback_server>`
     Where:
     - `<DynamicParameterName>` is the session variable name you defined in step 2.
     - `<fallback_server>` is a fallback server you choose from the servers defined for the relevant application, and which appears in the Web Servers tab of the Application properties dialog box.
   - Select the "Dynamic" option.
5. Once you finish configuring the rule, click OK to close the URL Change dialog box.

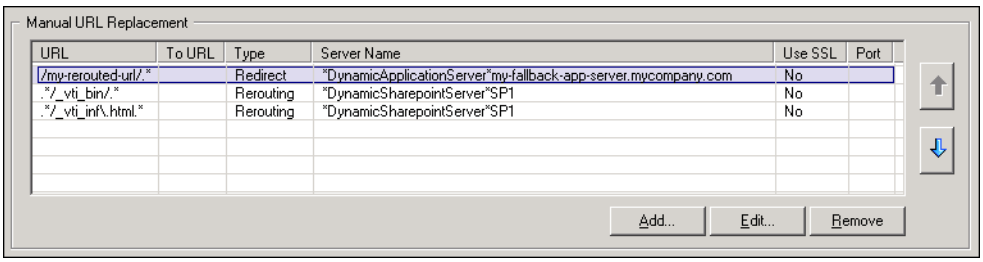
   The rule you defined is added in the "Manual URL Replacement" area. Once the trunk is activated, requests are forwarded dynamically to one of the servers defined in the hook, or to the fallback server.

Note

- The server to which requests are forwarded when the first request is sent is used for the entire session. The dynamic forwarding rule is applied again when a new session is opened.
- If you change the application servers in the Web Settings tab of the Application Properties dialog box, make sure that you adapt both the dynamic parameter in the authentication hook and the dynamic forwarding rule accordingly.

For Example:

In the Application Access Portal tab of the Advanced Trunk Configuration window, the dynamic forwarding rule that is highlighted below is defined:



In the `WhalePortal1PostPostValidate.inc` file, the following lines define the dynamic parameter:

```
<%
If SomeRuntimeCheck Then
     SetSessionParam g_cookie, "DynamicApplicationServer",
     "my-first-app-server.mycompany.com"
Else
     SetSessionParam g_cookie, "DynamicApplicationServer",
     "my-second-app-server.mycompany.com"
End If
%>
```

When a request to a URL that begins with `/my-rerouted-url/` is handled, the dynamic forwarding rule refers the request to one of the dynamically assigned server hostnames, depending on whether the runtime check succeeds or fails:

`my-first-app-server.mycompany.com` or `my-second-app-server.mycompany.com`

If the `DyanmicApplicationServer` session variable cannot be resolved, the request is sent to the fallback server:
`my-fallback-app-server.mycompany.com`

Advanced Configuration in the Configuration Files

In addition to the AAP options you can configure in the Application Access Portal tab, you can configure the following options in the AAP configuration files:

- Additional tags that will be parsed by the content-type parsers in responses. By default, the content-type parsers are configured so that all tags that can contain links are parsed during data manipulation. You can configure additional tags that will be parsed, for the following parsers:
  - HTML parser, described on page 280.
  - JavaScript parser, described on page 287.
- Data change in responses, including token-to-token replacement and signature injection, described in "Data Change" on page 289.
- Cookie manipulation in responses, described in "Cookie Handling" on page 300.
- Header filtering in requests, described in "Header Filtering" on page 314.
- Define servers that will not be manipulated by the IAG HAT mechanism, as described in "Excluding Servers from the Manipulation Process" on page 321.
- Define application-types on which the content-type parser will not run, in either the body of the request, the response, or both, in order to fine-tune system performance. This option is described in "Application Aware Skip Body Parsing" on page 324.
- Define application-types on which the search and replace parser will run, on the body of the response, as described in "Application Aware Search and Replace in Content" on page 331.

Note

The AAP configuration files are XML files. In order to edit them, you need to have a working knowledge of XML technology.

The following sections describe the AAP configuration files and the elements you can edit in order to streamline your application. Before you edit the files, be sure to read the section "Before You Begin" on page 279; it holds important information you should note.

Before You Begin

The files that control the configuration of the AAP are located under the following folder:

…\Whale-Com\e-Gap\Von\Conf\SRATemplates

For HTTP trunks:

WhlFiltSecureRemote_HTTP.xml

For HTTPS trunks:

WhlFiltSecureRemote_HTTPS.xml

When the configuration is activated and the trunk's internal website is created, the files are copied under the following location:

…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf

In order to make changes to the default settings, copy the file you wish to edit under the following custom folder, and make your changes there; if this folder does not exist, you need to create it:

…\Whale-Com\e-Gap\Von\Conf\Websites\<Trunk_Name>\Conf\CustomUpdate

The changes you make in the custom file are merged with the definitions of the file that is supplied with the system.

Caution

You can use the files to configure **only** the options described in this section and in the section "Troubleshooting the Application Access Portal" on page 337. **Do not** attempt to configure any of the other AAP options via the configuration files.

The AAP configuration files follow these conventions:

- Tag names are uppercase:

  <TAG>…</TAG>

- Attribute names are lowercase:

  <TAG **attribute**=value>…</TAG>

- Attribute values are case insensitive:

  <TAG attribute=**value**>…</TAG>

  is identical to:

  <TAG attribute=**VALUE**>…</TAG>

- Some XML elements take regular expressions in the data. When regular expressions are used, this is indicated in the description of the element usage. When using regular expressions, note the following:
  - Use the syntax described in Appendix B: “Regex++, Regular Expression Syntax”
  - In order to use a character as is, make sure it is preceded by a slash. For example, to use the asterisk (`*`) wildcard in data that takes regular expressions, enter `\*`

Once You Edit a File

Once you edit an AAP configuration file, in order for the changes to go into effect, take the following steps:

1. At the Configuration program, click to activate the configuration.
2. In the Activate Configuration window, select the option “Apply changes made to external configuration settings”, and click Activate >.

Configuring the HTML Parser

The HTML parser is used in responses only. It is defined in the AAP configuration file in the element <PARSER_HTML>, described below.

<PARSER_HTML>

Description

Defines all the HTML tags and event handlers that are parsed; the links that are found within those tags and event handlers are manipulated, as described in “Host Address Translation” on page 255. By default, the parser is configured to parse all standard tags and event handlers that contain links. You can edit the file to add other, non-standard tags or event handlers, which might contain links in your application.

Usage

One, and only one, <PARSER_HTML> element must reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <PARSER_HTML> can contain one <EVENT_HANDLERS> element, described on page 281.
- <PARSER_HTML> can contain an unlimited number of <TAG> elements, described in <TAG> on page 283.

<EVENT_HANDLERS>

Description

Child element of <PARSER_HTML>. Defines HTML and DHTML event handlers that will be parsed. The event handlers are associated with the HTML tags via tag attributes. When an event handler is configured to be parsed, it is parsed in all the tags that the parser handles (defied in the <TAG> elements).

- For each event handler you can configure a list of “exclude” tags, in which the handler will not be parsed.
- In order to parse an event handler in a selected tag or tags only, define it as part of the tag definition, in an <ATTR> element, described on page 284.

Usage

- Optional
- One <EVENT_HANDLERS> element can reside under <PARSER_HTML>.

Attributes & Attribute Values

None.

Child Elements

<EVENT_HANDLERS> can contain an unlimited number of <HANDLER> elements, described below.

<HANDLER>

Description

Child element of <EVENT_HANDLERS>. Defines the HTML or DHTML event handler that will be parsed.

Usage

An unlimited number of <HANDLER> elements can be nested under <EVENT_HANDLERS>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
parser	PARSER_EXCEPTION PARSER_SAR	Mandatory. Defines the parser that will parse the script within the event handler: • PARSER_EXCEPTION: parses JavaScript • PARSER_SAR: parses absolute URLs

Child Elements

- <HANDLER> must contain one <NAME> element, described below.
- <HANDLER> can contain an unlimited number of <EXCLUDE> elements, described on page 283.

<PARSER_HTML> ➤ <EVENT_HANDLERS> ➤ <HANDLER> ➤ <NAME>

<NAME>

Description

Child element of <HANDLER>. Defines the name of the event handler that will be parsed.

Usage

One, and only one, <NAME> element must be nested under each <HANDLER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<EXCLUDE>

Description

Child element of <HANDLER>. Defines an HTML tag in which the event handler, defined in <NAME>, will not be parsed.

Usage

- <EXCLUDE> is optional.
- An unlimited number of <EXCLUDE> elements can be nested under each <HANDLER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<TAG>

Description

Child element of <PARSER_HTML>. Defines the tag that will be parsed.

Usage

- At least one <TAG> element must reside under <PARSER_HTML>.
- An unlimited number of <TAG> elements can reside under <PARSER_HTML>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
replace_att	true, false	Optional. Searches for and manipulates links inside the defined tag attributes.
sar	true, false	Optional. Searches for and manipulates links in the entire tag.
tag_delimiter	Delimiter of the tag defined in the <NAME> child element.	Optional. Must be used if sar=true. **Examples:** • The delimiter of the tag <SCRIPT> is </SCRIPT> • The delimiter of the tag <META is >

Note

In each <TAG>, at least one of the following attributes must have a `true` value:

- `replace_att`
- `sar`

In each <TAG>, both attributes can be `true`.

Child Elements

- <TAG> must contain one <NAME> element, described below.
- <TAG> can contain only one <NAME> element.
- When `replace_att=true`, <TAG> must contain at least one <ATTR> element, described on page 284.
- When `replace_att=true`, <TAG> can contain an unlimited number of <ATTR> elements.

<PARSER_HTML> ➤ <TAG> ➤ <NAME>

<NAME>

Description

Child element of <TAG>. Defines the name of the tag to be parsed.

Usage

One, and only one, <NAME> element must reside under <TAG>.

Attributes & Attribute Values

None.

Child Elements

None.

<PARSER_HTML> ➤ <TAG> ➤ <ATTR>

<ATTR>

Description

Child element of <TAG>. Used only when, in the definition of <TAG>, `replace_att=true`. Defines the attribute that will be parsed within the tag defined in <NAME>, described on page 284.

Usage

- If, in the definition of <TAG>, `replace_att=true`, one <ATTR> element must reside under <TAG>.
- If, in the definition of <TAG>, `replace_att=true`, an unlimited number of <ATTR> elements can reside under <TAG>.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <PARSER_HTML>

```
<PARSER_HTML>
   <!--This tag holds the configuration of the html parser.
    For each tag there are three actions:
    1.   Event handlers that contain JavaScript executable code.
    2.   replace_attr (in which the attr to replace should be
         specified)
    3.   sar (in which case the delimiter of the tag should be
         specified)
   <EVENT_HANDLERS>
    <!-- Each event handler can have an exclude list of all
         the HTML tags in which the event handler will not be
         parsed. In the following example, the OnClick
         handler will not be parsed by the HTML parser when
         it appears in the tag a:
         <HANDLER parser="PARSER_JAVASCRIPT">
          <NAME>OnClick</NAME>
          <EXCLUDE>a</EXCLUDE>
         </HANDLER>
    -->
    <!-- HTML event handlers -->
    <HANDLER parser="PARSER_JAVASCRIPT">
         <NAME>OnClick</NAME>
    </HANDLER>
    <HANDLER parser="PARSER_JAVASCRIPT">
         <NAME>OnDblClick</NAME>
    </HANDLER>
    <HANDLER parser="PARSER_JAVASCRIPT">
         <NAME>OnMouseDown</NAME>
    </HANDLER>
    <!-- DHTML event handlers -->
    <HANDLER parser="PARSER_JAVASCRIPT">
         <NAME>OnBounce</NAME>
    </HANDLER>
    <HANDLER parser="PARSER_JAVASCRIPT">
         <NAME>OnScroll</NAME>
    </HANDLER>
   </EVENT_HANDLERS>
```

Sample Code: <PARSER_HTML> (Cont'd)

```
    <!--
    ==========================================================
    ==========================================================
    TAGS which have a list of replaceable attributes
    ==========================================================
    ==========================================================
    -->
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>a</NAME>
      <ATTR>href</ATTR>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>area</NAME>
      <ATTR>href</ATTR>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>base</NAME>
      <ATTR>href</ATTR>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>blockquote</NAME>
      <ATTR>cite</ATTR>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>body</NAME>
      <ATTR>background</ATTR>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>button</NAME>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>del</NAME>
      <ATTR>cite</ATTR>
    </TAG>
    <TAG replace_attr="true" sar="false" tag_delimiter="">
      <NAME>embed</NAME>
      <ATTR>pluginspage</ATTR>
      <ATTR>src</ATTR>
    </TAG>
</PARSER_HTML>
```

Configuring the JavaScript Parser

The JavaScript parser is used in responses only. It is defined in the AAP configuration file in the element <PARSER_EXCEPTION>, described below.

Tip

When you configure the manipulation of a tag in the parser, the parser searches for the tag in all responses. If you wish to apply the manipulation to a limited number of URLs, you can apply token-to-token replacement to the specific URLs, thus enhancing system performance. For details refer to "Data Change" on page 289.

<PARSER_EXCEPTION>

Description

Defines all the JavaScript commands that are parsed; the links that are found within those commands are manipulated, as described in "Host Address Translation" on page 255. By default, the parser is configured to parse all standard commands that contain links. You can edit the file to add other, non-standard commands, which might contain links in your application.

Usage

One, and only one, <PARSER_EXCEPTION> element must reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <PARSER_EXCEPTION> must contain at least one <SEARCH> element, described on page 288.
- <PARSER_EXCEPTION> can contain an unlimited number of <SEARCH> elements.

<SEARCH>

Description

Child element of <PARSER_EXCEPTION>. Defines the command to be parsed and the command's operation character.

Usage

- At least one <SEARCH> element must be nested under <PARSER_EXCEPTION>.
- An unlimited number of <SEARCH> elements can be nested under <PARSER_EXCEPTION>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
opr_char	(=	Mandatory. Defines the operation character: function opening or implementation, respectively.
encoding	base64	Optional.

Child Elements

None.

Sample Code

The following sample configures the JavaScript parser to search for links in the command `openNewWindow`, with the following attributes:

- The command character is directly followed by an opening parenthesis, therefore the operation character is (
- Encoding is not used

```
<SEARCH start="(" encoding="">openNewWindow</SEARCH>
```

Sample Code: <PARSER_EXCEPTION>

```
<PARSER_EXCEPTION>
     <!-- This TAG holds Java Script configuration -->
     <SEARCH opr_char="(" encoding="">openNewWindow</SEARCH>
     <SEARCH opr_char="(" encoding="">openInnewWindow</SEARCH>
     <SEARCH opr_char="(" encoding="">openWin</SEARCH>
     <SEARCH opr_char="(" encoding="">open</SEARCH>
     <SEARCH opr_char="(" encoding="">load</SEARCH>
     <SEARCH opr_char="(" encoding="">url</SEARCH>
     <SEARCH opr_char="(" encoding="">location.replace</SEARCH>
     <SEARCH opr_char="=" encoding="">href</SEARCH>
     <SEARCH opr_char="=" encoding="">location</SEARCH>
     <SEARCH opr_char="=" encoding="">location.href</SEARCH>
     <SEARCH opr_char="=" encoding="">selecter1.action</SEARCH>
     <SEARCH opr_char="=" encoding="">url</SEARCH>
     <SEARCH opr_char="=" encoding="">pathname</SEARCH>
     <SEARCH opr_char="=" encoding="">src</SEARCH>
     <SEARCH opr_char="=" encoding="">lowsrc</SEARCH>
</PARSER_EXCEPTION>
```

Data Change

Data change is used in responses, and is preformed prior to content parsing. It can be used as follows:

- Token-to-token replacement is used to replace text strings within text buffers. **For example:** since in <SCRIPT> tags the filter can only manipulate absolute paths if they are contained in a standard function, use this type of replacement in server-specific scripts, where URLs are contained within non-standard function calls.
- Signature injection is used to inject the IAG unique signature before or after a specific text string. **For example:** if, within an unknown XML element, the text buffer includes a relative link, use this method to add the signature before the link.

You configure data change in the AAP configuration file, using the <DATA_CHANGE> element, described below.

<DATA_CHANGE>

Description

Used to manipulate the content of the response before data is sent to the client, as follows:

- Search and replace data strings within text buffers
- Inject the IAG unique signature before or after data strings within text buffers

Can be applied per server or a number of servers, or per application-type or types.

Usage

- <DATA_CHANGE> is optional.
- Only one <DATA_CHANGE> element can be configured in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <DATA_CHANGE> can contain an unlimited number of <SERVER> elements, described on page 291.
- <DATA_CHANGE> can contain an unlimited number of <APPLICATION> elements, described on page 291.

<SERVER>

Description

Child element of <DATA_CHANGE>. Defines the server and URL in which data is changed.

Usage

An unlimited number of <SERVER> elements can be nested under <DATA_CHANGE>.

Attributes & Attribute Values

None.

Child Elements

<SERVER> must contain the following:

- One <SERVER_NAME> element, described on page 292.
- One or more <URL> elements, described on page 294.

Optionally, <SERVER> can also contain one <PORT> element, described on page 293.

<APPLICATION>

Description

Child element of <DATA_CHANGE>. Defines the application-type in which data is changed.

Usage

An unlimited number of <APPLICATION> elements can be nested under <DATA_CHANGE>.

Attributes & Attribute Values

None.

Child Elements

<APPLICATION> must contain the following:

- One <APPLICATION_TYPE> element, described on page 293.
- One or more <URL> elements, described on page 294.

<SERVER_NAME>

Description

Child element of <SERVER>. Defines the server in which data is changed.

Usage

- One, and only one, <SERVER_NAME> element must be nested under each <SERVER> element.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the "mask" attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

    Note: In this syntax, enter the actual IP address, not regular expressions.

Tip
For examples of the use of both methods in a <SERVER_NAME> element, refer to "Sample Code: <DATA_CHANGE>" on page 299.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<PORT>

Description

Child element of <SERVER>. Specifies the port number, on the server defined in <SERVER_NAME>, on which data is changed.

Usage

- Optional.
- Only one <PORT> element can be nested under each <SERVER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Type of application where data is changed.

Note

For instructions on where you can find the name of the application-type, as it should be entered here, refer to "Finding the Application-Type" on page 44.

Usage

One, and only one, <APPLICATION_TYPE> element must be nested under each <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<URL>

Description

Child element of <SERVER> or of <APPLICATION>. Defines the URL and the required data change.

Usage

- At least one <URL> element must be nested under each <SERVER> or <APPLICATION> element.
- An unlimited number of <URL> elements can be nested under each <SERVER> or <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

<URL> must contain the following:

- One <NAME> element, described on page 294.
- One or more of the following elements or element-pairs:
  - <ADD_SIGNATURE> element, described on page 295.
  - <SEARCH> and <REPLACE> element-pair, described in "<SEARCH>" on page 296 and in "<REPLACE>" on page 297.

Note

<SEARCH> and <REPLACE> elements always come in pairs: for every <SEARCH> element there must be a <REPLACE> element.

<NAME>

Description

Child element of <URL>. Defines the name of the URL.

Usage

- One, and only one, <NAME> element must be nested under each <URL> element.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

<ADD_SIGNATURE>

Description

Child element of <URL>. Defines the injection of the IAG unique signature by defining a data string, where the signature is added before or after the string.

Usage

- An unlimited number of <ADD_SIGNATURE> elements can be nested under each <URL> element.
- You can use plain text or regular expressions to define the data string where the signature is added. To use regular expressions, apply the `mode` attribute described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional.
location	before, after	Mandatory. Location where the signature is added, in relation to the string you define in this element.
mode	regex	Optional. Indicates that regular expressions are used to define the string.

Tip
In order to edit encoded text, use the Editor to convert from and to Base64, as described in "Editor" on page 40.

Child Elements

None.

<SEARCH>

Description

Child element of <URL>. Defines a string to search.

Usage

- An unlimited number of <SEARCH> elements can be nested under each <URL> element.
- For every <SEARCH> element there must be a <REPLACE> element, described in <REPLACE> on page 297.
- You can search for plain text, or use regular expressions. To use regular expressions, apply the `mode` attribute as described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. Used when an entire script, not only the URL, is searched. **Tip:** In order to edit encoded text, use the Editor to convert from and to base64, as described in "Editor" on page 40.
mode	regex, regexparam	Optional. Indicates that regular expressions are used to define the string. • **regex:** the entire string is replaced. • **regexparam:** some parameters in the string are retained, while other items in the string are replaced. See below for an example of how to use the "regexparam" value.

Example of how the "regexparam" value is used:

This example describes how you use a single search and replace action to identify these strings:

```
ThisWillRemain12_REPLACEME_ThisWillAlsoRemain56
ThisWillRemain22_REPLACEME_ThisWillAlsoRemain66
ThisWillRemain32_REPLACEME_ThisWillAlsoRemain76
```

and replace them with these corresponding strings:

```
*NEW*ThisWillRemain12*NEWTEXT*ThisWillAlsoRemain56*MORENEW*

*NEW*ThisWillRemain22*NEWTEXT*ThisWillAlsoRemain66*MORENEW*

*NEW*ThisWillRemain32*NEWTEXT*ThisWillAlsoRemain76*MORENEW*
```

In order to achieve this, you define the <SEARCH> and <REPLACE> elements as follows:

- In the <SEARCH> element:
  - Enclose the items that will not be changed within parentheses, including the regular expressions used to identify the string.

> **Tip**
> This example demonstrates the use of the `\d+` expression; you can use any regular expression here.

  - Leave the text to be replaced outside the parentheses.

  The above example is implemented using this <SEARCH> element:

```
<SEARCH encoding="" mode="regexparam">
(ThisWillRemain\d+)_REPLACEME_(ThisWillAlsoRemain\d+)
</SEARCH>
```

- In the corresponding <REPLACE> element, type the new text, interspersed by $1, $2... $n, where $1 corresponds to the contents of the first set of parentheses, $2 corresponds to the contents of the second set of parentheses $2, and so on.

  The above example is implemented using this <REPLACE> element:

```
<REPLACE encoding="">*NEW*$1*NEWTEXT*$2*MORENEW*
</REPLACE>
```

Child Elements

None.

<DATA_CHANGE> ➤ <SERVER> ¦ <APPLICATION> ➤ <URL> ➤ <REPLACE>

<REPLACE>

Description

Child element of <URL>. Defines the string that replaces the string defined in <SEARCH>.

Usage

- An unlimited number of <REPLACE> elements can be nested under each <URL> element.

- For every <REPLACE> element there must be a <SEARCH> element, described in <SEARCH> on page 296.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. Used when an entire script, not only the URL, is replaced.

Tip
In order to edit encoded text, use the Editor to convert from and to base64, as described in "Editor" on page 40.

Child Elements

None.

Sample Code: <DATA_CHANGE>

```
<DATA_CHANGE>
     <SERVER>
          <SERVER_NAME mask="">192\.168\.1\.75</SERVER_NAME>
          <PORT>443</PORT>
          <URL>
               <NAME>/whale\.html</NAME>
               <ADD_SIGNATURE encoding="" location="before">test/id/
               user</ADD_SIGNATURE>
               <ADD_SIGNATURE encoding="" location="after"
               mode="regex">
               ows_FileRef=AA(\d+);#</ADD_SIGNATURE>
               <ADD_SIGNATURE encoding="" location="before"
               mode="regex">
               test/\d+/id/user\d+/file\.htm</ADD_SIGNATURE>
          </URL>
     </SERVER>
     <SERVER>
          <SERVER_NAME mask="255.255.255.0">192.168.1.0<SERVER_NAME>
          <URL>
               <NAME>/whale\.html</NAME>
               <SEARCH encoding="">/images/</SEARCH>
               <REPLACE encoding="">http://192.168.1.75/images/
               </REPLACE>
          </URL>
     </SERVER>
     <APPLICATION>
          <APPLICATION_TYPE>OWA2003</APPLICATION_TYPE>
          <URL>
               <NAME>/exchange/.*/Inbox/.*\.EML\?cmd=preview
               </NAME>
               <ADD_SIGNATURE encoding="" location="before">
               /exchweb/bin/redir.asp
               </ADD_SIGNATURE>
          </URL>
          <URL>
               <NAME>/exchange/.*/Inbox/.*\.EML\?cmd=preview</NAME>
               <SEARCH encoding="" mode="regexparam">
               (ThisWillRemain\d+)_REPLACE_(ThisWillAlsoRemain\d+)
               </SEARCH>
               <REPLACE encoding="">*NEW*$1*NEWTEXT*$2*MORENEW*</
               REPLACE>
          </URL>
     </APPLICATION>
</DATA_CHANGE>
```

Cookie Handling

You can configure the AAP configuration file to manipulate the following types of cookie headers in responses:

- Set-Cookie
- Set-Cookie2

Note

In the following sections, references to Set-Cookie headers apply to both Set-Cookie and Set-Cookie2 header types.

You can remove cookies from these headers, and add, remove, or change any of the cookie attributes, thus changing the application's behavior. You can implement the changes for a specific application, or for an application-type.

The following examples demonstrate a few of the possible changes enabled via cookie manipulation; you can use this capability to implement any other changes according to your needs.

- Adding a `Discard` attribute to a cookie makes it non-persistent, while you can add an `Expires` attribute to make the cookie persistent.
- Adding a `Secure` attribute turns the cookie into a secure cookie, which will not be sent over an HTTP connection, only over HTTPS.

Cookie manipulation is defined in the AAP configuration file in the element <COOKIES_HANDLING>, described below.

Tip

- For a full description of cookie headers, refer to RFC2965.
- For information on troubleshooting cookies that contain a "path" or "domain" attribute, refer to "Maintaining Original Cookie Names" on page 356.

<COOKIES_HANDLING>

Description

Defines the handling of cookie in responses.

Usage

Only one <COOKIES_HANDLING> element can reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <COOKIES_HANDLING> can contain an unlimited number of <SERVER> elements, described on page 301.
- <COOKIES_HANDLING> can contain an unlimited number of <APPLICATION> elements, described on page 301.

<COOKIES_HANDLING> ➤ <SERVER>

<SERVER>

Description

Child element of <COOKIES_HANDLING>. Defines the manipulation of cookies sent by the defined server.

Usage

An unlimited number of <SERVER> elements can be nested under <COOKIES_HANDLING>.

Attributes & Attribute Values

None.

Child Elements

Each <SERVER> element must contain the following:

- One <SERVER_NAME> element, described on page 302.
- One or more <Set-Cookie> elements, described on page 304.

In addition, each <SERVER> element can optionally contain one each of the following:

- For each <SERVER_NAME> element, a <PORT> element, described on page 303.
- <URL> element, described on page 303.

<COOKIES_HANDLING> ➤ <APPLICATION>

<APPLICATION>

Description

Child element of <COOKIES_HANDLING>. Defines the manipulation of cookies sent by the defined application-type.

Usage

An unlimited number of <APPLICATION> elements can be nested under <COOKIES_HANDLING>.

Attributes & Attribute Values

None.

Child Elements

Each <APPLICATION> element must contain the following:

- One <APPLICATION_TYPE> element, described on page 303.
- One or more <Set-Cookie> elements, described on page 304.

In addition, each <APPLICATION> element can optionally contain one <URL> element, described on page 303.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <SERVER_NAME>

<SERVER_NAME>

Description

Child element of <SERVER>. Name of the server from which the manipulated cookie or cookies is sent.

Usage

- One, and only one, <SERVER_NAME> element must be nested under each <SERVER> element.
- Can be defined using one of the following methods:
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the “mask” attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

    Note: In this syntax, enter the actual IP address, not regular expressions.

Tip

For examples of the use of both methods in a <SERVER_NAME> element, refer to “Sample Code: <HEADER_FILTER>” on page 320

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <PORT>

<PORT>

Description

Child element of <SERVER>; applicable only if a <SERVER_NAME> element is used. Port number of the server from which the manipulated cookie or cookies is sent.

Usage

- Optional.
- One, and only one, <PORT> element can be nested under each <SERVER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<COOKIES_HANDLING> ➤ <APPLICATION> ➤ <APPLICATION_TYPE>

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Type of application from which the manipulated cookie or cookies is sent.

Note

For instructions on where you can find the name of the application-type, as it should be entered here, refer to “Finding the Application-Type” on page 44.

Usage

- One, and only one, <APPLICATION_TYPE> element must be nested under each <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ¦ <APPLICATION> ➤ <URL>

<URL>

Description

Child element of <SERVER> or of <APPLICATION>. URL of the server from which the manipulated cookie or cookies is sent.

Usage

- Optional.
- One, and only one, <URL> element can be nested under each <SERVER> or <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ¦ <APPLICATION> ➤ <Set-Cookie>

<Set-Cookie>

Description

Child element of <SERVER> or of <APPLICATION>. Defines the Set-Cookie header that is manipulated and the manipulation.

Usage

- At least one <Set-Cookie> element must be nested under each <SERVER> or <APPLICATION> element.
- An unlimited number of <Set-Cookie> elements can be nested under each <SERVER> or <APPLICATION> element.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove the entire header.

Child Elements

<Set-Cookie> must contain one <NAME> element, described on page 305.

The elements listed below define the actual manipulation on the cookie. <Set-Cookie> can contain a number of different elements, defining a number of manipulations on one cookie.

- <Comment>, described on page 306.
- <CommentURL>, described on page 306.
- <Discard>, described on page 307.
- <Domain>, described on page 308.
- <Max-Age>, described on page 309.
- <Path>, described on page 309.
- <Port>, described on page 310.
- <Secure>, described on page 311.
- <Version>, described on page 311.
- <Expires>, described on page 312.

Note
The element <VALUE> is not used at this time.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <Set-Cookie> ➤ <NAME>

<NAME>

Description

Child element of <Set-Cookie>. Defines the name of the cookie that is manipulated.

Usage

- One, and only one, <NAME> element must be nested under <Set-Cookie>.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <Set-Cookie> ➤ <Comment>

<Comment>

Description

Child element of <Set-Cookie>. Adds, removes, or changes a Comment attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <Comment> element can be nested under <Set-Cookie>.
- Characters in the value of this element must be in UTF-encoding (RFC2279).
- To add a Comment attribute to the Set-Cookie header, or to change the value of the existing attribute, enter the required value.
- To remove an existing Comment attribute from the Set-Cookie header, use the optional "remove" attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Comment attribute from the Set-Cookie header.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <Set-Cookie> ➤ <CommentURL>

<CommentURL>

Description

Child element of <Set-Cookie>. Adds, removes, or changes a CommentURL attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <CommentURL> element can be nested under <Set-Cookie>.

- Takes an HTTP URL.
- To add a CommentURL attribute to the Set-Cookie header, or to change the value of an existing attribute, enter the required value.
- To remove an existing CommentURL attribute from the header, use the optional “remove” attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing CommentURL attribute from the Set-Cookie header.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <Set-Cookie> ➤ <Discard>

<Discard>

Description

Child element of <Set-Cookie>. Adds or removes a Discard attribute to or from the Set-Cookie header.

Usage

- Optional.
- Only one <Discard> element can be nested under <Set-Cookie>.
- If the <Discard> element contains any value, a Discard attribute is added to the Set-Cookie header.
- To remove an existing Discard attribute from the header, use the optional “remove” attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Discard attribute from the Set-Cookie header.

Child Elements

None.

<Domain>

Description

Child element of <Set-Cookie>. Adds or removes a Domain attribute to or from the Set-Cookie header.

> **Tip**
> For an example of the use of the <Domain> element in the AAP configuration file, refer to "Sample Code: <COOKIES_HANDLING>" on page 313.

Usage

- Optional.
- Only one <Domain> element can be nested under <Set-Cookie>.
- To add a Domain attribute to the Set-Cookie header, or to change the value of an existing attribute, enter the required value.
- If the specified value does not start with a dot, the user agent adds a leading dot to the domain name.
- To remove an existing Domain attribute from the header, use the optional "remove" attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Domain attribute from the Set-Cookie header.

None.

Child Elements

None.

<Max-Age>

Description

Child element of <Set-Cookie>. Adds, removes, or changes a Max-Age attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <Max-Age> element can be nested under <Set-Cookie>.
- To add a Max-Age attribute to the Set-Cookie header, or to change the value of the existing attribute, enter the required value. The value is delta-seconds, and must be a decimal non-negative integer.
- To remove an existing Max-Age attribute from the Set-Cookie header, use the optional “remove” attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Max-Age attribute from the Set-Cookie header.

Child Elements

None.

<Path>

Description

Child element of <Set-Cookie>. Adds, removes, or changes a Path attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <Path> element can be nested under <Set-Cookie>.
- To add a Path attribute to the Set-Cookie header, or to change the value of the existing attribute, enter the required value.
- To remove an existing Path attribute from the Set-Cookie header, use the optional “remove” attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Path attribute from the Set-Cookie header.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <Set-Cookie> ➤ <Port>

<Port>

Description

Child element of <Set-Cookie>. Adds, removes, or changes a Port attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <Port> element can be nested under <Set-Cookie>.
- To add a Port attribute to the Set-Cookie header, or to change the value of the existing attribute, enter the required value.
- Takes a list of one or more ports. The port list must be in quotation marks, even if there is only one port number in the list:
  `Port[="portlist"]`
- To remove an existing Port attribute from the Set-Cookie header, use the optional “remove” attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Port attribute from the Set-Cookie header.

Child Elements

None.

<Secure>

Description

Child element of <Set-Cookie>. Adds or removes a Secure attribute to or from the Set-Cookie header.

Tip

For an example of the use of the <Secure> element in the AAP configuration file, refer to "Sample Code: <COOKIES_HANDLING>" on page 313.

Usage

- Optional.
- Only one <Secure> element can be nested under <Set-Cookie>.
- If the <Secure> element contains any value, a Secure attribute is added to the Set-Cookie header.
- To remove an existing Secure attribute from the header, use the optional "remove" attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Secure attribute from the Set-Cookie header.

Child Elements

None.

<Version>

Description

Child element of <Set-Cookie>. Adds, removes, or changes a Version attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <Version> element can be nested under <Set-Cookie>.

- To add a Version attribute to the Set-Cookie header, or to change the value of the existing attribute, enter the required value. The value must be a decimal integer.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Version attribute from the Set-Cookie header.

Child Elements

None.

<COOKIES_HANDLING> ➤ <SERVER> ➤ <Set-Cookie> ➤ <Expires>

<Expires>

Description

Child element of <Set-Cookie>. Adds, removes, or changes an Expires attribute in the Set-Cookie header.

Usage

- Optional.
- Only one <Expires> element can be nested under <Set-Cookie>.
- To add an Expires attribute to the Set-Cookie header, or to change the value of an existing Expires attribute, configure the value of the <Expires> element in one of the following, using one of the following syntax options:
  - A date, in the following format:
    `<Expires>Sun, 15-Feb-2003 10:50:00 GMT</Expires>`
    This element sets the value of the Expires attribute, and the cookie expiration time, to:
    Sun, 15-Feb-2003 10:50:00 GMT
  - Number of seconds after which the cookie will expire.
    For example, to set the value of the Expires attribute to one hour, set the value of the Expires attribute to 3,600 seconds:
    `<Expires>3600<Expires>`
- To remove an existing Expires attribute from the header, use the optional “remove” attribute, described below.

Attributes & Attribute Values

Attribute	Values	Type/Comments
remove	true, false	Optional. Used to remove an existing Expires attribute from the Set-Cookie header.

Child Elements

None.

Sample Code: <COOKIES_HANDLING>

```
// The following sample demonstrates how you can turn a domain
// cookie into a regular cookie. In this example, the filter will
// remove the Domain attribute from the cookie my_cookie, which
// can arrive from any server in the domain .whalecom.com, and the
// cookie will be sent to the browser without this attribute. When
// the browser sends this cookie to the filter, the filter will use
// the value of the <Domain> element defined here to determine
// which domain should receive the cookie.
// This example also uses the <Secure> element to add a Secure
// attribute to my_cookie, making it a secure cookie.

<COOKIES_HANDLING>
     <SERVER>
          <SERVER_NAME mask=””>.*\.whalecom\.com</SERVER_NAME>
          <Set-Cookie>
               <NAME>my_cookie</NAME>
               <Domain remove="true">.whalecom.com</Domain>
               <Secure remove="">true</Secure>
          </Set-Cookie>
     </SERVER>
</COOKIES_HANDLING>
```

Header Filtering

You can configure the AAP to filter headers in requests, so that the defined headers are sent only to the configured applications or application-types. Header filtering is defined in the element <HEADER_FILTER>, described below.

<HEADER_FILTER>

Description

Defines which headers are filtered, and the applications to which each header is sent.

Usage

Only one <HEADER_FILTER> element can reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

<HEADER_FILTER> must contain one, and only one, <INCOMING> element, described below.

<HEADER_FILTER> ➤ <INCOMING>

<INCOMING>

Description

Child element of <HEADER_FILTER>. Determines the filtering of request headers.

Usage

One, and only one, <INCOMING> element must be nested under <HEADER_FILTER>.

Attributes & Attribute Values

None.

Child Elements

<INCOMING> can contain an unlimited number of <HEADER> elements, described below.

<HEADER>

Description

Child element of <INCOMING>. Defines the header that is filtered and the applications to which the header is sent.

Usage

- At least one <HEADER> element must be nested under <INCOMING>.
- An unlimited number of <HEADER> elements can be nested under <INCOMING>.

Attributes & Attribute Values

None.

Child Elements

<HEADER> must contain the following:

- One <NAME> element, described on page 315.
- One or more <APPLICATION> elements, described on page 316.

<NAME>

Description

Child element of <HEADER>. Defines the header to be filtered.

Usage

- One <NAME> element must be nested under <HEADER>.
- Case insensitive.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION>

Description

Child element of <HEADER>. Defines the application or applications, or application-type or types, to which the header defined in <NAME>, above, is sent.

Usage

- At least one <APPLICATION> element must be nested under <HEADER>.
- An unlimited number of <APPLICATION> elements can be nested under <HEADER>.

Attributes & Attribute Values

None.

Child Elements

<APPLICATION> must contain at least one of the following:

- <SERVER_NAME> element, described on page 316.
- <APPLICATION_TYPE> element, described on page 318.

It can contain a combination of both <SERVER_NAME> and <APPLICATION_TYPE> elements.

<APPLICATION> can also contain any of the following optional elements:

- For each <SERVER_NAME> element, a <PORT> element, described on page 317.
- <VALUE>, described on page 318.
- <URL>, described on page 319.

<SERVER_NAME>

Description

Child element of <APPLICATION>. Defines a server or a subnet to which the header is sent.

Usage

- One, and only one. <SERVER_NAME> element must be nested under <APPLICATION>.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.

- Subnet, using the “mask” attribute. For example:

  For subnet class A:
  ```
  <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
  ```
  For subnet class B:
  ```
  <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
  ```
  For subnet class C:
  ```
  <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
  ```
  Note: In this syntax, enter the actual IP address, not regular expressions.

> **Tip**
> For examples of the use of both methods in a <SERVER_NAME> element, refer to “Sample Code: <HEADER_FILTER>” on page 320.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<HEADER_FILTER> ➤ <INCOMING> ➤ <HEADER> ➤ <APPLICATION> ➤ <PORT>

<PORT>

Description

Child element of <APPLICATION>; applicable only if a <SERVER_NAME> element is used. Specifies the port number to which the header is sent.

For example: to define that, on the server defined in <SERVER_NAME>, the header will only be sent to port 80, add the following element:

```
<PORT>80</PORT>
```

Usage

- Optional.
- Only one <PORT> element can be nested under <APPLICATION>.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Defines the type of application to which the header is sent.

Note

For instructions on where you can find the name of the application-type, as it should be entered here, refer to "Finding the Application-Type" on page 44.

Usage

One, and only one, <APPLICATION_TYPE> element must be nested under each <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<VALUE>

Description

Child element of <APPLICATION>. Only if the value defined here is found in the header, the header is sent to the application.

Usage

- Optional.
- Only one <VALUE> element can be nested under <APPLICATION>.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

<URL>

Description

Child element of <APPLICATION>. Defines the URL directory that specifies the application.

For example: to define that, on the server defined in <SERVER_NAME>, the header will only be sent to URLs beginning with `header_URLs`, add the following element:

```
<URL>/header_URLs/.*</URL>
```

Usage

- Optional.
- Only one <URL> element can be nested under <APPLICATION>.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <HEADER_FILTER>

```
// In this example, the header "xxx" is sent only to the following
// applications:
// Application 1:
//   Server: www.x.com
//   URL: /x/y/
// Application 2:
//   Server: www.y.com
//   Port: 6001
//   URL: /z/y/
// Whereas the header "yyy", with the value "zzz", is sent only to
// the servers in the following subnet:
//   IP 201.201.1.5
//   Mask 255.255.0.0
//   URL directory /x/y

<HEADER_FILTER>
 <INCOMING>
   <HEADER>
    <NAME>xxx</NAME>
    <APPLICATION>
     <SERVER_NAME>www\.x\.com</SERVER_NAME>
     <URL>/x/y/.*</URL>
    </APPLICATION>
    <APPLICATION>
     <SERVER_NAME>www\.y\.com</SERVER_NAME>
     <PORT>6001</PORT>
     <URL>/z/y/.*</URL>
    </APPLICATION>
   </HEADER>
   <HEADER>
    <NAME>yyy</NAME>
    <APPLICATION>
     <VALUE>zzz</VALUE>
     <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
     <URL>/x/y/.*</URL>
    </APPLICATION>
   </HEADER>
 </INCOMING>
</HEADER_FILTER>
```

Excluding Servers from the Manipulation Process

This section describes how you define servers so that they are not manipulated by the IAG HAT mechanism. **For example**: if you use regular expressions to define an application, but want to exclude a specific server from that application.

<DONT_SIGN_SERVERS>

Description

Defines the servers that are excluded from the manipulation process.

Usage

Only one <DONT_SIGN_SERVERS> element can reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

<DONT_SIGN_SERVERS> must contain one or more <SERVER> elements, described below.

<DONT_SIGN_SERVERS> ➤ <SERVER>

<SERVER>

Description

Child element of <DONT_SIGN_SERVERS>. Defines a server that is excluded from the manipulation process.

Usage

- At least one <SERVER> element must be nested under <DONT_SIGN_SERVERS>.
- An unlimited number of <SERVER> elements can be nested under <DONT_SIGN_SERVERS>.

Attributes & Attribute Values

None.

Child Elements

- One and only one <SERVER_NAME> element must be nested under <SERVER>.
- One <PORT> element can optionally be nested under <SERVER>.

<SERVER_NAME>

Description

Child element of <SERVER>. Name of the server that is excluded from the manipulation process.

Usage

- One, and only one, <SERVER_NAME> element must be nested under each <SERVER> element.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the "mask" attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

    Note: In this syntax, enter the actual IP address, not regular expressions.

Tip
For examples of the use of both methods in a <SERVER_NAME> element, refer to "Sample Code: <DONT_SIGN_SERVERS>" on page 323.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<PORT>

Description

Child element of <SERVER>. Specifies the port number, on the server defined in <SERVER_NAME>, that is excluded from the manipulation process.

Usage

- Optional.
- Only one <PORT>element can be nested under each <SERVER> element.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <DONT_SIGN_SERVERS>

```
<DONT_SIGN_SERVERS>
     <SERVER>
          <SERVER_NAME>www\.treemenu\.com</SERVER_NAME>
     </SERVER>
     <SERVER>
          <SERVER_NAME>www\.myserver\.com</SERVER_NAME>
          <PORT>8080</PORT>
     </SERVER>
     <SERVER>
          <SERVER_NAME mask="255.255.255.0">192.168.1.0</
          SERVER_NAME>
     </SERVER>
     <SERVER>
          <SERVER_NAME>www\.gnu\.org</SERVER_NAME>
     </SERVER>
     <SERVER>
          <SERVER_NAME>www\.whalecommunications\.com</SERVER_NAME>
     </SERVER>
</DONT_SIGN_SERVERS>
```

Application Aware Skip Body Parsing

In order to fine-tune IAG performance, you can configure URLs where body parsing is skipped. For example: pages that contain no links. You can also skip body parsing in pages where all the links are relative path URLs, where there is no need for link replacement, since the server name does not appear in the URL.

This section describes how you define the "Skip Body Parsing" option per application-type in the AAP configuration file. If you wish to implement this option per application server, you can do this in the Configuration program, as described in "Skip Body Parsing per Application Server" on page 265.

The definitions in the AAP configuration file are made separately for requests and responses, using the corresponding XML elements:

- For requests, use <URLS_WITHOUT_BODY_PARSING_REQ>, described on page 324.
- For responses, use <URLS_WITHOUT_BODY_PARSING_RES>, described on page 325.

<URLS_WITHOUT_BODY_PARSING_REQ>

Description

Defines the URLs of servers and/or application types where body parsing is skipped in requests.

Usage

Only one <URLS_WITHOUT_BODY_PARSING_REQ> element can reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <URLS_WITHOUT_BODY_PARSING_REQ> must contain at least:
  - One <SERVER> element, described on page 325.

  Or,
  - One <APPLICATION> element, described on page 328.
- <URLS_WITHOUT_BODY_PARSING_REQ> can contain an unlimited number of <SERVER> and <APPLICATION> elements.

<URLS_WITHOUT_BODY_PARSING_RES>

Description

Defines the URLs of servers and/or application types where body parsing is skipped in responses.

Usage

Only one <URLS_WITHOUT_BODY_PARSING_RES> element can reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <URLS_WITHOUT_BODY_PARSING_RES> must contain at least:
  - One <SERVER> element, described on page 325.

  Or,
  - One <APPLICATION> element, described on page 328.
- <URLS_WITHOUT_BODY_PARSING_RES> can contain an unlimited number of <SERVER> and <APPLICATION> elements.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <SERVER>

<SERVER>

Description

Child element of <URLS_WITHOUT_BODY_PARSING_REQ> or of <URLS_WITHOUT_BODY_PARSING_RES>. Defines a server where body parsing in requests or responses is skipped, respectively.

Tip

You can define the "Skip Body Parsing" option for specific URLs directly in the Configuration program, as described in "Skip Body Parsing per Application Server" on page 265.

Usage

- Optional.
- An unlimited number of <SERVER> elements can be nested under <URLS_WITHOUT_BODY_PARSING_REQ> or <URLS_WITHOUT_BODY_PARSING_RES>.

Attributes & Attribute Values

None.

Child Elements

<SERVER> must contain the following:

- One and only one <NAME> element, described on page 326.
- One or more <URL> elements, described on page 327.

Optionally, <SERVER> can also contain one <PORT> element, described on page 327.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <SERVER> ➤ <NAME>

<NAME>

Description

Child element of <SERVER>. Name of the server where body parsing in requests or responses is skipped.

Usage

- One, and only one, <NAME> element must be nested under each <SERVER> element.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the “mask” attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

    Note: In this syntax, enter the actual IP address, not regular expressions.

Tip

For examples of the use of both methods in a <NAME> element, refer to “Sample Code: <URLS_WITHOUT_BODY_PARSING_REQ> and <URLS_WITHOUT_BODY_PARSING_RES>” on page 330.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <SERVER> ➤ <URL>

<URL>

Description

Child element of <SERVER>. Defines the URL of the server where body parsing in requests or responses is skipped.

Usage

- At least one <URL> element must be nested under each <SERVER> element.
- An unlimited number of <URL> elements can be nested under each <SERVER> element.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <SERVER> ➤ <PORT>

<PORT>

Description

Child element of <SERVER>. Specifies the port number, on the server defined in <SERVER>, where body parsing of requests or responses is skipped.

Usage

- Optional.

- Only one <PORT> element can be nested under each <SERVER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <APPLICATION>

<APPLICATION>

Description

Child element of <URLS_WITHOUT_BODY_PARSING_REQ> or of <URLS_WITHOUT_BODY_PARSING_RES>. Defines an application-type where body parsing is skipped in requests or responses, respectively.

Usage

- Optional.
- An unlimited number of <APPLICATION> elements can be nested under <URLS_WITHOUT_BODY_PARSING_REQ> or <URLS_WITHOUT_BODY_PARSING_RES>.

Attributes & Attribute Values

None.

Child Elements

<APPLICATION> must contain the following:

- One and only one <APPLICATION_TYPE>, described on page 328.
- One or more <URL> elements, described on page 329.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <APPLICATION> ➤ <APPLICATION_TYPE>

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Type of application where body parsing in requests or responses is skipped.

Note

For instructions on where you can find the name of the application-type, as it should be entered here, refer to "Finding the Application-Type" on page 44.

Usage

- One, and only one, <APPLICATION_TYPE> element must be nested under each <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<URLS_WITHOUT_BODY_PARSING_REQ> | <URLS_WITHOUT_BODY_PARSING_RES> ➤ <APPLICATION> ➤ <URL>

<URL>

Description

Child element of <APPLICATION>. Defines a URL where body parsing in requests or responses is skipped, for the application-type defined in <APPLICATION_TYPE>.

Usage

- At least one <URL> element must be nested under each <APPLICATION> element.
- An unlimited number of <URL> elements can be nested under each <APPLICATION> element.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <URLS_WITHOUT_BODY_PARSING_REQ> and <URLS_WITHOUT_BODY_PARSING_RES>

```
<URLS_WITHOUT_BODY_PARSING_REQ>
     <SERVER>
          <NAME mask="">192\.168\.1\.20</NAME>
          <PORT>8080</PORT>
          <URL>/scripts/upload\.asp</URL>
          <URL>/upload/</URL>
     </SERVER>
     <SERVER>
          <NAME mask="255.255.255.0">192.168.1.0</NAME>
          <URL>.*/upload.asp</URL>
     </SERVER>
     <APPLICATION>
          <APPLICATION_TYPE>OWA2003</APPLICATION_TYPE>
          <URL>/exchange/[^/\*'"`|]*/deleted items/[^/]+\.asp/
          </URL>
     </APPLICATION>
</URLS_WITHOUT_BODY_PARSING_REQ>

<URLS_WITHOUT_BODY_PARSING_RES>
     <SERVER>
          <NAME mask="">192\.168\.1\.20</NAME>
          <PORT>8080</PORT>
          <URL>/work/documents/.*\.(htm|html)</URL>
     </SERVER>
     <SERVER>
          <NAME mask="255.255.255.0">192.168.1.0</NAME>
          <URL>/management/docs/.*</URL>
          <URL>/home/.*\.(htm|html)</URL>
     </SERVER>
     <APPLICATION>
          <APPLICATION_TYPE>OWA2003</APPLICATION_TYPE>
          <URL>/exchange/.*/Inbox/.*\.EML\?cmd=preview</URL>
     </APPLICATION>
</URLS_WITHOUT_BODY_PARSING_RES>
```

Application Aware Search and Replace in Content

The AAP search and replace engine manipulates absolute URLs, in order to hide link names in body data that is not otherwise handled by the content-specific parser, for example URLs that appear within HTML text tags, or Java comments. The AAP search and replace engine runs on the entire HTTP data in the application or in the trunk, including all tags. It is applicable for responses only.

This section describes how you define the "Search and Replace in Content" option per application-type in the AAP configuration file. If you wish to implement this option per application server, you can do this in the Configuration program, as described in "Search and Replace in Content per Application Server" on page 269.

<URLS_WITH_SEARCH_AND_REPLACE_RES>

Description

Defines the AAP search and replace engine.

Usage

Only one <URLS_WITH_SEARCH_AND_REPLACE_RES> element can reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <URLS_WITH_SEARCH_AND_REPLACE_RES> must contain at least:
  - One <SERVER> element, described on page 331.

  Or,
  - One <APPLICATION> element, described on page 334.
- <URLS_WITH_SEARCH_AND_REPLACE_RES> can contain an unlimited number of <SERVER> and <APPLICATION> elements.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <SERVER>

<SERVER>

Description

Child element of <URLS_WITH_SEARCH_AND_REPLACE_RES>. Defines a server on which the AAP search and replace engine runs.

Tip
You can define the "Search and Replace in Content" option for specific URLs directly in the Configuration program, as described in "Search and Replace in Content per Application Server" on page 269.

Usage

- Optional.
- An unlimited number of <SERVER> elements can be nested under <URLS_WITH_SEARCH_AND_REPLACE_RES>.

Attributes & Attribute Values

None.

Child Elements

<SERVER> must contain the following:

- One <NAME> element, described on page 332.
- One or more <URL> elements, described on page 333.

Optionally, <SERVER> can also contain one <PORT> element, described on page 334.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <SERVER> ➤ <NAME>

<NAME>

Description

Child element of <SERVER>. Name of the server on which the AAP search and replace engine runs.

Usage

- One, and only one, <NAME> element must be nested under each <SERVER> element.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the "mask" attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

Note: In this syntax, enter the actual IP address, not regular expressions.

> **Tip**
> For examples of the use of both methods in a <NAME> element, refer to "Sample Code: <URLS_WITH_SEARCH_AND_REPLACE_RES>" on page 336.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <SERVER> ➤ <URL>

<URL>

Description

Child element of <SERVER>. Defines the URL of the server on which the AAP search and replace engine runs.

Usage

- At least one <URL>element must be nested under each <SERVER> element.
- An unlimited number of <URL> elements can be nested under each <SERVER> element.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <SERVER> ➤ <PORT>

<PORT>

Description

Child element of <SERVER>. Specifies the port number, on the server defined in <SERVER>, on which the AAP search and replace engine runs.

Usage

- Optional.
- Only one <PORT> element can be nested under each <SERVER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <APPLICATION>

<APPLICATION>

Description

Child element of <URLS_WITH_SEARCH_AND_REPLACE_RES>. Defines an application-type on which the AAP search and replace engine runs.

Usage

- Optional.
- An unlimited number of <APPLICATION> elements can be nested under <URLS_WITH_SEARCH_AND_REPLACE_RES>.

Attributes & Attribute Values

None.

Child Elements

<APPLICATION> must contain the following:

- One <APPLICATION_TYPE> element, described on page 335.
- One or more <URL> elements, described on page 335.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <APPLICATION> ➤ <APPLICATION_TYPE>

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Type of application on which the AAP search and replace engine runs.

Note

For instructions on where you can find the name of the application-type, as it should be entered here, refer to "Finding the Application-Type" on page 44.

Usage

- One, and only one, <APPLICATION_TYPE> element must be nested under each <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<URLS_WITH_SEARCH_AND_REPLACE_RES> ➤ <APPLICATION> ➤ <URL>

<URL>

Description

Child element of <APPLICATION>. Defines a URL on which the AAP search and replace engine runs, for the application-type defined in <APPLICATION_TYPE>.

Usage

- At least one <URL> element must be nested under each <APPLICATION> element.
- An unlimited number of <URL> elements can be nested under each <APPLICATION> element.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <URLS_WITH_SEARCH_AND_REPLACE_RES>

```
<URLS_WITH_SEARCH_AND_REPLACE_RES>
     <SERVER>
          <NAME>localhost</NAME>
          <URL>/internalsite/sslvpnclient\.asp.*</URL>
          <URL>/internalsite/validatecontinue\.asp</URL>
          <PORT>6001</PORT>
     </SERVER>
     <SERVER>
          <NAME>127\.0\.0\.1</NAME>
          <URL>/internalsite/sslvpnclient\.asp.*</URL>
          <URL>/internalsite/validatecontinue\.asp</URL>
          <PORT>6001</PORT>
     </SERVER>
     <SERVER>
          <NAME mask="255.255.255.0">192.168.1.0</NAME>
          <URL>/comments/.*</URL>
     </SERVER>
     <APPLICATION>
          <APPLICATION_TYPE>OWA2003</APPLICATION_TYPE>
          <URL>/exchange/.*/Inbox/.*\.EML\?cmd=preview</URL>
     </APPLICATION>
     <APPLICATION>
          <APPLICATION_TYPE>OWA2003SP1</APPLICATION_TYPE>
          <URL>/exchange/.*/Inbox/.*\.EML\?cmd=preview</URL>
     </APPLICATION>
     <APPLICATION>
          <APPLICATION_TYPE>SharePoint2003</APPLICATION_TYPE>
          <URL>.*/(search|morinfo|AclVer|SurvEdit|newsbweb|
          listedit|mngdisc|storman|newMWS|DispForm|EditForm)\.aspx.*
          </URL>
          <URL>.*/_vti_bin/_vti_aut/author\.dll</URL>
     </APPLICATION>
</URLS_WITH_SEARCH_AND_REPLACE_RES>
```

Troubleshooting the Application Access Portal

This section describes the following aspects of troubleshooting the AAP:

- Some applications send links that are absolute URLs, which cannot be signed by the AAP on their way out, such as dynamic links created by a web application, or links within unmanipulated formats such as PDF. Other links are unsigned by the AAP since the application that sends them runs outside a browser, such as browser-embedded applications. For these types off links, use the "HAT via Proxy" solution, described on page 338.
- "Dealing with Unrecognized URLs in Responses" on page 339 describes the options available to you for dealing with URLs that are not manipulated by the AAP in responses, and are, therefore, unrecognized when requested by the user.
- Some applications might require the manipulation of content-types that are not configured by default to be handled by the content parsers. You can add such content-types to the AAP configuration file, as described in "Adding Content-Types to the Content Parsers" on page 343.
- You can configure token-to-token replacement for header values, in headers that contain links that are not manipulated by the header parsers, as described in "Header Token-to-Token Replacement" on page 347.
- If the client-side script does not recognize a cookie, it might be caused by the AAP manipulation. For details, refer to "Maintaining Original Cookie Names" on page 356.
- "Error Reporting" on page 357 describes the AAP error reporting options.

HAT via Proxy

Use this solution for applications whose responses include absolute URLs that cannot be signed by the AAP. This includes web applications that create dynamic links, or links included in unmanipulated formats such as PDF. Use it also for applications that run outside a browser, that is, Client/Server and Legacy Applications and Browser-Embedded Applications.

The "HAT via Proxy" solution is implemented by a dedicated application that you add to the portal, the Enhanced HAT application. Once this application is added to the trunk, when an end-user accesses the portal, the following occurs:

- The SSL Wrapper client launches on the endpoint computer.

> **Note**
> For a full description, refer to the *Intelligent Application Gateway User Guide*, to Chapter 6: "SSL Wrapper".

- The SSL Wrapper client adds the addresses of the servers that are configured in the Enhanced HAT application to the Hosts file on the endpoint computer.
- The SSL Wrapper client listens on these addresses. When the browser sends a request for one of those addresses, the SSL Wrapper client redirects the request to the IAG filter. The filter manipulates the link, and then sends a redirect request to the originating client.
- The browser sends the request again. This time, however, the URL is signed, and, therefore, can be resolved and passed on to the target server. This is transparent to the end-user, and there is no change in the way the link is displayed on the endpoint computer.

To configure "HAT by Proxy":

1. Using the Add Application Wizard, from the "Client/Server and Legacy Applications" drop-down list, add the Enhanced HAT application to the trunk.
2. In the step "Server Settings":
   a) Define the servers that send absolute URLs that cannot be manipulated, which you wish to resolve using the "HAT via Proxy" solution. Define the servers using their DNS hostnames.
   b) Define the port number on which the SSL Wrapper client listens to requests for the servers you configure here.

> **Note**
> On Mac OS X operating systems, non-root accounts cannot run this application on any port between 0–1024, including port 80. If the application is accessed from computers running Mac OS X operating systems by non-root accounts, make sure to change the default port number.

c) Make sure that the option “Launch automatically on Start” is activated.

> **Note**
> If the Enhanced HAT application is not launched, the HAT via Proxy solution will not be implemented.

3. Complete the remaining wizard steps to add the application to the trunk.
4. Using the Add Application Wizard, from the "Web Applications" drop-down list, add the Generic Web App application to the trunk.
5. In the step "Web Servers", define the same servers that you defined for the Enhanced HAT application in step 2a.
6. Complete the remaining wizard steps to add the application to the trunk.

   Requests for the servers you configured for the Enhanced HAT and Generic Web App applications here will be transparently redirected with signed URLs.

Dealing with Unrecognized URLs in Responses

Sometimes, absolute links are sent to the browser without being manipulated on their way out. For example: dynamic links, links within Flash or PDF files, links that are sent from applications that run outside the browser, and more. Trying to activate an “unmanipulated” link will result in a request error.

> **Tip**
> In order to identify the link that caused the error, you can use any of the error reporting mechanisms described in “Error Reporting” on page 357.

You can prevent such errors by using one of the following options:

- Manual URL Replacement redirects or reroutes the unrecognized URL to a pre-defined location, from where the request is processed by the filter. This method is especially useful for links that appear in

content that is not manipulated by the system, such as Flash or PDF files. Configuration of this option is described in “Manual URL Replacement” on page 272.

- If the error is caused by a link that is contained in a JavaScript command which is not parsed by the JavaScript parser, you need to configure the parser to parse this type of commands, as described in “Configuring the JavaScript Parser” on page 287.
- Use token-to-token replacement if the error is caused by a link in JavaScript, which doesn’t appear directly after the operation character. For details, refer to “Data Change” on page 289.
- Responses that the server sends in encoded form cannot be manipulated by the content parsers. By default, the AAP configuration file is configured to remove content encoding from content-types that might contain links. You can add content-types to the types configured by default, as described in “Removing Encoding from Encoded Content” on page 340.
- If the request that contains the unrecognized URL includes a Referer header, it is possible to reroute the URL to the server from where the link originated, based in the content of the header, as described in “Rerouting Unrecognized URLs Based on a Referer Header” on page 342.
- If the unrecognized URL is contained in content-type that is not configured to be parsed by any of the content parsers, you can configure the AAP configuration file so that this type of content will be parsed. Refer to “Adding Content-Types to the Content Parsers” on page 343.
- If the request header contains—within the header value—links that are not manipulated by the request header parser, you can configure the AAP configuration file to replace the header values, as described in “Header Token-to-Token Replacement” on page 347.

Removing Encoding from Encoded Content

By default, the AAP is configured so that content encoding is removed from content that might contain links, and the content is sent to the browser in an un-encoded form. This configuration includes files ending with the following extensions: `html`; `htm`; `htc`; `css`; `js`; `xsl`; `xml`; `cgi`; `asp`; files with no extension.

If any of your application servers send compressed files of a type where content encoding is not removed by default, you can add the file type to the AAP configuration file, in the element <REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>, described on page 341.

Note

- Before you make any changes to the AAP configuration file, be sure to read the information provided in "Before You Begin" on page 279; it holds important information you should note.
- We recommend that you do not remove any of the existing content-types from the element <REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>.
- For each content-type for which you configure the removal of content encoding, make sure that the content-type is handled by the response parser. For details, refer to "Adding Content-Types to the Content Parsers" on page 343.

<REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>

Description

Defines types of content where content encoding is removed, and the content is sent to the browser in an un-encoded form.

Usage

Only one <REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT> element can be configured in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

<REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT> can contain an unlimited number of <EXTENSION> elements, described on page 341.

<REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT> ➤ <EXTENSION>

<EXTENSION>

Description

Child element of <REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>. Defines a content-type where content encoding is removed.

Usage

An unlimited number of <EXTENSION> elements can be nested under <REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>.

Attributes & Attribute Values

None.

Child Elements

None.

Sample Code: <REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>

```
<REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>
     <EXTENSION>html</EXTENSION>
     <EXTENSION>htm</EXTENSION>
     <EXTENSION>htc</EXTENSION>
     <EXTENSION>css</EXTENSION>
     <EXTENSION>js</EXTENSION>
     <EXTENSION>xsl</EXTENSION>
     <EXTENSION>xml</EXTENSION>
     <EXTENSION>cgi</EXTENSION>
     <EXTENSION>asp</EXTENSION>
     <EXTENSION>no_ext</EXTENSION>
</REMOVE_CONTENT_ENCODING_FROM_URL_WITH_EXT>
```

Rerouting Unrecognized URLs Based on a Referer Header

The Referer is an optional request header that specifies the URL of the document (or element within the document) from which the link in the request was obtained. For example:

```
Referer: http://www.whalecom.com/Overview.html
```

You can configure the IAG so that if a requested URL is unrecognized, and the request contains a Referer header, in which the URL **is** recognized, the requested URL will be rerouted to the URL that is specified in the header. In our example, to the following location:

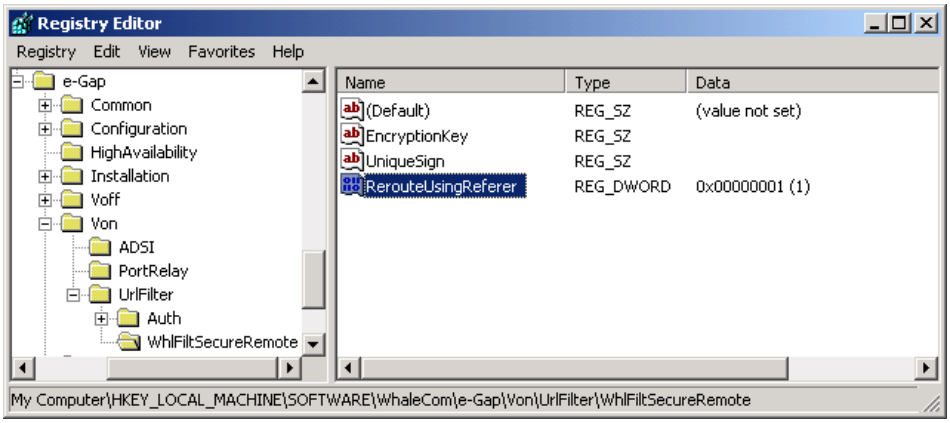
```
www.whalecom.com
```

Note

This option should be used only if the trunk is configured to use an authentication scheme, to eliminate the possibility of unauthorized users accessing the internal applications via the Referer header. For information on authentication, refer to the section "Authentication" on page 81.

To configure the IAG to reroute unrecognized URLs using the Referer header:

1. At the IAG, access the following Registry key:
   …\WhaleCom\e-Gap\Von\UrlFilter\WhlFiltSecureRemote
2. Create a new DWORD value named `RerouteUsingReferer`, and set the value to 1.



3. At the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

   Once the configuration is activated, unrecognized URLs in requests that contain a Referer header will automatically be rerouted to the URL from where they were referred.

Adding Content-Types to the Content Parsers

The AAP configuration file determines what types of content are manipulated by the HAT within the body of both responses and requests, and which parser will handle each content-type, in the tag <PARSERS_CONFIGURATION>. Content-type is defined in the value of the HTTP header "Content-Type".

Note

Before you make any changes to the AAP configuration file, be sure to read the information provided in "Before You Begin" on page 279; it holds important information you should note.

Each content-type that is defined in <PARSERS_CONFIGURATION> must also be on the global list of content-types that are handled by the filter, which is defined in the Application Customization tab.

If your application requires the manipulation of content of a type that is not configured by default to be handled by the content parsers, you can configure the required content-type as follows:

- Add the content-type to the global list, in the Application Customization tab. For details, refer to "Global Content-Type List" on page 210.

- Add the content-type to the parser configuration defined in the AAP configuration file, in the tag <PARSERS_CONFIGURATION>, as described in <PARSERS_CONFIGURATION> on page 344.

Tip

You can use the configuration in <PARSERS_CONFIGURATION> to create a subset of the global content-type list, for performance tuning. Content-types that are configured in the global list, and are not configured in the AAP configuration file, will not be manipulated by the HAT.

<PARSERS_CONFIGURATION>

Note

Whenever you configure an additional content-type in <PARSERS_CONFIGURATION>, make sure to also configure the same content-type in the global content-type list. If you fail to configure the new content-type in the global list, it will not be manipulated by the HAT even though it is configured in <PARSERS_CONFIGURATION>.

Description

Defines the content-types that are handled by the response and request content parsers.

Usage

One, and only one, <PARSERS_CONFIGURATION> element must reside in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

- <PARSERS_CONFIGURATION> can contain one <RESPONSE> element, described on page 345.
- <PARSERS_CONFIGURATION> can contain one <REQUEST> element, described on page 346.

Sample Code: <PARSERS_CONFIGURATION>

```
<PARSERS_CONFIGURATION>
     <RESPONSE>
          <CONTENT_TYPE parser="PARSER_HTML">text/html</
CONTENT_TYPE>
          <CONTENT_TYPE parser="PARSER_HTML">text/x-component
          </CONTENT_TYPE>
          <CONTENT_TYPE parser="PARSER_CSS">text/css</
CONTENT_TYPE>
          <CONTENT_TYPE parser="PARSER_SAR">text.*</CONTENT_TYPE>
          <CONTENT_TYPE parser="PARSER_JAVASCRIPT">
          application/x-javascript</CONTENT_TYPE>
     </RESPONSE>
     <REQUEST>
          <CONTENT_TYPE parser="PARSER_REQUEST_BODY_SAR">
          application/x-www-form-urlencoded</CONTENT_TYPE>
          <CONTENT_TYPE parser="PARSER_REQUEST_BODY_SAR">text/xml
          </CONTENT_TYPE>
     </REQUEST>
</PARSERS_CONFIGURATION>
```

<PARSERS_CONFIGURATION> ➤ <RESPONSE>

<RESPONSE>

Description

Child element of <PARSERS_CONFIGURATION>. Defines what content-types are handled by each of the content parsers in responses.

Usage

One, and only one, <RESPONSE> element must reside under <PARSERS_CONFIGURATION>.

Attributes & Attribute Values

None.

Child Elements

<RESPONSE> can contain an unlimited number of <CONTENT_TYPE> elements, described on page 346.

<PARSERS_CONFIGURATION> ➤ <REQUEST>

<REQUEST>

Description

Child element of <PARSERS_CONFIGURATION>. Defines what content-types are handled by each of the content parsers in requests.

Usage

One, and only one, <REQUEST> element must reside under <PARSERS_CONFIGURATION>.

Attributes & Attribute Values

None.

Child Elements

<REQUEST> can contain an unlimited number of <CONTENT_TYPE> elements, described on page 346.

<PARSERS_CONFIGURATION> ➤ <RESPONSE> ¦ <REQUEST> ➤ <CONTENT_TYPE>

<CONTENT_TYPE>

Description

Child element of <RESPONSE> and of <REQUEST>. Defines a content-type to be parsed in either responses or requests, respectively, and which parser will parse the content.

Usage

- An unlimited number of <CONTENT_TYPE> elements can be nested under <RESPONSE> and under <REQUEST>.
- Takes regular expressions.
- The order by which content-types are listed is significant, since content is parsed by the first parser on the list that matches it.

  For example:

  If, in responses, you wish HTML content to be parsed by the parser PARSER_HTML, make sure that, under the <RESPONSE> element, the element:

  ```
  <CONTENT_TYPE parser="PARSER_HTML">text/html</CONTENT_TYPE>
  ```

  appears before the element:

  ```
  <CONTENT_TYPE parser="PARSER_SAR">text.*</CONTENT_TYPE>
  ```

Attributes & Attribute Values

Attribute	Values	Type/Comments
parser	Parser name.	Mandatory. Defines the parser that will parse the data defined in the element.

Child Elements

None.

Header Token-to-Token Replacement

You can configure the AAP configuration file to replace header values in headers that contain links that are not manipulated by the header parsers.

For example:

If your application expects to receive a header from the application server, where:

- Header name is “Application1Server”
- Header value is the IP address of the application server, in this example: “192.168.1.166”

The filter will not manipulate this link because in response headers it only manipulates absolute URLs.

In this case, the client side script will receive the server name “192.168.1.166”, and not the external website name, thus exposing the real server name to the outside world. Token-to-token replacement on this header will solve the problem.

The following example shows the configuration of the replacement in the AAP configuration file, in the <HEADER_CHANGE> element, where:

- External website name is “192.168.1.1”, and
- The URL of the server that sent the header is “192.168.1.22”

A description of the <HEADER_CHANGE> element follows.

Note

Before you make any changes to the AAP configuration file, be sure to read the information provided in “Before You Begin” on page 279; it holds important information you should note.

<HEADER_CHANGE>

Description

Searches and replaces header values in requests, responses, or both. Can be applied per server or a number of servers, or per application-type or types.

Usage

- Optional.
- Only one <HEADER_CHANGE> element can be configured in the AAP configuration file.

Attributes & Attribute Values

None.

Child Elements

<HEADER_CHANGE> can contain one <REQUEST> element and one <RESPONSE> element, described below.

<HEADER_CHANGE> ➤ <REQUEST>

<REQUEST>

Description

Child element of <HEADER_CHANGE>. Defines the replacement of header values in requests.

Usage

Only one <REQUEST> element can be nested under <HEADER_CHANGE>.

Attributes & Attribute Values

None.

Child Elements

- <REQUEST> can contain an unlimited number of <SERVER> elements, described on page 349.
- <REQUEST> can contain an unlimited number of <APPLICATION> elements, described on page 350.

<HEADER_CHANGE> ➤ <RESPONSE>

<RESPONSE>

Description

Child element of <HEADER_CHANGE>. Defines the replacement of header values in responses.

Usage

Only one <RESPONSE> element can be nested under <HEADER_CHANGE>.

Attributes & Attribute Values

None.

Child Elements

- <RESPONSE> can contain an unlimited number of <SERVER> elements, described on page 349.
- <RESPONSE> can contain an unlimited number of <APPLICATION> elements, described on page 350.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <SERVER>

<SERVER>

Description

Child element of <REQUEST> and of <RESPONSE>. Defines the server and URL in which data is searched and replaced.

Usage

An unlimited number of <SERVER> elements can be nested under <REQUEST> and under <RESPONSE>.

Attributes & Attribute Values

None.

Child Elements

<SERVER> must contain the following:

- One <SERVER_NAME> element, described on page 350.
- One or more <URL> elements, described on page 352.

Optionally, <SERVER> can also contain one <PORT> element, described on page 351.

<APPLICATION>

Description

Child element of <REQUEST> and of <RESPONSE>. Defines the application-type and URL in which data is searched and replaced.

Usage

An unlimited number of <APPLICATION> elements can be nested under <REQUEST> and under <RESPONSE>.

Attributes & Attribute Values

None.

Child Elements

<APPLICATION> must contain the following:

- One <APPLICATION_TYPE> element, described on page 352.
- One or more <URL> elements, described on page 352.

<SERVER_NAME>

Description

Child element of <SERVER>. Defines the server in which the header value is searched and replaced.

Usage

- One, and only one, <SERVER_NAME> element must be nested under each <SERVER> element.
- Can be defined using one of the following methods:
  - Server name or a range of names, using regular expressions.
  - Subnet, using the “mask” attribute. For example:

    For subnet class A:
    ```
    <SERVER_NAME mask="255.0.0.0">63.0.0.0</SERVER_NAME>
    ```

    For subnet class B:
    ```
    <SERVER_NAME mask="255.255.0.0">160.12.0.0</SERVER_NAME>
    ```

    For subnet class C:
    ```
    <SERVER_NAME mask="255.255.255.0">192.168.1.0</SERVER_NAME>
    ```

    Note: In this syntax, enter the actual IP address, not regular expressions.

> **Tip**
> For examples of the use of both methods in a <SERVER_NAME> element, refer to "Sample Code: <HEADER_CHANGE>" on page 356.

Attributes & Attribute Values

Attribute	Values	Type/Comments
mask	subnet mask	Optional. Used to define a subnet mask.

Child Elements

None.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <SERVER> ➤ <PORT>

<PORT>

Description

Child element of <SERVER>. Specifies the port number, on the server defined in <SERVER_NAME>, on which data is searched and replaced.

Usage

- Optional.
- Only one <PORT> element can be nested under each <SERVER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION_TYPE>

Description

Child element of <APPLICATION>. Specifies the type of application on which data is searched and replaced.

Note

For instructions on where you can find the name of the application-type, as it should be entered here, refer to "Finding the Application-Type" on page 44.

Usage

One, and only one, <APPLICATION_TYPE> element must be nested under each <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

None.

<URL>

Description

Child element of <SERVER> or of <APPLICATION>. Defines the URL and the header where the value is searched and replaced.

Usage

- At least one <URL> element must be nested under each <SERVER> or <APPLICATION> element.
- An unlimited number of <URL> elements can be nested under each <SERVER> or <APPLICATION> element.

Attributes & Attribute Values

None.

Child Elements

<URL> must contain the following:

- One <NAME> element, described on page 353.
- One or more <HEADER> elements, described on page 353.

<NAME>

Description

Child element of <URL>. Specifies the URL that sent the header.

Usage

- One, and only one, <NAME> element must be nested under each <URL> element.
- Takes regular expressions.

Attributes & Attribute Values

None.

Child Elements

None.

<HEADER>

Description

Child element of <URL>. Defines the header where the value is replaced, the value that is replaced, and the replacement value.

Usage

One, and only one, <HEADER> element must be nested under each <URL> element.

Attributes & Attribute Values

None.

Child Elements

<HEADER> must contain the following:

- One <NAME> element, described on page 354.
- One or more <SEARCH> elements, described on page 354.
- One or more <REPLACE> elements, described on page 355.

Note

<SEARCH> and <REPLACE> elements always come in pairs—for every <SEARCH> element there must be a <REPLACE> element.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <SERVER> ¦ <APPLICATION> ➤ <URL> ➤ <HEADER> ➤ <NAME>

<NAME>

Description

Child element of <HEADER>. Defines the name of the header where the value is replaced.

Usage

One, and only one, <NAME> element must be nested under each <HEADER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <SERVER> ¦ <APPLICATION> ➤ <URL> ➤ <HEADER> ➤ <SEARCH>

<SEARCH>

Description

Child element of <HEADER>. Defines the header value that is replaced.

Usage

- At least one <SEARCH> element must be nested under each <HEADER> element.
- An unlimited number of <SEARCH> elements can be nested under each <HEADER> element.
- For every <SEARCH> element there must be a <REPLACE> element, described on page 355.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. Used when an entire script, not only the URL, is searched.

> **Tip**
> In order to edit encoded text, use the Editor to convert from and to base64, as described in “Editor” on page 40.

Child Elements

None.

<HEADER_CHANGE> ➤ <REQUEST> ¦ <RESPONSE> ➤ <SERVER> ¦ <APPLICATION> ➤ <URL> ➤ <HEADER> ➤ <REPLACE>

<REPLACE>

Description

Child element of <HEADER>. Defines the replacement value.

Usage

- At least one <REPLACE> element must be nested under each <HEADER> element.
- An unlimited number of <REPLACE> elements can be nested under each <HEADER> element.
- For every <REPLACE> element there must be a <SEARCH> element, described on page 354.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. Used when an entire script, not only the URL, is searched.

> **Tip**
> In order to edit encoded text, use the Editor to convert from and to base64, as described in “Editor” on page 40.

Child Elements

None.

Sample Code: <HEADER_CHANGE>

```
<HEADER_CHANGE>
     <RESPONSE>
           <SERVER>
              <SERVER_NAME mask="255.255.255.0">192.168.1.06
              </SERVER_NAME>
              <PORT>80</PORT>
              <URL>
                 <NAME>/192\.168\.1\.22</NAME>
                 <HEADER>
                       <NAME>Application1Server</NAME>
                       <SEARCH encoding="">192.168.1.166</SEARCH>
                       <REPLACE encoding="">"192.168.1.1"</REPLACE>
                 </HEADER>
              </URL>
           </SERVER>
     </RESPONSE>
</HEADER_CHANGE>
```

Maintaining Original Cookie Names

When a Set-cookie header contains a “path” or “domain” attribute, the AAP might manipulate the cookie name. In some cases, if a client-side script uses the cookie, it will be unable to find the cookie once its name changes. In this case, you can configure the AAP configuration file so that is does not manipulate the cookie name.

Note

Before you make any changes to the AAP configuration file, be sure to read the information provided in “Before You Begin” on page 279; it holds important information you should note.

Cookie manipulation is handled by the <COOKIES_HANDLING> element, described in “Cookie Handling” on page 300. In order to prevent the manipulation of the cookie name, add the following element in the AAP configuration file, under the <COOKIES_HANDLING> element:

```
<COOKIES_HANDLING>
     <SERVER>
          <SERVER_NAME mask=””>internal_server_name</SERVER_NAME>
          <Set-Cookie remove=””>
               <NAME>cookie_name</NAME>
               <Path remove="true">path</Path>
               <Domain remove="true">internal_domain_name</Domain>
          </Set-Cookie>
     </SERVER>
</COOKIES_HANDLING>
```

The following sample code shows how you configure the AAP so that it does not manipulate the name of the cookie:

Set-Cookie: a=b; Path = /Internal; Domain = .whalecom.com

```
<COOKIES_HANDLING>
     <SERVER>
          <SERVER_NAME mask="">www\.whalecom\.com</SERVER_NAME>
          <Set-Cookie remove="">
               <NAME>a</NAME>
               <Path remove="true">/Internal</Path>
               <Domain remove="true">.whalecom.com</Domain>
          </Set-Cookie>
     </SERVER>
</COOKIES_HANDLING>
```

Error Reporting

When users try to navigate to an unrecognized URL, the filter replies with a “Request Error” message, and denies the user access to the URL. In order to identify the source of the error, that is, the unrecognized URL, you can do one of the following:

- View the event in the Web Monitor, either in the Event Viewer or by generating a log query. The type of message that this error generates is “Can’t Associate URL with Application”. For a full description of the Web Monitor refer to the *Intelligent Application Gateway User Guide*, to “Web Monitor” on page 258.
- Configure a trace, to record the activities that led to the error, as described in “Configuring an Application Access Portal Trace” on page 358.
- Activate a Dump utility, and view the request before and after the filter’s manipulation. Refer to “Activating the Dump Utility” on page 359 for details.

Configuring an Application Access Portal Trace

This section describes the options available to you when you configure a trace for the AAP, and provides you with a sample trace, configured with the values that are recommended for debugging purposes. For a description of the Trace mechanism, refer to the *Intelligent Application Gateway User Guide*, to "Error Logging and Process Tracing" on page 307.

Application Access Portal Trace Options

Trace syntax: `[Trace\WhlFilter\WHLFILTSECUREREMOTE]`

Available trace options (reporters):

- `GENERAL`: general AAP traces.
- `MEMORY_MECHANISM`: reports allocation and enables identification of memory leaks.
- `LOCK_MECHANISM`: reports the start and end of functions; can be used to identify deadlocks.
- `SCRAMBLE_SERVER`: responsible for the signing and resolving of links.
- `GENERAL_MANAGER`: receives requests from the filter and forwards them to the appropriate parsers.
- `PARSER_MANAGER`: manages the specific parser.
- `PARSER_BASE`: base class of each parser.
- `PARSER_REQUEST_HEADER`: parses request headers, identifies unrecognized URLs, rerouted URLs, etc.
- `PARSER_RESPONSE_HEADER`: parses response headers.
- `PARSER_SAR`: Search and Replace parser.
- `PARSER_HTML`: HTML parser; parses responses' HTML buffers.
- `PARSER_CSS`: Cascading Style Sheet parser.
- `PARSER_EXCEPTION`: parses JavaScript code.
- `PARSER_FORM`: parses forms.
- `PARSER_HEADER`: parses header elements that are common to both requests and responses.

A Sample Application Access Portal Trace

Following is a sample trace, which is recommended for use for debugging purposes:

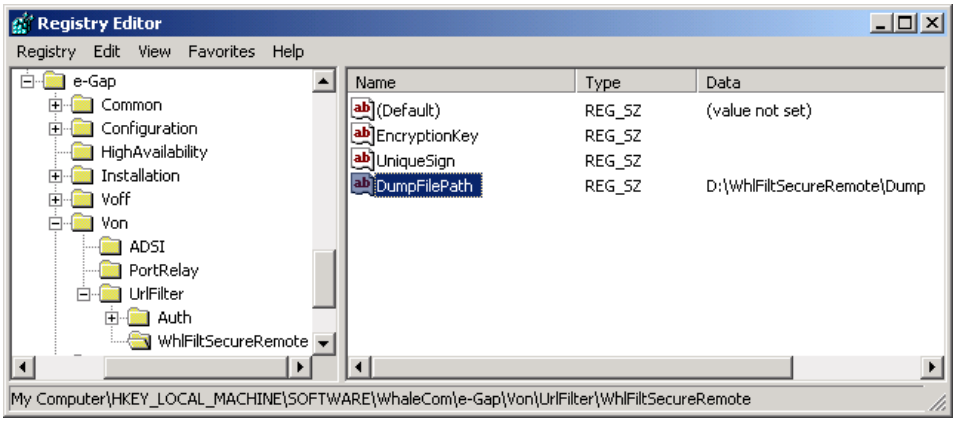
```
[Trace\WhlFilter\WHLFILTSECUREREMOTE]
*=light
PARSER_REQUEST_HEADER=heavy
PARSER_RESPONSE_HEADER=heavy
PARSER_HTML=heavy
GENERAL_MANAGER=xheavy
```

Activating the Dump Utility

Once activated, the Dump utility dumps the entire request data into the location you define during the configuration of the utility. Request data is dumped twice: before and after the filter's manipulation, so that you can keep track of the changes applied by the filter.

To configure and activate the Dump utility:

1. At the IAG, access the following Registry key:
   `…\WhaleCom\e-Gap\Von\UrlFilter\WhlFiltSecureRemote`
2. Create a new String value and name it `DumpFilePath`.
3. Modify the string value data, entering the path of the folder where you wish to dump the files.



4. At the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

Once the configuration is activated, the location you have entered is displayed in the Registry Editor, under Data. The dump files will be sent to the defined folder. The dump file name will match the URL name, except for the characters listed in the following table, which will be replaced as described in the table:

Table 34. Characters Replacement in Dump Files

Character	Replaced by
/	_Whl1_
\	_Whl2_
:	_Whl3_
*	_Whl4_
?	_Whl5_
“	_Whl6_
<	_Whl7_
>	_Whl8_
|	_Whl9_

Chapter 9

Configuring the High Availability Array

Note

All Intelligent Application Gateway (IAG) servers in the High Availability array must use the same encryption key with an identical passphrase. For details, refer to "Creating Encryption Keys" on page 20.

High Availability configuration enables you to deploy identical configuration settings throughout the IAG servers in the array. Once sets of configuration parameters are defined in the first of the IAG servers, Export and Import interfaces assist you to quickly and accurately deploy the configuration across the array, ensuring identical settings for all IAG servers. The identical configuration settings enable each server in the array to take over the functionality of any other IAG server as necessary.

You go through the following stages when configuring a High Availability array:

- (Optional) You can pre-configure the first IAG server using the Service Policy Manager. If you do so, make sure to pre-configure all the IP addresses and port numbers you will use on all trunks throughout the array. Configuration of the Service Policy Manager is described in the *Intelligent Application Gateway Advanced Configuration* guide.
- If you are using the Resonate Central Dispatch™ software as a software-based load balancing tool:
  - Configure the Resonate software, and define IP Persistence scheduling rules.
  - If you are using the Network Connector option, you must unbind the "Whale Network Connector" network connection on all IAG servers in the High Availability array.

Note

To unbind the connection:

- In the Windows desktop of the IAG, click Start, then point to **Settings > Network Connections**, and click **Whale Network Connector**.
- In the Whale Network Connector Properties dialog box, disable the item **Resonate RXP Driver** and click OK.

- On the first IAG server, in the Configuration program, you set up a list of the IAG servers in the High Availability array and define the IP addresses that are unique for each server on the array, as described in "Defining the High Availability Parameters" on page 362.
- When you have completed setting up the High Availability configuration on the first IAG server, you export the entire configuration from that server and import it into all the other IAG servers in the array, as described in "Deploying the High Availability Configuration" on page 365.

Defining the High Availability Parameters

You set up the High Availability configuration on the first IAG server in two stages, in the Configuration program:

- Prepare a list of the IAG servers in the array, as described in "Preparing the List of IAG Servers" on page 362.
- Configure, for each trunk, the relevant IP addresses for each of the listed IAG servers, as described in "Configuring Trunk IP Addresses" on page 364.

Note

If you use the Resonate Central Dispatch software as your load balancing tool, make sure the Central Dispatch site is set up before you set up the High Availability configuration. This will ensure that all the IAG servers in the High Availability array are synchronized.

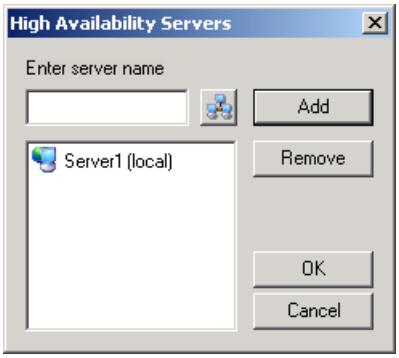
Preparing the List of IAG Servers

On the first IAG server, you prepare a list of all the IAG servers that are part of the High Availability array. You will only be able to deploy the High Availability configuration to the servers that you list here.

To prepare the list of IAG servers:

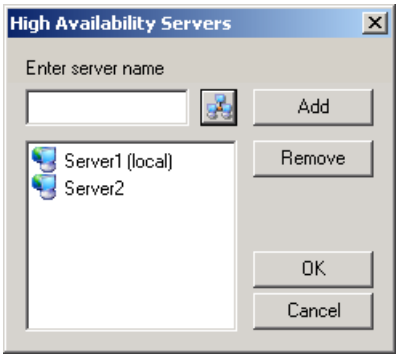
1. In the Configuration program, on the **Admin** menu, click **High Availability Servers...**

   The High Availability Servers dialog box is displayed. The first IAG server on which you are defining the array is listed by default as (local) in the list of servers.



2. To add an IAG server to the list of servers, do one of the following:
   - Enter the server's name in the text box and click Add.
   - Click to browse for the server.

   The server is added to the list of IAG servers.



3. Add all the IAG servers that are part of the High Availability array to the list, and then click OK.

   The High Availability dialog box closes. Proceed to "Configuring Trunk IP Addresses" on page 364.

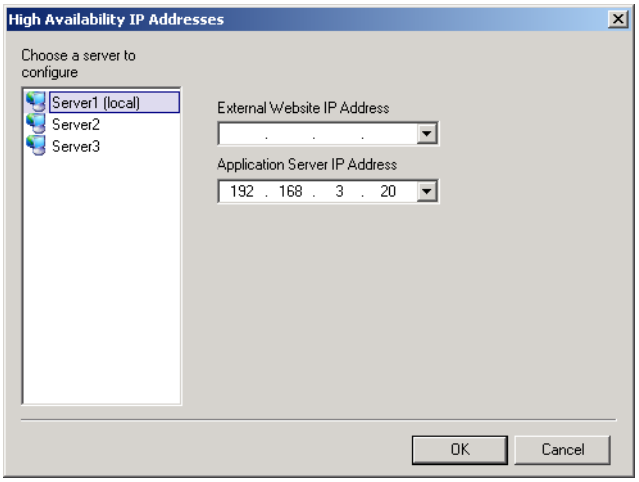
Configuring Trunk IP Addresses

For each trunk, you configure an IP address for each of the IAG servers in the High Availability list. You do this on the first IAG server only. Later, you will deploy the entire High Availability configuration across the array.

To configure the High Availability trunk IP addresses:

1. On the first IAG server, in the Configuration program, select a trunk for which you wish to configure the High Availability option.
2. In the lower-left area of the Configuration pane, next to the "High Availability" option, click Configure...

   The High Availability IP Addresses dialog box is displayed, with the first IAG on which you are defining the array selected and defined as (local) (Server1 in the following example screenshot).
3. In the list of servers, select the IAG server you wish to configure. The types of IP addresses that are displayed vary according to the type of trunk you are configuring; the following is an example of the High Availability configuration interface for a Webmail trunk.



4. Assign the following IP addresses for this trunk:
   - Portal Trunk: External Website
   - Webmail or Basic trunks:
     - External Website
     - Application Server
5. Repeat steps 3–4, as required, to configure the next server on the list. When you finish configuring all the servers that are part of the High Availability array for this trunk, click OK in the High Availability IP Addresses dialog box.

   The High Availability IP Addresses dialog box closes.

6. Repeat steps 1–5 to configure the High Availability option for all the trunks in your IAG configuration.

7. In the Configuration window, click and follow the instructions on the screen to save and activate the configuration.

   The High Availability configuration is saved, and can be deployed across the other IAG servers in the High Availability array, as described in "Deploying the High Availability Configuration" on page 365.

   Once the High Availability configuration is deployed, the definitions you configured in the preceding steps will be reflected in all IAG servers in the array, in the Configuration program.

Deploying the High Availability Configuration

Once you have completed all definitions in the first IAG server, you deploy them across the High Availability array, as follows:

- From the first IAG server, export the configuration settings into a project in a location that is accessible to all the other servers in the array, as described in "Exporting a Configuration Project" on page 365.
- Import the configuration into all the other servers in the array, as described in "Importing a Configuration Project" on page 366.

Note

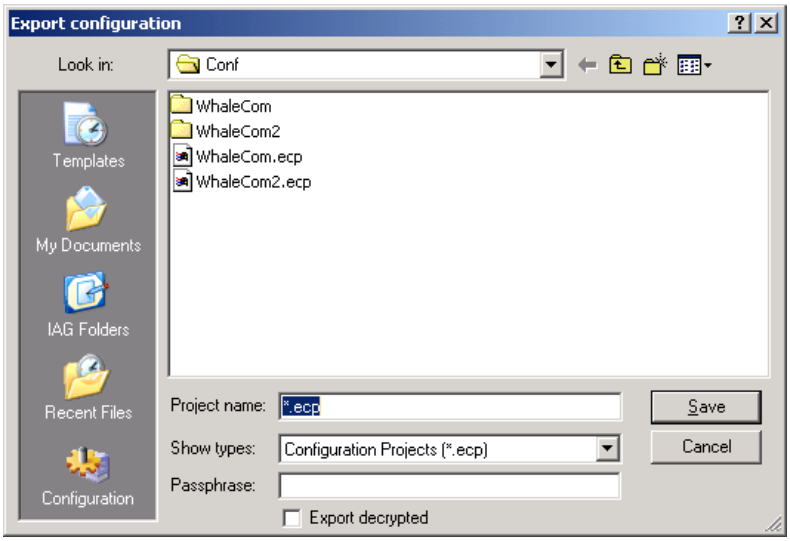
- Verify that the same encryption key and passphrase were used for the IAG server from which you export the configuration project, and all the IAG servers into which you import the configuration project. If the servers were defined with different keys, you can use a key import utility to export the keys from one IAG server and import them to another.
- Every time you change and activate the IAG configuration in one of the IAG servers in the High Availability array, you must deploy the configuration across the entire array. This includes changes in the Configuration program and changes to external configuration settings.
- If you add an additional IAG server to the High Availability array, you must deploy the configuration across the entire array again.

Exporting a Configuration Project

To export the configuration:

1. On the first IAG server, access the Configuration program, and on the **File** menu, click **Export...**

   The Export configuration box is displayed.

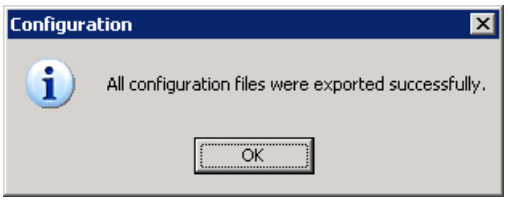


2. Enter the following:
   - Select the location to where you wish to export the project, and enter the project name.
   - Enter your passphrase.
   - By default, the configuration file is encrypted when it is exported. If you do not wish the file to be encrypted, select the option

     Export decrypted ☑ Export decrypted.

   Click Save.

   The Export configuration dialog box closes, and the configuration is exported to the selected location. The following message is displayed.



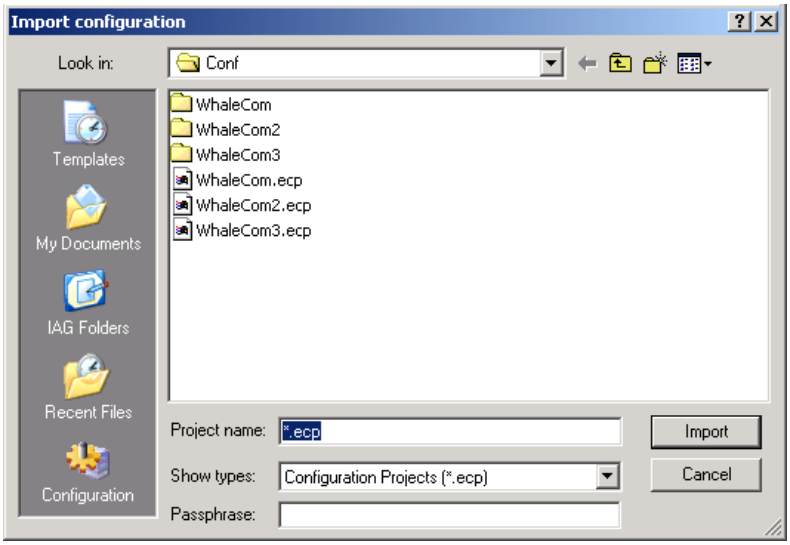
Importing a Configuration Project

You now import the configuration project that you exported from the first IAG server into other IAG servers, as described in the following procedure.

To import the configuration:

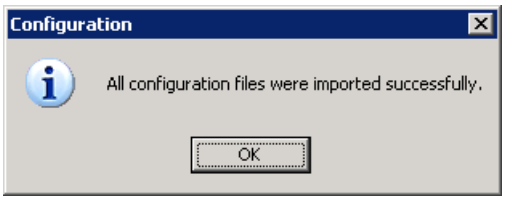
1. On the target IAG server, access the Configuration program, and on the **File** menu, click **Import...**

   The Import configuration dialog box is displayed.



2. Select the folder that contains the file you exported from the first IAG server, select the configuration file, enter your passphrase, and then click Import.

   The Import configuration dialog box closes. The project, including all its configuration settings, is imported into the target IAG server. The following message is displayed.



3. Click OK to dismiss the message. In the Configuration program, click and follow the instructions on the screen to save and activate the configuration.

   The configuration you created on the first IAG server is applied to the target IAG server.

Appendix A

Unicode Support

While users are accessing the internal network via the Intelligent Application Gateway (IAG), the IAG filter performs various operations that require text manipulation, such as host address or server name translation, URL inspection, and more. By default, the IAG supports manipulation of text that is encoded using the Unicode UTF-8 encoding format. However, in order to enable manipulation of text that is encoded using a different Unicode encoding format—UTF-16 or UTF-32—additional configuration is required.

This Appendix describes how you enable the manipulation of pages that use UTF-16 or UTF-32 encoding. If any of the pages in your site, with which users interact, use either of those encoding formats, follow the instructions in this Appendix to ensure the smooth operation of the IAG.

Enabling UTF-16 and UTF-32 Encoding

In order to enable the manipulation of text that is encoded using the UTF-16 or UTF-32 encoding formats, you take the following steps on the IAG:

- Add a dedicated Encoding filter extension to the Whale Filter Extension List. Note the following:
  - The Encoding filter extension must be first in the Extension list.
  - After you edit the Filter Extension List you have to restart the Web service in the IIS.

  For details, refer to “Activating the Encoding Filter Extension” on page 370.
- Create an encoding configuration file, and configure it to include each of the pages where UTF-16 or UTF-32 encoding is used. Note the following:
  - After the initial configuration, whenever you add files that use UTF-16 or UTF-32 encoding to your site, you need to update the configuration file to include the new files.
  - Whenever you update the configuration file, you need to activate the configuration.

For details, refer to "Creating and Updating the Encoding Configuration File" on page 371.

The steps described here enable the manipulation of UTF-16 or UTF-32 encoded text for one site, accessed via one trunk. You need to repeat them for each trunk that includes pages that use these types of encoding.

Activating the Encoding Filter Extension

When you configure a trunk, the Whale Filter Extension List, `WFEList.xml`, is copied under the trunk's folder on the IAG:

```
…\Whale-Com\e-Gap\Von\Conf\WebSites\<Trunk_Name>\Conf
```

The Extension List controls the activation of the filter extensions listed in it. In order to activate the Encoding extension, you have to add it to the list, as described in this section.

To activate the Encoding extension:

1. At the IAG, access the `CustomUpdate` folder under the trunk's `Conf` folder; if it does not exist, create it:

   ```
   …\Whale-Com\e-Gap\Von\Conf\WebSites\<Trunk_Name>\
   Conf\CustomUpdate
   ```

2. Copy the file `WFEList.xml` from the trunk's `Conf` folder to the `CustomUpdate` folder; if such a file already exists, use the existing file.
3. In the Whale Filter Extension List, `WFEList.xml`, add the Encoding extension to the list by adding the following line directly under the tag `<DLL_NAMES>`:

   ```
   <DLL active="1" dll_name="WhlFiltEncoding.dll"/>
   ```

Note

Make sure the Encoding extension is first in the list, that is, the line you add here is first in the extension list, as shown in the example below.

```
<DLL_LOADER ver="1.0">
     <DLL_NAMES>
           <DLL active="1" dll_name="WhlFiltEncoding.dll"/>
           <DLL active="1" dll_name="WhlFiltSecureRemote.dll"/>
           <DLL active="1" dll_name="WhlFiltSNT.dll"/>
           <DLL active="1" dll_name="WhlFiltRecorder.dll"/>
           <DLL active="1" dll_name="WhlFiltAppWrap.dll"/>
           <DLL active="1" dll_name="WhlFiltSSLVPN.dll"/>
           <DLL active="1" dll_name="WhlFiltLogOff.dll"/>
           <DLL active="1" dll_name="WhlFiltRuleSet.dll"/>
           <DLL active="1" dll_name="WhlFiltFormLogin.dll"/>
           <DLL active="1" dll_name="WhlFiltAuthorization.dll"/>
     </DLL_NAMES>
</DLL_LOADER>
```

4. Restart the Web service in the IIS. For details, refer to the *Intelligent Application Gateway User Guide*, to "Restarting the Web Service in the IIS" on page 321.

Creating and Updating the Encoding Configuration File

The encoding configuration file must include an entry for each of the pages in your site where UTF-16 or UTF-32 encoding is used. Whenever you add files that use these encoding methods to your site, you need to update the configuration file accordingly.

To configure the encoding configuration file:

1. Create the following file:
   - For HTTP trunks: `WhlFiltEncoding_HTTP.xml`
   - For HTTPS trunks: `WhlFiltEncoding_HTTPS.xml`

   Place the file in the following location:

   `…\Whale-Com\e-Gap\Von\Conf\WebSites\<Trunk_Name>\Conf`
2. Edit the file in the following format:

```
<WHLFILTENCODING ver="3.0">
  <URLS>
    <URL encoding="encoding_format" case_sensitive="true/false">URL</URL>
  </URLS>
</WHLFILTENCODING>
```

Tip

You can see sample files under the following folder:

`…\Whale-Com\e-Gap\Von\samples\Encoding`

Where:

- For each page that uses UTF-16 or UTF-32 encoding, you need to add one `<URL>` element.
- `URL` defines the page where UTF-16 or UTF-32 encoding is used. Note the following:
  - URLs can be defined using regular expressions; for details, refer to Appendix B: "Regex++, Regular Expression Syntax".
  - The Encoding extension receives the URL in its non-canonicalized form.
  - The URL must include the absolute path of the page. Since the Encoding extension is the first extension that is activated on the URL, the URL must also include the HAT signature that is added to the URL during link manipulation. For details, refer to "Link Manipulation" on page 256.

Tip

You can use the `.*` wildcard instead of adding the actual signature.

- The `encoding` attribute defines the encoding format of the URL, as follows:
  - `UTF-16`: UTF-16 with Byte Order Mark (BOM)
  - `UTF-16BE`: UTF-16 without BOM, serialized in big-endian
  - `UTF-16LE`: UTF-16 without BOM, serialized in little-endian
  - `UTF-32`: UTF-32 with BOM
  - `UTF-32BE`: UTF-32 without BOM, serialized in big-endian
  - `UTF-32LE`: UTF-32, without BOM, serialized in little-endian
- The `case_sensitive` attribute defines whether URL matching is case sensitive or not.

3. Once you finish editing the file, at Configuration program, click to activate the configuration.
4. In the Activate Configuration window, select the option "Apply changes made to external configuration settings".

Appendix B

Regex++, Regular Expression Syntax

This Appendix covers the regular expression syntax used by the Regex++ library, which you use in the configuration of the IAG. It is an excerpt from the documentation of the Regex++, Regular Expression Syntax, published by Dr. John Maddock at the following URL:

http://www.boost.org/libs/regex/doc/syntax.html

Literals

All characters are literals except: ".", "*", "?", "+", "(", ")", "{", "}", "[", "]", "^", "$", and "\". These characters are literals when preceded by a "\".

Wildcard

The dot character "." matches any single character.

Repeats

A repeat is an expression that is repeated an arbitrary number of times. An expression followed by "*" can be repeated any number of times including zero. An expression followed by "+" can be repeated any number of times, but at least once. An expression followed by "?" may be repeated zero or one times only. When it is necessary to specify the minimum and maximum number of repeats explicitly, the bounds operator "{}" may be used, thus "a{2}" is the letter "a" repeated exactly twice, "a{2,4}" represents the letter "a" repeated between 2 and 4 times, and "a{2,}" represents the letter "a" repeated at least twice with no upper limit. Note that there must be no white-space inside the {}, and there is no upper limit on the values of the lower and upper bounds.

All repeat expressions refer to the shortest possible previous sub-expression: a single character; a character set, or a sub-expression grouped with "()" for example.

Examples:

- "ba*" will match all of "b", "ba", "baaa" etc.
- "ba+" will match "ba" or "baaaa" for example but not "b".
- "ba?" will match "b" or "ba".
- "ba{2,4}" will match "baa", "baaa" and "baaaa".

Parenthesis

Parentheses serve two purposes, to group items together into a sub-expression, and to mark what generated the match. For example the expression "(ab)*" would match all of the string "ababab".

Alternatives

Alternatives occur when the expression can match either one sub-expression or another, each alternative is separated by a "|". Each alternative is the largest possible previous sub-expression; this is the opposite behavior from repetition operators.

Examples:

- "a(b|c)" could match "ab" or "ac".
- "abc|def" could match "abc" or "def".

Sets

A set is a set of characters that can match any single character that is a member of the set. Sets are delimited by "[" and "]" and can contain literals, character ranges, character classes, collating elements and equivalence classes. Set declarations that start with "^" contain the complement of the elements that follow.

Examples:

- Character literals:
  - "[abc]" will match either of "a", "b", or "c".
  - "[^abc] will match any character other than "a", "b", or "c".
- Character ranges:
  - "[a-z]" will match any character in the range "a" to "z".
  - "[^A-Z]" will match any character other than those in the range "A" to "Z".

Character Classes

Character classes are denoted using the syntax "[:classname:]" within a set declaration, for example "[[:space:]]" is the set of all whitespace characters. The available character classes are:

Table 35. Regex++ Character Classes

Class	Description
alnum	Any alpha numeric character.
alpha	Any alphabetical character a-z and A-Z. Other characters may also be included depending upon the locale.
blank	Any blank character, either a space or a tab.
cntrl	Any control character.
digit	Any digit 0-9.
graph	Any graphical character.
lower	Any lower case character a-z. Other characters may also be included depending upon the locale.
print	Any printable character.
punct	Any punctuation character.
space	Any whitespace character.
upper	Any upper case character A-Z. Other characters may also be included depending upon the locale.
xdigit	Any hexadecimal digit character, 0-9, a-f and A-F.
word	Any word character - all alphanumeric characters plus the underscore.
unicode	Any character whose code is greater than 255, this applies to the wide character traits classes only.

There are some shortcuts that can be used in place of the character classes, provided the flag `regbase::escape_in_lists` is set then you can use:

- \w in place of [:word:]
- \s in place of [:space:]
- \d in place of [:digit:]
- \l in place of [:lower:]
- \u in place of [:upper:]

Sample Set

To include a literal "-" in a set declaration, make it the first character after the opening "[" or "[^", the endpoint of a range. To include a literal "[" or "]" or "^" in a set, make them the endpoint of a range.

Line Anchors

An anchor is something that matches the null string at the start or end of a line: "^" matches the null string at the start of a line, "$" matches the null string at the end of a line.

Back References

A back reference is a reference to a previous sub-expression that has already been matched, the reference is to what the sub-expression matched, not to the expression itself. A back reference consists of the escape character "\" followed by a digit "1" to "9", "\1" refers to the first sub-expression, "\2" to the second etc. For example the expression "(.*)\1" matches any string that is repeated about its mid-point for example "abcabc" or "xyzxyz". A back reference to a sub-expression that did not participate in any match, matches the null string: NB this is different to some other regular expression matchers. Back references are only available if the expression is compiled with the flag `regbase::bk_refs` set.

Characters by Code

This is an extension to the algorithm that is not available in other libraries, it consists of the escape character followed by the digit "0" followed by the octal character code. For example "\023" represents the character whose octal code is 23. Where ambiguity could occur, use parentheses to break the expression up: "\0103" represents the character whose code is 103, "(\010)3 represents the character 10 followed by "3". To match characters by their hexadecimal code, use \x followed by a string of hexadecimal digits, optionally enclosed inside {}, for example \xf0 or \x{aff}, notice the latter example is a Unicode character.

Word Operators

The following operators are provided for compatibility with the GNU regular expression library:

- "\w" matches any single character that is a member of the “word” character class, this is identical to the expression "[[:word:]]".
- "\W" matches any single character that is not a member of the “word” character class, this is identical to the expression "[^[:word:]]".
- "\<" matches the null string at the start of a word.
- "\>" matches the null string at the end of the word.
- "\b" matches the null string at either the start or the end of a word.
- "\B" matches a null string within a word.

The start of the sequence passed to the matching algorithms is considered to be a potential start of a word unless the flag `match_not_bow` is set. The end of the sequence passed to the matching algorithms is considered to be a potential end of a word unless the flag `match_not_eow` is set.

Buffer Operators

The following operators are provide for compatibility with the GNU regular expression library, and Perl regular expressions:

- "\`" matches the start of a buffer.
- "\A" matches the start of the buffer.
- "\'" matches the end of a buffer.
- "\z" matches the end of a buffer.
- "\Z" matches the end of a buffer, or possibly one or more new line characters followed by the end of the buffer.

A buffer is considered to consist of the whole sequence passed to the matching algorithms, unless the flags `match_not_bob` or `match_not_eob` are set.

Escape Operator

The escape character "\" has several meanings.

- Inside a set declaration the escape character is a normal character.
- The escape operator may introduce an operator, for example: back references, or a word operator.
- The escape operator may make the following character normal, for example "*" represents a literal "*" rather than the repeat operator.

Single Character Escape Sequences

The following escape sequences are aliases for single characters:

Table 36. Regex++ Single Character Escape Sequences

Escape Sequence	Character Code	Meaning
\a	0x07	Bell character
\f	0x0C	Form feed.
\n	0x0A	Newline character.
\r	0x0D	Carriage return.
\t	0x09	Tab character.
\v	0x0B	Vertical tab.
\e	0x1B	ASCII Escape character.
\0dd	0dd	An octal character code, where *dd* is one or more octal digits.
\xXX	0xXX	A hexadecimal character code, where XX is one or more hexadecimal digits.

Table 36. Regex++ Single Character Escape Sequences (Cont'd)

Escape Sequence	Character Code	Meaning
\x{XX}	0xXX	A hexadecimal character code, where XX is one or more hexadecimal digits, optionally a unicode character.
\cZ	z-@	An ASCII escape sequence control-Z, where Z is any ASCII character greater than or equal to the character code for '@'.

Miscellaneous Escape Sequences

The following are provided mostly for perl compatibility, but note that there are some differences in the meanings of \l \L \u and \U:

Table 37. Regex++ Miscellaneous Escape Sequences

Escape Sequence	Meaning
\w	Equivalent to [[:word:]].
\W	Equivalent to [^[:word:]].
\s	Equivalent to [[:space:]].
\S	Equivalent to [^[:space:]].
\d	Equivalent to [[:digit:]].
\D	Equivalent to [^[:digit:]].
\l	Equivalent to [[:lower:]].
\L	Equivalent to [^[:lower:]].
\u	Equivalent to [[:upper:]].
\U	Equivalent to [^[:upper:]].

Table 37. Regex++ Miscellaneous Escape Sequences (Cont'd)

Escape Sequence	Meaning
\C	Any single character, equivalent to '.'.
\X	Match any Unicode combining character sequence, for example "a\x 0301" (a letter a with an acute).
\Q	The begin quote operator, everything that follows is treated as a literal character until a \E end quote operator is found.
\E	The end quote operator, terminates a sequence begun with \Q.

What Gets Matched?

The regular expression library will match the first possible matching string, if more than one string starting at a given location can match then it matches the longest possible string, unless the flag `match_any` is set, in which case the first match encountered is returned. Use of the `match_any` option can reduce the time taken to find the match—but is only useful if the user is less concerned about what matched—for example it would not be suitable for search and replace operations. In cases where their are multiple possible matches all starting at the same location, and all of the same length, then the match chosen is the one with the longest first sub-expression, if that is the same for two or more matches, then the second sub-expression will be examined and so on.

Copyright © 1998-2001

Dr. John Maddock[1]

1. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee. Dr. John Maddock makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Appendix C

Form Authentication Engine

The Intelligent Application Gateway (IAG) Form Authentication Engine can be used in conjunction with Web and Browser-Embedded applications, to handle HTML forms sent by the application to the end-user, for the following implementations:

- Login forms: in this implementation, the Form Authentication Engine automatically replies to authentication requests from the application. This option is used for applications that request users to authenticate using an HTML form, when the option "Automatically Reply to Application-Specific Authenticating Requests" is activated for the application, and the selected request type is "HTML Form" or "Both".
  - For Portal trunks, you configure this option in the Authentication step of the Add Application Wizard.
  - For Webmail and Basic trunks, you configure this option in the Application Login step of the Create New Trunk Wizard.
  - Once you add the application to a Portal trunk, or create a Webmail or Basic trunk, you can change the selected configuration in the Web Settings tab of the Application Properties dialog box. For details, refer to the *Intelligent Application Gateway User Guide*, to the section "Web Settings Tab" on page 73.
- Change Password forms: this option is used to update the IAG's internal credentials database with the new password, when a user changes the password during a session using an application's Change Password form. In this implementation, when a user changes the password, the Form Authentication Engine stores the new password in the internal credentials database. It can be used for applications that enable users to change their password during a session using an HTML form, and is implemented for both user-initiated and application-initiated password changes.

Note

The Form Authentication Engine does not disrupt the application's inherent login or change password flows.

The Form Authentication Engine is configured by default to handle Login and Change Password HTML forms for out-of-the-box applications that use either of these form-types.

For example:

- Since the Citrix NFuse® FR3 application requires users to login using HTML forms, the Form Authentication Engine is configured to handle Login forms for this application.
- For the Microsoft Outlook Web Access 2003 application, the Form Authentication Engine is configured to handle both Login and Change Password forms, since the application uses HTML forms for both interactions.

For generic applications, that are not supported out-of-the-box, and use HTML forms, you have to manually configure the Form Authentication Engine to handle the forms. If required, you can also change the default configuration settings for out-of-the-box applications.

This Appendix describes:

- The manner in which the Form Authentication Engine handles Login and Change Password forms, in "Functional Operation" on page 382.
- How you configure the Engine, in "Configuring the Form Authentication Engine" on page 384.

Functional Operation

This section describes how the Form Authentication Engine processes HTML forms, as follows:

- "How Login Forms Are Processed" on page 382.
- "How Change Password Forms Are Processed" on page 383.

How Login Forms Are Processed

This section describes how the Form Authentication Engine processes a Login HTML form, in order to automatically reply to an application's authentication request. Since all transactions between the application server and the endpoint computer pass through the IAG, the Engine can intercept the form whenever necessary, in order to complete the authentication process with no user intervention.

Login forms are usually processed as follows:

- The user tries to login by sending the login request to the application server.

- The application server responds by sending the Login form.
- The Form Authentication Engine intercepts the form on its way to the browser, and processes it as follows:
  - Depending on the configuration of this feature, the Engine can insert real or dummy values in input fields.
  - The Engine injects “autosubmit” and Onload Handler scripts, which will cause the form to be automatically returned from the browser to the application.

  It then sends the form to the endpoint browser.
- The Login form is displayed on the endpoint browser, and is then returned to the application without user intervention.
- The Form Authentication Engine intercepts the form on its way from the browser, and processes it as follows:
  - Retrieves the user credentials from the IAG’s internal credentials database.
  - Inserts the real values where required, replacing dummy values inserted previously.

  It then sends the filled-in form to the application server.
- The application server sends a response. If the authentication is successful, the user is logged in.
- Depending on the configuration, the Form Authentication Engine can intercept the response and analyze it before transferring it to the endpoint browser, in order to determine whether the authentication was successful.

If the process fails at any stage, the Form Authentication Engine sends the original Login form to the endpoint browser, so that the user can manually log into the application.

How Change Password Forms Are Processed

This section describes how the Form Authentication Engine processes a Change Password HTML form, in order to update the internal credentials database with the new password. Since all transactions between the application server and the endpoint computer pass through the IAG, the Engine can intercept the form whenever required in order to retrieve the new credentials from the form.

Change Password forms are usually processed as follows:

- When a password change is initiated, either by the user or by the application, the application server sends a Change Password form to the endpoint browser. At this stage, the Form Authentication Engine can partially fill the form, such as fill in the user name or domain name, depending on the configuration of this option.
- The user fills in the required fields and submits the form, and the endpoint browser returns the Change Password form to the application server.
- The Form Authentication Engine intercepts the form on its way to the application server, and analyzes it. If the user's credentials have changed, the Engine updates the internal credentials database with the new credentials. These credentials are used then by the IAG for the remainder of the session, or until the credentials are changed again.
- At this stage, some applications might require the user to re-authenticate. In this case, the following takes place:
  - If the option "Automatically Reply to Application-Specific Authenticating Requests" is activated for the application, the Form Authentication Engine replies to the authentication request with the user's new credentials.
  - If the option "Automatically Reply to Application-Specific Authenticating Requests" is not activated for the application, the authenticating request is forwarded to the user. The user replies to the request with the new credentials.

  Once the authentication is accepted by the application, the user can continue to work with the application.

Configuring the Form Authentication Engine

Note

- You configure the Form Authentication Engine using an XML file. The configuration process requires a working knowledge of XML technology.
- In order to edit the "autosubmit" script, working knowledge of JavaScript is required.
- Prior to configuring the Engine, be sure to read the section "Before You Begin" on page 386; it holds important information you should note.

The stages you go through in order to configure the Form Authentication Engine include:

- Optionally configure browsers, browser groups, or scripts in the Form Authentication Engine's data definition file, as described in "Form Authentication Engine Custom Data Definition File" on page 387. The Form Authentication Engine's configuration file refers to the data definition file whenever browsers or scripts must be defined.
- Create and configure a custom configuration file, as described in "Form Authentication Engine Custom Configuration File" on page 400.
- Optionally, and for Login forms only, define your own "autosubmit" script, as described in "Autosubmit Script" on page 431. These scripts can be contained in stand-alone files, or as part of the data definition file.
- If the HTML form requires client-side processing, define it as described in "Client-Side Processing" on page 434.
- If the HTML form includes multi-value variables such as checkboxes, define them as described in "Multi-Value Variables" on page 436.

Conventions Used in the Engine's Data Definition and Configuration Files

The Form Authentication Engine data definition and configuration files follow these conventions:

- Tag names are uppercase:

  <TAG>...**</TAG>**

- Attribute names are lowercase:

  <TAG **attribute**=value>...</TAG>

- Element values are either lower or upper case:

  <TAG>**value**</TAG>

  <TAG>**VALUE**</TAG>

  The case for each element is included in the element description, where relevant.

- Attribute values are lower case:

  <TAG attribute=**value**>...</TAG>

- Some XML elements take regular expressions in the data. When regular expressions are used, this is indicated in the description of the element usage. When using regular expressions, note the following:

- Use the syntax described in Appendix B: **“Regex++, Regular Expression Syntax”**
- In order to use a character as is, make sure it is preceded by a slash. For example, to use the asterisk (`*`) wildcard in data that takes regular expressions, enter `\*`

- For the following characters, CDATA type should be used: ampersand (`&`); angle brackets (`<` and `>`), quotation marks (`"`).

  For example: `<NAME><![CDATA[p&ssword]]></NAME>`

Before You Begin

Before you begin, you should examine the application or applications for which you are configuring the Form Authentication Engine, and identify the following for each application and for each form:

- Application-type:
  - If you are configuring the Engine for a generic application, make sure the type you define here is identical to the type you will enter when you add the application to the trunk, in the Add Application Wizard, in the step “Application Setup”.
  - If you are changing the configuration of an out-of-the-box application, you can use the Editor to find the definition of the application-type in the IAG, as described in “Finding the Application-Type” on page 44.
  - For Basic trunks, the application-type is always `OtherWeb`.

Note

The application type must be alphanumeric, without spaces and without punctuation marks. It is recommended that the name is significant so that it can be easily identified.

- Primary host URL, that is, location of the page containing the Login or Change Password form.
- If required, one or more secondary host URLs, that is, alternative location or locations of the page containing the form.
- Whether different forms are used for different browsers.
- Name or ID used to identify the form.
- Method used to send the form.
- Whether the form contains an onload handler in the <BODY> tag, and whether different handler types are used for different browsers (Login forms only).

- How the application server responds to the success or failure of the form submission, in order for the Engine to be able to evaluate the result (optional).
- Names of the following controls:
  - For Login forms: controls that the Form Authentication Engine fills in automatically in the form.
  - For Change Password forms: controls that the user fills in the form.
- Whether the form requires client-side processing in order to fill in any of the controls, and the names of the relevant controls.

Form Authentication Engine Custom Data Definition File

This section describes how you create and define a custom Form Authentication Engine data definition file. This file is used to:

- Define browsers so that they are identified by the Form Authentication Engine. The default file includes all browsers that are fully supported by the IAG. You may add browsers to the file, but the IAG support for such browsers is limited, as described in the *Intelligent Application Gateway User Guide*, in "Supported Browsers" on page 19.
- Create groups of browsers to which different form definitions in the configuration file refer.
- Hold scripts that are referred to from the configuration file.

The steps you need to take in order to prepare the data definition file are described in "Creating a Custom Form Authentication Engine Data Definition File" on page 388.

The contents of the data definition file, and explanations of how to define the file, are described in "Syntax of the Form Authentication Engine Data Definition File" on page 389.

Creating a Custom Form Authentication Engine Data Definition File

This section describes how you prepare and configure a custom Form Authentication Engine data definition file.

Note

- You can add new definitions to the data definition file, as described in "Syntax of the Form Authentication Engine Data Definition File" on page 389. It is recommended that you do not make changes to the existing definitions.
- Once you create a custom file, the definitions in this file override the definitions in the file that is supplied with the system. Thus, new updates will not be applied as they become available.

To create a custom Form Authentication Engine data definition file:

1. Access the following location on the IAG:

   `…\Whale-Com\e-Gap\Von\Conf`

2. Under the `Conf` folder, create the following subfolder:

   `CustomUpdate`.

   If such a folder already exists, use the existing folder.

3. From the `Conf` folder, copy the following file to the `CustomUpdate` subfolder:

   `FormLoginDataDefinitions.xml`

Caution

Do not make any changes to the default files located under the `Conf` folder.

4. Edit the file as required, as described in "Syntax of the Form Authentication Engine Data Definition File" on page 389.

5. At the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

Syntax of the Form Authentication Engine Data Definition File

Tip

- For a description of the conventions used in the data definition file, refer to "Conventions Used in the Engine's Data Definition and Configuration Files" on page 385.
- For a description of how you prepare the file where you configure the engine, refer to "Creating a Custom Form Authentication Engine Data Definition File" on page 388.

The data definition file contains the following:

- Browser definitions, described in "<USER_AGENTS>" on page 389.
- Browser group definitions, described in "<USER_AGENT_GROUPS>" on page 393.
- Scripts, described in "<SCRIPTS>" on page 396.

Caution

Do not change the root element in the data definition file: <WHLFILTFORMLOGIN_DATA_DEFINITIONS ver="x.x">

<USER_AGENTS>

Description

Defines the browsers that are supported by the Form Authentication Engine. The default data definition file includes the browsers that are fully supported by the IAG. You may add additional browsers to the list, but the IAG support for such browsers is limited, as described in the *Intelligent Application Gateway User Guide*, in "Supported Browsers" on page 19. It is recommended that before attempting to add new browsers to the list, you contact technical support.

Caution

Do not edit or erase any of the <USER_AGENT> elements nested under <USER_AGENTS> in the default data definition file.

Usage

Only one <USER_AGENTS> element can reside in the data definition file.

Attributes & Attribute Values

None.

Child Elements

<USER_AGENTS> must contain one or more <USER_AGENT> elements.

<USER_AGENTS> ➤ <USER_AGENT>

<USER_AGENT>

Description

Each <USER_AGENT> element defines a single browser, including:

- A unique ID to which the engine's configuration file refers, and which is used within the data definition file to create groups of browsers.
- The name of the browser.
- The method by in which the browser is identified by the USER_AGENT header value, in the header sent by the browser.

Usage

An unlimited number of <USER_AGENT> elements can reside under <USER_AGENTS>.

Attributes & Attribute Values

Attribute	Values	Type/Comments
id	An alphanumeric string	Mandatory. A short and unique ID that identifies the browser, for example: `<USER_AGENT id="IntExplorer">` **Note**: The string should not include spaces or special characters. Upper and lower case characters can be used, but these must be replicated exactly when you refer to the ID.

Child Elements

<USER_AGENT> must contain one each of the following elements:

- <NAME>, described on page 391.
- <SIGNATURE>, described on page 391.

<NAME>

Description

The browser name.

Usage

- One, and only one, <NAME> element must be nested under each <USER_AGENT> element.
- Can contain special characters, mixed case, and spaces.

  For example: `<NAME>Internet Explorer</NAME>`

Attributes & Attribute Values

None.

Child Elements

None.

<SIGNATURE>

Description

The browser's signature in the header sent by the browser, in the USER_AGENT header value.

Usage

- One, and only one, <SIGNATURE> element must be nested under each <USER_AGENT> element.
- Can take an exact string or regular expressions, depending on the value of the **check_by** attribute.

Attributes & Attribute Values

Attribute	Values	Type/Comments
check_by	search, regex	Mandatory. The method used to identify the USER_AGENT header value in the header sent by the browser: • `search`—an exact string • `regex`—regular expressions

Child Elements

None.

Sample USER_AGENTS Code

In this sample code, note the following:

- The Internet Explorer browser is identified by the ID “IntExplorer”, which is used to refer to this <USER_AGENT> from within the data definition file, and from the configuration file.
- In the first <USER_AGENT> element, the Internet Explorer browser is identified by an exact string.
- In the second <USER_AGENT> element, the Mozilla® browser is identified by regular expressions.

```
<USER_AGENTS>
     <USER_AGENT id="IntExplorer">
          <NAME>Internet Explorer</NAME>
          <SIGNATURE check_by="search">MSIE</SIGNATURE>
     </USER_AGENT>
     <USER_AGENT id="Mozilla">
          <NAME>Mozilla</NAME>
          <SIGNATURE check_by="regex">
          Mozilla/.*\(Macintosh;.*\).*Gecko\/.*
          </SIGNATURE>
     </USER_AGENT>
</USER_AGENTS>
```

<USER_AGENT_GROUPS>

Description

Defines groups of browsers to which the configuration file refers. In cases where the form defined in the configuration file supports more than one browser, this mechanism enables you to enter a single value, instead of listing all of the relevant browsers. A browser group can include any browsers defined as <USER_AGENT> elements, described on page 390.

Caution

You can add new groups and edit some of the existing groups in the data definition file, but **should not** edit the following groups:

- ie
- Netscape
- any

Usage

Only one <USER_AGENT_GROUPS> element can reside in the data definition file.

Attributes & Attribute Values

None.

Child Elements

<USER_AGENT_GROUPS> must contain one or more <USER_AGENT_GROUP> elements.

<USER_AGENT_GROUPS> ➤ <USER_AGENT_GROUP>

<USER_AGENT_GROUP>

Description

Defines a browser group.

Usage

- An unlimited number of <USER_AGENT_GROUP> elements can reside under <USER_AGENT_GROUPS>.
- The name you define here is used in the configuration file to refer to the browser group.

Attributes & Attribute Values

Attribute	Values	Type/Comments
name	An alphanumeric string	Mandatory. A short and unique ID that identifies the browser group. The string should not include spaces or special characters. Upper and lower case characters can be used, but these must be replicated exactly when you refer to the ID.

Child Elements

Each <USER_AGENT_GROUP> element must contain one or more <USER_AGENT_ID> elements.

<USER_AGENT_GROUPS> ➤ <USER_AGENT_GROUP> ➤ <USER_AGENT_ID>

<USER_AGENT_ID>

Description

The ID of a browser to be included in the browser group.

Usage

- An unlimited number of <USER_AGENT_ID> elements can reside under <USER_AGENT_GROUP>.
- The value you enter must be a browser ID, as defined in the **id** attribute of the <USER_AGENT> element, described on page 390.
- The same browser ID can be used in different browser groups.

Attributes & Attribute Values

None.

Child Elements

None.

Sample USER_AGENT_GROUPS Code

Caution

Do not edit or delete the USER_AGENT_GROUP elements that are highlighted in the following sample code:

- `ie`
- `Netscape`
- `any`

```
<USER_AGENT_GROUPS>
      <USER_AGENT_GROUP name="ie">
            <USER_AGENT_ID>IntExplorer</USER_AGENT_ID>
      </USER_AGENT_GROUP>
      <USER_AGENT_GROUP name="Netscape">
            <USER_AGENT_ID>NetscapeBrowser</USER_AGENT_ID>
      </USER_AGENT_GROUP>
      <USER_AGENT_GROUP name="any">
            <USER_AGENT_ID>IntExplorer</USER_AGENT_ID>
            <USER_AGENT_ID>NetscapeBrowser</USER_AGENT_ID>
      </USER_AGENT_GROUP>
      <USER_AGENT_GROUP name="ie_based">
            <USER_AGENT_ID>IntExplorer</USER_AGENT_ID>
      </USER_AGENT_GROUP>
      <USER_AGENT_GROUP name="mozilla_based">
            <USER_AGENT_ID>NetscapeBrowser</USER_AGENT_ID>
            <USER_AGENT_ID>Firefox</USER_AGENT_ID>
            <USER_AGENT_ID>Safari</USER_AGENT_ID>
            <USER_AGENT_ID>Camino</USER_AGENT_ID>
            <USER_AGENT_ID>Mozilla</USER_AGENT_ID>
      </USER_AGENT_GROUP>
      <USER_AGENT_GROUP name="all_supported">
            <USER_AGENT_ID>IntExplorer</USER_AGENT_ID>
            <USER_AGENT_ID>NetscapeBrowser</USER_AGENT_ID>
            <USER_AGENT_ID>Firefox</USER_AGENT_ID>
            <USER_AGENT_ID>Safari</USER_AGENT_ID>
            <USER_AGENT_ID>Camino</USER_AGENT_ID>
            <USER_AGENT_ID>Mozilla</USER_AGENT_ID>
      </USER_AGENT_GROUP>
</USER_AGENT_GROUPS>
```

<SCRIPTS>

Description

This element can contain Java or Visual Basic® scripts to which you refer from the configuration file.

Caution

You can add new scripts, but **do not** edit or delete these predefined scripts:

- `WhaleHandler`
- `WhaleSubmitStandard`

Usage

Only one <SCRIPTS> element can reside in the data definition file.

Attributes & Attribute Values

None.

Child Elements

<SCRIPTS> must contain one or more <SCRIPT> elements.

<SCRIPTS> ➤ <SCRIPT>

<SCRIPT>

Description

Contains a script to which the configuration file refers.

Usage

- An unlimited number of <SCRIPT> elements can reside under <SCRIPTS>.
- The **name** attribute is used in the configuration file to refer to this script.

Attributes & Attribute Values

Attribute	Values	Type/Comments
name	An alphanumeric string	Mandatory. A short and unique ID that identifies the browser. The string should not include spaces or special characters. Upper and lower case characters can be used, but these must be replicated exactly when you refer to the ID. **Caution, do not** use the following: • Names already in use in other scripts in the data definition file. • The names `WhaleSubmit` or `WhaleOnload`, as these are reserved for use by the IAG.

Child Elements

<SCRIPT> must contain one each of the following elements:

- <LANGUAGE>, described on page 397.
- <BODY>, described on page 398.

<SCRIPTS> ➤ <SCRIPT> ➤ <LANGUAGE>

<LANGUAGE>

Description

The script language.

Usage

- One, and only one, <LANGUAGE> element must be nested under each <SCRIPT> element.
- The values you can give include:
  - `javascript`—Javascript
  - `vbscript`—Visual Basic

Attributes & Attribute Values

None.

Child Elements

None.

<BODY>

Description

Contains the script.

Usage

One, and only one, <BODY> element must be nested under each <SCRIPT> element.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	cdata, base64	Mandatory. Defines whether the string uses plain text or encoding: • `cdata`—plain text • `base64`—base64 encoding **Tip:** Use the Editor to view and edit encoded strings. For a description refer to "Editor" on page 40.

Child Elements

None.

Sample SCRIPTS Code

Caution

Do **not** edit or delete the `WhaleSubmitStandard` and `WhaleHandler` scripts.

```
<SCRIPTS>
     <SCRIPT name="WhaleHandler">
          <LANGUAGE>javascript</LANGUAGE>
          <BODY encoding="cdata">
               <![CDATA[
                var gSafeOnload = new Array();
                function WhaleOnload()
                {
                  for (var i=0; i < gSafeOnload.length; i ++)
                  {
                     gSafeOnload[i]();
                  }// for i
                }// WhaleOnload
                  if (window.onload)
                {
                  gSafeOnload[0] = window.onload;
                  gSafeOnload[gSafeOnload.length] = WhaleSubmit;
                  window.onload = WhaleOnload;
                }
                else
                {
                  window.onload = WhaleSubmit;
                } // if window.onload
               ]]>
          </BODY>
     </SCRIPT>
     <SCRIPT name="WhaleSubmitStandard">
          <LANGUAGE>javascript</LANGUAGE>
          <BODY encoding="cdata">
               <![CDATA[
               function WhaleSubmit()
               {
                formsCol = document.forms;
                if (formsCol.length == 1)
                {
                  document.forms[0].submit();
                }
                else
                {
                  alert("more than one form");
                }
                return false;
                }
               ]]>
          </BODY>
     </SCRIPT>
</SCRIPTS>
```

Form Authentication Engine Custom Configuration File

This section includes the following:

- The steps you need to take in order to prepare a custom configuration file, in "Creating a Custom Form Authentication Engine Configuration File" on page 400.
- Description of the configuration file, in "Syntax of the Form Authentication Engine Configuration File" on page 402.
- Sample code for both Login and Change Password configurations, in "Sample Login Form Configuration" on page 427.

Note

Some of the elements in the configuration file are related to pre-defined or user-defined items located in the data definition file. For more information on the contents and definitions of this file, refer to "Form Authentication Engine Custom Data Definition File" on page 387.

Creating a Custom Form Authentication Engine Configuration File

This section describes how you prepare and configure a custom configuration file. Use this file to handle HTML forms for custom applications, as well as to change the default settings of out-of-the-box applications. The same file is used for both Login and Change Password forms, for all relevant applications.

Note

The changes you make in the custom file are merged with the definitions of the file that is supplied with the system. If you change the default configuration of an out-of-the-box application, the settings you configure here will take precedence over the default settings. Thus, new updates will not be applied to the application as they become available.

To create a custom configuration file:

1. Access the following location on the IAG:

   `…\Whale-Com\e-Gap\Von\Conf\WizardDefaults\FormLogin`
2. Under the `FormLogin` folder, create the following subfolder:

   `CustomUpdate`. If such a folder already exists, use the existing folder.
3. From the `FormLogin` folder, copy the following file to the `CustomUpdate` subfolder:

   `FormLoginCustom.xml`

4. Rename the file you have just copied `FormLogin.xml`

Caution

Do not make any changes to the default pages and scripts located under the `FormLogin` folder.

5. The file you have copied in step 3 contains one sample <APPLICATION> XML element. For each application you are configuring here, you need to create a dedicated <APPLICATION> element, and edit the values of each element as required.

   - The file contains one sample element. Replicate this element for

Tip

For out-of-the-box applications, you can copy the relevant <APPLICATION> section from the default configuration file, and edit the default settings as required. The default configuration file, `FormLogin.xml`, is located under the `FormLogin` folder.

   each relevant application, and edit the values as required.
   - Under the <APPLICATION> element, for each Login or Change Password form used by the application, you have to configure a dedicated <USAGE> element.
   - For a full description of the configuration file and the <APPLICATION> element, refer to "Syntax of the Form Authentication Engine Configuration File" on page 402.

Caution

After you edit the file as required, make sure to delete all unused elements. Note that, though the "autosubmit" script is not usually used with Change Password forms, for compatibility reasons, the elements that are related to this script must be defined for both form-types. Thus, the <SCRIPT_NAME> and <POLICY> elements must be included in the configuration of all forms.

6. When you finish editing the configuration file, at the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

Syntax of the Form Authentication Engine Configuration File

Tip

- For a description of the conventions used in the data definition file, refer to "Conventions Used in the Engine's Data Definition and Configuration Files" on page 385.
- For a description of how you prepare a custom configuration file, refer to "Creating a Custom Form Authentication Engine Configuration File" on page 400.

The Form Authentication Engine configuration file is structured as follows:

- For each custom application that requires form authentication, one <APPLICATION> element is nested under the root element. For a description of this element, including the elements nested under it, refer to <APPLICATION>, on page 402.
- The same <APPLICATION> element handles all Login and Change Password forms for the application, as applicable. For each form, one <USAGE> element is nested under the <APPLICATION> element.

Caution

Do not change the root element in the file:
<WHLFILTFORMLOGIN ver="x.x">

<APPLICATION>

Description

Provides the Engine with the information required to handle an application's Login and Change Password forms.

Usage

- An unlimited number of <APPLICATION> elements can reside in the configuration file.
- One <APPLICATION> element is required for each application that uses custom form authentication.

Attributes & Attribute Values

None.

Child Elements

- <APPLICATION_TYPE>, described on page 403.
- <USAGE>, described on page 403.

<APPLICATION_TYPE>

Description

The application type.

Note

Unless you specifically want to override the form authentication definitions of one of the out-of-the-box applications, do not use any of the <APPLICATION_TYPE> values used in the IAG's default configuration file, `FormLogin.xml`, located in the `FormLogin` folder.

Usage

- One, and only one, <APPLICATION_TYPE> element must be nested under the <APPLICATION> element.
- For a description of the application-type, refer to "Before You Begin" on page 386.

Attributes & Attribute Values

None.

Child Elements

None.

<USAGE>

Description

Defines the configuration for a specific usage, either a Login form or a Change Password form.

Usage

- An unlimited number of <USAGE> elements can reside under each <APPLICATION> element.
- One <USAGE> element is required for each HTML form that is processed by the application:
  - If an application uses both Login and Change Password forms, you must define <USAGE> elements for both form-types.
  - If an application uses multi-forms, for example different Login forms for various user groups, you must define an individual <USAGE> element for each form.

Attributes & Attribute Values

Attribute	Values	Type/Comments
description	change_password, form_login	Mandatory. Defines the type of form being processed.

Child Elements

<USAGE> contains the following child elements:

- Mandatory <PRIMARY_HOST_URL>, described on page 404.
- Optional <SECONDARY_HOST_URL>, described on page 405.
- Mandatory <SCRIPT_NAME>, described on page 405.
- Mandatory <USER_AGENT>, described on page 406.
- <MULTIPLE_LOGIN>, described on page 411, is optional for Login forms, and mandatory for Change Password forms.
- Mandatory <LOGIN_FORM>, described on page 412.

<APPLICATION> ➤ <USAGE> ➤ <PRIMARY_HOST_URL>

<PRIMARY_HOST_URL>

Description

The primary URL of the form. If there is more than one form in the current application, for example a Login form and a Change Password form, each form must have a unique <PRIMARY_HOST_URL> value.

Usage

- One, and only one, <PRIMARY_HOST_URL> element must be nested under each <USAGE> element.
- All <PRIMARY_HOST_URL> elements nested under the same <APPLICATION> element must have unique values.
- Must be a regular expression.

Note
Tune the URL as strictly as possible.

Attributes & Attribute Values

None.

Child Elements

None.

<SECONDARY_HOST_URL>

Description

Additional, secondary URL or URLs for the form.
For example: for an application where the primary URL is used for the first attempt to access the application, and the secondary URL is used for all subsequent attempts.

Usage

- <SECONDARY_HOST_URL> is optional.
- An unlimited number of <SECONDARY_HOST_URL> elements can be nested under each <USAGE> element.
- All <SECONDARY_HOST_URL> elements nested under the same <APPLICATION> element must have unique values.
- Must be a regular expression.

Note
Tune the URL as strictly as possible.

Attributes & Attribute Values

None.

Child Elements

None.

<SCRIPT_NAME>

Description

Name of the “autosubmit” script that causes the form to be returned from the browser to the Form Authentication Engine without user intervention. You can create a custom script, or use the default `WhaleSubmitStandard` script provided in the data definition file.

Note
Though the “autosubmit” script is not usually used with Change Password forms, for compatibility reasons, this element must be included in the configuration of all forms.

- For a description of where the “autosubmit” script is used in the login process, refer to “How Login Forms Are Processed” on page 382.

- For a detailed description of the script, and of how to create a custom script, refer to “Autosubmit Script” on page 431.

Usage

- One and only one <SCRIPT_NAME> element must be nested under each <USAGE> element.
- You can use the same script for different forms, or create a different script with a different name for each form.
- The location of the script is indicated by the **source** attribute.

Attributes & Attribute Values

Attribute	Values	Type/Comments
source	data_definition, file	Mandatory. Defines the location where the script is stored: • `data_definition`—the script resides in the data definition file. The value of the element is the name of the script as defined in the value of the **name** attribute of the <SCRIPT> element, described on page 396. Make sure you use the exact value, including upper and lower case usage. For example: `<SCRIPT_NAME source= "data_definition"> WhaleHandler </SCRIPT_NAME>` • `file`—the script resides in a separate file. The value of the element is the name of the file. The file extension is mandatory and must be either `.js` or `.vbs`. For example: `<SCRIPT_NAME source="file"> Autosubmit.js</SCRIPT_NAME>`

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <USER_AGENT>

<USER_AGENT>

Description

Defines the following:

- Method used for script injection in the form.

- Supported browser or browsers. That is, the browser or browsers where the IAG supports the application for which you are configuring Form Authentication.

Note

For a list of the browsers on which IAG Form Authentication is fully supported, refer to the *Intelligent Application Gateway User Guide*, to "Supported Browsers" on page 19. Other browsers can be used, with limited support, as described in the same section.

Usage

- <USER_AGENT> elements can refer to single browsers or to groups of browsers, which are defined in the data definition file under "<USER_AGENTS>" on page 389 and "<USER_AGENT_GROUPS>" on page 393, respectively.
- At least one <USER_AGENT> element must be nested under each <USAGE> element.
- Any number of <USER_AGENT> elements can be nested under each <USAGE> element. The number of <USER_AGENT> elements is determined as follows:
  - If only one browser-type is supported, use a single <USER_AGENT> element for that browser.
  - If more than one browser-type is supported, and the form's behavior is identical for all browser-types, use either a single <USER_AGENT> element pointing to an appropriate browser group.

Tip

You can use one of the pre-defined groups, or create your own groups in the data definition file, as described in "<USER_AGENT_GROUPS>" on page 393.

  - If more than one browser-type is supported, and the form's behavior differs between the browser-types. In this case, define multiple <USER AGENT> elements, one for each behavior.

Attributes & Attribute Values

None.

Child Elements

<USER_AGENT> must contain one of each of the following elements:

- <AGENT_TYPE>, described on page 408.
- <POLICY>, described on page 409.

- If the value of the <POLICY> element is “multiplatform”, the <USER_AGENT> element must also contain one <SCRIPT_NAME> element, described on page 410.

<APPLICATION> ➤ <USAGE> ➤ <USER_AGENT> ➤ <AGENT_TYPE>

<AGENT_TYPE>

Description

Defines the browser or browser group for which form authentication is supported for this form.

Usage

- One, and only one, <AGENT_TYPE> element must be nested under each <USER_AGENT> element.
- <AGENT_TYPE> includes the name of a single browser or of a group of browsers, as defined in the data definition file, as follows:
  - For a single browser, the value of <AGENT_TYPE> is the value defined for the **id** attribute of the appropriate <USER_AGENT> element, described on page 408.
  - For a browser group, the value of <AGENT_TYPE> is the value defined for the **name** attribute of the appropriate <USER_AGENT_GROUP> element, described on page 412.

  Make sure you use the exact value, including upper and lower case usage.

Attributes & Attribute Values

Attribute	Values	Type/Comments
search	specific, group	Mandatory. Defines whether this <AGENT_TYPE> element refers to a single browser or to a group of browsers: • `specific`—a single browser • `group`—a browser group

Child Elements

None.

<POLICY>

Description

Method by which the "autosubmit" and "onload" handler scripts are injected into the form, for the browser or browser groups determined by the <AGENT_TYPE> element or elements.

Note

Though the "autosubmit" script is not usually used with Change Password forms, for compatibility reasons, this element must be included in the configuration of all forms.

Usage

- One, and only one, <POLICY> element must be nested under each <USER_AGENT> element.
- <POLICY> values must be lower case.
- The value of <POLICY> can be one of the following:

Value	Description
limited	No "autosubmit" script is used. • Login forms: use when the form should not be submitted automatically, but has to be presented on the endpoint browser for the user to complete. **For example:** when a token device such as SecurID or VASCO is used. • This value is usually used for all Change Password forms.
multiplatform	Used for all forms, except for forms that are limited.
standard, extended	These policies are intended for internal IAG use only, and should not be used in the custom configuration file.

Attributes & Attribute Values

None.

Child Elements

None.

<SCRIPT_NAME>

Description

Use if the value of the <POLICY> element is “multiplatform”. Defines the onload handler script to be injected. For more information on the multiplatform policy, see “<POLICY>” on page 409.

Usage

- One <SCRIPT_NAME> element must be nested under each <USER_AGENT> element where the value of the <POLICY> element is “multiplatform”.
- You can create a custom script, or use the default script provided with the IAG inside the data definition file, under the element `<SCRIPT name="WhaleHandler">`
- The script can reside in a separate file, or as a <SCRIPT> element in the data definition file. The location of the script is indicated by the **source** attribute.
- <SCRIPT_NAME> includes the name the script, as follows:
  - If the script resides in the data definition file, the name of the script as defined in the value of the **name** attribute of the <SCRIPT> element, as described on page 396. Make sure you use the exact value, including upper and lower case usage. For example:
    ```
    <SCRIPT_NAME source= "data_definition">WhaleHandler
    </SCRIPT_NAME>
    ```
  - If the script resides in a separate file, the name of the file, including the extension. The extension must be either `.js` or `.vbs`, and is mandatory. For example:
    ```
    <SCRIPT_NAME source="file"> Autosubmit.js</SCRIPT_NAME>
    ```

Attributes & Attribute Values

Attribute	Values	Type/Comments
source	data_definition, file	Mandatory. Defines the location where the script is stored: • `data_definition`—the script resides in the data definition file. • `file`—the script resides in a separate file.

Child Elements

None.

<FUNC_TO_INJECT>

Description

This element is intended for internal IAG use only, and should not be used in new <APPLICATION> elements in the configuration file.

<MULTIPLE_LOGIN>

Description

The <MULTIPLE_LOGIN> element defines whether the Form Authentication Engine is required to handle the form each time a user accesses it during a session, or only once in a session, at the initial access.

- Login forms: use the <MULTIPLE_LOGIN> element to define which method is used for multiple login attempts. When users open multiple instances of an application, the application can handle the multiple login attempts by one of two methods:
  - Using the same Login form for all the application login attempts. In this case, the Form Authentication Engine is required to handle the form each time the user logs into the application during the session.
  - Using one Login form for the initial login attempt, and a different Login form for all subsequent login attempts in a session. In this case, the Form Authentication Engine is required to handle the form only at the initial login attempt.
    For example: when accessing the Domino iNotes application, users are only required to authenticate during the initial login attempt. Subsequent attempts are automatically handled by the application, using a different form.
- Change Password forms: since the same form is used for all password change instances, use the <MULTIPLE_LOGIN> element to define the method whereby the Form Authentication Engine processes the form each time the user changes the password

Usage

- For Login forms, one <MULTIPLE_LOGIN> element can optionally be nested under each <USAGE> element.
- For Change Password forms, one <MULTIPLE_LOGIN> element must be nested under each <USAGE> element.
- <MULTIPLE_LOGIN> values must be lower case.

- The value of <MULTIPLE_LOGIN> can be one of the following:

Value	Description
true	The Form Authentication Engine processes the form at all login or password change attempts during the session. Use this value for all Change Password forms, and for Login forms where the application uses the same form for all login attempts.
false	Default behavior when a <MULTIPLE_LOGIN> element is not defined under the <USAGE> element. The Form Authentication Engine processes the form only once in a session, the first time the form in accessed. Use this value only for Login forms where the application uses a different form for subsequent login attempts.

Note

Since if no <MULTIPLE_LOGIN> element is defined, the Form Authentication Engine processes the form only the first time it is accessed during a session, this element must always be defined for Change Password forms, with a "true" value. If <MULTIPLE_LOGIN> is not defined for a Change Password form, Form Authentication will only work the first time a user changes the password. If, during a session, the user changes the password more than once, the form will not be processed in the subsequent password changes.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM>

<LOGIN_FORM>

Description

Defines the form that the Form Authentication Engine processes, for both Login and Change Password forms.

Usage

One, and only one, <LOGIN_FORM> element must be nested under each <USAGE> element.

Attributes & Attribute Values

None.

Child Elements

- <LOGIN_FORM> must contain one of the following elements:
  - <NAME>, described on page 414.

  Or,

  - <ID>, described on page page 414.
- <LOGIN_FORM> must contain one, and one only, <METHOD> element, described on page 415.
- <LOGIN_FORM> must contain at least one <CONTROL> element, described on page 415. Define <CONTROL> elements for all the controls on the form that are affected by or used by the Form Authentication Engine.

Caution

Do not define <CONTROL> elements for controls that are not affected by or used by the Form Authentication Engine.

For example:

  - For Login forms, define the fields that the Form Authentication Engine automatically fills in.
  - For Change Password forms, define the fields where the Form Authentication Engine retrieves the value entered by the user in order to update the internal credentials database, such as the password.
- <LOGIN_FORM> can contain <LOGIN_EVALUATOR> elements, described on page 423, as follows:
  - For Login forms, the <LOGIN_EVALUATOR> is optional.
  - For Change Password forms, at least one <LOGIN_EVALUATOR> must be defined.
  - For both form-types, a maximum of two <LOGIN_EVALUATOR> can reside under <LOGIN_FORM>, one for a positive evaluation and one for a negative evaluation.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <NAME>

<NAME>

Description

Name of the form.

Usage

Either a <NAME or an <ID> element must be nested under the <LOGIN_FORM> element.

Note

There is no need to define both a name and an ID for the form, even if both exist in the application. If both are defined, the <NAME> element is dominant.

Attributes & Attribute Values

None

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <ID>

<ID>

Description

The ID of the form.

Usage

Either an <ID or a <NAME> element must be nested under the <LOGIN_FORM> element.

Note

There is no need to define both a name and an ID for the form, even if both exist in the application. If both are defined, the <NAME> element are dominant.

Attributes & Attribute Values

None

Child Elements

None.

<METHOD>

Description

HTTP method used to send the form.

Usage

- One, and only one, <METHOD> element must be nested under the <LOGIN_FORM> element.
- <METHOD> values must be lower case.
- The value of <METHOD> can be either “GET” or “POST”.

Attributes & Attribute Values

None.

Child Elements

None.

<CONTROL>

Description

Each <CONTROL> element defines a form “control”, such as a form field or a drop-down list. This control is handled by the Form Authentication Engine before the form is submitted to the end-user. The Engine supports all HTML control types.

Usage

- At least one <CONTROL> element must be nested under <LOGIN_FORM>.
- An unlimited number of <CONTROL> elements can be nested under <LOGIN_FORM>.
- Child-element definitions must be unique for each of the <CONTROL> elements nested under the same <LOGIN_FORM> element.
- Define <CONTROL> elements for all the controls on the form that are affected by or used by the Form Authentication Engine.

  For example:

  - For Login forms, define the fields that the Form Authentication Engine automatically fills in.

- For Change Password forms, define the fields where the Form Authentication Engine retrieves the value entered by the user in order to update the internal credentials database, such as the password.

- If you are defining a multi-value control, such as a multiple checkbox, you must define a separate <CONTROL> element for each value. For details, see "Multi-Value Variables" on page 436.

Caution

Do not define <CONTROL> elements for controls that are not affected by or used by the Form Authentication Engine.

Attributes & Attribute Values

Mandatory **handling** attribute defines how the control is handled, as follows:

Attribute Value	Description
dummy_value	Normally used for Login forms only. The Form Authentication Engine sends a dummy, read-only value to the browser, and replaces it with the real value when the browser returns the form to the application server. The value itself is defined in the <DEF_VALUE> child-element, described on page 421. **Note:** • We recommend that you use this value in all cases where the real values should not be exposed on the endpoint browser. • In most cases, we recommend that you do not use this value for User Name fields, since users might find it confusing. Use it only in cases where exposing the user name may cause a security problem.
app_default	Normally used for Login forms only. The Form Authentication Engine does not change control's default value, and sets it to read-only. **For example:** use this value when the application uses a cookie to automatically set the control's value. This type of behavior is typical for applications that automatically fill in the User Name field, such as email applications.

Attribute Value	Description
real_value	The Form Authentication Engine retrieves the real value from the IAG's internal credentials database and sends it to the browser, set as a read-only value. **Tip:** Use this value for Change Password forms, for controls where the value of the <TYPE> child element is "USER_NAME".
conf_default	Normally used for Login forms only. The Form Authentication Engine attempts to retrieve a value from the IAG's internal credentials database. If no value is available, the Form Authentication Engine sets the value to the dummy value defined in the <DEF_VALUE> child-element, described on page 421.
user_input	The Form Authentication Engine leaves the control blank when it sends it to the browser, and the value is provided by the end-user. In this case, set the value of the <POLICY> element, described on page 409, to "limited". **Tip:** Use this value for Change Password forms, for controls where the value of the <TYPE> child element is "PASSWORD".

Child Elements

<CONTROL> must contain one each of the following elements:

- <TYPE>, described on page 418.
- When the <TYPE> is "DOMAIN_USER", the <CONTROL> element must also contain one <SEPARATOR> element, described on page 420.
- <NAME>, described on page 420.
- <DEF_VALUE>, described on page 421.
- <PROCESSING_SIDE>, described on page 422.
- When the <PROCESSING_SIDE> is "client", the <CONTROL> element must also contain one <VARIABLE_NAME> element, described on page 423.

<TYPE>

Description

Logical control-type; defines the usage of the data. **For example:** if the type is "PASSWORD", the Form Authentication Engine treats the control value as a password.

Note

The type you define here is not connected in any way to the HTML TYPE.

Usage

- At least one <TYPE> element must be nested under each <CONTROL> element.
- The number of <TYPE> elements that can be nested under each <CONTROL> element depends on the value of <TYPE>:
  - If the value is any value other than "USER_PROVIDED", only one <TYPE> element can be nested under each <CONTROL> element.
  - If the value is "USER_PROVIDED", an unlimited number of <TYPE> elements can be nested under each <CONTROL> element.
- <TYPE> values must be upper case.
- The value of <TYPE> can be one of the following:

Value	Description
USER_NAME	The control contains the user name.
PASSWORD	The control contains the password.
DOMAIN	The control contains the domain name.

DOMAIN_USER	The control contains both the domain name and the user name. For this control-type, the <CONTROL> element must also include a <SEPARATOR> element. The control should be in this format:
	`DOMAIN separator USER`
	where `separator` is the character defined in the <SEPARATOR> element, for example:
	`MyDomain\MyUserName`

Value	Description
NEW_PASSWORD	Used in Change Password forms only. The control contains the new password. For this control-type, the value of the **handling** attribute of the <CONTROL> element must be "user_input".

USER_PROVIDED	This control-type should be used only in one of these cases:
	• If the user must fill in a field, for example a token such as VASCO or SecurID, or a multi-value field.
	• For a value that exists in the internal credentials database, for example the department name or region. The value of the control is retrieved from the IAG's internal credentials database, which is the IAG Session Manager.
	In this case, the value must be defined in the IAG Session Manager, as a Session Resource parameter. You do this by adding the relevant code in one of the authentication hooks, for example in the file `PostPostValidate.asp`, using the following format:
	`SetSessionResourceParam g_cookie, "<ApplicationID>", "<PARAMETER>", "<PARAMETER VALUE>"`
	For example:
	`SetSessionResourceParam g_cookie, "9786CBF4BA0743416C219DB", "mailbox", "name@company.com"`
	For more information about authentication hooks, see "Adding Functionality to the Default Pages" on page 99.
	Note:
	• The identical value must be used in the Session Manager and in the USER_PROVIDED control
	• If the value cannot be retrieved from the IAG's internal credentials database, the policy is downgraded to "limited", the "autosubmit" script is not used, and the form is submitted to the endpoint browser for the end-user to fill. For detailed information about the limited policy, see "<POLICY>" on page 409.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <CONTROL> ➤ <SEPARATOR>

<SEPARATOR>

Description

Used when the value of the <TYPE> element is “DOMAIN_USER”, to define the separator between the domain and the user.

Usage

When the value of <TYPE> is “DOMAIN_USER”, one <SEPARATOR> element must be nested under <CONTROL>.

The separator must be a single character, and can only be one of these three characters: `\ / @`

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <CONTROL> ➤ <NAME>

<NAME>

Description

Name of the control, as it appears in the form.

Usage

One, and only one, <NAME> element must be nested under each <CONTROL> element.

Attributes & Attribute Values

None.

Child Elements

None.

<DEF_VALUE>

Description

Used as the control's default value, which is sent to the endpoint browser, for the following controls:

- When the **handling** attribute of the <CONTROL> element is defined as "dummy_value".
- When the **handling** attribute of the <CONTROL> element is defined as "conf_default", and the Form Authentication Engine cannot retrieve a value from the IAG's internal credentials database.

Note

Although the default value is not used for controls where the **handling** attribute value is "real_value" or "app_default", for compatibility reasons, a <DEF_VALUE> element must be defined for all controls.

Usage

- One, and only one, <DEF_VALUE> element must be nested under each <CONTROL> element.
- <DEF_VALUE> should comply with the application requirements. For example, the application might require that the password include a certain number of characters, or that it must be a combination of letters and numerals.
- When the value of <TYPE> is "DOMAIN_USER":
  - The value of <DEF_VALUE> must be in the following format:

    `DOMAIN separator USER`

    where `separator` is the character defined in the <SEPARATOR> element, as described in "<SEPARATOR>" on page 420, for example: `whldomain/whluser`
  - The <CONTROL> element must contain a <SEPARATOR> child element.

Attributes & Attribute Values

None.

Child Elements

None.

<PROCESSING_SIDE>

Description

Defines whether this control is processed on the client-side, or on the server-side (IAG).

Usage

- One <PROCESSING_SIDE> element can be nested under each <CONTROL> element, as follows:
  - If client-side processing is required, <PROCESSING_SIDE> must be nested under each <CONTROL> element.
  - If client-side processing is not required, <PROCESSING_SIDE> is not required.
- When client-side processing is required, the <CONTROL> element must also include a <VARIABLE_NAME> element.
- The value of <PROCESSING_SIDE> can be one of the following:

Value	Description
client	Client-side processing is required, in order for the "autosubmit" script to retrieve data from the client before the form is submitted. **For example:** in cases where the authentication process has to query an external user database, and the query result should be submitted to the application server. For a full description of how you configure client-side processing refer to "Client-Side Processing" on page 434.
server	All processing is done at the server side. This is the default behavior, and if the <PROCESSING_SIDE> element is not used, all processing is done on the server-side. When defining server-side processing, you can use any of the available <POLICY> values.

Attributes & Attribute Values

None.

Child Elements

None.

<VARIABLE_NAME>

Description

Used when client-side processing of the control is required, that is, the value of the <PROCESSING_SIDE> element is “client”. Name of the variable that holds the value of the control, as defined in the “autosubmit” script, in WhaleSubmit function list.

For a full description of how you configure client-side processing for a control refer to “Client-Side Processing” on page 434.

Usage

When the value of the <PROCESSING_SIDE> element is “client”, one <VARIABLE_NAME> element must be nested under <CONTROL>.

Attributes & Attribute Values

None.

Child Elements

None.

<LOGIN_EVALUATOR>

Description

Examines the response sent by the application server after submission of the form. This is used for the following:

- Login forms: for diagnostic purposes, in order to define whether the authentication succeeded or failed.

Note

For Login forms that use both HTTP 401 requests and HTML forms in the same authentication flow, we recommend that you do not use the <LOGIN EVALUATOR> element.

- Change Password forms: evaluates whether the user changed the password successfully, and, accordingly, determines whether or not to complete the change password process and enable the user to access the application.

The examination can be either of the header or of the body of the response, depending on the child-elements you define.

Usage

- For Login forms, <LOGIN_EVALUATOR> is optional.
- For Change Password forms, at least one <LOGIN_EVALUATOR> element must be nested under <LOGIN_FORM>.
- For both form-types, a maximum of two <LOGIN_EVALUATOR> elements can be nested under each <LOGIN_FORM> element, one for positive evaluation and one for negative evaluation.

Attributes & Attribute Values

Attribute	Values	Type/Comments
indicate	success, failure	Mandatory. Defines how the Form Authentication Engine determines whether form-handling succeeded or failed, if the strings defined in the <HEADER> or <SEARCH> child-elements are found in the response: • `success`—if the strings are found it indicates that the form was submitted to the application successfully. • `failure`—if the strings are found it indicates a failure; the form was not submitted successfully.

Child Elements

<LOGIN_EVALUATOR> must contain one or both of the following child elements:

- One <HEADER> element, described on page 424 (optional).
- One or more <SEARCH> elements, described on page 426.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <LOGIN_EVALUATOR> ➤ <HEADER>

<HEADER>

Description

Response header sent by the application server after the submission of the form. Used to evaluate the success or failure of the authentication.

Usage

- <HEADER> is optional.
- Only one <HEADER> element can be nested under each <LOGIN_EVALUATOR> element.

Attributes & Attribute Values

None.

Child Elements

<HEADER> must contain the following elements:

- <NAME>, described on page 425.
- <VALUE>, described on page 425.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <LOGIN_EVALUATOR> ➤ <HEADER> ➤ <NAME>

<NAME>

Description

Name of the response header that is used to evaluate the results of the authentication.

Usage

One, and only one, <NAME> element must be nested under each <HEADER> element.

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <LOGIN_EVALUATOR> ➤ <HEADER> ➤ <VALUE>

<VALUE>

Description

Value of the response header that is used to evaluate the results of the authentication.

Usage

- One, and only one, <VALUE> element must be nested under each <HEADER> element.
- Must be a regular expression

Tip

If you want to check the existence of a header, and not the content, set the value to `.*`:

```
<VALUE>.*</VALUE>
```

Attributes & Attribute Values

None.

Child Elements

None.

<APPLICATION> ➤ <USAGE> ➤ <LOGIN_FORM> ➤ <LOGIN_EVALUATOR> ➤ <SEARCH>

<SEARCH>

Description

String in the response that is used to evaluate the results of the authentication.

Usage

The <LOGIN_EVALUATOR> element can contain an unlimited number of <SEARCH> elements.

Attributes & Attribute Values

Attribute	Values	Type/Comments
encoding	base64	Optional. **Tip:** Use the Editor to view and edit encoded strings. For a description refer to "Editor" on page 40.

Child Elements

None.

Sample Login Form Configuration

The sample code in this section handles form authentication for the Outlook Web Access 2003 application, for Login and Change Password forms:

- "Sample Login Form Code" on page 427
- "Sample Change Password Code" on page 429

For the clarity of the examples, each code snippet is nested under a separate <APPLICATION> element. In the configuration file, however, you define the handling of all forms for one application, including both Login and Change Password forms, under one <APPLICATION> element, with one <APPLICATION_TYPE> element.

Sample Login Form Code

The code in this example is used to define the processing of a Login form for the Outlook Web Access 2003 application. Note the following:

- In the <USAGE> element, the value of the **description** attribute is "form_login".
- No <SECONDARY_HOST_URL> is required.
- The default "autosubmit" script is used to automatically submit the form from the client to the application server. In this case, the script referred to is a stand-alone script, indicated by the **source** attribute with the value "file".
- The <POLICY> element uses the "multiplatform" policy. The subsequent <SCRIPT_NAME> element identifies the `WhaleHandler` script located inside the data definition file by using the attribute **source** with the value "data_definition".
- The value of <MULTIPLE_LOGIN> is "true", so that the Form Authentication Engine processes the form each time a user accesses it during a session, since the application uses the same form for all login attempts.
- The Form Authentication Engine automatically fills in the Password field. This is defined by a <CONTROL> element where the value of the attribute **handling** is "dummy_value".
- The evaluation method is defined to identify failed login attempts by searching for the string "You could not be logged" in the response.

```
<APPLICATION >
   <APPLICATION_TYPE>OWA2003</APPLICATION_TYPE>
   <USAGE description="form_login">
    <PRIMARY_HOST_URL><![CDATA[/exchweb/bin/auth/owalogon\.asp\
          ?url=https://[^/]+/exchange/?&reason=(0|3)]]>
    </PRIMARY_HOST_URL>
    <SCRIPT_NAME source="file">Autosubmit.js</SCRIPT_NAME>
    <USER_AGENT>
       <AGENT_TYPE search="group">any</AGENT_TYPE>
       <POLICY>multiplatform</POLICY>
       <SCRIPT_NAME source="data_definition">WhaleHandler
       </SCRIPT_NAME>
    </USER_AGENT>
    <MULTIPLE_LOGIN>true</MULTIPLE_LOGIN>
    <LOGIN_FORM>
       <NAME>logonForm</NAME>
       <METHOD>POST</METHOD>
       <CONTROL handling="dummy_value">
          <TYPE>DOMAIN_USER</TYPE>
          <NAME>username</NAME>
          <DEF_VALUE>whldomain/whluser</DEF_VALUE>
          <SEPARATOR>/</SEPARATOR>
       </CONTROL>
       <CONTROL handling="dummy_value">
          <TYPE>PASSWORD</TYPE>
          <NAME>password</NAME>
          <DEF_VALUE>whlpass</DEF_VALUE>
       </CONTROL>
       <LOGIN_EVALUATOR indicate="failure">
          <SEARCH encoding="">You could not be logged</SEARCH>
       </LOGIN_EVALUATOR>
    </LOGIN_FORM>
   </USAGE>
</APPLICATION>
```

Sample Change Password Code

The code in this example is used to define the processing of a Change Password form for the Outlook Web Access 2003 application. Note the following:

- In the <USAGE> element, the value of the attribute **description** is "change_password".
- The <SCRIPT_NAME> element identifies the `WhaleSubmitStandard` script located inside the data definition file by using the attribute **source** with the value "data_definition".
- The method used to inject the "autosubmit" script is identical for all browsers. Therefore, the <USER_AGENT> element is defined to use the browser group "all_supported" defined in the data definition file.
- The value of the <POLICY> element is "limited", since the end-user is required to provide the new password.
- The Form Authentication Engine automatically fills in two controls, with the user-defined values stored in the IAG's internal credentials database. This is handled by two <CONTROL> elements, where the value of the attribute **handling** is "real_value":
  - For the User Name field, the definition of the control-type is: `<TYPE>USER</TYPE>`.
  - For the Domain field, the definition of the control-type is: `<TYPE>DOMAIN</TYPE>`.
- For the New Password field that requires user input, the following is defined:
  - A <CONTROL> element where the value of the attribute **handling** is "user_input".
  - Control type: `<TYPE>NEW_PASSWORD</TYPE>`.
- The evaluation method is defined to identify failed login attempts by searching for the string "You could not be logged".
- The evaluation method is defined to identify the success of the password-change by searching for the string "successfully" in the response.

```
<APPLICATION >
     <APPLICATION_TYPE>OWA2003</APPLICATION_TYPE>
     <USAGE description="change_password">
         <PRIMARY_HOST_URL>/iisadmpwd/aexp2b\.htr\?htt(ps|p)://[^/]+/
         exchange/[^/]+/\?Cmd=close </PRIMARY_HOST_URL>
         <SCRIPT_NAME source="data_definition">WhaleSubmitStandard
         </SCRIPT_NAME>
         <USER_AGENT>
           <AGENT_TYPE search="group">all_supported</AGENT_TYPE>
           <POLICY>limited</POLICY>
         </USER_AGENT>
         <MULTIPLE_LOGIN>true</MULTIPLE_LOGIN>
         <LOGIN_FORM>
           <NAME>logform</NAME>
           <METHOD>POST</METHOD>
           <CONTROL handling="real_value">
              <TYPE>USER</TYPE>
              <NAME>acct</NAME>
              <DEF_VALUE>whluser</DEF_VALUE>
           </CONTROL>
           <CONTROL handling="real_value">
              <TYPE>DOMAIN</TYPE>
              <NAME>domain</NAME>
              <DEF_VALUE>whldomain</DEF_VALUE>
           </CONTROL>
           <CONTROL handling="user_input">
              <TYPE>NEW_PASSWORD</TYPE>
              <NAME>new</NAME>
              <DEF_VALUE>new</DEF_VALUE>
           </CONTROL>
           <LOGIN_EVALUATOR indicate="success">
              <SEARCH encoding="">successfully</SEARCH>
           </LOGIN_EVALUATOR>
         </LOGIN_FORM>
     </USAGE>
</APPLICATION>
```

Autosubmit Script

Note

- This script is usually used in conjunction with Login forms only.
- For a description of where in the login process the "autosubmit" script is used, refer to "How Login Forms Are Processed" on page 382.

When the Form Authentication Engine processes a Login form, it injects an "autosubmit" script into the form. This script causes the form to be returned to the Form Authentication Engine without user intervention. The "autosubmit" script can be used for:

- Automatic form submission.
- If required, client-side processing prior to submission, such as inserting a value that is stored on the endpoint computer into a control.

Tip

For a description of client-side processing, refer to "Client-Side Processing" on page 434.

In the Form Authentication Engine's configuration file, the "autosubmit" script that is used for the form is defined in the <SCRIPT_NAME> element, described on page 405. The JavaScript function that is defined in the script is injected into the HTML page that contains the form, immediately after the <HEAD> tag.

Caution

Do not change or delete the `WhaleHandler` script in the data definition file.

The IAG provides a default script that handles automatic form submission, but does not handle any client-side processing. This script is located in the data definition file, under the element `<SCRIPT name="WhaleSubmitStandard">`.

For each form that you configure in the configuration file, you can select to use this default script, or to create and use a custom script, depending on the requirements of the application. You can use the same script for multiple forms and applications, or create different scripts for different forms or applications, in order to perform form- or application-specific operations. In all cases, you must refer to the script in the configuration file, in the <SCRIPT_NAME> element.

It is recommended that you contain custom “autosubmit” scripts inside the Form Authentication Engine’s data definition file, under the element <SCRIPTS>. Alternatively, such scripts can be held in a stand-alone file.

This section provides you with:

- Instructions on how to configure the “autosubmit” script if you do not use the default script provided with the IAG, in “Configuring a Custom Autosubmit Script” on page 432.
- A sample script, in “Sample Autosubmit Script” on page 434. This sample code is used for automatic form submission, with no client-side processing; for a description of how you configure client-side processing, including changes to the “autosubmit” script, refer to “Client-Side Processing” on page 434.

Configuring a Custom Autosubmit Script

This section describes how you configure a custom “autosubmit” script if you do not use the default script provided with the IAG, and includes instructions for creating the script:

- Inside the Form Authentication Engine data definition file, as described in “Creating a Custom “autosubmit” Script in the Data Definition File” on page 432.
- As a stand-alone file, as described in “Creating a Stand-Alone Custom “autosubmit” Script” on page 433.

Note that the function you define in the script is injected into the form as is; the Form Authentication Engine does not examine the content of the file. A sample “autosubmit” script is provided in “Sample Autosubmit Script” on page 434.

Creating a Custom “autosubmit” Script in the Data Definition File

This is the recommended method for creating custom scripts.

Please note the following:

- You can use the default `<SCRIPT name="WhaleSubmitStandard">` element located in the data definition file as an example of how to insert an “autosubmit” script in the data definition file, but **take care not to make any modifications in this element**.
- **Do not** use the names `WhaleSubmit` or `WhaleOnload` for custom <SCRIPT> elements, as these are reserved for use by the IAG.

To create an "autosubmit" script in the data definition file:

1. Create a custom data definition file, as described in "Creating a Custom Form Authentication Engine Data Definition File" on page 388. If a custom data definition file already exists, use the existing file.
2. Under the <SCRIPTS> element, create a new <SCRIPT> element, as described in "<SCRIPTS>" on page 396.

   Make sure to adhere to the following guidelines:

   - The "autosubmit" script must be self contained and should not require external parameters.
   - Give your script a unique name.
3. Refer to the custom script in the relevant section or sections of the configuration file, in the element <SCRIPT_NAME>.

   For example: if you named your script `Customscript`, in the definition of all the forms where you wish to use this script, refer to it as follows:

   ```
   <SCRIPT_NAME source="data_definition">Customscript</SCRIPT_NAME>
   ```

   For a description of this element, refer to "<SCRIPT_NAME>" on page 405.
4. If required, repeat steps 3–4 to create additional custom scripts.

Creating a Stand-Alone Custom "autosubmit" Script

This section describes how you create a stand-alone custom "autosubmit" file. Please note the following:

- You can use the default `Autosubmit.js` file as an example of an "autosubmit" file, **but take care not to make any modifications in this file**.
- When you name your custom file, **do not** use any of the file names already in use in the FormLogin folder, for example `Autosubmit.js`.

To configure a stand-alone "autosubmit" script:

1. Access the following location on the IAG:

   `…\Whale-Com\e-Gap\Von\Conf\WizardDefaults\FormLogin`
2. Under the `FormLogin` folder, create the following subfolder:

   `CustomUpdate`. If such a folder already exists, use the existing folder.
3. Create a JavaScript file, containing the "autosubmit" script, and place it in `CustomUpdate` the folder.

   Make sure to adhere to the following guidelines:

   - Give your file a unique name.
   - The function name in the script must always be `WhaleSubmit()`.
   - The "autosubmit" script must be self contained and should not require external parameters.

4. Refer to the file containing the custom script in the relevant section or sections of the configuration file, in the element <SCRIPT_NAME>.

   For example: if you named your file `Customsubmit.js`, in the definition of all the forms where you wish to use this script, refer to it as follows:

   ```
   <SCRIPT_NAME source="file">Customsubmit.js</SCRIPT_NAME>
   ```

   For a description of this element, refer to "<SCRIPT_NAME>" on page 405.
5. If required, repeat steps 3–4 to create additional custom files.
6. When you finish creating the files, at the Configuration program, click to activate the configuration, select the option "Apply changes made to external configuration settings", and click Activate >.

Sample Autosubmit Script

The script in this example is used for automatic form submission without client-side processing.

```
function WhaleSubmit()
{
    formsCol = document.forms;
     if (formsCol.length == 1)
     {
        document.forms[0].submit();
     }
     else
     {
        alert("more than one form");
     }
     return false;
}
```

Client-Side Processing

This section describes how you configure the automatic processing of a form on the client side, if required. **For example:** in cases where the authentication process has to query an external user database, and the query result should be submitted to the application server.

The configuration involves both the configuration file and the “autosubmit” script, and refers only to the configuration settings relevant for the client-side processing option. For a full description of the files refer to:

- “Form Authentication Engine Custom Configuration File” on page 400.
- “Autosubmit Script” on page 431.

Caution

Do not make any changes in the default pages and scripts supplied with the IAG.

To configure client-side processing for a form:

1. Create a custom configuration file, as described in “Creating a Custom Form Authentication Engine Configuration File” on page 400. If a custom file already exists, use the existing file.
2. In the configuration file, set the value of the <POLICY> element to either “extended” or “standard”. For a description of this element refer to page 409.
3. Still in the configuration file, configure the relevant <CONTROL> element, where the client-side input is required, as follows:
   - Set the value of the **handling** attribute to one of the following, as applicable: “real_value”, “conf_default”, or “dummy_value”. For a description of these values, refer to page 416.

Note

Do not set the value of the **handling** attribute to **“**user_input**”** or **“**app_default**”**.

   - Add a <PROCESSING_SIDE> child-element, with the value “client”, as described on page 422.
   - Add a <VARIABLE_NAME> element, as described on page 423, containing the variable that holds the value of the control.

   The following sample code shows a <CONTROL> element that is configured for client-side processing, with a `whlDomain` variable:

```
<CONTROL handling="real_value">
     <TYPE>DOMAIN</TYPE>
     <NAME>domain</NAME>
     <DEF_VALUE>whale domain</DEF_VALUE>
     <PROCESSING_SIDE>client</PROCESSING_SIDE>
     <VARIABLE_NAME>whlDomain</VARIABLE_NAME>
</CONTROL>
```

4. Create a custom “autosubmit” script, as described in “Autosubmit Script” on page 431. If a custom script already exists, use the existing script.
5. In the custom “autosubmit” script, add the body of the `WhaleSubmit` function list, using the following format:

```
function WhaleSubmit ()
{
     // <FunctionBody>
}
```

> **Tip**
> Inside the function body, you can reference variables such as the `whlDomain` variable cited in the example in step 3.

6. When you finish the configuration process, at the Configuration program, click to activate the configuration, select the option “Apply changes made to external configuration settings”, and click Activate >.

Multi-Value Variables

This section describes how you enable the use of multi-value variables, such as checkboxes, in a form.

> **Note**
> The Form Authentication Engine does not support multi-choice selection lists at this time.

The configuration involves both the configuration file and the “autosubmit” script, and refers only to the configuration settings relevant for the multi-value variables option. For a full description of the files refer to:

- “Form Authentication Engine Custom Configuration File” on page 400.
- “Autosubmit Script” on page 431.

> **Caution**
> Do not make any changes in the default pages and scripts supplied with the IAG.

To configure multi-value variables for a form:

1. Create a custom configuration file, as described in "Creating a Custom Form Authentication Engine Configuration File" on page 400. If a custom file already exists, use the existing file.
2. In the configuration file, configure a <CONTROL> element for each value within the multi-value variable, with the following definitions:
   - Set the value of the **handling** attribute to "conf_default", as described on page 416.
   - Set the value of the <TYPE> child-element to "USER_PROVIDED". For a description of this element, refer to page 418.
   - The <NAME> child-element must be identical for all values within the multi-value variable. For a description of this element, refer to page 420.
   - The <DEF_VALUE> child-element must be unique for each value within the multi-value variable. For a description of this element, refer to page 421.
   - The <PROCESSING_SIDE> child-element must be identical for all values within the multi-value variable. For a description of this element, refer to page 422.
   - The <VARIABLE_NAME> child-element, if used, must be identical for all values within the multi-value variable. For a description of this element, refer to page 423.

   The following sample code shows a <CONTROL> element that is configured to handle a multi-value variable:

```
<CONTROL handling="conf_default">
     <TYPE>USER_PROVIDED</TYPE>
     <NAME>Check</NAME>
     <DEF_VALUE>Two</DEF_VALUE>
     <PROCESSING_SIDE>client</PROCESSING_SIDE>
     <VARIABLE_NAME>whlCheck</VARIABLE_NAME>
</CONTROL>
```

3. Create a custom "autosubmit" script, as described in "Autosubmit Script" on page 431. If a custom script already exists, use the existing script.
4. In the custom "autosubmit" script, add body of the function to the `WhaleSubmit` function list.

In this example, if the value of the `whlDomain` variable is “whalecom”, and you configured two <CONTROL> elements with default values of “two” and “three”, the function is injected as the following carry:

```
WhaleSubmit()
{
     var whlCheck=['TypeTwo', 'TypeThree'];
     // <FunctionBody>
}
```

5. When you finish the configuration process, at the Configuration program, click to activate the configuration, select the option “Apply changes made to external configuration settings”, and click Activate >.