CA SiteMinder®

Web Agent Configuration Guide

r12.5

CA Technologies Product References

This document references the following CA Technologies products:

- CA SiteMinder®
- CA Wily Introscope®
- CA Identity Manager
- CA SOA Security Manager

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Web Agents 15

How Web Agents Secure Resources 16
How Web Agents and the Policy Server Work Together 17
Considerations for Web Agents and Policy Servers in Different Time Zones 19
How the Agent Reads SiteMinder Cookies 20
Web Agents and Dynamic Key Rollovers 21
Key Stores 21
Framework and Traditional Agent Architectures 22
Parameters Requiring a Server Restart when Changed 23
Multiple Agent for IIS Directory Structures According to Operating Environment 26

Chapter 2: Agent Configuration Methods 29

Central Configuration 29
Implement Central Configuration 30
Local Agent Configuration 31
WebAgent.conf File Locations 32
WebAgent.conf file for Framework Agents 33
LocalConfig.conf File Locations (Framework Agents) 34
Parameters Found Only in Local Configuration Files 35
Implement Local Configuration 36
Central and Local Configuration Together 40

Chapter 3: Configuration Files used by the Web Agent 41

Agent Connection Manager Configuration File 41
Connection API Configuration File 42
Local Agent Configuration File 43
Trace Configuration File 44
Web Agent Trace Configuration File 45
SiteMinder Host Configuration File 46
Web Agent Configuration File 47

Chapter 4: Basic Agent Setup and Policy Server Connections 49

Default Settings of Web Agent Configuration Parameters 49
Set the AgentName and DefaultAgentName Values 50
Restrict Changes to Local Configuration Parameters 52

Ensure that Agent Names Match 53
Encrypt the Agent Name 53
How to Manage Web Agent and Policy Server Communication 53
Accommodate Network Latency 54
Manage Web Agents with Multiple Web Server Instances 55
Set the ServerPath Parameter for Windows Systems 56
Set the ServerPath Parameter for UNIX Systems 56
Additional Configurations Requiring the ServerPath Parameter 57

Chapter 5: Starting and Stopping Web Agents 59

Enable a Web Agent 59
Disable a Web Agent 60
Starting or Stopping Most Apache-based Agents with the apachectl Command 60

Chapter 6: User Protection 61

Change How Often an Agent Checks for Policy or Key Updates 61
User Tracking and URL Monitoring 62
Track User Identity Across Anonymous Realms 62
Track User Activities or Application Usage with Auditing 63
URL Monitoring Overview 63
Help Prevent Attacks 64
Protect Web Sites Against Cross-Site Scripting 65
Configure the Web Agent to Check For Cross Site-Scripting 65
Protect J2EE Applications against Cross-Site Scripting Attacks 66
Override the Default CSS Character Set 66
Specify Bad Query Characters 67
Specify Bad URL Characters 69
Enable Bad Form Characters 71
Help Prevent DNS Denial Of Service Attacks 72
Protect Resources Without Extensions 72
Disable POST Preservation 73
Secure Applications 73
Verify IP Addresses 74
Resolve Agent Identity by IP Address 74
Compare IP Addresses to Prevent Security Breaches 75
SiteMinder Browser Cookies 75
Require Cookies for Basic Authentication 76
Safeguard Information in Cookies with HTTP-Only Attribute 76
Set Secure Cookies 77
Control Identity Cookies 77
Set Persistent Cookies 78

Specify the Cookie Path for Agent Cookies 79
Force the Cookie Domain 81
Implement Cookie Domain Resolution 82
How CookiePathScope Settings Work 82
Configure Support for SDK Third-Party Cookies 83
Define HTTPS Ports 84
Decode Query Data in a URL 84
How to Protect Resources Without Periods or Extensions 85
Handle Complex URIs 86

Chapter 7: Use Platform for Privacy Preferences (P3P) Compact Policies with SiteMinder Agents 87

How to Support a P3P Compact Policy with your SiteMinder Web Agent 87
Configure your Web Agent to Accommodate P3P Compact Policies 88

Chapter 8: Session Protection 89

Apply SiteMinder Behavior to a Web Application Client 89
Web Application Client Response Introduced 90
Cookie Providers and the Web Application Client Response 92
How to Apply the Web Application Client Response to a Web Application 92
Modify the Session Grace Period 95
Modify the Session Update Period 96
Protect Session Cookies from Misuse with Validation Periods and Expired Cookie URLs 97
Prevent Session Cookie Creation or Updates 98
Prevent Session Cookie Creation or Updates Based on Method and URI 99
Store Session Cookies on the Session Store for Improved Security 100
Validate a Session Cookie Domain 101
Redirect a User after a Session Time-out 102
How to Enforce Timeouts across Multiple Realms 103

Chapter 9: Web Application Protection 105

Mechanisms for Developing Web Applications 105
REMOTE_USER Variable 105
Configure the Web Agent to set the REMOTE_USER Variable 106
IIS Web Servers and the REMOTE_USER Variable 107
How Response Attributes Work with Web Agents 108
Use SM_AGENT_ATTR_USRMSG Response for a Forms Challenge 109
Cache Response Attributes 110
SiteMinder Default HTTP Headers 111
HTTP Header and Cookie-Variables 113

Header Variables and End-User IP Address Validation 114
Preserve HTTP Headers 116
Custom Error Handling For Applications 122

Chapter 10: Configure Virtual Servers 127

How to Set Up Virtual Server Support 128
Assign Web Agent Identities for Virtual Servers 129
Specify Virtual Servers for the Web Agent to Ignore 130

Chapter 11: Forms Authentication 133

How Credential Collectors Process Requests 134
MIME Types for Credential Collectors 135
How to Configure Basic FCC Authentication 136
Set Up Credential Collectors for IIS and Domino Web Servers 137
Enable FCCs and SCCs to Use Agent Names as Fully Qualified Host Names 137
Configure the FCC to Use a Single Resource Target 138
Use a Relative Target for Credential Collector Redirects 138
Define Valid Target Domains 139
Tune the Performance of the FCC 139
Disable FCC Realm Context Confirmation to Improve Performance 140
Forms Cache 140
Specify an NTLM Credential Collector 142
Map URLs for FCC Redirects with a Domino Web Agent 142
Configure POST Preservation 142
Enable Post Preservation between Framework and Traditional Agents 144
Customize the POST Preservation Page 145
Disable POST Preservation 147
Using Credential Collectors Between 4.x Type and Newer Type Agents 147
Configure Credential Collectors in a Mixed Environment 147
Use FCCs and NTCs in a Mixed Environment 149
Use SCCs in a Mixed Environment 152
Configure Advanced FCC Settings 153
Specify Redirect URL Protocols with Lowercase Characters 153
Use a Special Forms Template for Passport Authentication 154
Use the safeword.fcc File for SafeWord Forms Authentication 155
How to use Forms with ACE Authentication 155
Encrypt Query String Parameters in Redirection URLs 156
FCC Directive for Encoding Query Strings of Redirect URLs 157
How to Configure Application Request Routing (ARR) for HTML Forms Authentication 157
How to Configure the FCC to Allow Windows Authentication 160
How to Allow the NTC to Encode URLs During Redirects to Protected Resources 162

Chapter 12: Agents and Password Services 171

How to Configure FCC Password Services 171
Password Services Implementations 171
FCC Password Services and URL Query Encryption 172
How to Localize FCC-based Password Services Change Forms 173
Use a Fully Qualified URL for Password Services Redirects 174
Configure SecureID Authentication with FCC Password Services 175
How to Enable User-Initiated Password Changes with FCCs 176
How to Enable User-Initiated Password Changes with FCCs (SecureURLs=Yes) 178
How to Enable User-Initiated Password Changes when using the SiteMinder X.509 Certificate and Basic Authentication Scheme 180

Chapter 13: Single Sign-On (SSO) 183

Allow Automatic Access to Resources that use the OPTIONS Method 183
How Single Sign-on Works in a Single Domain 184
Single Sign-On Across Multiple Domains 185
Hardware Load Balancers and Single Sign-On Across Multiple Cookie Domains 186
Single Sign-On and Authentication Scheme Protection Levels 188
Single Sign-on and Agent Key Management 188
How to Configure Single Sign-On 189
Restrict Cookie Provider Functions 190
Prevent Cookie Provider Replay Attacks 191
Set RequireCookies Parameter for Single Sign-On 192
Enabling Persistent Cookies for Single Sign-On 193
Specify the Cookie Domain 194
Enable IP Address Validation for Single Sign-On Environments 195
Modify the Session Update Period 196
Set Secure Cookies Across Multiple Domains 196
Ignore the Cookie Provider for Unprotected Resources 197
Ignore the Cookie Provider for POST Requests 197
Configure SecureUrls with Single Sign-on 198
Specify the Cookie Provider 199
Disable Cookie Providers 200

Chapter 14: Comprehensive Log Out 201

How Full Logoff Works 201
Configure Full Logoff 202
How to Configure Full Logoff for Single Sign-on 203
Configure Comprehensive Log Out using FCC Forms 205

Chapter 15: SSO Security Zones 207

Security Zones Overview 207
- Security Zone Definitions 208
- Security Zones Benefits 208
- Security Zone Basic Use Case 209
- User Sessions Across Security Zones 210
- Trusted Zone Order 210
- The Default Single Sign-On Zone and Trusted Zone List 212
- Request Processing with Multiple User Sessions 212
- Transitive Relationships Across Zones 213
- Other Cookies Affected by Single Sign-On Zones 213
- Single Sign-On Zones and Authorization 214

Configure Security Zones 215
- Specify the Single Sign-on Zone for the Agent 217
- The Order of Trust and Failover 218

Chapter 16: Advanced Configuration Settings 218

Agents and Proxy Servers 219
- Configure Agents that Sit behind Proxy Servers 220
- Customize the Cache-Control and ExpireForProxy Header Settings 222
- Proxy Header Usage Notes 224
- Security Considerations 225

Agents and Reverse Proxy Servers 226
- How Reverse Proxy Servers Work with SiteMinder 226
- SiteMinder Secure Proxy Server 227
- SiteMinder IIS 7.x Web Servers and Application Request Routing (ARR) 228
- SiteMinder Reverse Proxy Deployment Considerations 235

HTTP Header Settings 241
- Remove the Server HTTP Header if Using the URLScan Utility 241
- Ensure Custom Responses Comply with X-Frame Options 242

URL Settings 242
- Specify Redirect URL Protocols with Lowercase Characters 243
- Decode Query Data in a URL 243
- Set a Maximum URL Size 244

IIS Web Server Settings 244
- Configure Agents for IIS to Obtain User Credentials Without Redirecting to an NTLM Credential Collector (NTC) 244
- Record the User Name and Transaction ID in IIS Server Logs 246
- Use the NetBIOS Name or UPN for IIS Authentication 248
- Configure Agents for IIS to Support NT Challenge/Response Authentication 248
- How to Implement an Information Card Authentication Scheme 254

Configure an FCC Template for an Information Card Authentication Scheme 255
Control IIS 7.x Module Execution Order when using the SiteMinder Agent for IIS 256
Use an IIS Proxy User Account (IIS Only) 258
Enable Anonymous User Access 259
Disable Windows Security Context on Agents for IIS 259
Apache Web Server Settings 260
Use the HttpsPorts Parameter on Apache 2.x Servers 260
Use Legacy Applications with an Apache Web Agent 261
Use the HTTP HOST Request for the Port Number 262
Record the Transaction ID in Apache Web Server Logs 262
Choose How Content Types are Transferred in POST Requests 263
Restrict IPC Semaphore-Related Message Output to the Apache Error Log 263
Delete Certificates from Stronghold (Apache Agent Only) 264
Oracle iPlanet Web Server Settings 265
Restrict Directory Browsing on an Oracle iPlanet Web Server 265
Handle Multiple AuthTrans Functions for Oracle iPlanet Web Servers 266
Record the Transaction ID in Oracle iPlanet Web Server Logs 266
Domino Web Server Settings 268
Domino Agents Overview 269
Domino URL Syntax 270
Domino Aliases 271
Configure the Domino Web Agent 272
Configure Domino-Specific Agent Functions 272
Specify User Directories for Domino 272
Guidelines for Creating Policies on Domino Servers 273
Configure Policies for Domino 273
Create Rules for Domino Server Resources 274
Authenticate Users with the Domino Server 276
Authenticate as the Domino Super User 277
Authenticate as the Actual User or the Default User 278
Modify the Domino Default User and the Domino Super User 278
Use Encryptkey to Set the Domino Default or Super User 279
Force SiteMinder to Authenticate Users 280
Use a SiteMinder Header for Authentication 281
Disable Domino Session Authentication 281
Use an Anonymous SiteMinder Authentication Scheme with Domino 282
Enable a Domino Agent to Collect Credentials for Authentication 282
Map URLs for FCC Redirects with a Domino Web Agent 282
Disable URL Normailization 283
Control Access to Lotus Notes Documents 284
Convert Notes Document Names 285
Configure Full Logoff Support for Domino Agents 286

Use a Domino Agent with a WebSphere Application Server 287
Force Domino Server to Authenticate Unprotected SiteMinder Resources 287
Backward Compatibility Settings 287
Accommodate Legacy URL Encoding 288
Choose How Content Types are Transferred in POST Requests 288
Accommodate Testing Tools that do not send HOST Headers 289
Agent Setting for Federation Domains 290

Chapter 17: Performance 291

Set a Time-out for Saved Credentials 291
Web Agent Caches 292
Cache Anonymous Users 293
Set the Maximum Resource Cache Size 294
Set the Maximum User Session Cache Size 295
Control How Long Resource Enteries Remain Cached 296
Disable the Resource Cache 296
Monitoring Web Agents 296
Monitor Web Agents with the OneView Monitor 297
Use CA Wily Introscope to Monitor Web Agents 297
Ignore Unprotected Resources 298
Reduce Overhead by Ignoring File Extensions of Unprotected Resources 299
Specify Virtual Servers for the Web Agent to Ignore 300
Ignore Query Data in a URL 302
Allow Un-restricted Access to URIs 303

Chapter 18: Logging and Tracing 305

Logs of Start-up Events 305
Error Logs and Trace Logs 305
Parameter Values Shown in Log Files 307
Set Up and Enable Error Logging 308
Enable Transport Layer Interface (TLI) Logging 310
Limit the Number of Log Files Saved 310
How to Set Up Trace Logging 311
Configure Trace Logging 312
Trace Log Components and Subcomponents 314
Trace Message Data Fields 317
Trace Message Data Field Filters 319
Determine the Content of the Trace Log 320
Limit the Number of Trace Log Files Saved 322
Collect Detailed Agent Connection Data with an Agent Connection Manager Trace Log 323

Chapter 19: Troubleshooting 325

Chapter 20: Agent Error Codes 327

Agent for IIS Troubleshooting Log 327
Duplicate LLAWP Error Appears in Log File 327
Custom Error Pages not Appearing 328
Unable to initialize tracing message 329
Japanese Pages Rendered Improperly (153202, 153609) 330

Chapter 21: Agent Error Codes 331

00-0001 331
00-0002 332
00-0004 332
00-0005 332
00-0006 333
00-0007 333
00-0008 333
00-0009 334
00-0010 334
00-0011 334
00-0012 335
00-0013 335
00-0014 336
00-0015 336
00-0016 336
00-0017 337
10-0001 337
10-0002 337
10-0003 337
10-0004 338
10-0005 338
10-0007 338
20-0001 339
20-0002 339
20-0003 340
30-0026 340

Appendix A: Agent Parameters 341

List of Agent Configuration Parameters 341

Chapter 22: SiteMinder Support Matrix 347

Locate the Platform Support Matrix 347

Index 349

Chapter 1: Web Agents

This section contains the following topics:

How Web Agents Secure Resources (see page 16)
How Web Agents and the Policy Server Work Together (see page 17)
How the Agent Reads SiteMinder Cookies (see page 20)
Framework and Traditional Agent Architectures (see page 22)
Parameters Requiring a Server Restart when Changed (see page 23)
Multiple Agent for IIS Directory Structures According to Operating Environment (see page 26)

How Web Agents Secure Resources

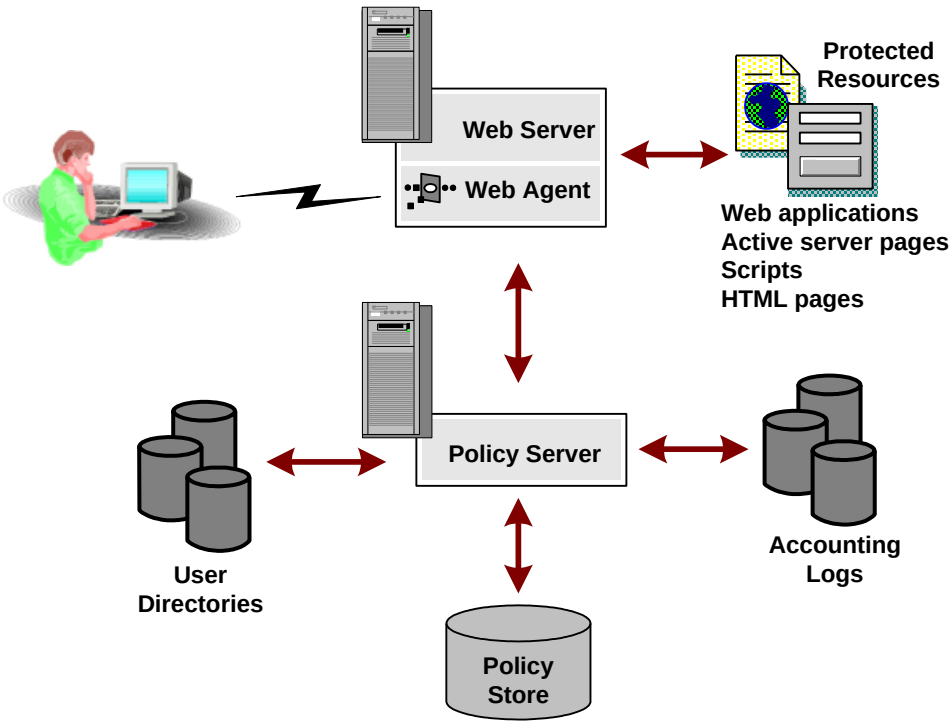
A SiteMinder Web Agent is a software component that controls access to any resource that can be identified by a URL. The Web Agent resides on a web server and intercepts requests for a resource to determine whether or not the resource is protected by SiteMinder. The Web Agent then interacts with the Policy Server to authenticate and authorize users who request access to the protected web server resources.

Web Agents perform the following tasks:

- Intercept access requests for protected resources and work with the Policy Server to determine whether or not a user should have access.
- Provide information to a Web application that dictates how content is presented to the user (policy-based personalization) and how to deliver access privileges.
- Ensure a user's ability to access information quickly and securely. Web Agents store contextual information about user access privileges in a session cache. You can optimize performance by modifying the cache settings.
- Enable single sign-on across Web servers in a single cookie domain or across multiple cookie domains without requiring users to re-authenticate.

For a list of SiteMinder Web Agents and supported Web server platforms, go to Technical Support and search for SiteMinder Support Matrix.

Web Agents reside on web servers as illustrated in the following diagram:



How Web Agents and the Policy Server Work Together

To enforce access control, the Web Agent interacts with the Policy Server, where all authentication and authorization decisions are made.

The Web Agent intercepts user requests for resources and checks with the Policy Server to see if the requested resource is protected. If the resource is unprotected, the access request proceeds directly to the web server. If the resource is protected, the following occurs:

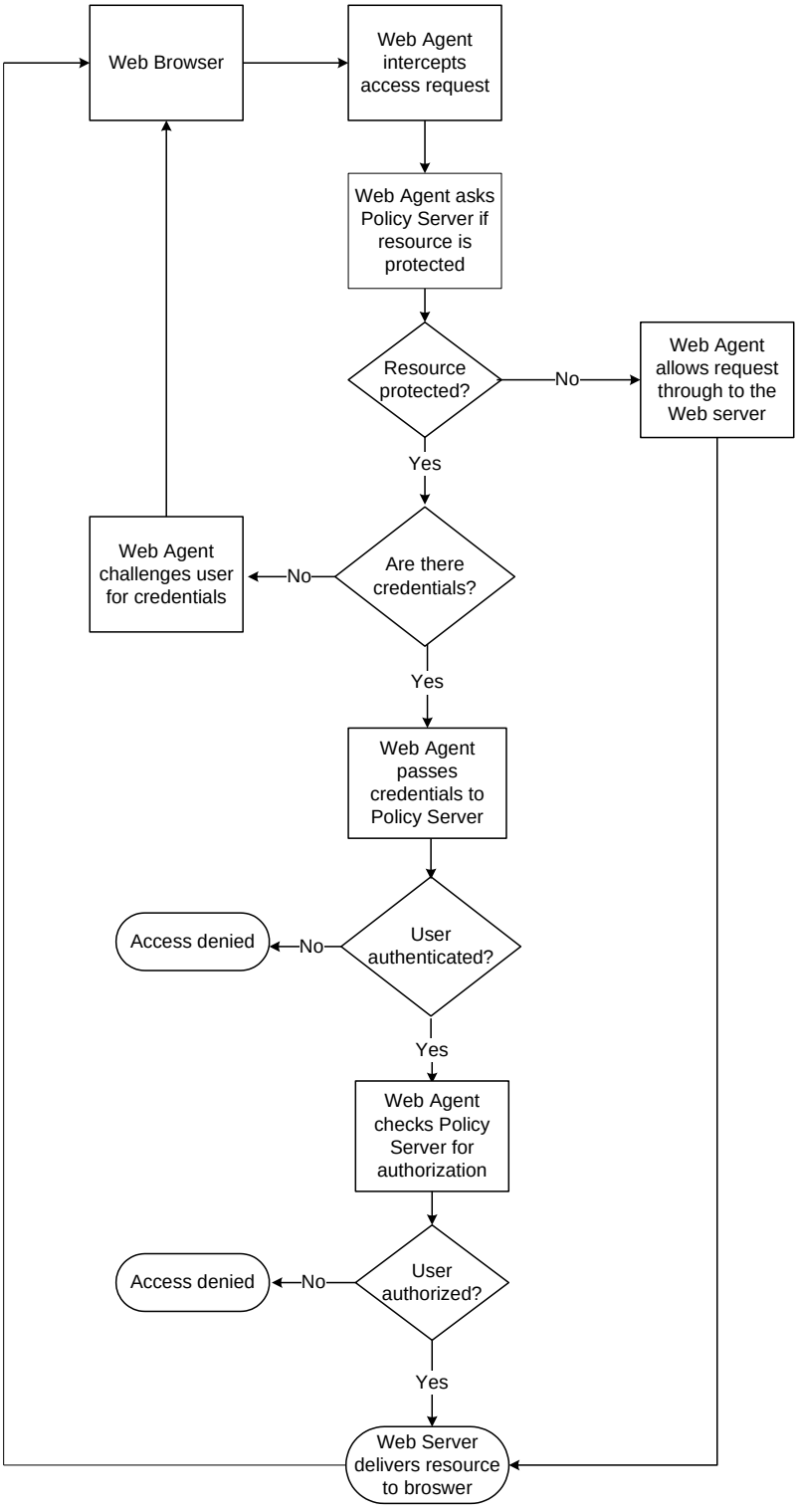
1. The Web Agent checks which authentication method is required for this resource. Typical credentials are a name and password, but other credentials, such as a certificate or a token card PIN, may be required.
2. The Web Agent challenges the user for credentials.

   The user responds with the appropriate credentials.
3. The Web Agent passes the credentials to the Policy Server, which determines if the credentials are correct.
4. If the user passes the authentication phase, the Policy Server determines if the user is authorized to access the resource. Once the Policy Server grants access, the Web Agent allows the request to proceed to the web server.

The Web Agent also receives user-specific attributes, in the form of a *response,* to enable Web content personalization and session management. A response is a personalized message or other user-specific information returned to the Web Agent from the Policy Server after authorizing the user. It consists of name-value attribute pairs that are added to HTTP headers by the Web Agent for use with Web applications. Examples of responses include the following:

- After authorizing a user to access a Web application, the Web Agent could also send information to the Web application dictating how long the user session can last.
- If the user is returning to a site where he or she previously registered, the Web Agent could return information about that user's buying preferences.

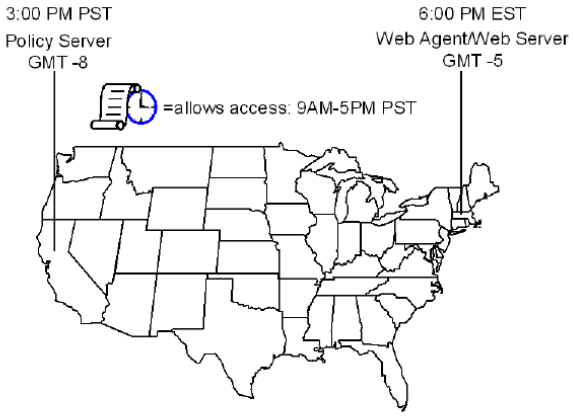
The following diagram shows the communication between the Web Agent and the Policy Server:


Web Browser
Web Agent intercepts access request
Web Agent asks Policy Server if resource is protected
Resource protected?
No
Web Agent allows request through to the Web server
Yes
Are there credentials?
No
Web Agent challenges user for credentials
Yes
Web Agent passes credentials to Policy Server
User authenticated?
No
Access denied
Yes
Web Agent checks Policy Server for authorization
User authorized?
No
Access denied
Yes
Web Server delivers resource to broswer

Considerations for Web Agents and Policy Servers in Different Time Zones

By default, the Policy Server and Web Agent calculate time relative to Greenwich Mean Time (GMT). Therefore, for each system that has a Policy Server or Web Agent installed, the system clock must be set for the time zone appropriate to that system's geographical location.

The following illustration shows how the Policy Server executes a policy relative to time. A resource is stored on a web server in Massachusetts and is protected by a Policy Server in California. The policy allows access to the resource between 9:00 AM and 5:00 PM. However, the user in Massachusetts can still access the resource at 6:00 PM because the policy is based on the Policy Server's time zone, Pacific Standard Time (PST), which is three hours behind the Web Agent's time zone, Eastern Standard Time (EST).



At 6:00 PM, the user in Massachusetts can still access the resource because it is only 3:00PM according to the Policy Server.

Note: For Windows systems, both the time zone setting and the time of day (set in the Date/Time control panel) must agree. For example, to reset a system in the U.S. from Eastern time to Pacific time, perform the following tasks in the order shown:

a. Set the time zone to Pacific time.

b. Verify that the system clock displays the correct time (three hours earlier than Eastern time).

If these settings differ, single sign-on across multiple domains and agent key management will not work properly.

How the Agent Reads SiteMinder Cookies

Web Agents use agent keys to encrypt and decrypt SiteMinder cookies so the data they contain can be read. The Agent uses the key to encrypt cookies before sending them to a user's browser and to decrypt cookies received from other Web Agents.

All Web Agents need to be aware of the same keys, and the keys must be set to the same value for all Agents communicating with a Policy Server. This rule is particularly important for Agents in a single sign-on environment. To ensure that the keys remain secure, the Policy Server performs a key *rollover*. A key rollover is the process of generating new keys, encrypting them, and distributing them to all Web Agents within a SiteMinder environment.

When a Web Agent starts up and makes a management call request, the Policy Server supplies the current set of keys. Each time the Web Agent polls the Policy Server, the agent again makes the management call. The Web Agent receives the updated keys.

The Policy Server provides the following types of keys:

Dynamic Keys

Refers to a key that is generated by a Policy Server algorithm and distributed to other connected Policy Servers and their associated Web Agents. Dynamic keys can be rolled over automatically at a regular interval, or they can be changed manually by using the Administrative UI.

Static Keys

Refers to a key that remains the same indefinitely, and can be generated by a Policy Server algorithm or configured manually. SiteMinder uses this type of key for a subset of features that requires information to be stored in cookies over extended periods.

Automated key changes ease the process of managing agent keys for large SiteMinder installations that share a single *key store*. A key store is a storage location for all key information. Policy Servers access the key store to obtain the current keys, which are then passed on to the Web Agents. For Agents that are configured for single sign-on, the key store must be replicated and shared across all Policy Servers in the single sign-on environment. Automating key changes also ensures the integrity of the keys.

Note: For more information, see the Policy Server documentation.

Web Agents and Dynamic Key Rollovers

You can use the Administrative UI to configure dynamic Agent key rollover. Web Agents poll the Policy Server for key updates at regular intervals. If keys have been updated, Web Agents pick up the changes during polling. The default polling time is 30 seconds, but you can customize it by changing the value of the PSPollInterval parameter for a Web Agent.

When a Web Agent detects that a key rollover has occurred, the Agent retrieves new values for the following Agent keys:

Old Key

Contains the last value used for the dynamic Agent key before the current value.

Current Key

Contains the value of the current dynamic Agent key.

Future Key

Contains the next value that will be used as the *current key* in a dynamic Agent key rollover.

Static Key

Contains a long-term key that the Agent can use for SiteMinder features that need to identify a user and maintain this information for long periods. Static keys also support cookie encryption for single sign-on when dynamic keys are not enabled.

Web Agents require multiple keys to preserve cookie data and ensure a smooth transition between old keys and new keys.

More information:

Change How Often an Agent Checks for Policy or Key Updates (see page 61)

Key Stores

When the Policy Server generates dynamic keys, it saves and maintains these keys in the key store. The key store is a repository from which all Policy Servers retrieve the most current keys. Web Agents obtain the current keys from the Policy Servers. The key store may be part of a SiteMinder policy store or maintained as a stand-alone key store.

Note: If an administrator issues multiple agent key rollovers in rapid succession, this action may invalidate all cookies issued for single sign-on and may disrupt single sign-on for all users currently logged in. After these users re-authenticate, single sign-on will operate normally.

Framework and Traditional Agent Architectures

All SiteMinder agents are based on *one* of the following architectures:

- Traditional agents are based on the original SiteMinder agent architecture.
- Framework Agents were introduced with SiteMinder version 5.x QMR 6.

Agent Features are primarily the same regardless of the architecture, but some minor differences exist. For example, framework agents use a different WebAgent.conf file and a LocalConfig.conf file. Traditional agents do *not* use those files.

Traditional agents are installed on the following web servers:

- Domino (all supported versions)

Framework Agents are installed on the following web servers:

- Microsoft Internet Information Services (IIS) 7.0, 7.5
- Apache 2.0.54, 2.2.x, and other Apache 2.0-based servers, such as the IBM HTTP Server and the HP Apache server
- Oracle iPlanet Web Server versions 6.1 and above.

  Note: The Oracle iPlanet Web Server was previously named "Sun Java Systems" or "SunONE".

More information:

Enable Post Preservation between Framework and Traditional Agents (see page 144)

Parameters Requiring a Server Restart when Changed

Some Agent parameters be updated dynamically. You must restart the web server to apply any changes to the following parameters:

AgentConfigObject

Defines the name of an Agent Configuration Object (stored on a policy server) in a local agent configuration file. This parameter is *not* used in Agent Configuration Objects.

Default: no default

CacheAnonymous

Specifies if the Web Agent caches anonymous user information. You may want to set this parameter in any of the following situations:

- If your web site gets mostly anonymous users and you want to store their session information.
- If your web site gets a mix of registered and anonymous users.

  You may want to disable this parameter to keep the anonymous user information from filling the cache and leaving no room for registered users.

Default: No

HostConfigFile

Specifies the path to the SMHost.conf file (in an IIS 6.0 or Apache agent) that is created after a trusted host computer has been successfully registered with a Policy server. All Web Agents on a computer share the SMHost.conf file.

Default: No default

MaxResourceCacheSize

Specifies the maximum number of entries that the Web Agent keeps in its resource cache. An entry contains the following information:

- A Policy Server response about whether a resource is protected
- Any additional attributes returned with the response

When the maximum is reached, new resource records replace the oldest resource records.

If you set this value to a high number, be sure that sufficient system memory is available.

If you are viewing Web Agent statistics using the OneView Monitor, you may notice that the value shown for the ResourceCacheCount is greater than the value you specified for the MaxResourceCacheSize parameter. This is not an error. The Web Agent uses the MaxResourceCacheSize parameter as a guideline and the values may at times differ because the MaxResourceCacheSize parameter represents the maximum number of average-sized entries in the resource cache. The actual cache entries are most likely larger or smaller than the pre-determined average size; therefore, the effective maximum number of entries may be more or less than the value specified.

Note: For Web Agents that use shared memory, such as the framework Agents, the cache is pre-allocated to a constant size based on the MaxResourceCacheSize value and will not grow.

Default: (Domino web servers) 1000

Default: (IIS and Sun Java System web servers) 700

Default: (Apache web servers) 750

MaxSessionCacheSize

Specifies the maximum number of users the Agent maintains in its session cache. The session cache stores the session IDs of users who authenticate successfully. Authenticated users accessing other resources within the realm during a session, are authenticated using the session cache instead of the Policy Server. When the maximum number is reached, the Agent replaces the oldest user records with new user records.

Base the value of this parameter on the number of users that you expect to access and use resources for a sustained period. If you set this value to a high number, verify that sufficient system memory is available.

Note: Regardless of the cache size, all entries in the session cache of the Web Agent *expire automatically* after one hour.

Default: (Domino web servers) 1000

Default: (IIS and Oracle iPlanet web servers) 700

Default: (Apache web servers) 750

PostPreservationFile

Enables the transfer of POST preservation data between Traditional and Framework Agents by specifying the path to *one* of the following POST-preservation-template files:

- tr2fw.pptemplate—Indicates that resources hosted on a server running a Traditional agent are protected by an FCC running on a Framework agent.
- fw2tr.pptemplate—Indicates that resources hosted on a server running a Framework agent are protected by an FCC running on a Traditional agent.

Default: No default

Example: *web_agent_home*/samples/forms/fw2tr.pptemplate

ResourceCacheTimeout

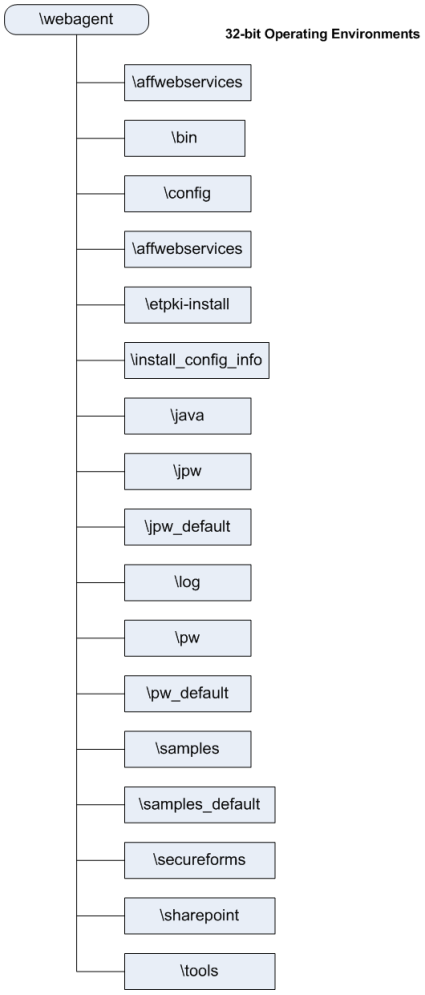
Specifies the number of seconds that resource entries remain in the cache. If a user tries to access a protected resource after the time interval has been exceeded, the Web Agent removes the cached entries and contacts the Policy server.

Default: 600 (10 minutes)

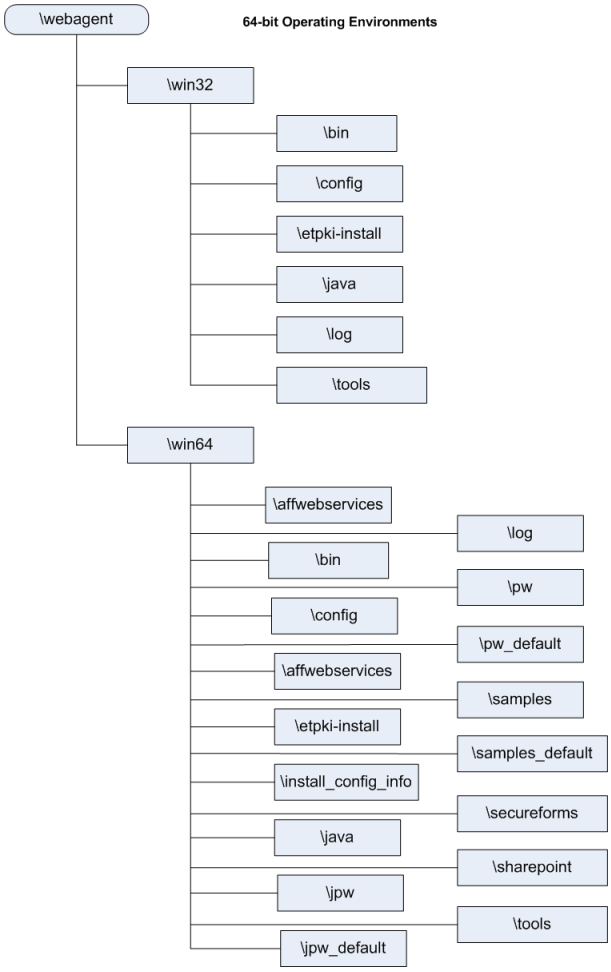
Multiple Agent for IIS Directory Structures According to Operating Environment

The directory structure added to your IIS web server for your Agent files varies according to the operating environment of your IIS web server. The following directory structures exist:

- SiteMinder Web Agents and [set AGENT value for your book]s for IIS use the directory structure shown in the following illustration:



- SiteMinder Agents for IIS installed on 64-bit operating environments use the directory structure shown in the following illustration:



Chapter 2: Agent Configuration Methods

This section contains the following topics:

Central Configuration (see page 29)
Local Agent Configuration (see page 31)
Central and Local Configuration Together (see page 40)

Central Configuration

A central agent configuration manages one or more Web Agents from an Agent Configuration Object in the Policy Server. The Agent Configuration Object that resides in the Policy Server contains the parameters used by the Web Agents. One advantage of central configuration is that you can update the parameter settings of several agents at once. Most parameter changes occur dynamically, but some Framework parameters require a web server restart after they are changed.

You create and edit an Agent Configuration Object with the Administrative UI. Each Web Agent communicating with the Policy Server must be associated with an Agent Configuration Object, but many Web Agents can use a single Agent Configuration Object.

Note: For more information about creating an Agent Configuration Object, see the Policy Server documentation.

More Information

Parameters Requiring a Server Restart when Changed (see page 23)

Implement Central Configuration

Central configuration is enabled by default. The agent uses the configuration settings from the existing Agent Configuration Object that you specified when you configured the agent with the configuration wizard. You can change the settings of the parameters to suit your needs at any time.

Follow these steps:

1. Log in to the Administrative UI.

   The Welcome screen appears.

2. Click Infrastructure, Agent Configuration Objects.

   A list of agent configuration objects appears.

3. Click the modify icon in the row of Agent Configuration Object you want.

   The Modify Agent Configuration window appears.

4. Verify that the value of the AllowLocalConfig parameter is set to no.

5. Use the Administrative UI to modify the settings of any other parameters according to your needs.

6. Click Submit.

   The Modify Agent Configuration window closes, and a confirmation message appears.

7. (Optional) Enter any comments about the change in the comment field for future reference.

8. Click Yes.

   A confirmation message appears. Central configuration is implemented. Most parameter changes occur dynamically, but some changes require a web server restart to take effect.

More information:

Parameters Requiring a Server Restart when Changed (see page 23)

Local Agent Configuration

Local Configuration

A local agent configuration manages a Web Agent using local files that are installed on the system hosting the web server. The parameter settings in the local file override any settings stored in an Agent Configuration Object on the Policy Server. The settings in the Agent Configuration Object do not change. Situations to consider local agent configuration include the following:

- When you have three Apache Web Agents, and the first two (A and B) use identical parameter settings, but you want the third Apache Agent (C) use the most of the settings from A and B while acting as a reverse proxy. To accomplish this, use central agent configuration for Apache Agents A and B, but use local configuration for Apache Agent C.

- When the Policy Server administrator is not the same person (or group) who configures an Agent. For example, the information technology department in a company maintains the Policy Server, but the finance department uses an Agent to control access to an accounting application. Someone from the information technology department enables local configuration for the Agent on the Policy Server, but another person from the finance department controls the specific configuration settings for the Agent that protects the accounting application.

Framework Web Agents use the following files for local configuration:

WebAgent.conf

Contains the core settings that the Framework Web Agent uses to start and connect to a Policy Server.

LocalConfig.conf

Contains the configuration settings for the Framework Web Agents.

Traditional Web Agents use the following file for local configuration:

WebAgent.conf

Contains all of the configuration settings for traditional Web Agents.

WebAgent.conf File Locations

The following table shows the locations of the WebAgent.conf file on various web servers:

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

Web Server	File Location
IIS	*web_agent_home*\bin\IIS
Oracle iPlanet (iPlanet/SunOne)	*Oracle_iPlanet_server_home*/https-*hostname*/config where *Oracle_iPlanet_home* is the location in which theOracle iPlanet web server is installed and *hostname* is the name of the server.
Apache, IBM HTTP Server Oracle HTTP Server	*web_server_home*/conf where *web_server_home* is the installed location of the web server
Domino	Windows: c:\lotus\domino UNIX: $HOME/notesdata

More information:

Enable a Web Agent (see page 59)
Disable a Web Agent (see page 60)

WebAgent.conf file for Framework Agents

In addition to the AgentConfigObject, HostConfigFile, and EnableWebAgent parameters, the following parameters are also added to the WebAgent.conf file of Framework Agents:

Important! Do not modify any sections of the file that refer to other SiteMinder products other than the Web Agent. However, you can change the values of the Web Agent parameters in the file.

LocalConfigFile

Specifies the location of the LocalConfig.conf file, where most of Agent configuration settings reside.

ServerPath

Identifies the web server directory (of Apache 2.0 and Oracle iPlanet web servers) to the Agent.

LoadPlugin

Specifies which plug-ins are loaded for Framework Agents. The plug-ins support different types of Agent functions. The following plug-ins are available:

HttpPlugin

Specifies whether the Web Agent operates as an HTTP agent.

Default: Enabled

SAMLAffiliatePlugin

Allows communication between the Web Agent and a SAML Affiliate Agent (if you have purchased Federation Security Services).

Default: Disabled

Affiliate10Plugin

Allows communication between the Web Agent and a 4.x Affiliate Agent.

Default: Disabled.

Limits: The SAML affiliate agent does not use this plug-in.

OpenIDPlugin

Lets the web agent use the OpenID authentication scheme (OIAS).

Default: Disabled

To enable the other LoadPlugin entries, remove the pound symbol (#) from the beginning of the line.

AgentIdFile

Specifies the path of the AgentId file which stores the unique ID string of the agent. The agent automatically generates the AgentId file, which must not be modified. Both on Windows and UNIX, the agent must have write permission to update the AgentId file. On Windows, the Web Agent configuration wizard grants the write permission automatically.

Default name: Agentid.dat

Path: WebAgent.conf directory/AgentId.dat

More Information

Manage Web Agents with Multiple Web Server Instances (see page 55)

LocalConfig.conf File Locations (Framework Agents)

When you install a Framework Web Agent, the SiteMinder installation program creates a LocalConfig.conf file in the following directory:

Windows

web_agent_home\config

UNIX

web_agent_home/config

Important! This file contains all of the default settings. Do not modify this file. We recommend creating a backup copy of this file for future reference or for recovery purposes.

When you configure the Web Agent, the configuration wizard copies the LocalConfig.conf file to the following directory:

IIS web server

web_agent_home\bin\IIS

Oracle iPlanet web server

Oracle_iPlanet_home/https-*hostname*/config

Apache web server

Apache_home/conf

The Web Agent retrieves its configuration settings from this copy of the LocalConfig.conf file.

Parameters Found Only in Local Configuration Files

For central Agent configurations, most of the parameters in the local configuration file are also in an Agent Configuration Object. The following parameters are used in the local configuration file only and are *not* found in Agent Configuration Objects:

AgentConfigObject

Defines the name of an Agent Configuration Object (stored on a policy server) in a local agent configuration file. This parameter is *not* used in Agent Configuration Objects.

Default: no default

EnableWebAgent

Activates a Web Agent and allows it to communicate with the Policy server. Set this parameter to yes only after you have finished changing all of the configuration parameters.

Default: No

HostConfigFile

Specifies the path to the SMHost.conf file (in an IIS 6.0 or Apache agent) that is created after a trusted host computer has been successfully registered with a Policy server. All Web Agents on a computer share the SMHost.conf file.

Default: No default

Implement Local Configuration

You can control whether local configuration is allowed with the following parameter:

AllowLocalConfig

Instructs the Agent Configuration Object on the Policy Server to read the local configuration file to obtain configuration parameters for the agent. This parameter is used only in Agent Configuration Objects.

Add multiple values for this parameter in the Agent Configuration Object to control which parameters can be changed in a local configuration file. When multiple values are set for this parameter, they are processed in the following order:

- If yes is used, *all* parameters can be set locally.
- No takes precedence over a list of parameters. No also overrides yes when both values are set together. This option lets you quickly disable local configuration entirely without having to remove any of the other configuration parameters from the Agent Configuration Object.

Default: No (local configuration prohibited).

Example: No, EnableAuditing, EnableMonitoring (all local configuration prohibited).

Example: No, Yes (all local configuration prohibited).

Example: EnableAuditing, EnableMonitoring (allows local control of the only the two previous parameters).

To implement local configuration

1. Log in to the Administrative UI.

   The Welcome screen appears.

2. Click the Infrastructure, Agent Configuration Objects.

   A list of agent configuration objects appears.

3. Click the modify icon in the row of the agent configuration object you want.

   The Modify Agent Configuration dialog appears.

4. Click the edit icon to the left of the AllowLocalConfig parameter.

   The Edit Parameter dialog appears.

5. Change the text in the Value field to yes, and then click OK.

   The Edit Parameter dialog closes.

6. Click Submit.

   A confirmation message appears.

7. (Optional) Enter any remarks about the change in the comment field for future reference.

8. Click Yes.

   Local configuration is enabled.

9. Open the appropriate local configuration file on your web server and change the parameter settings you want.
10. For traditional agents only, set the value of the EnableWebAgent parameter to yes.
11. Save and close the local configuration file.
12. For Framework agents only, do the following steps:
    a. Open the WebAgent.conf file.
    b. Set the value of the EnableWebAgent parameter to yes.
    c. Save and close the WebAgent.conf file.
13. Restart the web server.

    Local configuration is enabled and any updated parameters are changed.

More information:

Parameters Requiring a Server Restart when Changed (see page 23)
Enable a Web Agent (see page 59)

How to Edit an Agent Configuration File

The agent configuration file controls the settings of a locally configured Web Agent. To change those settings, use the following process:

1. Create a backup copy of WebAgent.conf (for a traditional agent) or the LocalConfig.conf file (for a Framework agent).
2. Open the original copy of the agent configuration file with a text editor.
3. Enable or disable parameters by doing any of the following tasks:
   - Removing the pound sign (#) from the beginning of the line to enable a parameter.
   - Adding the pound sign (#) to the beginning of the line to disable a parameter.
4. Change the values of parameters using the following guidelines:
   - Do not add spaces between the parameter names, the equal sign (=), and the parameter values.
   - Surround the parameter values with quotation marks.
   - The WebAgent.conf and LocalConfig.conf files are not case-sensitive. You do not have to match the case shown in the sample file that is installed with the agent.
   - Many values are shown in the file as descriptive variables, such as <Agent Name>,<IP Address>. Replace the angle brackets and text with the values you want.
   - In cases where the value is Empty, a blank is valid as the default. A default value applies only if there is no pound sign (#) preceding the parameter.
5. Set EnableWebAgent to yes only when you are done. Then save and close the file.

   All local configuration changes are effective. If you make more changes after an Agent has been enabled, restart your web server to apply those changes.

Restrict Changes to Local Configuration Parameters

With central agent configuration, you can restrict the configuration parameters which local web server administrators modify. We recommend this method when the SiteMinder administrator and the web server administrator are different people.

Follow these steps:

1. Log in to the Administrative UI.

   The Welcome screen appears.

2. Click the Infrastructure, Agent Configuration Objects.

   A list of Agent Configuration objects appears.

   Click the edit icon in the line Agent Configuration Object you want.

   The Modify Agent Configuration dialog appears.

3. Click the edit icon to the left of the AllowLocalConfig parameter.

   The Edit Parameter dialog appears.

4. Erase the text in the Value field, and then click the multivalue option button.

5. Click Add.

   An empty field appears.

6. Type the name of the parameter to which you want to allow access in the field. Separate multiple parameters with commas. Only those parameters in the list can be changed locally.

   Example: The following example shows how to allow only the EnableAuditing and EnableMonitoring parameters to be set on the local web server:

   AllowLocalConfig=EnableAuditing,EnableMonitoring

7. (Optional) Repeat Steps 5 and 6 to add more parameters.

8. Click OK.

   The Edit Parameter dialog closes, and the Modify Agent Configuration dialog appears.

9. Click Submit.

   The Modify Agent Configuration dialog closes, and a confirmation message appears.

10. (Optional) Enter any remarks about the change in the Comment field for future reference.

11. Click Yes.

    Your changes will be applied the next time the Web Agent polls the Policy Server.

Central and Local Configuration Together

If you have a large number of Web Agents that you want to configure centrally, but the settings of a few of those Web Agents need to be different than the others, you can use a combination of central and local configuration together.

For example, if you need to configure multiple cookie domain single sign-on across a SiteMinder network without configuring the Agents individually, you can use a central configuration for all of the agents, and local configuration settings for the smaller group that needs the different settings.

In the previous example, suppose the CookieDomain parameter in the Agent Configuration Object is set to example.com. However, for one Web Agent in your network, you want to set the CookieDomain parameter to .example.net, while still using all the other parameter values set in the Agent Configuration Object.

To implement the example configuration

1. With the Administrative UI, create an Agent Configuration Object with all the parameters that you want for your environment. Set the CookieDomain parameter to .example.com
2. Set the AllowLocalConfig parameter of the Agent Configuration Object to yes.
3. At one Web Agent, change *only* the local configuration file (on the web server) to use example.net as the value of the CookieDomain parameter. Do *not* modify any other parameters.

The value for the CookieDomain parameter in the lone Agent's local configuration file overrides the value in the Agent Configuration Object, while the Agent Configuration Object determines the settings for all the other parameters.

Chapter 3: Configuration Files used by the Web Agent

SiteMinder Web Agents use configuration files for certain settings. Some of these configuration files are installed on the web server with the Web Agent. Other configuration files are created the web server by the SiteMinder Web Agent Configuration Wizard where the Web Agent files are associated with the particular web server product installed on the computer hosting your web server.

For example, if you install a Web Agent on a 32-bit Windows system that runs an Apache Web Server, the Web Agent Configuration Wizard makes the changes in your Apache Web Server configuration required by the SiteMinder Web Agent..

Agent Connection Manager Configuration File

The Web Agent installation wizard installs the Agent Connection Manager Configuration file (AgentConMgr.conf), in the following location:

web_agent_home/config

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

The Agent Connection Manager Configuration file lets you create detailed trace logs about the connections the Web Agent makes while operating.

More information:

Collect Detailed Agent Connection Data with an Agent Connection Manager Trace Log (see page 323)

Connection API Configuration File

The Connection API file (conapi.conf) is used to configure services through the Connection API. These services include the OneView Monitor.

The Web Agent Installation wizard creates the Connection API Configuration file in the following location:

web_agent_home/config

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

Note: More information about using the OneView Monitor exists in the *DNA Always Current Scheduler*.

Local Agent Configuration File

The Web Agent Installation wizard installs a Local Agent Configuration File (LocalConfig.conf) in the following location:

`web_agent_home/config`

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

This file lets you set the Web Agent configuration parameters on the same web server where the Web Agent is installed, instead of using the parameter settings stored in the Agent Configuration Object on the associated Policy Server.

For IIS Web Agents, the Web Agent Configuration Wizard makes a duplicate copy of the Local Agent Configuration file in the following location:

web_agent_home\bin\IIS

More information:

Local Agent Configuration (see page 31)
LocalConfig.conf File Locations (Framework Agents) (see page 34)

Trace Configuration File

The Trace Configuration File (trace.conf) lets you configure trace logs for the following items:

- Connection API
- IPC provider
- TCP/IP transport
- Monitoring API

The Web Agent Installation Wizard creates the Trace Configuration file in the following location:

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

More information:

Restrict IPC Semaphore-Related Message Output to the Apache Error Log (see page 263)

Web Agent Trace Configuration File

The Web Agent Trace Configuration file lets you create trace logs for various aspects of the Web Agent operations. For example, you can create a trace log of any Web Agent operations related to the SiteMinder single-sign on (SSO) feature.

The Web Agent Installation wizard creates the Web Agent Trace Configuration file in the following location:

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

More information:

How to Set Up Trace Logging (see page 311)

SiteMinder Host Configuration File

The Web Agent Configuration wizard creates a Host Configuration File (SmHost.conf) on every web server on which you configure a SiteMinder Web Agent in the following location:

web_agent_home/config

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

The SmHost.conf file contains information that the Web Agent uses to make initial connections to the Policy Servers to which it is associated.

Note: For more information, see the *SiteMinder Web Agent Installation Guide*.

Web Agent Configuration File

The SiteMinder Web Agent Configuration Wizard creates a Web Agent Configuration file (WebAgent.conf) on every web server on which you configure a SiteMinder Web Agent in the following location:

web_agent_home\conf

web_agent_home

Indicates the directory where the SiteMinder Agent is installed.

Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

Default (UNIX/Linux installations): /opt/ca/webagent

This file is used to enable or disable (start or stop) the Web Agent.

For IIS Web Agents, the Web Agent Configuration Wizard makes a duplicate copy of the Local Agent Configuration file in the following location:

web_agent_home\bin\IIS

More information:

Enable a Web Agent (see page 59)
Disable a Web Agent (see page 60)

Chapter 4: Basic Agent Setup and Policy Server Connections

This section contains the following topics:

Default Settings of Web Agent Configuration Parameters (see page 49)
Set the AgentName and DefaultAgentName Values (see page 50)
Restrict Changes to Local Configuration Parameters (see page 52)
Ensure that Agent Names Match (see page 53)
Encrypt the Agent Name (see page 53)
How to Manage Web Agent and Policy Server Communication (see page 53)
Accommodate Network Latency (see page 54)
Manage Web Agents with Multiple Web Server Instances (see page 55)

Default Settings of Web Agent Configuration Parameters

The default settings for the Web Agent configuration parameters are always used unless a different value is specified.

If a parameter does not exist in the Agent Configuration Object or local configuration file, the default value is used.

Set the AgentName and DefaultAgentName Values

The AgentName parameter specifies the identity of the agent. The Policy Server uses this identity to tie policies to a Web Agent. You can define the name of an agent with the following parameters:

AgentName

Defines the identity of the web agent. This identity links the name and the IP address or FQDN of each web server instance hosting an Agent.

The value of the DefaultAgentName is used instead of the AgentName parameter if any of the following events occur:

- The AgentName parameter is disabled.
- The value of AgentName parameter is empty.
- The values of the AgentName parameter do *not* match any existing agent object.

Note: This parameter can have more than one value. Use the multivalue option when setting this parameter in an Agent Configuration Object. For local configuration files, add each value to a separate line in the file.

Default: No default

Limit: Multiple values are allowed, but each AgentName parameter has a 4,000 character limit. Create additional AgentName parameters as needed by adding a character to the parameter name. For example, AgentName, AgentName1, AgentName2.

Limits: Must contain 7-bit ASCII characters in the range of 32-127, and include one or more printable characters. Cannot contain the ampersand (&) and asterisk (*) characters. The value is not case-sensitive. For example, the names MyAgent and myagent are treated the same.

Example: myagent1,192.168.0.0 (IPV4)

Example: myagent2, 2001:DB8::/32 (IPV6)

Example: myagent,www.example.com

Example (multiple AgentName parameters): AgentName1, AgentName2, AgentName3. The value of each AgentName*number* parameter is limited to 4,000 characters.

DefaultAgentName

Defines a name that the agent uses to process requests. The value for DefaultAgentName is used for requests on an IP address or interface when no agent name value exists in the AgentName parameter.

If you are using virtual servers, you can set up your SiteMinder environment quickly by using a DefaultAgentName. Using DefaultAgentName means that you do not need to define a separate agent for each virtual server.

Important! If you do not specify a value for the DefaultAgentName parameter, then the value of the AgentName parameter requires every agent identity in its list. Otherwise, the Policy Server cannot tie policies to the agent.

Default: No default.

Limit: Multiple values are allowed.

Limits: Must contain 7-bit ASCII characters in the range of 32-127, and include one or more printable characters. Cannot contain the ampersand (&) and asterisk (*) characters. The value is not case-sensitive. For example, the names MyAgent and myagent are treated the same.

If you are configuring virtual server support, specify a value for either the AgentName or the DefaultAgentName parameter.

Follow these steps:

1. Specify an AgentName value by doing either of the following steps:
   - For central agent configurations, open the Agent Configuration Object on the Administrative UI, and then add the values that you want to the AgentName parameter.
   - For local agent configurations, open the local configuration file on your web server. Add the values that you want on separate lines in the file.
2. Specify a DefaultAgentName identity by doing either of the following steps:
   - For central agent configurations, open the Agent Configuration Object on the Administrative UI, and then add the value that you want to the DefaultAgentName parameter.
   - For local agent configurations, open the local configuration file on your web server. Add the values that you want to the DefaultAgentName parameter.

   The AgentName and DefaultAgentName values are set.

More Information

How to Set Up Virtual Server Support (see page 128)

Restrict Changes to Local Configuration Parameters

With central agent configuration, you can restrict the configuration parameters which local web server administrators modify. We recommend this method when the SiteMinder administrator and the web server administrator are different people.

Follow these steps:

1. Log in to the Administrative UI.

   The Welcome screen appears.

2. Click the Infrastructure, Agent Configuration Objects.

   A list of Agent Configuration objects appears.

   Click the edit icon in the line Agent Configuration Object you want.

   The Modify Agent Configuration dialog appears.

3. Click the edit icon to the left of the AllowLocalConfig parameter.

   The Edit Parameter dialog appears.

4. Erase the text in the Value field, and then click the multivalue option button.

5. Click Add.

   An empty field appears.

6. Type the name of the parameter to which you want to allow access in the field. Separate multiple parameters with commas. Only those parameters in the list can be changed locally.

   Example: The following example shows how to allow only the EnableAuditing and EnableMonitoring parameters to be set on the local web server:

   AllowLocalConfig=EnableAuditing,EnableMonitoring

7. (Optional) Repeat Steps 5 and 6 to add more parameters.

8. Click OK.

   The Edit Parameter dialog closes, and the Modify Agent Configuration dialog appears.

9. Click Submit.

   The Modify Agent Configuration dialog closes, and a confirmation message appears.

10. (Optional) Enter any remarks about the change in the Comment field for future reference.

11. Click Yes.

    Your changes will be applied the next time the Web Agent polls the Policy Server.

Ensure that Agent Names Match

SiteMinder rules and policies are tied to Agent names. If a request is made to a host with an Agent name that is unknown to the Policy Server, the Policy Server cannot implement policies. Therefore, the value for the Web Agent's DefaultAgentName or AgentName parameter must match the name of an Agent entry defined at the Policy Server.

You define an Agent at the Policy Server using the Administrative UI. The value you enter in the Name field of the Agent Properties dialog box is the value that must match the name defined for the DefaultAgentName or AgentName setting, whether the Web Agent is configured locally (Agent configuration file) or centrally from the Policy Server (Agent Configuration Object).

Encrypt the Agent Name

The Web Agent, by default, adds its name to the URL that redirects a user to a forms, SSL, or NTLM credential collector. You can control whether the Agent encrypts its name in the URL and whether the credential collector decrypts the name when it receives the URL with the EncryptAgentName parameter.

The default setting for the EncryptAgentName parameter is yes. You should set this parameter to no in either of the following situations:

- If a third-party application is working with the credential collector and it must be able to read the Agent name for processing.
- If you configure a Web Agent as a Forms Credential Collector (FCC) for forms authentication, and direct users to a single resource to be authenticated. The procedure to configure a single resource target requires an un-encrypted Agent name.

To encrypt the Web Agent name, set the EncryptAgentName parameter to yes.

More Information

Configure the FCC to Use a Single Resource Target (see page 138)

How to Manage Web Agent and Policy Server Communication

You can manage the communication between agents and the Policy Server using any of the following procedures:

- Accommodate network latency issues (see page 54).
- Manage agents with multiple web server instances (see page 55).

More information:

Monitoring Web Agents (see page 296)

Accommodate Network Latency

When network latency issues exist, the Web Agent cannot connect with the Policy Server. To avoid this problem, use the following parameter in the Agent Configuration Object or local configuration file:

AgentWaitTime

Specifies the number of seconds that the agent waits for the Low-level agent Worker process (LLAWP) to become available. When the interval expires, the agent tries to connect to the Policy Server.

Setting this parameter can help to resolve agent start-up errors that are related to the LLAWP connections. We recommend starting with the default value and then increasing the interval 5 seconds each time until the agent starts successfully.
If you are using local configuration, set this parameter in the WebAgent.conf file *instead of* the agent configuration object.

Default: 5

Example: Calculate a suggested value with the following formula:

(*The_number_of_Policy_Servers* x 30) + 10 = value of the AgentWaitTime parameter (in seconds).

For example, if you have five Policy Servers, then set value of the AgentWaitTime parameter to 160. [(5x30) + 10 = 160] (seconds).

Limit: (FIPS-compatability and FIPS-migration modes) minimum of 5.

Limit: (FIPS-only mode) minimum of 20.

Use a higher setting *only* if network latency issues exist. A high setting possibly causes unexpected web server behavior.

To accommodate any network latency, enable the AgentWaitTime parameter in your Agent Configuration Object or local configuration file. Then specify the number of seconds you want.

Manage Web Agents with Multiple Web Server Instances

If you configure a Web Agent on multiple web server instances, each server instance requires its own Web Agent cache, log file, and health monitoring resources. To verify that resources are unique, configure the following parameter in the WebAgent.Conf file:

ServerPath

Specifies a unique path to each web server instance when a Web Agent is configured to use multiple instances of a web server. The ServerPath creates a unique identifier for the caching, logging, and health-monitoring resources that the agents use.

This value must be an alphanumeric string unique among server instances running on the system. For example, if you have two server instances, the value of the ServerPath parameter for one instance could be MyAgent1 and the value for the second instance could be MyAgent2.

Default: Empty

Example: If there are four web server instances, each loading an agent, then set the ServerPath parameter in the WebAgent.conf file of each server instance to a unique value. You can set the ServerPath parameter to the directory where the log file of each server instance is stored, such as *server_instance_root*/logs.

To configure a Web Agent on multiple server instances, add a unique path to the ServerPath parameter in each WebAgent.conf file.

Set the ServerPath Parameter for Windows Systems

If there are multiple server instances in your Windows operating environment, specify a value for the following parameter in the WebAgent.conf file:

ServerPath

Specifies a unique path to each web server instance when a Web Agent is configured to use multiple instances of a web server. The ServerPath creates a unique identifier for the caching, logging, and health-monitoring resources that the agents use.

This value must be an alphanumeric string unique among server instances running on the system. For example, if you have two server instances, the value of the ServerPath parameter for one instance could be MyAgent1 and the value for the second instance could be MyAgent2.

Default: Empty

Example: If there are four web server instances, each loading an agent, then set the ServerPath parameter in the WebAgent.conf file of each server instance to a unique value. You can set the ServerPath parameter to the directory where the log file of each server instance is stored, such as *server_instance_root*/logs.

Note: Do not use a backslash (\) in the string you specify for the ServerPath parameter; all other characters are permitted.

The ServerPath parameter is *not* required for the following Windows platforms:

- IIS servers (there is always only one server instance).
- Apache 2.0 servers if there is only one web server instance. The parameter is supported on these systems, but it serves no purpose.
- The Oracle iPlanet or Domino web servers.

  These servers do *not* run in multiprocess mode on Windows.

Set the ServerPath Parameter for UNIX Systems

The ServerPath parameter is located in the WebAgent.conf file.

For web servers on UNIX platforms, we recommend that each server instance has its own Agent resources.

Set the ServerPath parameter for the following servers on UNIX:

- Any Apache 2.0 (including any Apache 2.0-based servers, such as IBM HTTP Server)
- Any Oracle iPlanet web server instance

Note: The ServerPath is *not* required for Domino web servers on UNIX systems.

The value you set for the ServerPath parameter must be an alphanumeric string unique among the server instances running on the system. For example, if you have two server instances, the value of the ServerPath parameter for one instance could be MyAgent1 and the value for the second instance could be MyAgent2.

Additional Configurations Requiring the ServerPath Parameter

The following list describes other use cases for requiring the ServerPath parameter:

- The Web Agent tracks shared memory using a semaphore. A semaphore is a value in operating system (or kernel) storage. Process running on the system examine this value to verify resource availability. Because the semaphore is not unique, multiple Agents would try to point to the same area of memory. Naming a server path gives the root of an instance, and the Agent finds the files that are used for creating unique keys for semaphores.
- For multiple server instances (all platforms except Windows), the Agent fails to execute one of the following operations:
  - To encrypt value of the AgentName parameter (00-0012 error).
  - To encrypt SMSESSION or SMIDENTITY cookies.
  - To update the agent encryption keys when the agent starts.
- With Apache (all platforms except Windows), the Agent does not release its six shared memory segments (semaphores) when Apache is restarted.
- If each Web Agent is configured for a different web server type on the same system, such as an Apache 2.0 server and an Oracle iPlanet server. Specify a unique ServerPath value for the configuration of each server. Different web server types *cannot* share agent resources.

Chapter 5: Starting and Stopping Web Agents

This section contains the following topics:

Enable a Web Agent (see page 59)
Disable a Web Agent (see page 60)
Starting or Stopping Most Apache-based Agents with the apachectl Command (see page 60)

More information:

WebAgent.conf File Locations (see page 32)

Enable a Web Agent

Configure your agent parameters and then enable the agent to protect the resources on the web server.

Note: *No* resources are protected until you also define policies in the SiteMinder Policy Server.

Follow these steps:

1. Open the WebAgent.conf file with a text editor.

   Note: SiteMinder r12.5 Agents for IIS installed on 64-bit Windows operating environments have *two* WebAgent.conf files. One file is associated with 32-bit Windows applications. The other file is associated with 64-bit Windows applications. Modify *both* WebAgent.conf files to start or stop the SiteMinder Agent for IIS on all 32-bit *and* 64-bit applications on a particular IIS web server.

2. Change the value of the EnableWebAgent parameter to yes.
3. Save and close the WebAgent.conf file.
4. Restart the web server (the web server itself, not the computer on which it runs).

   The Web Agent is enabled.

Disable a Web Agent

To stop the Web Agent from protecting the resources on your web server and stop communicating with the Policy Server, disable the Web Agent.

Follow these steps:

1. Open the WebAgent.conf file with a text editor.

   Note: SiteMinder r12.5 Agents for IIS installed on 64-bit Windows operating environments have *two* WebAgent.conf files. One file is associated with 32-bit Windows applications. The other file is associated with 64-bit Windows applications. Modify *both* WebAgent.conf files to start or stop the SiteMinder Agent for IIS on all 32-bit *and* 64-bit applications on a particular IIS web server.

2. Change the value of the EnableWebAgent parameter to no.
3. Save and close the WebAgent.conf file.
4. Restart the web server (the web server itself, not the computer on which it runs).

   The Web Agent is disabled.

More information:

WebAgent.conf File Locations (see page 32)

Starting or Stopping Most Apache-based Agents with the apachectl Command

Starting or stopping most Apache-based agents with the apachectl command on UNIX or Linux operating environments requires setting the environment variables for the product first.

Note: The Apache-based agents do *not* support the apachectl -restart option. This procedure does *not* apply to Apche-based IBM HTTP servers. Use this procedure instead.

Follow these steps:

1. For UNIX/Linux operating environments, set the environment variables by running the following script:

   ```
   ./ca_wa_env.sh
   ```

2. Use *one* of the following commands:

   ```
   apachectl -stop
   ```

   ```
   apachectl -start
   ```

Chapter 6: User Protection

Change How Often an Agent Checks for Policy or Key Updates

The Web Agent polls the Policy Server at regular intervals to check for the following items:

- Updated management information
- Updated policies
- Dynamically updated agent keys

You can change this interval according to your needs with the following parameter:

PSPollInterval

Specifies how often (in seconds) the Web Agent contacts the Policy Server to retrieve information about policy changes or dynamically updated keys. Higher numbers (longer intervals) decrease network traffic. Lower numbers (shorter intervals) increase network traffic.

Default: 30

Limit: 1

To change how often a Web Agent checks the Policy Server for updates, change the number of seconds in the PSPollInterval parameter.

Important! Increasing the PSPollInterval parameter also affects how quickly the Agents enforce SiteMinder policy changes. For example, suppose you change a Policy to revoke access for a terminated employee at 10:30, and your PSPollInterval parameter has a value of 3600 (the number of seconds in an hour). The Web Agents would not enforce the changed policy until as late as 11:30.

More information:

Web Agents and Dynamic Key Rollovers (see page 21)

User Tracking and URL Monitoring

SiteMinder agents can track users and can monitor URLs using the parameters described in the following procedures:

- Track user identity across anonymous realms (see page 62).
- Track user activities or application usage (see page 63).
- Overview of URL monitoring (see page 63).

Track User Identity Across Anonymous Realms

When an anonymous user accesses resources, that user is assigned an SMIDENTITY (anonymous) cookie. When the user moves to another domain, the user is challenged, logs in successfully, and is assigned an SMSESSION (logged in) cookie.

As this user accesses protected and "anonymous" resources, that is, resources in a realm that do not require a user to present credentials, the user may enter a domain that contains both cookies for a user. For resources protected by Web Agents starting at 5.x QMR 3 , the Web Agent uses the SMSESSION cookie to identify the user, not the SMIDENTITY cookie.

If the user goes from a thoroughly upgraded domain to a domain where older Agents use the SMIDENTITY cookie to identify the user, the cookie used depends on the version of the Web Agent handling the request.

Regarding separate cookie domains, when a master cookie domain contains protected resources and a second domain contains anonymous resources, a user who does the following tasks continues to be treated as an anonymous user in the anonymous domain:

1. Accesses the anonymous domain first
2. Moves to the master domain and logs in
3. Moves back to the anonymous domain

Track User Activities or Application Usage with Auditing

You can you can measure how often applications on your web site are used, or track user activities with auditing. Auditing is controlled with the following parameter:

EnableAuditing

Specifies whether the Web Agent logs all successful authorizations that are stored in the user session cache. When enabled, user authorizations are logged even when the Web Agent uses information from its cache instead of contacting the Policy Server. Web Agents log user names and access information in native web server log files when users access resources.

Default: No

Both the Policy Server and Web Agent audit user activity. The Web Agent sends a message to the Accounting service each time a user is authorized from cache to access resources. This action ensures that the Accounting service is tracking successful authorizations for the Web Agent and the Policy Server. If the Web Agent cannot successfully send an audit message to the Accounting service for an authorization, access to the resource is denied. You can then run a SiteMinder activity report from the Administrative UI. The reports from the Policy Server show user activity for each SiteMinder session.

Note: For more information, see the Policy Server documentation.

To track user activity or application usage with auditing, set the value of the EnableAuditing parameter to yes.

URL Monitoring Overview

The Web Agent can prevent attacks by malicious users trying to halt normal operation of a Web site or circumvent a site's security mechanisms to gain illegal access to information.

The Web Agent monitors URLs in resource requests and enforces the security policies for these resources. SiteMinder Web Agents interpret and parse URLs differently from the web servers where the resources reside. These differences can result in subtle performance and security issues that potentially allow unauthorized users to gain access to resources. You need to consider these issues in the design of your Web site and the configuration of the SiteMinder Web Agent.

Help Prevent Attacks

SiteMinder agents can help prevent attacks using the parameters described in the following procedures:

- Protect web sites against cross-site scripting (see page 65).
- Configure the agent to prevent cross-site scripting (see page 65).
- Protect J2EE applications against cross-site scripting (see page 66).
- Override the default CSS character set (see page 66).
- Prohibit certain characters in the query string portion of a URL (see page 67).
- Prohibit certain characters in a URL (see page 69).
- Prohibit certain characters in a form.
- Help prevent DNS denial of service attacks (see page 72).
- Protect resources or files that do not have extensions (see page 72).
- Disable POST preservation (see page 73).
- Secure applications (see page 73).

Protect Web Sites Against Cross-Site Scripting

A Cross Site Scripting (CSS) attack can occur when the input text from the browser (typically, data from a post or data from query parameters on a URL) is displayed by an application without being filtered for characters that may form a valid, executable script when displayed at the browser.

An attack URL can be presented to unsuspecting users. When a user clicks on the URL, an application may return a display to the browser that includes the input characters, along with an error message about bad parameters on the query string. The display of these parameters at the browser can lead to an unwanted script being executed on the browser.

For example, when a user types news into a search engine web page, the application normally might return a blank field, or a response, such as:

Your search for news returned the following:

In response to an attack URL, the browser might receive a response, such as:

news<script>BadProgram</script>

The BadCSSChars parameter does not interpret the double quotation mark (") if it is entered as an ASCII character. To include the double quotation mark as a bad cross-site scripting character, enter the hexidecimal equivalent of the ASCII character, which is %22. For example:

BadCSSChars="%22"

Configure the Web Agent to Check For Cross Site-Scripting

To instruct the Web Agent to check a URL for characters that may be part of an executable script, set the CSSChecking parameter to yes. By enabling this parameter, the Web Agent scans a full URL, including the query string, for escaped and unescaped versions of the following default character set:

- left and right angle brackets (< and >)
- single quote (')

Protect J2EE Applications against Cross-Site Scripting Attacks

You can prevent an attacker from bypassing cross-site scripting protection by using noncanonical (overlong) unicode characters in a request.

Follow these steps:

1. Set the value of the CSSChecking parameter to yes.
2. Set the value of the following parameter to yes:

   DisallowUTF8NonCanonical

   Prevents attackers from sending noncanonical (overlong) Unicode (utf-8) characters in requests and attempting to bypass cross-site scripting protection. When the value of this parameter is yes, the agent blocks requests for URLs containing noncanonical (overlong) Unicode characters.

   Default: No

Override the Default CSS Character Set

To override the default cross-site scripting character set, enter a character set of your choice for the BadCSSChars parameter. Include the entire string of characters that you want. For example, if you set the BadCSSChars parameter to <,>, the Web Agent scans only for the left and right angle brackets.

If the Web Agent detects a problem related to the character set, it returns an Access Denied message to the user, and the logs the following message in the Agent error log:

> Caught Possible Cross Site Scripting Violation in URL. Exiting with HTTP 403 ACCESS FORBIDDEN.

Some applications require the use of the quote characters in the query string, irrespective of the web server platform. For example, some Domino applications, such as iNotes Web Access, require the use of single quotes.

To use applications that require quotes in the query string, remove quotation marks from the BadCssChars parameter.

If you leave do not use this parameter, the Web Agent checks for the default character set.

Note: For more information about cross-site scripting, go to CERT Advisory.

Specify Bad Query Characters

To prevent certain characters the query string portion of a URL, set the following parameter:

BadQueryChars

Specifies characters that the Web Agent prohibits in the query string portion (following the '?') in a URL.

Default: Empty (any characters allowed in query strings)

Limits:

- The default hexadecimal numbers apply to English characters. For other languages, remove any hexadecimal values that correspond to the characters of the language that you want to allow. Examples of such languages include (but are not limited to), Brazilian Portuguese, French, Japanese, and Chinese.
- You can specify characters literally. You can also enter the URL-encoded form of that character. For example, you can enter the letter a, or you can enter the encoded equivalent of %61.
- You can specify a maximum number of 4096 characters (including commas that are used for separating characters).
- You can specify ranges of characters that are separated with hyphens. The syntax is: *starting_character-ending_character*. For example, you can enter a-z as a range of characters.
- Specify any quotation marks (") with the URL-encoded equivalent of %22. Do *not* use ASCII.

Example: %25 blocks URL-encoded characters in queries.

Web Agents search for prohibited characters in query strings by comparing the characters in the query string of the URL with the ASCII values of the characters defined in the BadQueryChars parameter. For an example, see the following process;

1. The BadQueryChars parameter contains the URL-encoded value for the percent symbol (%) as shown in the following example:

   ```
   %25
   ```

2. The Web Agent receives an HTTP request that contains the following query string:

   ```
   xxx=%0d
   ```

3. The Web Agent examines the URL in the previous example, but does *not* decode the URL-encoded values. For example, the Web Agent interprets the previous example (in Step 2) as the literal string %0d, and *not* as a carriage return.
4. The Web Agent examines the values in the BadQueryChars parameter, and converts them to their ASCII values. For example, the %25 in Step 1 is converted to a percent symbol (%).
5. The Web Agent compares each character in the URL against the decoded ASCII values from the BadQueryChars parameter.
6. The Web Agent blocks the request, because the ASCII percent symbol (%) exists in both of the following places:
   - The query string of the URL.
   - The decoded (ASCII) value in the BadQueryChars parameter.

To block certain characters from query strings, set the value of the BadQueryChars parameter to include the characters you want to block.

Specify Bad URL Characters

You can list a set of character sequences that cannot be part of a URL request. These are treated by the Agent as bad URL characters. The Web Agent will refuse URL requests that contain any of the characters or strings of characters that you include in this list. The checking is done on the URL before the "?" character. The Web Agent rejects URL requests that include such characters because a malicious Web client might use such characters to evade SiteMinder rules.

When a Web Agent refuses a URL request containing a Bad URL character, the web server responds with one of the following messages:

- Internal Server Error
- Web Page not Found (404) Error

Check your Web Agent logs for information on how the Agent is handling requests.

You specify the characters with the following parameter:

BadUrlChars

Specifies the character sequences that cannot be used in URL requests. The Web Agent checks the characters in the URL that occur before the "?" character against the list in this parameter. If any of the specified characters are found, the Web Agent rejects the request.

You can specify the following characters:

- a backward slash (\)
- Two forward slashes (//)
- Period and a forward slash (./)
- Forward slash and a period (/.)
- Forward slash and an asterisk (/*)
- An asterisk and a period (*.)
- A tilde (~)
- %2d
- %20
- %00-%1f
- %7f-%ff
- %25

Separate multiple characters with commas. Do *not* use spaces.

You can use the bad URL characters in CGI parameters if the question mark (?) precedes the bad URL characters.

Default: //,./,/.,/*,*.,~,\,%00-%1f,%7f-%ff,%25

Limits:

- The default hexadecimal numbers apply to English characters. For other languages, remove any hexadecimal values that correspond to the characters of the language that you want to allow. Examples of such languages include (but are not limited to), Brazilian Portuguese, French, Japanese, and Chinese.
- You can specify characters literally. You can also enter the URL-encoded form of that character. For example, you can enter the letter a, or you can enter the encoded equivalent of %61.
- You can specify a maximum number of 4096 characters (including commas that are used for separating characters).
- You can specify ranges of characters that are separated with hyphens. The syntax is: *starting_character-ending_character*. For example, you can enter a-z as a range of characters.
- Specify any quotation marks (") with the URL-encoded equivalent of %22. Do *not* use ASCII.

To specify Bad URL characters, edit the value of the BadURLChars parameter to include the characters that you want to block.

Note: When configuring the Apache 2.0 Reverse Proxy Server and Outlook Web Access (OWA), be sure to turn off the BadURLChars parameter. OWA allows unrestricted characters in the email subject that might be listed in the BadURLChars parameter.

Enable Bad Form Characters

The following characters are commonly used in cross-site scripting attacks:

- Left and right brackets (< >)
- ampersand (&)
- quotation marks (")

If you want to use scripting code for presenting forms to a user during an authentication challenge, enable the following parameter to configure the Web Agent to block any special characters before sending them to an HTML form:

BadFormChars

Specifies the characters that the Web Agent blocks before using them as output on a form. If enabled and if the agent name part of the URL has one or more characters that are specified in this parameter, then the login page returns the following error message:

```
Internal Server Error
```

Default: Disabled (characters are not blocked)

Examples: <, >, &, %22

Limits:

- You can specify the characters literally.
- You can specify a maximum number of 4096 characters (including commas that are used for separating the characters).
- You can specify ranges of characters that are separated with hyphens. The syntax is: starting_character-ending_character. For example, you can enter a-z as a range of characters.
- Specify the quotes (") with the URL-encoded equivalent of %22. Do not use ASCII.

Follow these steps:

1. Log in to the Administrative UI.
2. Open the Agent Configuration Object in which you want to enable this parameter.
3. Enable the BadFormChars parameter by removing the # character in front of it.

   The BadFormChars parameter is enabled with the default values.
4. (Optional) Remove any characters that you do not want to use from the list. You can add any other character to this list. Verify that the characters are separated from one another by a comma.

Help Prevent DNS Denial Of Service Attacks

If a web server receives HTTP requests with false IP addresses, the Web Agent tries to resolve the IP addresses to fully qualified domain names. For large volumes of HTTP requests, a denial-of-service condition could affect the Web Agent and possibly the DNS servers. The following parameter controls whether the Web Agent performs DNS lookups:

DisableDNSLookup

Prevents the Web Agent from performing DNS lookups.

Follow these steps:

1. Verify that the DisableDNSLookup parameter does *not* end with an s. Some earlier versions of the ACO templates and LocalConfig.conf files possibly contain this error. The correct parameter ends with a p.
2. Set the value of the DisableDNSLookup parameter to yes.

Important! When the value of this parameter is set to yes, fully qualified domain names are required for cookie-based functions to work properly.

Protect Resources Without Extensions

To prevent unauthorized users from gaining access to resources without extensions, you can use the following parameter:

OverrideIgnoreExtFilter

Specifies a list of strings you want the Web Agent to match against all URIs. This helps you protect resources whose extensions are normally ignored by the Web Agent, or any files or applications that do not have extensions. If the URI matches one of the strings in the list, the Web Agent checks with the Policy Server to determine if the resource is protected.

It is better to specify more general strings instead of exact paths. You can also include a partial string to protect a group of resources. For example, the string /servlet/ protects the following resources:

- /dira/app1/servlet/app
- /dirb/servlet/app1
- /dirc/mydir/servlet/app2

Default: No default

To protect resources without extensions, add strings for the resources (without periods) that you want to protect to the value of the OverrideIgnoreExtFilter parameter. If you are using an Agent Configuration Object, use the multi-value option to add the strings. If you are using a local configuration file, add each string on its own line.

Disable POST Preservation

If you do not need to use POST preservation, you may disable it with the following parameter:

PreservePostData

Specifies whether the Web Agent preserves POST data when redirecting requests. When the user is challenged for advanced authentication, such as forms or certificate authentication, the post data is preserved during the authentication phase.

Default: Yes

To disable POST preservation, set the value of the PreservePostData parameter to no.

Secure Applications

An unauthorized user can append a false file name that contains an extension that the Web Agent is configured to ignore to the end of a URL . The Agent then allows the unauthorized user access to the resource. To have the Web Agent deny access to such attempts, use the following parameter:

SecureApps

Prevents the Agent from authorizing URLs from an unauthorized user. If your Web Agent is configured to ignore requests for files ending with certain extensions, an attacker may attempt to access resources by creating a false URL.

For example, if you have a resource with the following URL:

```
/scripts/myapp
```

An attacker may attempt to gain access by creating a false URL like the one in the following example:

```
/scripts/myapp/junk.jpg
```

If the value of the SecureApps parameter is set to no, the request for /scripts/myapp/junk.jpg would be automatically authorized if the Web Agent was set to ignore requests for .jpg files.

If the value of the SecureApps parameter is set to yes, the Web Agent attempts to discover if the resource is legitimate or if the URL is false.

Default: No

To secure applications, set the value of the SecureApps parameter to yes.

Verify IP Addresses

SiteMinder agents can verify IP addresses using the parameters described in the following procedures:

- Resolve agent identity by IP address (see page 74).
- Compare IP addresses to prevent security breaches (see page 75).

Resolve Agent Identity by IP Address

On virtual web servers, when IP addresses and host names are used to resolve the Agent name, the Web Agent can potentially use an incorrect value for AgentName to evaluate the request. This situation would allow unauthenticated users to access protected resources.

You can force the Web Agent to resolve the Agent name based on the physical IP address of the virtual server, with the following parameter:

UseServerRequestIp

Instructs the Web Agent to resolve the AgentName according to the physical IP address of a virtual web server. Use this parameter to increase security if a web server uses IP addresses for virtual server mappings. If this parameter is set to no, the Web Agent resolves the AgentName according to the host name in the HTTP Host header of the client's request.

For Domino servers, this parameter is supported only for Domino 6.x. If this parameter is enabled for an Agent on other Domino versions, the Web Agent uses the default Agent name.

For IIS Web Agents configured for SSL communication and virtual hosts, you must set this parameter to yes. IIS does not allow virtual host mappings using host names with SSL enabled.

Default: No

To resolve a Web Agent's identity using the IP Address, set the UseServerRequestIp parameter to yes.

Compare IP Addresses to Prevent Security Breaches

An unauthorized system can monitor packets, steal a cookie, and use that cookie to gain access to another system. To prevent a breach of security by an unauthorized system, you can enable or disable IP checking with persistent and transient cookies.

The IP checking feature requires agent to compare the IP address stored in a cookie from the last request against the IP address contained in the current request. If the IP addresses do not match, the agent rejects the request.

The two parameters that are used to implement IP checking are PersistentIPCheck and TransientIPCheck. Set them as follows:

- If you enabled PersistentCookies, set PersistentIPCheck to yes.
- If you did not enable PersistentCookies, set TransientIPCheck to yes.

SiteMinder identity cookies are unaffected by IP checking.

More Information

Set Persistent Cookies (see page 78)
Control Identity Cookies (see page 77)

SiteMinder Browser Cookies

To manage the cookies that are associated with the SiteMinder agent, use the parameters in any of the following procedures:

- Require cookies for the basic authentication scheme (see page 76).
- Safeguard information in cookies with the HTTP-only attribute (see page 76).
- Set secure cookies within a domain (see page 77).
- Control identity cookies (see page 77).
- Set persistent cookies (see page 78).
- Specify the cookie path for agent cookies (see page 79).
- Force the agent to use a cookie domain (see page 81).
- Implement cookie domain resolution (see page 82).
- How cookie path scope settings work (see page 82).

Require Cookies for Basic Authentication

You can control whether SiteMinder requires cookies with the following parameter:

RequireCookies

Specifies whether SiteMinder requires cookies. SiteMinder requires cookies for the following functions:

- Securing single sign-on environments.
- Enforcing session timeouts.
- Enforcing idle timeouts.

When the value of this parameter is yes, the agent requires one of the following cookies to process HTTP requests:

- SMCHALLENGE
- SMSESSION

When the value of this parameter is no, the following conditions could occur:

- Users are challenged for credentials unexpectedly.
- Timeouts are not strictly enforced.

Important! If the agent requires cookies, instruct your users to accept HTTP cookies in their browsers. Otherwise, the users are denied access to all protected resources.

Default: Yes

To require cookies, set the value of the RequireCookies parameter to yes.

Safeguard Information in Cookies with HTTP-Only Attribute

To help protect against cross-site scripting attacks, you can make the Web Agent set the HTTP-Only attribute for any cookies it creates using the following parameter:

UseHTTPOnlyCookies

Instructs the Web Agent to set the HTTP-only attribute on the cookies it creates. When a Web Agent returns a cookie with this attribute to a user's browser, the contents of the cookie cannot be read by a script, even a script from the web site which originally set the cookie. This helps prevent any sensitive information in the cookie from being sent to an unauthorized third party through a script.

Default: No

To safeguard the information in cookies, set the value of the UseHTTPOnlyCookies parameter to yes.

Set Secure Cookies

You can specify that session cookies are only sent between a protected web server and the requesting browser over secure (HTTPS) connections using the following parameter:

UseSecureCookies

Sends cookies to web servers using secure (HTTPS) connections. Enable this parameter to increase security between browsers and web servers.

When this setting is enabled, users in single sign-on environments who move from an SSL web server to a non-SSL web server will have to reauthenticate. Secure cookies cannot be passed over traditional HTTP connections.

Default: No

To send cookies over SSL connections, set the UseSecureCookies parameter to yes.

More information:

Set Secure Cookies Across Multiple Domains (see page 196)

Control Identity Cookies

The TransientIDCookies parameter specifies whether or not the Agent identity cookie (SMIDENTITY) is transient or persistent.

Persistent cookies are written to a client system's hard disk. Prior to Web Agent 5.x QMR1, persistent cookies remained valid for 7 days. Beginning with Web Agent 5.x QMR1, persistent cookies remain valid for the configured maximum session timeout plus 7 days. (The maximum session timeout is set in the Administrative UI.) Typically, a persistent cookie is deleted from a Web browser's cookie file after the cookie expires; however, browsers may handle persistent cookies differently. By default, the Web Agent does not use persistent cookies. It uses transient cookies.

If you want to use single sign-on for multiple browser sessions, use persistent cookies. If you set persistent cookies, a user can end their browser session before a SiteMinder session expires then start a new browser session and still have single sign-on capability.

Whereas persistent cookies are written to a hard disk, transient cookies are not written to the hard disk and they are not subject to configured session time-outs. Transient cookies will remain in your cookie folder.

Set TransientIDCookies to no, if you want the identity cookie to be persistent. Leave the default value set to yes, if you want the identity cookie to be transient.

Be sure to set the corresponding IP Check.

More Information

Compare IP Addresses to Prevent Security Breaches (see page 75)

Set Persistent Cookies

If you want to use single sign-on for multiple browser sessions, use persistent cookies. The following steps describe one possible use for persistent cookies:

1. Users authenticate with SiteMinder, but end their browser sessions before the SiteMinder session expires.
2. Users start new browser sessions later, but the persistent cookie maintains their single-sign on capability.

Persistent cookies remain valid for the configured maximum session time-out *plus* seven days. Many browsers delete the cookie file of the web browser after the cookie expires. Some browsers possibly handle persistent cookies differently.

Follow these steps:

1. Set the PersistentCookies parameter to yes.

   The SMSESSION cookies are persistent.
2. Set the TransientIDCookies parameter to no.

   The SMIDENTITY cookies are persistent.

Specify the Cookie Path for Agent Cookies

When a Web Agent creates a cookie, the web agent automatically uses the root (/) directory as the cookie path. The domain and path attributes of cookies are compared to the URL of a request. If the cookie is valid for the domain and the path, the client sends the cookie to the server. When the cookie path uses the root value, the client sends the cookie to the server with all requests in the domain.

You can set SiteMinder cookies to a given set of paths to eliminate the web traffic caused when cookies are sent for unprotected resources. For example, if a cookie path is set to /mypackage, the client only sends the cookie for requests in a particular package in the domain.

To specify the cookie path for agent cookies

1. Open your Agent Configuration Object or your local agent configuration file.
2. Set the Cookie Path for the Cookie Provider in the following parameter:

   MasterCookiePath

   Specifies the path for the primary-domain session cookies created by the cookie provider. For example, if this parameter is set to /siteminderagent, all session cookies that the cookie provider creates will have the /siteminderagent path. If this parameter is not set in the Cookie Provider Agent, the default value is used.

   Default: / (root)
3. Set the cookie path for the secondary agents in the following parameter:

   CookiePath

   Specifies the cookie path for the following secondary agent browser cookies:

   - xxSESSION
   - xxIDENTITY
   - xxDOMINODATA
   - xxCHALLENGE (including SSL_CHALLENGE_DONE)
   - xxDATA
   - xxSAVEDSESSION

   For example, setting this parameter to /BasicAuth, all of the secondary agents in the previous list are created using /BasicAuth as the path. If not specified, the default value is used.

   The CookiePath is *not* added to credential cookies (such as xxxxCRED) to maintain backwards compatibility with 4.x agents.

   The following cookies will *always* use the root (/) path:

   - ONDENIEDREDIR

- TRYNO

If the CookiePathScope parameter is greater than zero, the CookiePath parameter settings are overriden.

Default: / (root)

4. (Optional) If you want the Web Agent to extract the cookie path from the URL instead of using the CookiePath value, set the following parameter to a number greater than zero:

CookiePathScope

Specifies the scope of the cookie path for the following secondary agent cookies:

- xxSESSION
- xxIDENTITY
- xxDOMINODATA
- xxCHALLENGE (including SSL_CHALLENGE_DONE)
- xxDATA
- xxSAVEDSESSION

Using a CookiePathScope greater than zero in this parameter overrides the setting of the CookiePath parameter.

Default: 0

More information:

Configure Full Logoff (see page 202)

Force the Cookie Domain

Using fully qualified domain names helps ensure that cookies work properly. You can force the agent to append its cookie domain to the host name in a URL request that meets either of the following conditions:

- The request does not specify a domain.
- The request contains only an IP address.
- Agents can be forced to use a cookie domain by setting the following parameters:

  ForceCookieDomain

  Forces the Web Agent to append its cookie domain to the host name in a URL request that does not specify a domain or contains only an IP address. This parameter works together with the ForceFQHost parameter for added functionality.

  Default: No

  ForceFQHost

  Forces an agent to use a fully qualified domain name. This parameter uses configured Domain Name System (DNS) services to append the cookie domain to the host name in a URL request through DNS services and not an Agent. If the Web Agent receives a request containing a partial URL, then the agent redirects the request back to the same destination resource specified in the original URI. The redirect request uses the fully qualified host name, which the agent determines using the configured DNS services. Use this parameter with the ForceCookieDomain parameter for added functionality.

  Default: No

  Example: When the agent receives a request from http://host1/page.html, it responds with http://host1.myorg.com/page.html. If the agent receives a request such as http://123.113.12.1/page.html, it responds with http://host1.myorg.com/page.html.

  Note: These examples only work with proper DNS lookup tables. Request containing partial domain names that DNS cannot resolve could possibly generate errors.

Follow these steps:

1. Set the value of the ForceCookieDomain parameter to yes.
2. Set the value of the ForceFQHost parameter to yes.

   The agent appends its cookie domain to the host name when necessary.

Implement Cookie Domain Resolution

To implement automatic domain resolution, comment out the CookieDomain parameter or set it to none to cause the Web Agent to create cookies that are good only for the server from which they were issued.

You can further define the cookie domain by adding a value to the CookieDomainScope parameter. The scope determines the number of sections, separated by periods, that make up the domain name. (A domain always begins with a ".")

A CookieDomainScope value of 0 instructs the agent to use the most specific scope for a given host. A value of 1 (resulting, for example, in a cookie domain of .com) is not allowed by the HTTP specification. The value 2 instructs the agent to use the most general scope.

The following table shows some domain names and CookieDomainScope values.

Domain Name	Cookie Domain Scope value	Cookie Domain
server.myorg.com	2	.myorg.com
server.division.myorg.com	3	.division.myorg.com
	2	.myorg.com
server.subdivision.division.myorg.com	4	.subdivision.division.myorg.com
	3	.division.myorg.com
	2	.myorg.com

For example, the domain division.myorg.com has a scope of 3. By default, the Web Agent assumes a scope of 2; cookie domains cannot have a scope of 1.

How CookiePathScope Settings Work

The following table shows how the value of the CookiePathScope parameter works with the following settings:

- A URL such as http://*fqdn*/path1/path2/path3/path4/index.html
- A CookiePath parameter value of /BasicA

If the CookiePath value is:	And the CookiePathScope value is:	Then the following path is used:
/BasicA	0	/BasicA

If the CookiePath value is:	And the CookiePathScope value is:	Then the following path is used:
/BasicA	1	/Path1
/BasicA	2	/Path1/Path2
/BasicA	3	/Path1/Path2/Path3
/BasicA	4	/Path1/Path2/Path3/Path4
/BasicA	5	/Path1/Path2/Path3/Path4
/BasicA	99	/Path1/Path2/Path3/Path4
/ or "undefined"	0	/
/ or "undefined"	1	/Path1
/ or "undefined"	2	/Path1/Path2
/ or "undefined"	3	/Path1/Path2/Path3
/ or "undefined"	4	/Path1/Path2/Path3/Path4
/ or "undefined"	5	/Path1/Path2/Path3/Path4
/ or "undefined"	99	/Path1/Path2/Path3/Path4

These settings also affect simple SSO. For example, if the value of the CookiePathScope is set to 1 or higher,users will get challenged for credentials for both /BasicA/Index.html and /BasicB/Index.html since the SESSION cookie with a path /BasicA will not be valid for /BasicB/Index.html request.

Configure Support for SDK Third-Party Cookies

If you use non-SiteMinder Web Agents in your organization, you can configure them to support single sign-on with the following parameter:

AcceptTPCookie

Allows the Web Agent to accept session (SMSESSION) cookies created by third-party (non-SiteMinder) Web Agents. Third-party agents generate and read SMSESSION cookies using the SiteMinder SDK.

Default: No default

Note: For more information, see the SiteMinder Developer's documentation.

To allow the Web Agent to accept session cookies created by non-SiteMinder Web Agents, set the AcceptTPCookie parameter to yes.

Define HTTPS Ports

If you are using an SSL connection to the web server (HTTPS) to keep your requests more secure, specify the HTTPS port numbers with the following parameter:

HttpsPorts

Specifies the secure ports the Web Agent listens on if you are using an SSL connection to the web server. If you specify a value for this parameter, you must include all the ports for all the web servers that serve secure requests. If you do not specify a value, the Web Agent reads the HTTP scheme from the web server's context.

If a server is behind an HTTPS accelerator (which converts HTTPS to HTTP), the requests are treated as SSL connections by your browser.

Default: Empty

Example: 443

Example: (multiple ports) 443,7002

To define your HTTPS ports, set the value of the HttpsPorts parameter to the port numbers that use SSL. Use commas to separate multiple port numbers.

Decode Query Data in a URL

To have the Web Agent's Base64 algorithm decode a URL's query data before calling the Policy Server (so the Policy Server sees the proper resource), use the following parameter:

DecodeQueryData

Specifies whether the Web Agent decodes the query data in a URL before calling the Policy Server. Set this parameter to yes if you need do any of the following tasks in your environment:

- If you need to ensure the rules filer acts against the proper string.
- If you need to or write rules against the data in a query string.

Default: No

To have the Web Agent decode the query data in a URL before calling the Policy Server, set the value of the DecodeQueryData parameter to yes.

How to Protect Resources Without Periods or Extensions

Some URLs, such as servlets, do not have peroids. Other URLs may not have extensions. Both of these situations pose security risks. The following process demonstrates these risks:

1. Your environment contains a directory called /mydir/servlets that is a protected resource.
2. Your Web Agent is configured to ignore requests for resources with the .gif extension.
3. An unauthorized user appends the name of a nonexistent file along with a .gif extension to the end of the URL as shown in the following example:

   /mydir/servlets/file.gif
4. The Web Agent ignores the .gif extension and grants the unauthorized user access to the /mydir/servelets directory.

If you are most concerned about the security risks, do not allow the Agent to ignore any extensions, but consider the following consequences:

- Performance may decrease because the Web Agent will evaluate every image URL on a page.
- Behavior of your Web site may change because users may be challenged for resources that formerly did not require authentication.

The following options are available to protect URLs that do not have periods:

- Configure the Agent to use the OverrideIgnoreExtFilter feature.
- Make sure that protected resources do not have extensions that the Web Agent is configured to ignore.

Handle Complex URIs

The DisableDotDotRule parameter determines whether or not the Web Agent automatically authorizes a URI that contains two dots separated by a slash (/).

Default: No

If the DisableDotDotRule is set to yes, the Agent *does not* apply the double dot rule. For example, if the URI is:

- /dir1/app.pl/file1.gif

  The Web Agent uses the IgnoreExt parameter to determine if the resource should be automatically authorized.

- /dir1/okay.button.gif

  The Agent can ignore this URI because the two dots are not separated by a slash (/). The double-dot rule is not applicable in this case.

If the DisableDotDotRule is set to no, the default, the Web Agent *applies* the double-dot rule. The Web Agent challenges requests for the following URIs, passing the request to the Policy Server:

- /dir1/app.pl/file1.gif

  This URI falls under the double-dot rule because the two dots are separated by a slash.

  The web server may consider /dir1/app.pl as the target resource, and /file1.gif as extra path information, typically viewable in CGI headers as PATH_INFO.

- /dir1/okay.button.gif

  The Agent may ignore this URI because even though the double-dot rule is being enforced, the two dots are not separated by a slash (/), so the rule is not applicable.

Important! Avoid creating the possibility for unauthorized access when you use the IgnoreExt and DisableDotDotRule parameters together. For example, if you want to protect /dir1/app.pl, but you set the DisableDotDotRule parameter to yes, the Agent ignores the URI /dir1/app.pl/file1.gif because you have disabled the double-dot rule and included .gif in the IgnoreExt parameter. Consequently, an unauthorized user may access the protected application /dir1/app.pl.

Chapter 7: Use Platform for Privacy Preferences (P3P) Compact Policies with SiteMinder Agents

SiteMinder agents can support P3P compact policies using the parameters described in the following procedures:

- How to support P3P compact policies (see page 87).
- Configure your agent to support P3P compact policies (see page 88).

This section contains the following topics:

How to Support a P3P Compact Policy with your SiteMinder Web Agent (see page 87)
Configure your Web Agent to Accommodate P3P Compact Policies (see page 88)

How to Support a P3P Compact Policy with your SiteMinder Web Agent

CA SiteMinder supports P3P Compact policies on most types of Web Agents, *except* for the following types:

- Domino

Note: For more information about P3P, see the P3P page of the World Wide Web Consortium web site.

To configure your Web Agent to support a P3P Compact policy, use the following process:

1. Configure a P3P Compact policy on your web server.

   Note: For more information, see the documentation provided by your web server vendor.

2. Configure your Web Agent to accommodate your P3P Compact policy.

Configure your Web Agent to Accommodate P3P Compact Policies

You can determine whether the custom responses from your Web Agent comply with P3P response headers with the following parameter:

P3PCompactPolicy

Determines whether custom responses comply with the Platform for Privacy Preferences Project (P3P) response headers. P3P compact policies use tokens representing the specific elements from the P3P terminology. If you set the P3PCompactPolicy parameter to the appropriate policy syntax, it ensures that custom responses are set with the correct P3P response header when a P3P compact policy is specified for the Web Agent.

Default: No default

Example: NON DSP COR CURa TAI (these represent: none, disputes, correct, current/always, and tailoring, respectively)

To accommodate P3P compact policies, add an appropriate policy syntax to the P3PCompactPolicy parameter.

Chapter 8: Session Protection

This section contains the following topics:

Apply SiteMinder Behavior to a Web Application Client (see page 89)
Modify the Session Grace Period (see page 95)
Modify the Session Update Period (see page 96)
Protect Session Cookies from Misuse with Validation Periods and Expired Cookie URLs (see page 97)
Prevent Session Cookie Creation or Updates (see page 98)
Prevent Session Cookie Creation or Updates Based on Method and URI (see page 99)
Store Session Cookies on the Session Store for Improved Security (see page 100)
Validate a Session Cookie Domain (see page 101)
Redirect a User after a Session Time-out (see page 102)
How to Enforce Timeouts across Multiple Realms (see page 103)

Apply SiteMinder Behavior to a Web Application Client

Some web applications use script engines, which execute in the context of a web browser, to request resources and display content. Similar to requests standard web browsers send, the requests originating from the script engine can trigger SiteMinder generated behavior, such as HTTP redirects or challenges.

Unless properly integrated with the web application, this behavior can result in the web application client reaching an indeterminate state.

The web application client response (WebAppClientResponse) ACO parameter lets you:

- Configure SiteMinder to identify requests originating from the script engine that is executing in the context of the web browser.
- Use a customized response to integrate SiteMinder generated behavior, including a challenge, with the functionality of the web application client.

  Note: If you are using the WebAppClientResponse parameter to integrate the session management features ofSiteMinder (such as idle or session timeouts), configure the OverLookSessionFor ACO parameter also.

  While the OverLookSessionFor parameters prevent web application client requests from keeping user sessions active indefinitely, the WebAppClientResponse parameter lets you integrate the required SiteMinder functionality to redirect users after a session timeout.

More information:

Prevent Session Cookie Creation or Updates (see page 98)
Prevent Session Cookie Creation or Updates Based on Method and URI (see page 99)
Redirect a User after a Session Time-out (see page 102)

Web Application Client Response Introduced

You use the WebAppClientResponse ACO parameter to implement the functionality of the web application client, while maintaining SiteMinder security.

The parameter is comprised of the following default attributes:

```
Resource=|Method=|Status=|Body=|ContentType=|Charset=
```

Consider the following factors:

- This ACO parameter requires at least one attribute with a valid value.
- All additional attributes are optional.
- If you must identify requests from multiple web applications, a single ACO parameter can include multiple values for each attribute.

Example: WebAppClientResponse ACO parameter

The example shows the parameter with a valid value for each attribute. A description of each attribute follows the example:

```
WebAppClientResponse:Resource=/web20/dir/*|Method=GET,POST|Status=200
|Body=C:\location\custombody_1.txt|Content–Type=application/xml|Charset=us–ascii
```

Resource

Specifies the URI to which the web application client is making requests. If the URI of a request matches this value, SiteMinder identifies the request as originating from the web application client. The resource can contain a wildcard (*) for prefix and suffix matching.

Default: No value. If this value is omitted, all resources that the Web Agent is protecting apply to the parameter.

Limit: Regular expressions are not supported.

Example: Resource=/web20/dir/*

Example: Resource=/web20/dir/*.xml

Method

Specifies the HTTP method with which the web application client is making the request. If the HTTP method of a request matches this value, SiteMinder identifies the request as originating from the web application client.

Default: No value. If this value is omitted, the parameter applies all HTTP methods.

Separate multiple methods with a comma (,).

Example: GET, POST

Status

Specifies the HTTP status that SiteMinder must send back to the web application client request.

Default: No value. If this value is omitted, an HTTP status of 200 applies to the parameter.

Body

Specifies the fully qualified name of the file containing the custom body that is to function as the response to the web application client request. This file resides on the Web Agent host system and can:

- Be text–based or contain binary data.
- Contain any custom body that is designed by the application owner.
- Contain a custom body that can be used to forward a SiteMinder reason and redirect URL.

Default: No value. If this value is omitted, SiteMinder forwards the response to the web application client without a body.

ContentType

Specifies the MIME type of the data present in the file that contains the response.

Default: No value. If this value is omitted, a MIME type of text/plain applies to the parameter.

If the custom body contains SiteMinder generated responses, the content type of the data must be one of the following types:

- text/*
- application/xml
- application/*+xml

Charset

Specifies the character set of the data present in the body file.

Default: No value. If this value is omitted, the parameter applies a character set type of us–ascii.

Cookie Providers and the Web Application Client Response

Considering the following factors when setting the WebAppClientResponse parameter:

- If a user accesses a Web 2.0 resource, SiteMinder does not update the session cookie on the cookie provider.
- When a user accesses a non–Web 2.0 resource, such as .html, .jsp, .asp, and .cgi, SiteMinder updates the session cookie on the cookie provider as normal.

How to Apply the Web Application Client Response to a Web Application

Applying the web application client response with a web application lets you implement the functionality of the web application client, while maintaining SiteMinder security. Complete the following steps to apply the web application client response:

1. Configure the web application client response (WebAppClientResponse) ACO parameter.
2. Configure a custom response.
3. Configure the web application to handle a custom response.

Configure a Web Application Client Response

To configure a web application client response.

1. Do one of the following tasks:
   - Open the Agent Configuration Object (ACO) in the Administrative UI and uncomment WebAppClientResponse.
   - Open the local agent configuration file and uncomment WebAppClientResponse.
2. Enter a value for one or more of the following default attributes:
   - Resource
   - Method
   - Status
   - Body
   - Content–Type
   - Charset

Note: Consider the following limitations:

- This ACO parameter requires a valid value in at least one attribute.
- All additional attributes are optional.
- If you must identify requests from multiple web applications, a single ACO parameter can include multiple values for each attribute.

3. Do one of the following tasks:
   - Save the ACO in theAdministrative UI.
   - Save the local agent configuration file.

Configure a Customized Response

The application owner configures a customized response in the body of a file that resides on the Web Agent host system. When a web application client request triggers SiteMinder functionality, the Web Agent returns the body as a response to the web application client.

Consider the following factors:

- The file can contain any custom body as designed by the application owner.
- The file can be text–based. If the file is text–based, SiteMinder parses the body of the file for $$Reason$$ and $$URL$$ before sending the response to the web application client.

  If the response is to include a SiteMinder generated behavior:

  - The content MIME type of the data must be one of the following types:
    - text/*
    - application/xml
    - application/*+xml
  - The following placeholder values must appear in the body:

    ```
    SiteminderReason=$$Reason$$
    SiteminderRedirectURL=$$URL$$
    ```

    SiteMinder parses the body for these values and inserts the triggered SiteMinder functionality and redirect URL. The following parameters or policy response types define the functions and URLs:

    - IdleTimeoutURL
    - MaximumTimeoutURL
    - ForceFQHost
    - LogOffRedirectURL
    - ExpiredCookieURL

- OnAuthAcceptRedirect
- OnAuthRejectRedirect
- OnAccessAcceptRedirect
- OnAccessRejectRedirect
- Challenge

Example: Suppose that a web application client request triggers an idle timeout. SiteMinder replaces the placeholder values with IdleTimeoutURL and the URL specified in the value of the IdleTimeoutURL parameter.

- The file can contain binary data. If the file contains binary data, SiteMinder forwards the body of the file to the web application client without parsing it.

Configure the Web Application to Handle a Custom Response

If the custom response includes a SiteMinder reason and redirect URL, configure the web application separately to handle the custom response.

The Web Agent installation wizard installs sample applications in *web_agent_home*/samples. Extrapolate from the samples for your specific environment and situation.

web_agent_home

Specifies the Web Agent installation path.

Modify the Session Grace Period

Web pages usually consist of many resources, all of which are potentially protected by the Web Agent. For each resource associated with a single request, a session cookie is generated. To eliminate the overhead of generating multiple session cookies for a single user request, set the following parameter:

SessionGracePeriod

Specifies the number of seconds during which a SiteMinder session (SMSESSION) cookie will not be regenerated. Cookies are not regenerated when *all* of the following conditions are met:

- There is no URL SMSESSION cookie.
- The difference between the current time and the last access time of the received SMSESSION cookie is less than or equal to the SessionGracePeriod.
- The amount of time between the current time and the time when the received cookie would have been idle exceeds two grace periods. For example, if your grace period is 25 minutes and the idle time-out is 60 minutes, SiteMinder regenerates a session cookie after 10 minutes, because then there are less than two grace periods (50 minutes) of time left before the session goes idle.

Default: 30

To modify the session grace period

1. Change the value of the SessionGracePeriod parameter.
2. If you increased the setting for the SessionGracePeriod parameter in Step 1, use the Administrative UI to ensure both of the following values in all of your realms do *not* exceed the value of the SessionGracePeriod parameter:
   - Session timeout value
   - Idle timeout value

   The session grace period is changed.

   Note: Session timeouts are part of configuring a realm, which you do using the Administrative UI. For further instructions on configuring session timeouts, see the Policy Server documentation.

Modify the Session Update Period

You can specify how often the Web Agent redirects a request to the Cookie Provider to set a new cookie with the following parameter:

SessionUpdatePeriod

Specifies how often (in seconds) a Web Agent redirects a request to the Cookie Provider to set a new cookie. Refreshing the master cookie decreases the possibility that it will expire due to an idle time-out of the SiteMinder session.

Default: 60

To modify the session update period

1. Make sure the CookieProvider parameter is defined.
2. Change the number of seconds in the SessionUpdatePeriod parameter to reflect the interval you want.

   The session update period is changed.

Protect Session Cookies from Misuse with Validation Periods and Expired Cookie URLs

SiteMinder uses time-based session cookie parameters that can substantially reduce the possibility of a SiteMinder session cookie being compromised by administrators or other users who have access to the following items:

- Web server logs
- SiteMinder Web Agent logs
- Potentially compromised proxy servers sitting between domains in the case of cross-domain single sign-on

These time-based session cookie parameters add the concept of "born dates" to session cookies. Agents receiving a session cookie as a result of a redirect (URL session cookie) will look for the cookie born date name/value pair and compare this value with the value set for the CookieValidationPeriod configuration parameter. If the value of the born date and the CookieValidationPeriod parameter value exceed the current time, the cookie is rejected.

To protect session cookies from misuse, set the following parameters:

CookieValidationPeriod

Specifies the time period (in seconds) in which the receiving agent will accept the session cookie. After this time passes, the session cookie will not be accepted. If this field is not used or is set to zero, the session cookie expires when the Idle Timeout and Max Session Timeout values are met.

Default: Empty.

ExpiredCookieURL

(Optional) Specifies a URL that the agent redirects the user to after any session cookie has expired. If neither the born date nor the CookieValidationPeriod are configured, the agent ignores the settings and processes the cookie as usual (backwards compatibility).

Prevent Session Cookie Creation or Updates

Some Web applications, such as Microsoft Outlook Web Access, make HTTP requests behind the scenes even when a user is not actively using the application. For example, the Web Access application makes HTTP requests even when the user is not actively checking for new email on the server.

These requests may update the SMSESSION cookie so that the session never expires, even though the user has been idle. You can prevent the Web Agent from creating or updating session cookies during these background requests so that sessions expire normally.

To prevent creating or updating SMSESSION cookies

1. Configure one or both of the following parameters:

   OverlookSessionForMethods

   Specifies whether the Web Agent compares the request method of all HTTP requests against the methods listed in this parameter. If a match occurs, the Web Agent does not create or update an SMSESSION cookie. Also, cookie providers (if configured) are not updated for that request.

   Default: No default

   OverlookSessionForUrls

   Specifies whether the Web Agent compares the URLs from all HTTP requests against the URLs listed in this parameter. If a match occurs, the Web Agent does not create or update an SMSESSION cookie. Also, cookie providers (if configured) are not updated for that request.

   Default: No default

   Example: Use a relative URL, such as /MyDocuments/index.html. Do not use an absolute URL (http://fqdn.host/MyDocuments/index.html)

Note: If you configure both of the previous parameters, the methods are processed before the URLs.

Prevent Session Cookie Creation or Updates Based on Method and URI

Some Web applications, such as Microsoft Outlook Web Access, make HTTP requests behind the scenes even when a user is not actively using the application. For example, the Web Access application makes HTTP requests even when the user is not actively checking for new email on the server.

These requests update the SMSESSION cookie so that the session never expires, even though the user has been idle. You can prevent the Web Agent from creating or updating session cookies during these background requests so that sessions expire typically.

To prevent creation or updates based on method and URI

1. Set all the following parameters:

   OverlookSessionForMethods

   Specifies whether the Web Agent compares the request method of all HTTP requests against the methods listed in this parameter. If a match occurs, the Web Agent does not create or update an SMSESSION cookie. Also, cookie providers (if configured) are not updated for that request.

   Default: No default

   OverlookSessionForMethodUri

   Specifies whether the Web Agent compares the method and the URI from all HTTP requests against the method and URI listed in this parameter. If a match occurs, the Web Agent does not create or update an SMSESSION cookie. Cookie providers (if configured) are not updated for that request.

   Default: No default.

   Limits: Specify a relative URI

   Example: POST, /directory/file prevents updates to the SMSESSION cookie for POST requests to /directory/resource.

   Note: Methods are processed before URIs.

Store Session Cookies on the Session Store for Improved Security

Session cookies are stored on the client computer of the end user. You can increase the security of your environment by having SiteMinder create session cookies that are stored in the SiteMinder session store. Storing session cookies in the SiteMinder session store prevents anyone with access to the following items from copying a session cookie from a client computer and then attempting a replay attack:

- Web server logs.
- SiteMinder Web Agent logs.
- Potentially compromised proxy servers sitting between domains (with single-sign on across multiple domains).

You can control where SiteMinder stores its session cookies by setting the following parameter:

StoreSessioninServer

Specifies whether session cookies are stored on the client computer, or in the SiteMinder session store. When the value of the StoreSessioninServer parameter is yes, a session cookie is created and stored on the session store. Cookie providers and Web Agents access the cookie from the session store.

Cookie providers and Web Agents replace the session cookie in a URL with a GUID that corresponds to the session cookie stored on the session store.

When the value of the StoreSessioninServer parameter is no, the session cookie is passed directly in the URL.

Default: No

Follow these steps:

1. Verify that your environment meets the following conditions:
   - Upgrade your Web Agents and cookie providers to use SiteMinder 6.0 SP5 QMR1 or higher.
   - Use a value for the DefaultAgentName parameter in the Web Agents and cookie provider.
   - Your Policy Server is configured with a valid session store.
2. In your Web Agents and cookie provider, set the value of the StoreSessioninServer parameter to yes.

Validate a Session Cookie Domain

You can reduce the risk that unauthorized users may hijack and attempt to reuse SiteMinder session cookies by having SiteMinder validate the domain of a session cookie with the following parameter:

TrackSessionDomain

Instructs the Web Agent to encrypt and store the intended domain of a session cookie within the session cookie itself. During subsequent requests, the Web Agent compares the intended domain stored within the session cookie against the domain of the requested resource. If the domains do *not* match, the Web Agent rejects the request.

For example, when the value of this parameter is set to yes, session cookies intended for operations.example.com are rejected when presented at finance.example.com.

In SiteMinder environments using SSO, set this parameter on the Web Agent that creates the encrypted session cookie. For example, suppose your SSO environment has domains named a.example.com and b.example.com. If the Web Agent protecting a.example.com encrypts the session cookie, set the value of the TrackSessionDomain parameter of the associated Web Agent. When the Web Agent protecting b.example.com receives the cookie, it compares the intended domain stored in the cookie against the domain of the requested resource.

Default: No

To have SiteMinder validate the domain of a session cookie, set the value of the TrackSessionDomain parameter to yes.

Redirect a User after a Session Time-out

Session time-outs are set when you configure a realm with the Administrative UI. When a user's SiteMinder session times out, the Web Agent does one of the following actions:

- Rechallenges the user for credentials
- Redirects the user to another URL

If a redirect URL is specified, the user is sent to that destination page. If the page is unprotected, the user is granted direct access to that page. If the page is protected, the user is challenged for credentials before being granted access to the page. If no redirection URL has been specified, the Web Agent rechallenges the user for credentials after a session time-out.

You can redirect users whose sessions time out to a URL with a customized web page, which explains why their session has been terminated and how they can reestablish it. For example, you can create a custom web page that displays a message such as, "You have been logged out automatically as a security precaution. Please login again to continue."

If the user is not redirected to another page after a session times out, SiteMinder challenges the user again. This may confuse users because they may not understand why they are being asked to reauthenticate.

To redirect users to different URLs after session time-outs

1. Add the following parameters to your Agent Configuration Object or your local configuration file:

   IdleTimeoutURL

   Specifies the URL where the Web Agent should redirect the user when the idle time-out for the session occurs.

   Example: http://example.mycompany.com/sessionidletimeoutpage.html

   Note: IdleTimeoutURL should only be used for non-persistent sessions; it has no effect if configured for persistent sessions.

   MaxTimeoutURL

   Specifies the URL where the Web Agent should redirect the user when the maximum time-out for the session occurs.

   Example: http://example.mycompany.com/maxtimeoutpage.html

   Default: No default

2. Enter one URL for each of the previous parameters. You can use the same URL for all of the parameters, or you may use different URLs for each.

If the idle timeout and maximum timeout values for a session (set in the Policy Server) occur at the same time and the IdleTimeoutURL and MaxTimeoutURL parameters are set, the user is redirected to the URL specified in the MaxTimeoutURL parameter when a time-out occurs.

How to Enforce Timeouts across Multiple Realms

User session timeouts are governed by the realm that the user first logs into. If a user enters a new realm through single sign-on, the time-out values for the new realm are still governed by the session that was established by the initial login at the first realm. If you have different time-out values for different realms, and you want to have each realm use its own time-out values, you can override the time-outs of the original realm.

A user who has already timed out cannot log in to another realm without being rechallenged. For example, if the Idle Timeout in Realm1 is 15 minutes and the Idle Timeout in Realm2 is 30 minutes, a user who accumulates 20 idle minutes in Realm1 will be challenged upon logging in to Realm2.

To override the time-outs of the original realm, configure your Web Agent and realms as described in the following process:

1. Set the value of the EnforceRealmTimeouts parameter to yes.
2. Use the Administrative UI to do the following tasks:
   a. For each realm where you want to supersede the original time-outs (any realm that SSO functionality allows the user to access), do the following:
      - To override the Maximum Timeout value, create a response using the WebAgent-OnAuthAccept-Session-Max-Timeout response attribute.
      - To override the Idle Timeout value, create a response using the WebAgent-OnAuthAccept-Session-Idle-Timeout response attribute.
   b. Bind each of the previous responses to an OnAuthAccept rule.

   Note: For information about creating responses, see the *Policy Server Configuration* Guide.

Chapter 9: Web Application Protection

This section contains the following topics:

Mechanisms for Developing Web Applications (see page 105)
REMOTE_USER Variable (see page 105)
How Response Attributes Work with Web Agents (see page 108)
SiteMinder Default HTTP Headers (see page 111)

Mechanisms for Developing Web Applications

SiteMinder provides the following methods for protecting web applications:

- REMOTE_USER variable for passing the authenticated user name to applications (see page 105).
- Response attributes (see page 108).
- HTTP headers (see page 111).
- Custom error pages (see page 122).

REMOTE_USER Variable

The REMOTE_USER variable holds the name of the user authenticated by the web server. When the agent is installed on a web server, SiteMinder replaces the native authentication of the web server. The REMOTE_USER variable is blank.

If your applications use the REMOTE_USER variable, then set REMOTE_USER variable is set.

If your web server does not use the REMOTE_USER variable, the HTTP_SM_USER header provides an alternate method of passing a user name to an application.

More Information

Configure the Web Agent to set the REMOTE_USER Variable (see page 106)

Configure the Web Agent to set the REMOTE_USER Variable

Configure the Web Agent to set the REMOTE_USER variable as follows:

- To set the REMOTE_USER to the value of the SiteMinder log-in user name, set the Web Agent's SetRemoteUser parameter to yes.

  The default for this parameter is no, which leaves the REMOTE_USER variable blank.

  Note: Prior to SiteMinder Web Agent 5.x QMR 2, the SetRemoteUser parameter affected only IIS web servers; Apache and Oracle iPlanet Agents always set REMOTE_USER to the SiteMinder logged-in user name. If you install or upgrade from Agents prior to 5.x QMR 2, note that REMOTE_USER is no longer enabled by default.

- To set the REMOTE_USER variable based on a specific user account instead of the logged-in user's credentials:

  - Enable the SetRemoteUser parameter by setting it to yes.
  - Set the RemoteUserVar parameter. This parameter instructs the Agent to populate the REMOTE_USER variable based on the value from an HTTP-WebAgent-Header-Variable response attribute. Use this to integrate with legacy applications.

    To configure the RemoteUserVar parameter, enter only the name of the response variable. For example, to return an HTTP-WebAgent-Header-Variable such as "user=ajohnson", set the RemoteUserVar parameter to the value user.
  - Bind the header variable to an OnAuthAccept rule. Do not use an existing HTTP header variable response; create a new one.

    Note: For more information, see the Policy Server documentation.

- To revert to the default, which leaves REMOTE_USER blank, return the SetRemoteUser parameter to no.

Note: Be sure to take security consequences into consideration before configuring SetRemoteUser or RemoteUserVar.

IIS Web Servers and the REMOTE_USER Variable

Your IIS web server requires basic authentication to use the REMOTE_USER variable with SiteMinder.

When Basic authentication is enabled and a user requests a SiteMinder-protected resource, the agent attempts to set the HTTP_Authorization header of the IIS web server with a user name only. Not with a password.

The Basic authentication mechanism of the IIS web server takes precedence over any other authentication challenge when an HTTP_Authorization header is used. Therefore, the IIS web server thinks that the user is responding to its own challenge.

If your agent operates as an ISAPI filter (using the Classic Pipeline mode for IIS), the agent does the following tasks:

- Sets the user context of the request.
- Sets a value for the REMOTE_USER header.

The Agent populates the REMOTE_USER header when the value of the SetRemoteUser parameter is yes and any of the following settings are used:

- DefaultUsername and DefaultPassword—together these parameters control the (privileged) proxy user account that the agent uses for most activities.
- ForceIISProxyUser—overrides the normal behavior and forces the agent to instruct the IIS web server to run as the proxy user.
- UseAnonAccess—instructs the agent to provide no user context for the request at all, leaving any existing user context unchanged.
- Run in Authenticated User's Security Context—the agent instructs the IIS web server to use the credentials stored in the persistent session.

Be cautious when using the SetRemoteUser parameter and the UseAnonAccess parameter together.

The following table shows how these parameters work together:

If these parameters are set as follows:	Then this result occurs...
SetRemoteUser=yes UseAnonAccess=yes	The REMOTE_USER variable cannot be set because the agent does *not* pass along a user security context. The lack of a user security context forces the IIS web server to use the credentials from the HTTP_Authorization header that the agent modified. But this header is incomplete because it only contains the user name.

If these parameters are set as follows:	Then this result occurs...
SetRemoteUser=yes UseAnonAccess=no	The agent can pass along a user context of some type, depending on how other parameters are set, such as DefaultUserName, DefaultPassword, or ForceIISProxyUser. If the agent passes a security context to the IIS web server, then the IIS web server uses the credentials of the agent. The IIS web server ignores the incomplete HTTP_Authorization header.

How Response Attributes Work with Web Agents

SiteMinder response attributes instruct applications how to collect user data and apply that information to display personalized content for each user.

SiteMinder provides configurable response attributes as a means of delivering data to applications and customizing the user experience.

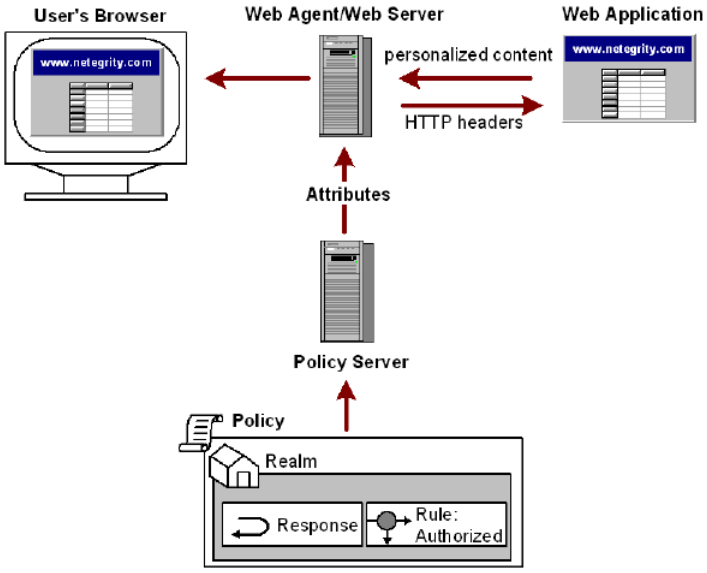
You configure responses using the Administrative UI, and then associate them with specific rules in a policy. When a request triggers a rule with a configured response, the Policy Server sends the response data to the Agent, which interprets the information and makes it available to Web applications.

When you configure a response, you associate the response with an Agent action. You can associate HTTP header and cookie response attributes with the actions GET and POST. These attributes can also be tied to Authentication or Authorization events. The Policy Server can send a response if the user is accepted or rejected for either of these events.

Note: When configuring response attributes note that the maximum buffer size for the web server for agent responses is 32 KB. There is no length limit of a response other than the total buffer size.

Response attributes other than the header and cookie attributes can *only* be used when an authentication or authorization event occurs (whether or not the user is accepted or rejected for either of these events). For example, you can select an **Authorization event** action for a rule and then configure a **WebAgent-OnReject-Redirect** response attribute. If a user is rejected during the authorization process by SiteMinder, the Agent redirects the user to another page that could display a message indicating why that user was rejected.

The following illustration shows how response attributes are sent from the Policy Server to the web server:



To simplify the task of maintaining responses, define a separate response for each type of event. For example, define one response for an OnAccept event and another response for an OnReject event. Creating a separate response makes it easier to find attributes when you need to modify response values.

Use SM_AGENT_ATTR_USRMSG Response for a Forms Challenge

The SM_AGENTAPI_ATTR_USERMSG response enables developers of custom SiteMinder authentication schemes to return custom text to their client applications, as part of a user challenge or for some other purpose.

Beginning with v5 QMR3, the Web Agent has the ability to convert the text from an SM_AGENTAPI_ATTR_USERMSG response to an SMUSRMSG cookie when performing a forms challenge.

To ensure the SMUSRMSG cookie is removed after the challenge is complete, the FCC consumes the cookie (deletes it from the browser) after a successful POST request, as follows:

- In native SiteMinder mode, the Agent deletes the cookie after a successful login, while redirecting back to the target URL.
- In SiteMinder 4.x compatibility mode, the Agent deletes the cookie after generating the FORMCRED cookie, while redirecting back to the target URL.

Note: The SMUSRMSG cookie will be stored for a period of time in the user's browser, and could possibly be transmitted over non-secure HTTP connections. As a result, sensitive data should be avoided.

Web Agents will URL-encode text that is placed in the SMUSRMSG cookie during a forms challenge, to make it safe for HTTP transmission, eliminating spaces and other harmful characters. The FCC decodes this text before making it available to the environment for use in custom FCC functionality.

Note: URL encoding is not implemented unless the text is placed in the SMUSRMSG cookie.

To implement the new functionality, custom authentication scheme developers must generate custom forms-based authentication schemes. When an Sm_AgentApi_Login() call returns SM_AGENTAPI_CHALLENGE, the Agent challenges the requesting user by redirecting to the authentication scheme URL provided by the response to Sm_AgentApi_IsProtected().

When the Web Agent handles an authentication scheme that uses the HTML Forms authentication scheme template, the Agent looks for a SM_AGENTAPI_ATTR_STATUS_MESSAGE response attribute. If the attribute is found, the Agent generates the appropriate SMUSRMSG cookie, while redirecting to the authentication scheme URL. The FCC may then use this cookie during form generation, if appropriate directives are placed in the desired .FCC source file.

Note: For more information, see the Policy Server documentation.

Cache Response Attributes

You can instruct a SiteMinder Agent to cache response attributes or expire attributes that contain dynamic data, forcing the Agent to contact the Policy Server and update the information. If you configure a static response attribute, the Policy Server only allows you to cache the value. By definition, static values do not change, so there is no need to recalculate them. If you configure user, DN, or active attributes, you can either cache the value or recalculate the value at specific intervals to ensure that the data is current.

SiteMinder Default HTTP Headers

SiteMinder default HTTP headers instruct applications how to collect user data and apply that information to display personalized content for each user.

As part of the Web application environment, the SiteMinder Agent submits default HTTP headers to the web server, and the web server makes them available for Web applications. You can use these headers to include functions and enable your Web applications to personalize content. Headers can store information such as a user's name and the type of action a user is authorized to perform.

The Agent sends these headers regardless of whether or not they are called from a Web application; however, you can disable some of these headers so that they do not use up header space.

The following SiteMinder default HTTP headers are available for Web Agents:

HTTP_SM_AUTHDIRNAME

Indicates the name of the directory against which the Policy Server authenticates the user. The administrator specifies this directory with the Administrative UI.

HTTP_SM_AUTHDIRNAMESPACE

Identifies the directory namespace against which the Policy Server authenticates the user. The administrator specifies this namespace with the Administrative UI.

HTTP_SM_AUTHDIROID

Indicates the directory object identifier (OID) from the Policy Server database.

HTTP_SM_AUTHDIRSERVER

Indicates the directory server against which the Policy Server authenticates the user. The administrator specifies this directory server with the Administrative UI.

HTTP_SM_AUTHREASON

Indicates the code the Web Agent returns to the user after a failed authentication attempt or secondary authentication challenge.

HTTP_SM_AUTHTYPE

Indicates the type of authentication scheme the Policy Server uses to verify the user's identity.

HTTP_SM_DOMINOCN

Identifies the user's Domino canonical name if a Domino LDAP directory is used to authenticate users.

Example: HTTP_SM_DOMINOCN="CN=jsmith/O=netegrity."

HTTP_SM_REALM

Indicates the SiteMinder realm in which the resource exists.

HTTP_SM_REALMOID

Indicates the realm object ID that identifies the realm where the resource exists. This ID is may be used by third party applications to make calls to the Policy Server.

HTTP_SM_SDOMAIN

Indicates the Agent's local cookie domain.

HTTP_SM_SERVERIDENTITYSPEC

Indicates the Policy Server identity ticket that keeps track of the user identity. The Web Agent uses this to access content protected by anonymous authentication schemes so that it can personalize the content for the user.

HTTP_SM_SERVERSESSIONID

Indicates a unique string that identifies a user session.

HTTP_SM_SERVERSESSIONSPEC

Indicates the ticket that contains user session information. Only the Policy Server knows how to decode this information.

HTTP_SM_SESSIONDRIFT

Indicates the amount of time the Web Agent can keep a session active using the information in its cache before validating the session with the Policy Server. The session server at the Policy Server must be enabled and a session validation period must be configured for this header to be set.

HTTP_SM_TIMETOEXPIRE

Indicates the amount of time remaining for a SiteMinder session.

HTTP_SM_TRANSACTIONID

Indicates the agent-generated unique ID for each user request.

HTTP_SM_UNIVERSALID

Identifies the Policy Server-generated universal user ID. This ID is specific to the customer and identifies the user to the application, but it is not the same as the user login.

HTTP_SM_USER

Indicates the login name of the authenticated user. If a user does not provide a user name at log in, such as certificate-based authentication, then this variable is not set.

HTTP_SM_USERDN

Identifies an authenticated user's distinguished name as determined by the Policy Server.

For anonymous authentication schemes, this returns a Globally Unique Identifier (GUID).

HTTP_SM_USERMSG

Identifies the text that the Agent presents to the user after an authentication attempt. Some authentication schemes supply challenge text or a reason why an authentication has failed.

More Information

Disable Default HTTP Header Variables (see page 121)

HTTP Header and Cookie-Variables

The HTTP-Header-Variable and HTTP-Cookie-Variable attributes enable a Web Agent to pass a static or dynamic list of name/value pairs to a Web application. The name/value pairs are specific to the user requesting a resource, which enables the application to customize what the user sees.

For example, an administrator configures the WebAgent-HTTP-Header-Variable response attribute to store the full name of the user. When the user is authorized to access the protected resource, the Web Agent passes the user's full name to the Web application. The user's name is then displayed by the application, which helps to establish a relationship with the customer.

Be aware that in a Web application environment, the HTTP-Header-Variable response attribute appears as an HTTP_*attribute_name* variable, where *attribute_name* is the name of the HTTP variable, for example USERFULLNAME. You do not have to have an underscore in the name as the underscores cause problems with some application servers.

Note: The server may convert any dash (-) in the attribute name to an underscore (_), and all alphabetic characters to uppercase.

Header Variables and End-User IP Address Validation

When a SiteMinder Web Agent receives a request that follows an initial request by that same user, the Agent validates the session cookie sent with the subsequent request by comparing the IP address of the requesting user with the IP address encrypted inside the session cookie. The address inside the cookie is generated by the Agent during the user's initial request.

Mechanisms used to balance and manage incoming network traffic, such as firewalls, load balancers, cache devices, and proxies can alter the user's IP address or make it appear as if all incoming requests are coming from a single or small group of IP addresses. As a result, the Web Agent's IP checking becomes ineffective. The Web Agent can now perform IP checking in these network environments using a custom HTTP header and a configurable list of safe proxy IP addresses.

The following table lists the terminology for new IP checking functionality.

Term	Definition
HTTP Request Header	A name/value pair that describes a single element of an HTTP request.
Custom IP Header	A user-defined HTTP request header used by intermediate HTTP network applications or hardware devices to store the requestor's IP address.
IP Checking	Feature that enables the Web Agent to check requests for authenticity by comparing the REMOTE_ADDR in the request with the REMOTE_ADDR value stored in the SMSESSION cookie, after an initial request. This feature is also known as IP validation.
REMOTE_ADDR	web server variable representing the IP address of the HTTP client making a request to the web server. Also known as REMOTE_IP or CLIENT_IP. This differs from the Requestor IP Address when a proxy server, NAT firewall, or other network service or device sits between the requestor and the target web server.
Requestor	The initiator of an HTTP request, typically a user at a browser.
Requestor IP Address	The IP address of the user making the original HTTP request.
Single Sign-on	Feature that requires a user to enter credentials for secure access to a protected Web site only once during a session.
SMSESSION cookie	HTTP mechanism used by Web Agents to track single sign-on state.

How Custom Headers Validate IP Addresses

The Web Agent can now use a custom HTTP header to determine a user's IP address instead of using the REMOTE_ADDR variable. If a proxy or other device sets a custom client IP header and the Web Agent is configured to look for that header on an incoming request, the Agent uses that header as the source of the client IP information.

In addition to configuring a custom header, you can set up a list of proxy IP addresses. If the REMOTE_ADDR matches an address in the proxy list, the Web Agent retrieves the user's IP address from the custom header. Otherwise, the user's IP address is obtained from the REMOTE_ADDR.

After the Web Agent resolves the requestor's IP address, the address is stored and used for request processing. If an address cannot be resolved, the IP address is set to unknown.

The Web Agent logs where the client IP address was resolved from to facilitate any debugging that may be necessary.

Configure IP Address Validation

You can implement IP-address checking using the following parameters:

CustomIpHeader

Specifies an HTTP header for which the agent searches to find the IP address of the requestor. If no value is specified for this parameter, the default is an empty string. No maximum length is enforced and the value can be any string that contains a valid HTTP header value.

Default: No

Example: HTTP_ORIGINAL_IP

ProxyDefinition

Specifies the IP address of a proxy (such as a cache device) that requires the use of a custom HTTP header. This custom header helps the agent resolve the IP addresses of the requester.

Default: No default

Limits: The string must contain an IP address. Do *not* use server names or fully qualified DNS host names.

RequireClientIP

Specifies if the agent validates the IP address of the client. When this value is set to yes, the agent validates that the IP address in the browser cookie matches the IP address of the client. If the addresses do *not* match, a 403 error message appears in the browser of the user. If the cookie does not contain an IP address, then users are prompted for their credentials.

Default: No (client IP addresses not validated).

Note: These settings are independent of the TransientIPCheck and PersistentIPCheck parameters.

IP Address Validation with Previous Web Agent Releases

In an environment of 6.x QMR 2 or 3 Web Agents and older Agents, single sign-on may be affected if IP checking is configured.

Web Agents prior to v6.x QMR 2 and 5.x QMR 7 will not be able to resolve the requestor IP address and as a result, SMSESSION cookies created by those Web Agents may be discarded by 6.x QMR 2 or 3 Web Agents. This includes custom Agents using the SDK to generate SMSESSION cookies, Application Server Agents, and any other SiteMinder Agents in a single sign-on environment that use SMSESSION cookies.

Conversely, 6.x QMR 2 and 3 Web Agents may resolve a requestor's IP address, which then differs from the address resolved by an older Agent.

Preserve HTTP Headers

You can configure the Web Agent to save existing HTTP headers instead of replacing them when new headers are created. This feature is useful for applications that generate multiple SiteMinder responses with the same name but with different values, and need to be included in headers. If there are multiple instances of the same HTTP header, the web server handles this by generating a single header with all the relevant header values separated by commas.

By default, the Web Agent does not preserve headers as a precaution against applications using the wrong header values. For Oracle iPlanet, Domino, and Apache Web Agents to preserve HTTP headers, set the PreserveHeaders parameter to yes. The default value is no.

Control How HTTP Header Resources are Cached

You can control how the Web Agent handles cache-related request headers by setting the following parameter:

AllowCacheHeaders

Specifies whether the web agent removes the following cache-related HTTP headers from protected-resource requests *before* passing those requests to the web server:

- if-modified-since
- if-none-match

This setting affects whether a browser uses cached pages. This setting does *not* affect auto-authorized resources (including values in the IgnoreExt parameter). The settings of the web server and browser determine whether auto-authorized resources are cached.

This parameter uses the following values:

- Yes—The agent does *not* remove any cache-related HTTP headers. The SMSESSION cookies are still tracked to validate the session. When the session expires, the web agent sends an updated SMSESSION cookie with a 304 "not modified" response. This response is applied to the unmodified resources that are stored in the cache. The time indicated in the if-modified-since HTTP header determines when this behavior occurs.

Important! When this parameter is set to yes, pages of personalized applications lacking the appropriate cache control headers could possibly be cached. This situation introduces unexpected behavior and allows a browser to save sensitive data to the disk.

- No—The agent removes the cache-related HTTP headers only from *protected* resource-requests only. The web server treats the request as unconditional. The contents of the cache are not validated.
- None—The Web agent removes all cache-related headers for protected *and* unprotected resources.

For terminated sessions, the browser does *not* use cached content. The value of the AllowCacheHeaders parameter is ignored.

The settings of this parameter affect the following parameters:

- LogOffUri—Set the value of the AllowCacheHeaders parameter to no when using the LogOffUri parameter. Otherwise these sessions do not terminate properly. A cached log-out page could possibly be served to a user.

Default: No

Limits: Yes, No, None

To remove all cache related headers from protected and unprotected resources, set the value of the AllowCacheHeaders parameter to none.

Note: For more information about HTTP 1.1 caching mechanisms, see RFC 2616, Section 13 "Caching in HTTP."

Set the HTTP Header Encoding Spec

The HTTPHeaderEncodingSpec setting affects the encoding of all HTTP header values and all custom HTTP-COOKIE responses.

Use this parameter to support Web applications expecting localized text in specific encodings. Since cookies pass back and forth between the browser and portal through the HTTP protocol, you should use the RFC-2047 HTTPWrapSpec if your chosen encoding puts characters that HTTP traffic considers illegal into the cookies.

For example, some Shift-JIS characters will cause undesirable results if not further encoded by RFC-2047.

For Kanji characters, you can use SECP932, which is a superset of SHIFT-JIS. Though SHIFT-JIS can be used for most Kanji encoding and decoding, CP932 covers an even larger character set.

When HTTPWrapSpec is used, first the data is encoded according to the HTTPHeaderEncodingSpec, then the data is further encoded following the RFC-2047 specification.

The syntax for the parameter is:

encoding_spec, *wrapping_spec*

The *encoding_spec* is a text string that represents one of the following encoding types: UTF-8, Shift-JIS, EUC-J, or ISO-2022 JP. Specify the encoding type you want the Agent to use.

The *wrapping_spec* is the wrapping specification, which must be RFC-2047. Although this variable is optional, we strongly recommend that you include the wrapping specification because the encoding type you choose may generate byte codes that are not compatible with the HTTP protocol.

This is especially true if you use custom HTTP Cookie responses that contain double-byte encoded data. For example, some Shift-JIS characters cause undesirable results if they are not further encoded by RFC-2047. The wrapping also tells the receiving application the type and nature of the encoding so the application can better interpret the encoded text. For example, you may set the parameter to Shift-JIS,RFC-2047.

When RFC-2047 is used, the Agent first encodes the data based on the chosen encoding specification and then further encodes the data following the RFC-2047 specification.

Note: If you leave the HTTPHeaderEncodingSpec setting blank, the default is UTF-8 with no wrapping.

Disable Conformance to RFC 2047

By default, the Web Agent conforms to RFC 2047. However, you can disable this conformance by setting the ConformToRFC2047 parameter to no.

If this parameter does not exist or is set to yes, the Web Agent conforms to RFC 2047.

Use Lower Case HTTP in Headers (for Oracle iPlanet, Apache, and Domino web servers)

If you have server applications that are case-sensitive, you can specify the case of the Agent's HTTP headers. The Web Agent defaults to lower case headers.

For example, Oracle iPlanet web servers, by default, provide the HTTP header variables in lower-case, such as http_sm_user.

Note: IIS Web Agents do not benefit from this feature, because IIS forces all headers to an upper case format.

To use lower case headers, set the LowerCaseHTTP parameter to yes. If you require upper-case header variables, set LowerCaseHTTP to no.

More information:

Record the Transaction ID in Oracle iPlanet Web Server Logs (see page 266)

Enable Legacy Variables for HTTP Headers

You can specify which naming convention the Web Agent uses for HTTP headers with the following parameter:

LegacyVariables

Specifies if the Web Agent uses underscores in HTTP header names. With some web servers (such as the Sun Java System), using the underscore character in the HTTP headers causes problems with some applications.

When this parameter is set to no, the HTTP headers will not have underscores, as shown in the following example:

SM*HeaderName*

When this parameter is set to yes, the HTTP headers will use underscores, as shown in the following example:

SM_*HeaderName*

Default: (traditional agents) Yes

Default: (framework agents) No

To enable legacy variables and have the Web Agent use underscores in the HTTP header names, set value of the LegacyVariables parameter to yes.

Disable Default HTTP Header Variables

Many system platforms have an HTTP header limit of 4096 bytes. To avoid exceeding this limit and to allow space for custom response variables, you can disable some of SiteMinder's default HTTP header variables.

The default variables are grouped into the following categories:

Note: You cannot disable individual variables. You can only disable a category of several variables.

- Authentication source variables
  - SM_AUTHDIRNAME
  - SM_AUTHDIRSERVER
  - SM_AUTHDIRNAMESPACE
  - SM_AUTHDIROID
- User Session variables
  - SM_SERVERSESSIONID
  - SM_SERVERSESSIONSPEC
  - SM_SERVERIDENTITYSPEC
  - SM_SESSIONDRIFT
  - SM_TIMETOEXPIRE
- User Name variables
  - SM_USER
  - SM_USERDN
  - SM_DOMINOCN

To disable the default use of HTTP header variables do any of the following tasks:

- To disable authentication source variables, set the value of the DisableAuthSrcVars parameter to yes.
- To disable user session variables, set the value of the DisableSessionVars parameter to yes.

  Default: No
- To disable user name variables, set the value of the DisableUserNameVars parameter to yes.

  Note: If you are using CA Identity Manager, or any application that might use the variables in this category, ensure the value of this parameter is set to no (enabled).

Custom Error Handling For Applications

Custom error handling allows you to make error information relevant to your application. To customize applications for users, you can add the HTML text displayed by HTTP 500, HTTP 401, and HTTP 403 error pages or, with the exception of 401 errors, you can redirect the user to a URL that points to a custom error page or application.

You can configure customized handling for the following types of errors:

- Server errors—the Agent uses the ServerErrorFile for error pages that result from HTTP 500 web server errors. These error codes are passed to the custom error pages and include:
  - Problems because the Web Agent cannot read values from required HTTP headers.
  - Advanced authentication cookies cannot be parsed or contain an error status.
  - Connectivity problems between the Web Agent and the Policy Server.
- Access Denied errors—the Agent uses the file specified in the following parameter:

  Custom401ErrorFile

  Specifies the customized HTML page to display when users receive a 401 (insufficient privileges) browser error. These errors occur when a user does not have the appropriate privileges to access a resource.

  Note: Some web servers append text of their own to the custom text that you choose. So the response pages for these servers are not customizable.

  Default: No default (blank).
- Require cookies errors—if the RequireCookies parameter is set, the Web Agent sets a cookie during basic authentication. If this cookie is not returned by the browser with the basic credentials, the error page designated by the ReqCookieErrorFile parameter is returned, and the Agent denies the user access to the web server.
- Cross-site scripting errors—the Agent uses the file specified in the CSSErrorFile parameter for error pages that result from HTTP 403 cross-site scripting errors. Cross-site scripting can compromise the security of a Web site.

After you create these HTML files or applications, direct the Web Agent to the custom error pages or URLs.

Note: For an Apache server being used as a proxy or reverse proxy server, the Apache Agent will not return custom SiteMinder error pages, but will return the standard Apache HTTP 500 and 403 error pages.

How to Set Up Error Handling

To customize how your applications display error messages for users, do any of the following tasks:

- Add HTML text that the browser displays for the following HTTP errors:
  - 500
  - 401
  - 403
- Redirect the user to a URL that points to a custom error page or application.

For HTTP 500 and 403 errors only: If you configure the agent to redirect the user to a URL, the agent appends the error code to the URL. See the following example of an appended URL:

```
?SMError=error_code,
```

If you add standard HTML error text, you can only specify HTML code between the following tags:

```
<body>
</body>
```

To direct the agent to the custom error pages or URLs, do *one* of the following tasks:

- Specify the path where the text files reside.

Enter the URL in the value of the respective agent configuration parameter.

Errors and other events and the respective agent configuration parameters are listed in the following table:

Set a custom response for this type of error:	**With the value of this configuration parameter:**
Server errors	ServerErrorFile
Access denied errors	Custom401ErrorFile
Cookie required errors	ReqCookieErrorFile
CSS characters errors	CSSErrorFile

The error files can reside anywhere in your application.

Important! Leave any URL you configure as a custom error page unprotected.

Note: If the URLs of your applications require HTML tags, encode the characters in the tags. For information about encoding HTML characters, see: http://www.cert.org/tech_tips/.

The following examples show a file path and a URL to an error file. The syntax in the example is for a local configuration file. You can also set these parameters in an agent configuration object.

File Path:

```
CSSErrorFile="C:\error\error.txt"

ReqCookieErrorFile="C:\custompages\error.txt"

ServerErrorFile="C:\error\error.txt"

Custom401ErrorFile="C:\error\accessdenied.txt"
```

URL:

```
CSSErrorFile="http://www.mycompany.com/error.jsp"

ReqCookieErrorFile="http://www.myorg.com/error.asp"

ServerErrorFile="http://www.mycompany.com/error.jsp"
```

More Information

Custom Error Handling For Applications (see page 122)

Notes for Custom 401 Pages

- Do not set the Custom401errorfile parameter to a URL.
- If a value (usable or not) for Custom401errorfile exists, the Agent will check every 60 seconds to see if the file has changed. However, the response is intended to be static in nature. You cannot, for example, insert a "*user_name* denied" type of dynamic message.

  Because re-checking is triggered by the existence of the Custom401errorfile value rather than its usability, you can correct an error without restarting the agent. The correction will be picked up on the next check.
- The customized message file text will not be exposed by other errors. The file pathname will be logged at startup and in the case of error.

- The extent of customization may be limited by the web server, which may add text of its own to the response.
- The size of the customized text file is limited only by the system file size limit.

Chapter 10: Configure Virtual Servers

This section contains the following topics:

How to Set Up Virtual Server Support (see page 128)
Assign Web Agent Identities for Virtual Servers (see page 129)
Specify Virtual Servers for the Web Agent to Ignore (see page 130)

How to Set Up Virtual Server Support

A virtual server is a logical entity that you configure on a physical server. This logical entity acts as an independent server. Virtual servers let you host multiple websites on one physical server. For example, using virtual servers, you could set up a server to host both www.mysite.com and www.yoursite.com.

You can assign any of the following to a virtual server:

- A unique IP address
- An IP address that is shared with the physical server
- An IP address that is shared with another virtual server

Although you configure only one Web Agent per web server, you can configure Agent identities to protect your virtual servers. If one user accesses the server through www.mysite.com and another user accesses the server through www.yoursite.com, each server is protected by an agent identity. The advantage of creating an agent identity for each virtual server is that you can define unique realms and rules for each site.

The settings that you define for the Web Agent apply to all virtual servers that you define for that web server instance; however, each virtual server processes requests independently and the Policy Server treats each virtual server request separately. For more information about virtual servers and how to configure them, see the documentation for your web server.

To configure support for virtual servers, do *one* of the following tasks:

- Define and add an Agent identity for each virtual server, specify a value for the AgentName parameter, and assign it the IP address or host header name of a virtual server.
- Define an Agent identity only for virtual servers that need to be uniquely identified.
- Set a Default Agent Name.

Note: If you have more than one instance of the Oracle iPlanet web server, such as a server for HTTP communication and a server for HTTPS communication, two WebAgent.conf files exist. Each file can have multiple agent identities. (The name Oracle iPlanet refers to the web server that was formerly called Sun ONE and iPlanet.)

Assign Web Agent Identities for Virtual Servers

Additional Web Agents for each virtual server are not actually *defined*, but are *assigned* a Web Agent identity. To protect virtual servers that have unique access requirements or to protect distinct realms, assign each server a unique Agent identity and use the default agent name for all other virtual servers. The advantage of this option is that you can configure your SiteMinder installation quickly, yet still guard virtual servers hosting realms that require separate protection.

The AgentName parameter and its associated IP address provide mapping for web server interfaces to agent names as defined in the policy store. Web Agents need to make Agent API calls in the proper agent name context in order for the correct set of rules and policies to apply. If no Agent name or IP address is assigned for mapping to the policy store, then the Web Agent uses the value of the DefaultAgentName parameter only for a virtual server.

To protect virtual servers using unique Agent identities, add a Web Agent for each virtual server in the AgentName parameter. Adding separate Web Agents for each virtual server lets you define unique realms and rules for each virtual server.

To assign a Web Agent identity

1. Enter the name of the agent and the IP address, separated by a comma.
2. Specify the port number associated with the IP address (for example: 112.12.12.1:8080) if your virtual servers share the same IP address, but use different ports. If you are using default ports, port numbers are not required.
3. To add more than one Agent, put each entry on a separate line, as in the following example:

   ```
   agentname="agent1,123.123.12.12:8080"
   agentname="agent2,123.123.12.12:8081"
   agentname="agent3,123.123.12.13"
   ```

4. If you add an Agent Identity, you must define it in the Administrative UI with the same configuration. Make sure that the Agent Identity is defined in Administrative UI exactly as it is defined for the Agent configuration.

If it finds no entries in the AgentName parameter, SiteMinder uses the value of the DefaultAgentName only for a virtual server.

Note: If you change the DefaultAgentName, make sure that it is defined in the Administrative UI exactly as it is defined for the Agent.

Specify Virtual Servers for the Web Agent to Ignore

If a web server at your site supports several virtual servers, there may be resources on these virtual servers that you do not want to protect with the Web Agent. To simplify how the Web Agent distinguishes which portions of a web server's content it protects, use the following parameter:

IgnoreHost

Specifies the fully qualified domain names of any virtual servers that you want the web Agent to ignore. Resources on such virtual servers will be auto-authorized, and the Web Agent always grants access to them regardless of which client makes the request. The authorization decision is based on the configuration of the Web Agent instead of being based on a policy.

The list of ignored hosts is checked first before any other auto-authorization checks, such as the IgnoreExt and IgnoreURL settings. Therefore, the double-dot rule will not trigger an authorization call to the Policy Server for resources on an ignored host but would not be ignored by extension.

The host portion of the URL entries for the IgnoreHost parameter must exactly match what the Web Agent reads for the host header of the requested resource.

Note: This value is case-sensitive.

If the URL uses a specific port, then the port must specified.

For centrally-managed agents, use a multi-value parameter in the Agent Configuration Object to represent several servers. For agents configured with a local configuration file, list each host on a separate line in the file.

Example: (URL shown with port specified)

IgnoreHost="myserver.example.org:8080"

Example: (local configuration file)

IgnoreHost="my.host.com"

IgnoreHost="your.host.com"

Default: No default

To specify virtual servers for the Web Agent to Ignore, do either of the following tasks:

- For central configuration, add the servers you want to ignore to your agent configuration object. For more than one server, use the multi-value setting for the parameter.
- For local configuration, add a separate line for each server in the local configuration file.

Resources using the specified URLs are ignored by the Web Agent and access to those resources is granted automatically.

More Information

Handle Complex URIs (see page 86)

Chapter 11: Forms Authentication

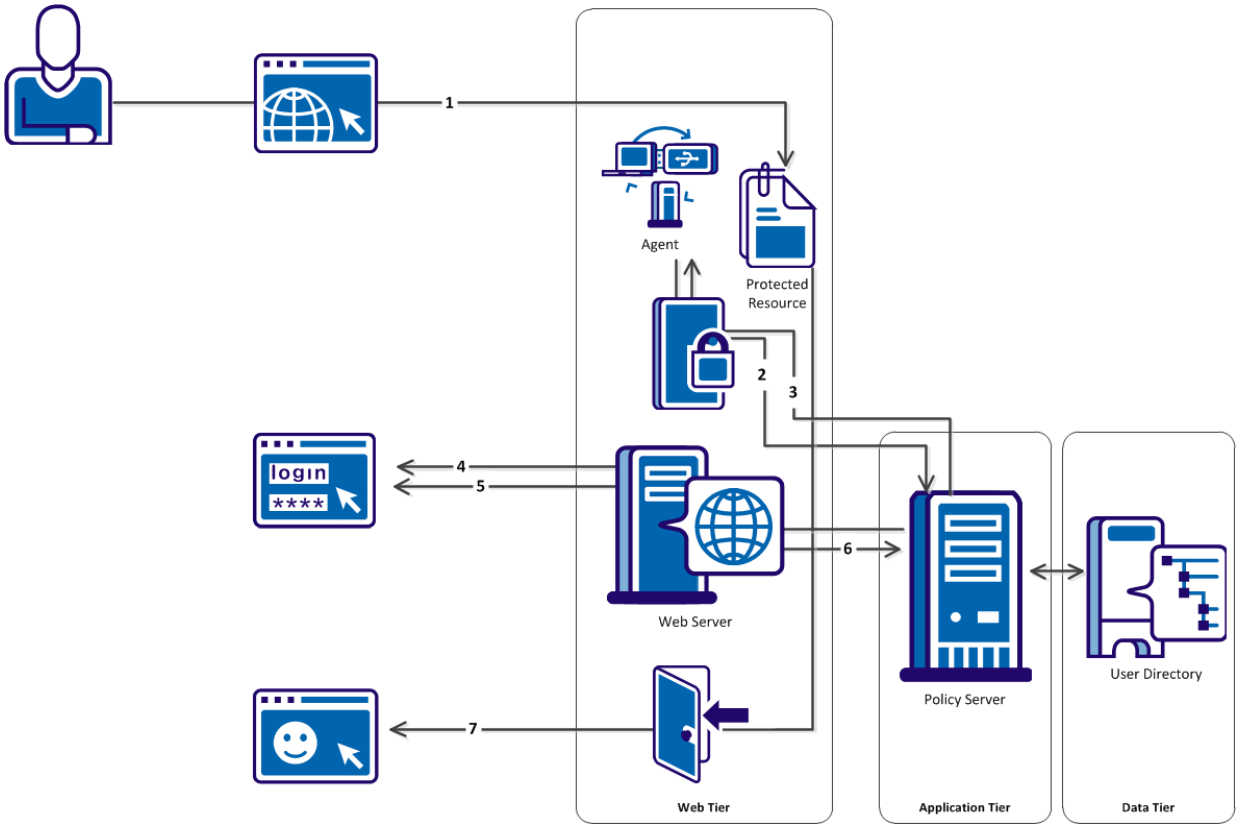
This section contains the following topics:

How Credential Collectors Process Requests (see page 134)
MIME Types for Credential Collectors (see page 135)
How to Configure Basic FCC Authentication (see page 136)
Tune the Performance of the FCC (see page 139)
Specify an NTLM Credential Collector (see page 142)
Map URLs for FCC Redirects with a Domino Web Agent (see page 142)
Configure POST Preservation (see page 142)
Using Credential Collectors Between 4.x Type and Newer Type Agents (see page 147)
Configure Advanced FCC Settings (see page 153)

How Credential Collectors Process Requests

The following illustration describes how forms credential collectors (FCCs) process requests for protected resources:

Note: Cookie providers use a different process for single-sign on.



The process shown in the previous illustration describes the following steps:

1. A user requests access to a resource.
2. The agent contacts the Policy Server to determine whether the resource is protected.
3. The Policy Server informs the agent that a credential collector protects the resource, and specifies the type of credential collector in use.
4. The agent adds query data, the target resource and an encrypted agent name to the URL of the credential collector. The agent then redirects the user to the appropriate credential collector.
5. *One* of the following actions occurs, depending on the type of credential collector:
   - The FCC displays the form and then collects the credentials of the user.
   - The NTC collects the NT credentials of the user.
   - The SCC collects the credentials of the user.
   - If no certificate is available, the SFCC displays the form. The SFCC collects the user credentials.
6. The credential collector logs the user directly in to the Policy Server. The Policy Server creates a session.
7. The agent validates the session and grants the user access to the resource.

Note: For more information about SSL Authentication Schemes, see the Policy Server documentation.

MIME Types for Credential Collectors

Associated with each credential collector is a MIME type. The MIME type indicates which collector presents the authentication challenge when a user requests a resource. The following table shows each type.

Credential Collector	MIME Type
Forms Credential Collector	.fcc
SSL Credential Collector	.scc
Cookie Provider	.ccc
NTLM Credential Collector	.ntc
SSL Forms Credential Collector	.sfcc
Kerberos Credential Collector	.kcc

When you configure an authentication scheme that uses a credential collector, or set up single sign-on across multiple cookie domains, the relevant MIME type is used as a file extension for a file referenced by the authentication scheme or single-sign-on configuration, for example:

- When configuring single sign-on across multiple cookie domains, you enter a URL like the following to identify the cookie provider:

  http://myserver.company.com:80/siteminderagent/SmMakeCookie.ccc

  SmMakeCookie.ccc is the default cookie provider name. You can use this name or create a name of your own; however, it must have the .ccc extension to initiate single sign-on.

- For Windows authentication, the default target file to enable this scheme is:

  /siteminderagent/ntlm/creds.ntc

  Again, you must use a file with the correct MIME type as the extension.

The FCC and SFCC are the only credential collectors that require actual files to exist on the web server where the Agent is installed. These collectors are for forms-based authentication schemes. The .fcc and .sfcc templates are required to define the HTML form presented to the user.

How to Configure Basic FCC Authentication

To authenticate users with forms, Follow these steps:

1. If you are using an IIS web server or a Domino web server, then set up your credential collectors (see page 137).

   Note: The agent configuration wizard automatically sets up the proper MIME types that SiteMinder credential collectors use for the following types of web servers:

   - Apache and Apache-based web servers.
   - Oracle iPlanet web servers.

2. Map your agent identities and web servers for use by FCCs or SCCs.
3. Configure any of the following additional settings:
   - Configure the FCC to use a single resource target (see page 138).
   - Use a relative target for credential collector redirects (see page 138).
   - Enable FCCs and SCCs to use Agent Names as fully qualified host names (see page 137).
   - Define valid target domains (see page 139).
   - Define valid federation target domains.

Set Up Credential Collectors for IIS and Domino Web Servers

IIS and Domino web servers require mapping of MIME types (represented as file extension parameters) in your Web Agent configuration. To configure credential collectors for IIS and Domino web servers.

Follow these steps:

1. Map the specific MIME types for use with each credential collector. We recommend using the default values shown in the following table:

Agent Configuration Parameter	Credential Collector	MIME Type
CCCExt	Cookie Provider	.ccc
FCCExt	Forms Credential Collector	.fcc
SCCExt	SSL Credential Collector	.scc
SFCCExt	SSL Forms Credential Collector	.sfcc
NTCExt	NTLM Credential Collector	.ntc
KCCExt	Kerberos Credential Collector	.kcc

Note: If you do not want to use the default extensions or the defaults are already in use for other purposes, enter the extensions that you want instead. For example, if you set FCCExt to .myext for the FCC, and rename the FCC template to use this extension, for example, login.myext, the agent recognizes URLs ending in .myext as forms authentication requests.

The credential collectors are set.

Enable FCCs and SCCs to Use Agent Names as Fully Qualified Host Names

To enable the forms and SSL credential collectors to use the fully qualified host name of the target URL as an Agent name, define the AgentNamesAreFQHostNames configuration parameter.

For example, if the AgentNamesAreFQHostNames parameter is set to Yes, the www.nete.com portion of the following URL string serves as the Web Agent name:

```
url?A=1&Target=http://www.nete.com/index.html
```

The credential collector uses this parameter in the following situations:

- If no Agent name is appended to the URL from the target agent. (Sometimes the case with third-party agents.)
- You have not configured agent-to-host name mappings in the AgentName parameter.

If the AgentNamesAreFQHostNames parameter is set to No, the credential collector uses the value of the DefaultAgentName parameter as the name of the target Web Agent.

Configure the FCC to Use a Single Resource Target

To configure the FCC to direct users to a single resource, hard-code the target in the login.fcc template file.

Follow these steps:

1. Open the login.fcc file, which is located in *agent_home*/Samples.
2. Add @target=*target_resource* to the FCC.
3. Add the following entry:

   @smagentname=*agent_name_protecting_resource*

   For example: @smagentname=mywebagent
4. Set the EncryptAgentName parameter to no. This parameter is required because no method exists to encrypt the agent name after you hard code it in the file.
5. Set the EncryptAgentName to no for any other agent using this FCC.

Note: For more information, see the Policy Server documentation.

Use a Relative Target for Credential Collector Redirects

Optionally, instruct an agent to use a relative URI instead of a fully qualified URL when directing requests to a credential collector and target resource. Using a relative URI prevents credential collectors on other systems with Web Agents from processing requests.

Note: This setting applies to all credential collectors *except* the cookie credential collector (CCC). The CCC must use a fully-qualified domain name for this parameter. OnAuthAccept responses will not work properly with a CCC if a relative URI is used.

Typically, a fully qualified URL is appended to the credential collector URL. For example:

url?A=1&Target=http://www.nete.com/index.html.

To use only a relative URI, set the TargetAsRelativeURI parameter to yes. If set to yes, the target parameter that is appended to the credential collector URL is a relative target, such as url?A=1&Target=/index.html. In turn, when the credential collector redirects back to the Web Agent protecting the target resource, it is a relative redirect. Also, the Web Agent rejects any target that does not begin with a forward slash (/).

The default value for this parameter is no, so a fully qualified URL is always used.

Define Valid Target Domains

To configure SiteMinder Agents to help protect your resources from phishing attempts that could redirect users to a hostile website, set the following configuration parameter:

ValidTargetDomain

Specifies the domains to which a credential collector is allowed to redirect users. If the domain in the URL does not match the domains set in this parameter, the redirect is denied.

Default: No.

All advanced authentication schemes, including forms credential collectors (FCCs) support this parameter.

The ValidTargetDomain parameter identifies the valid domains for the target during processing. Before the user is redirected, the agent compares the values in the redirect URL against the domains in this parameter. Without this parameter, the agent redirects the user to targets in any domain.

The ValidTargetDomain parameter can include multiple values, one for each valid domain.

For local Web Agent configurations, specify an entry, one entry per line, for each domain, for example:

```
validtargetdomain=".xyzcompany.com"

validtargetdomain=".abccompany.com"
```

Tune the Performance of the FCC

You can configure any of the following settings to help improve the performance of your credential collectors:

- Disable the FCC realm context confirmation to improve performance (see page 140).
- Use the forms cache (see page 140).

Disable FCC Realm Context Confirmation to Improve Performance

During forms authentication, the Web Agent makes an IsProtected call to the Policy Server to determine if the requested resource is protected. After this first call, the Web Agent typically makes an additional IsProtected call to the Policy Server. This second call establishes a realm context so that the Web Agent can log a user in with an FCC to access a protected resource. You can control whether the Web Agent makes this additional call using the following parameter:

FCCForceIsProtected

Specifies whether the Web Agent makes an additional IsProtected call to the Policy Server to establish a realm context so that the Web Agent can log a user in to access a protected resource.

When this parameter is set to no, the Web Agent uses the realm information obtained from its initial IsProtected call to the Policy Server instead.

Default: Yes

To improve performance by disabling the FCC realm context confirmation, set the value of the FCCForceIsProtected parameter to no.

Forms Cache

The forms cache stores form template data. Storing template data improves performance because the agent no longer reads the .fcc files multiple times for the same data. When a resource with an FCC extension is accessed, the FCC reads and processes the corresponding template file. An agent performs hundreds of these read operations each second.

The form cache relieves the FCC by storing form template files in memory where they can be read easily. Because virtual memory access is faster than disk access, allowing FCC components to process forms more quickly with reduced strain on the host server.

The improved processing time increases the capacity of the FCC for serving requests for each web server. Forms authentication becomes more efficient.

Form Cache Data

The data stored in the form cache consists of the form template text, which is parsed beforehand into data structures. These data structures optimize FCC processing.

These data structures include:

- Form locale data for internationalization
- An ordered list of data objects containing raw text in UTF-8 format, template directive information and function/variable information for substitution from the request environment.

Directives, functions, and variables are processed from the top of the FCC file down.

Configure the Form Cache

Forms can be cached to improve performance and reduce unnecessary network traffic. You can control the settings of form cache with the following parameters:

EnableFormCache

Controls the forms template cache. Setting this parameter to yes, improves the performance of forms authentication. To disable the cache, set this parameter to no.

Default: Yes

FormCacheTimeout

Specifies the number of seconds that an object may reside in cache before being considered invalid. When the timeout interval expires, the date and time of the form template file is compared against the time that the cache object was created. If the object in the cache is stored more recently than the file on disk, the timeout is reset for another interval. Otherwise, the object is removed from the cache.

Default: 600

Follow these steps:

1. Set the value of the EnableFormCache parameter to yes.
2. If you want to change the timeout interval for the form cache, set the value of the FormCacheTimeout value to the number of seconds you want.

   The form cache is configured.

Specify an NTLM Credential Collector

The NTLM credential collector (NTC) is an application within the Web Agent. The NTC collects NT credentials for resources that the Windows authentication scheme protects. This scheme applies to resources on an IIS web server that are accessed by Internet Explorer browsers.

Each credential collector has an associated MIME type. For IIS, the NTC MIME TYPE is defined in the following parameter:

NTCExt

Specifies the MIME type that is associated with the NTLM credential collector. This collector gathers NT credentials for resources that the Windows authentication scheme protects. This scheme applies to resources on IIS web servers that only Internet Explorer browser users access.

You can have multiple extensions in this parameter. If you are using an Agent Configuration Object, select the multivalue option. If you are using a local configuration file, separate each extension with a comma.

Default: .ntc

If your environment already uses the default extension that the NTCExt parameter specifies, you can specify a different MIME type.

To change the extension that triggers the credential collector, add a different file extension to the NTCExt parameter.

Map URLs for FCC Redirects with a Domino Web Agent

To protect Domino view (.nsf) resources with a forms authentication scheme, map the URLs before they are redirected to the forms credential collector.

Follow these steps:

1. Set the value of the DominoNormalizeUrls parameter to yes.
2. Set the value of the DominoMapUrlForRedirect parameter to yes.

   Domino URLS are mapped before redirection to the FCC.

Configure POST Preservation

SiteMinder automatically preserves the data that a user posts to an FCC form. This preservation mechanism prevents the data on the form from loss if a timeout or other interruption occurs during the POST operation.

If you are using a combination of traditional and framework agents in your environment, the following additional configuration steps are required:

- Enable POST preservation between Framework and Traditional agents (see page 144)
- Customize the POST preservation page (see page 145)

If you do not want to use POST preservation, you can disable (see page 73) it.

POST preservation is not supported in the following situations:

- ACE authentication.
- Any custom authentication scheme that posts to an FCC.

Enable Post Preservation between Framework and Traditional Agents

Framework Agents handle POST preservation data differently than Traditional Agents do. If your SiteMinder environment uses a combination of Framework and Traditional agents, and resources hosted by one type of Agent are protected by Forms Credential Collectors (FCCs) hosted on the other type of agent, you must specify the proper template file with the following parameter:

PostPreservationFile

Enables the transfer of POST preservation data between Traditional and Framework Agents by specifying the path to *one* of the following POST-preservation-template files:

- tr2fw.pptemplate—Indicates that resources hosted on a server running a Traditional agent are protected by an FCC running on a Framework agent.
- fw2tr.pptemplate—Indicates that resources hosted on a server running a Framework agent are protected by an FCC running on a Traditional agent.

Default: No default

Example: *web_agent_home*/samples/forms/fw2tr.pptemplate

To enable post preservation between Framework and Traditional agents

1. Determine which resources are protected by FCCs running on a different type of Agent.
   a. Create a list of Traditional Agents hosting resources that are protected by FCCs running on Framework Agents.
   b. Create a list of Framework Agents hosting resources that are protected by FCCs running on Traditional Agents.
2. For any traditional Agents hosting resources (those you listed previously in step 1a), set the value of the PostPreservationFile parameter to the path of the tr2fw.pptemplate file.
3. For any Framework Agents hosting resources (those you listed previously in step 1b), set the value of the PostPreservationFile parameter to the path of the fw2tr.pptemplate file.

4. For all of your Framework Web Agents that communicate with Traditional Agents, set the value of the following parameter to yes:

   LegacyPostPreservationEncoding

   Specifies whether the Web Agent encodes any POST preservation data in a way that is compatible with the older, Traditional, Web Agents, or with the newer, Framework Web Agents. When the value of this parameter is set to yes, the encoding is compatible with the Traditional Web Agents. When the value of this parameter is set to no, the encoding is compatible *only* with the Framework Web Agents.

   Default: No

5. Restart the web servers hosting your resources.

   POST preservation is between Framework and Traditional agents is enabled.

Customize the POST Preservation Page

When a timeout or other interruption occurs during a POST operation, the POST preservation page is displayed. In most cases, the POST preservation page appears for less than a second. However, the Post Preservation page can be displayed for as long as 5 seconds when the amount of form data being posted is large.

By default, the POST preservation page displays the following text:

```
This page is used to hold your data while you are being authorized for your request.
You will be forwarded to continue the authorization process. If this does not happen
automatically, please click the Continue button below.
```

The POST preservation page also displays a Continue button that allows the user to repost the data to the application.

To customize the POST preservation page, create a POST preservation template file.

The general structure of the default page is as follows:

```
<HTML><HEAD><TITLE></TITLE></HEAD><BODY onLoad="document.AUTOSUBMIT.submit();">
This page is used to hold your data while you are being authorized for your
request.<BR><BR>
You will be forwarded to continue the authorization process. If this does not happen
automatically, please click the Continue button below.
<FORM NAME="AUTOSUBMIT" METHOD="POST" ACTION="$$smpostlocation$$">
<$$smpostdata$$>
<INPUT TYPE="SUBMIT" VALUE="Continue">
</FORM></BODY></HTML>
```

The POST preservation template must include the following two elements which the Web Agent expands when rendering the POST preservation page:

$$smpostlocation$$

Expanded to the credential collector URL during the first phase of POST preservation. Expanded to the protected resource URL during the second phase of POST preservation.

$$smpostdata$$

Expanded to contain HTML which results in the correct form data being posted to either location respective to the phase of POST preservation.

Do not remove or alter these elements.

However, you can change other elements. For example, to remove the Continue button, remove the <INPUT> element that defines that button:

```
<INPUT TYPE="SUBMIT" VALUE="Continue">
```

Two sample POST preservation template files, fw2tr.pptemplate and tr2fw.pptemplate, are included in the following location:

- **UNIX**: *web_agent_home*/samples_default/forms/
- **Windows**: *web_agent_home*\samples_default\forms\

web_agent_home

Indicates the directory where the Web Agent is installed on your web server.

To configure the Web Agent to use your POST preservation template file, define the PostPreservationFile agent configuration parameter to specify the path of the template file.
For example:

```
PostPreservationFile="/app/netegrity/webagent/samples_default/forms/nosubmitbutto
n.pptemplate"
```

Disable POST Preservation

If you do not need to use POST preservation, you may disable it with the following parameter:

PreservePostData

Specifies whether the Web Agent preserves POST data when redirecting requests. When the user is challenged for advanced authentication, such as forms or certificate authentication, the post data is preserved during the authentication phase.

Default: Yes

To disable POST preservation, set the value of the PreservePostData parameter to no.

Using Credential Collectors Between 4.x Type and Newer Type Agents

Older versions of the SiteMinder agent objects used a security model that featured a shared secret that is stored on the Policy Server and in the WebAgent.conf file. These agents are named 4.x type agents. You can specify support for 4.x agent functions when creating an agent object in the SiteMinder Administrative UI.

Later versions of SiteMinder use a trusted host object on the Policy Sever instead of the shared secret security model.

SiteMinder supports using credential collectors between 4.x type and later agents. This usage of credential collectors is named mixed mode. Additional configuration steps are required for mixed mode deployments.

Configure Credential Collectors in a Mixed Environment

From SiteMinder r6.x to SiteMinder r12.5, the credential collectors operate differently than the older 4.x type credential collectors do. 4.x type credential collectors placed a cookie in the browser of the user, and then redirected the user back to the original agent.

In the newer SiteMinder versions, the credential collector logs the user in to the Policy Server on behalf of the agent protecting the requested resource. Cookies are *not* used.

Note: We recommend using credential collectors to log users in directly rather than setting cookies. Using credential collectors to log users in better secures user credentials because these credentials are not being passed around the network in cookies.

A credential collector requires the following information to log a user in:

- The name of the agent protecting the requested resource.
- The credentials that are supplied by the user.

To learn the Agent name, a credential collector uses the following process:

1. Use the SMAGENTNAME query parameter that the original Agent adds to the query string of the URL as it redirects to the credential collector.
2. If no Agent name is appended to the URL, use the mappings defined in the AgentName configuration parameter that is associated with the credential collector.

   Each mapping in the AgentName parameter specifies the name and IP address of a host using that collector for its protected resources.
3. If no Agent name mappings are configured, use the fully qualified host name of the target URL as the Agent name. This behavior is determined by enabling the AgentNamesAreFQHostNames configuration parameter.

   This parameter is disabled by default, so the credential collector uses the value of the DefaultAgentName parameter as the agent name.

Consider the previous implications before configuring credential collectors in a mixed environment.

Use FCCs and NTCs in a Mixed Environment

To process requests, the FCC and NTC rely on the user credentials and the name of the Web Agent that is protecting the requested resource. However, 4.x agents and third-party agents posting to the FCC and NTC do not pass the Agent name on the URL they send.

The following configuration options help FCCs and NTCs to operate with 4.x Web Agents:

Use Compatibility Mode—to enable a r5.x, r6.x, or r12.5 FCC/NTC to serve up forms for resources that are protected by 4.x agents or third-party applications, then enable the FCCCompatMode parameter. Traditional Web Agents have the FCCCompatMode parameter is enabled by default. Framework Agents have the FCCCompatMode parameter is disabled by default.

Enabling this parameter makes a r5.x, r6.x, or r12.5 Agent handle forms and NTLM credential collection like a 4.x Agent. This setting which means that a form or NTLM credential cookie is written to the browser of the user is redirected back to the Agent before logging in. This configuration permits the agents to interoperate.

When the value of the FCCCompatMode parameter is set to no, compatibility with 4.x Agents is disabled. In an r12.5 environment, set the value of the parameter to no.

Important! Setting this parameter to no removes support for version 4.x of the Netscape browser.

- Specify Agent name mappings—FCC only: If you disable backward compatibility, map the AgentName parameter to the name and IP address of each host using that FCC for its protected resources. Set up these mappings in the configuration settings of the FCC.

  Example mappings:

  myagent, 123.1.12.1

  myagent, www.sitea.com

- Use Host Names as Agent Names—FCC only**:** If the first two options in the algorithm are not optimal, you can set the value of the AgentNamesAreFQHostNames parameter to yes. This setting instructs the FCC to use the fully qualified host name in the target URL as the Agent name. For example, if the URL string includes:

  url?A=1&Target=http://www.nete.com/index.html

  The www.nete.com portion of the Target string serves as the Agent name.

  By default, this parameter is set to no. Consequently, the value of the DefaultAgentName parameter is used as the Agent name.

The following tables list guidelines for configuring r5.x, r6.x, or r12.5 and 4.x FCCs and NTCs, and describes how each behaves in a mixed environment:

Notes:

- NTLM credential collectors can redirect users from non-IIS Web Servers to IIS Web Servers.
- For framework Web Agents, refer only to the instructions where FCC compatibility mode is disabled.

Web Agent Protecting Resources	r5.x, r6.x, or r12.5 FCC in FCC Compatibility Mode	r5.x, r6.x, or r12.5 FCC - FCC Compatibility Mode Disabled
r5.x, r6.x, or r12.5	■ FCC issues a credential cookie. ■ Certificate **and** Forms authentication are disabled. ■ Certificate **or** Forms authentication are disabled.	■ FCC issues a session cookie ■ Certificate **and** Forms authentication works. ■ Certificate **or** Forms authentication works.

Web Agent Protecting Resources	4.x QMR 2/3/4 FCC
4.x QMR 5 or 4.x QMR 6	■ Agent issues a credential cookie ■ Certificate and Forms authentication are disabled. ■ Certificate or Forms authentication works
r5.x, r6.x, or r12.5	■ Agent issues a credential cookie ■ Certificate and Forms authentication are disabled. ■ Certificate or Forms authentication works

Note: For more information about SSL Authentication Schemes, see the Policy Server documentation.

Web Agent Protecting Resources	r5.x, r6.x, or r12.5 FCC in FCC Compatibility Mode	r5.x, r6.x, or r12.5 FCC - FCC Compatibility Mode Disabled
4.x QMR 5 or 4.x QMR 6	■ NTC issues a credential cookie.	■ NTC issues a session cookie

Web Agent Protecting Resources	r5.x, r6.x, or r12.5 FCC in FCC Compatibility Mode	r5.x, r6.x, or r12.5 FCC - FCC Compatibility Mode Disabled
r5.x, r6.x, or r12.5	■ NTC issues a credential cookie.	■ NTC issues a session cookie
Web Agent Protecting Resources	**4.x QMR 2/3/4 NTC**	
4.x QMR 5, 4.x QMR 6	■ Agent issues a credential cookie	
r5.x, r6.x, or r12.5	■ Agent issues a credential cookie	

Use SCCs in a Mixed Environment

To enable 4.x type Web Agents and r5.x, r6.x, or r12.5 SCCs to interoperate, do one of the following tasks:

- Specify Agent name mappings: Map the AgentName parameter to the host name and IP address of each host using that SCC for its protected resources. Create these mappings in the agent configuration parameters of the SCC.
- Use Host Names as Agent Names: If you do not specify Agent name mappings, you can set the AgentNamesAreFQHostNames parameter to Yes. This setting instructs the SCC to use the fully qualified host name in the target URL as the Agent name.

  For example, if the URL string is:

  ```
  url?A=1&Target=http://www.nete.com/index.html
  ```

  The www.nete.com portion of the Target string serves as the Agent name.

  By default, this parameter is set to no. Consequently, the value of the DefaultAgentName parameter is used as the Agent name.

The following table shows how 4.x and r5.x, r6.x, or r12.5 Agents acting as SCCs operate in a mixed environment:

Web Agent Version	4.x QMR 2/3/4 SCC	r5.x, r6.x, or r12.5 SCC
4.x QMR 5 or 4.x QMR 6	■ Agent issues an SSL credential cookie. ■ Certificates cannot be collected without redirecting requests, even if the original connection from the browser to server is over SSL.	■ Create mappings in the AgentName parameter or set AgentNamesAreFQHostNames to Yes. ■ SCC issues a session cookie ■ Certificates cannot be collected without redirecting requests, even if the original connection from the browser to server is over SSL.
r5.x, r6.x, or r12.5	■ Agent issues an SSL credential cookie. ■ Certificates can be collected without redirecting requests.	■ SCC issues a session cookie ■ Certificates can be collected without redirecting requests.

Note: For more information about SSL Authentication Schemes, see the Policy Server documentation.

Configure Advanced FCC Settings

You can configure any of the following advanced credential collector settings to suit your needs:

- Specify the protocol portion of URLs using lowercase characters (see page 153).
- Authenticate Passport users with a form (see page 154).
- Authenticate SafeWord users with a form (see page 155).
- Authenticate ACE users (see page 155).
- Encrypt Query Strings in redirection URLs (see page 156).
- FCC directive for encoding query strings in redirection URLs (see page 157).
- Configure the FCC to allow Windows authentication (see page 160).
- Using Application Request Routing with FCCs (see page 157).

Specify Redirect URL Protocols with Lowercase Characters

If you protect legacy applications that do not confirm to RFC 2396 with a forms-based authentication scheme, and you need the protocol portions of URLs to be lowercase, then set the following parameter:

LowerCaseProtocolSpecifier

Specifies whether the scheme (protocol) portion of a redirect URL uses only lowercase characters. This configuration parameter accommodates legacy applications that do not confirm to RFC 2396. This RFC states that applications must handle the protocol portion of a URL in both uppercase and lowercase. Change this parameter in any of the following situations:

- You use legacy applications that do not confirm to RFC 2396.
- Your redirect URLS contain query data.
- You use an HTML-forms (FCC) authentication scheme.

Default: No (uppercase characters are used HTTP, HTTPS).

Example: Yes (lowercase characters are used http, https).

To specify lowercase protocols for the URLs in your environment, set the value of the LowerCaseProtocolSpecifier parameter to yes.

Use a Special Forms Template for Passport Authentication

Beginning with Web Agent 5.x QMR1, n FCC file named loginusername.fcc was provided for use with the Passport authentication scheme. If you configure SiteMinder to use this form, when a user requests a protected resource, SiteMinder will:

1. Recognize a signed-in Passport user as a mapped user from the SiteMinder user directory.
2. Present the form, which:
   - Automatically displays the user name of the mapped user
   - Prompts for the corresponding password

To use the loginusername.fcc file:

1. Edit the value of the IgnoreExt Web Agent parameter by removing the .fcc entry from the list of extensions that the Agent should ignore.
2. Protect loginusername.fcc, using the Passport (Custom) authentication scheme.

   Note: For more information, see the Policy Server documentation.
3. For each realm protected by the Passport authentication scheme, create a response on the Policy Server. For each response, configure a Web Agent response attribute as follows:
   a. Select WebAgent-HTTP-Header-Variable from the Attribute drop-down list.
   b. Select the User Attribute radio button from the Attribute Kind group box.
   c. In the Attribute Name field, enter the name of the user directory attribute that corresponds to the user name or user id. For example, if an LDAP directory contains the users that are mapped to Passport holders, enter uid.
   d. In the Variable Name field, enter a name for the response variable, such LDAPUID.

      Note: For more information, see the Policy Server documentation.
4. Edit the loginusername.fcc form to reflect the Variable Name value. Continuing with this example, the variable name is LDAPUID.

You can add these advanced features to the Agent configuration file or an Agent Configuration Object.

Use the safeword.fcc File for SafeWord Forms Authentication

The Policy Server can authenticate users against a SafeWord authentication server, including users who are logging in via SafeWord hardware tokens.

One of the prerequisites for using the SafeWord forms-based authentication scheme is to have a customized safeword.fcc file residing on a web server where the SiteMinder Web Agent is installed. This web server must be in the cookie domain in which you implement HTML Forms authentication.

The safeword.fcc file defines the forms that a user sees during SafeWord authentication. Depending on the value of the authentication code sent by the Policy Server to the credential collector, the form that the user is asked to fill out changes. In the safeword.fcc file you can see the different text for each authentication code, as indicated by the directive smauthreason.

To customize the safeword.fcc file for your enterprise, you can modify the HTML layout of the form but not the type of credentials that the user must provide for a particular form. You may also want to modify the form logo. The file uses ISO-8859-1 encoding.

The sample safeword.fcc file is located in the directory:

web_agent_home/Samples/Forms

Note: For more information, see the Policy Server documentation.

How to use Forms with ACE Authentication

If you want to use a SiteMinder form together with an ACE authentication scheme, do one of the following:

- If you are creating a new authentication scheme, use the following form as a template:

  web_agent_home\samples_default\forms\smaceauth.fcc

- If you are modifying an existing authentication scheme to use ACE, then add the following directive below the other directives in the forms.fcc file:

  ```
  @smacefcc=1
  ```

Encrypt Query String Parameters in Redirection URLs

The following parameter enables the Web Agent to encrypt all SiteMinder query parameters in a redirect URL:

SecureURLs

Specifies whether the Web Agent encrypts the SiteMinder query parameters in a redirect URL. You can use this setting to provide additional security for requested resources protected by an advanced authentication scheme, Password Services, or when a request invokes the Cookie Provider.

Important! The Web Agent only encrypts data sent between SiteMinder components. The data sent for redirects to non-SiteMinder applications is not encrypted.

The following SiteMinder credential collectors and applications support the SecureUrls functionality:

- HTML Forms authentication
- Cert And Forms authentication
- SSL Authentication
- Cert or Forms authentication
- NTLM authentication
- ACE authentication
- SafeWord authentication
- User self registration
- Multi-domain Single Sign-on with Cookie Provider
- FCC-based Password Services (not CGI- or JSP-based)

Default: No

Follow these steps:

1. Set the value of the SecureURLs parameter to yes.
2. To encrypt query string parameters in redirection URLs within a single sign-on environment, ensure that all Web Agents in the single sign-on environment have the SecureURL parameter set to the same value.
3. If you are using custom FCCs, add the smquerydata directive with the other FCC directives (such as TARGET) to the custom FCC.

   Query string parameters are encrypted in SiteMinder redirection URLs.

FCC Directive for Encoding Query Strings of Redirect URLs

You can encrypt the query strings of redirect URLs for credential collectors. The credential collectors provide the keys that are used to encrypt the query data.

For forms authentication schemes, the query string directive, smquerydata, is part of the FCC template. The agent serving the FCC uses this directive to send the encrypted query data to the target agent when the FCC is posted.

The following directive is used:

```
<INPUT type='hidden' name='smquerydata' value='$$smquerydata$$>
```

Note: If you are using custom FCCs, add the smquerydata directive with other FCC directives, such as TARGET to the custom FCC.

SiteMinder r12.5 agents with the SecureUrls parameter enabled can operate only with credential collectors served from other agents that support this functionality.

How to Configure Application Request Routing (ARR) for HTML Forms Authentication

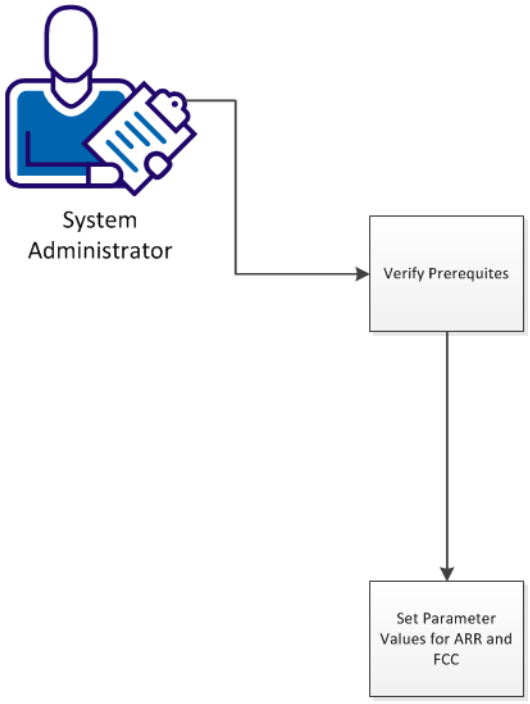
Application Request Routing (ARR) is an optional feature that is available for Microsoft Internet Information Services (IIS). ARR directs requests to other servers, much like a proxy server.

The IIS web server processes cookies differently with ARR. This configuration affects how SiteMinder cookies are processed with FCC authentication schemes.

This scenario describes the additional configuration settings that <stmdnr> agents require in any of the following situations:

- ARR is used with FCC.
- ARR is used with SiteMinder and Arcot.

The following illustration shows how a system administrator configures SiteMinder for ARR with FCC:

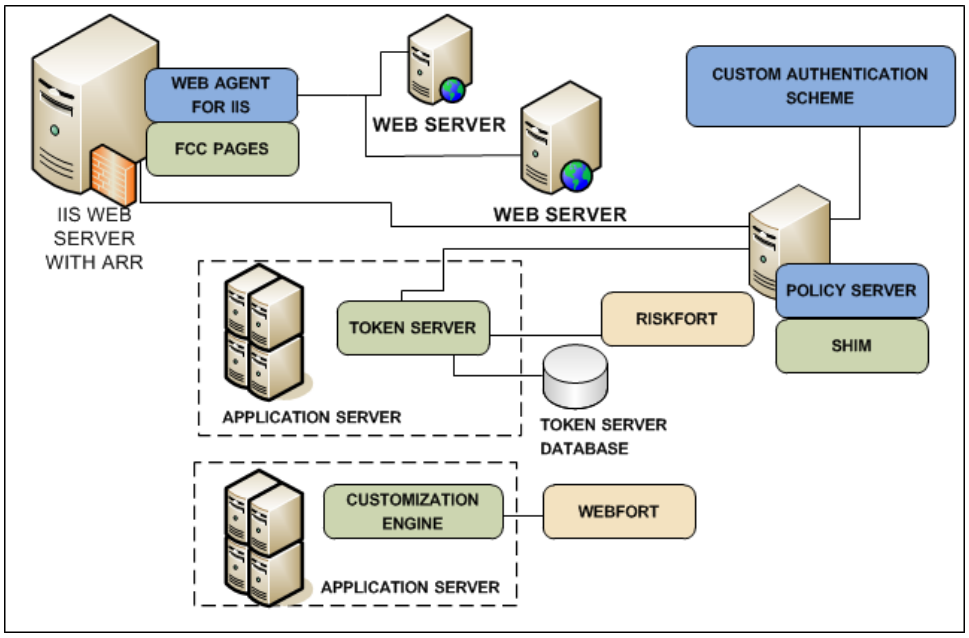


To configure your SiteMinder agents to use ARR and FCCs together, follow these steps:

1. Verify that your environment meets the prerequisites (see page 159).
2. Set the parameter values for ARR and FCC (see page 160).

Verify Prerequisites

The following illustration describes the components and prerequisites of your environment:



Verify that your SiteMinder and CA DataMinder environments meet the following requirements:

- A complete SiteMinder environment is installed and configured, with the following components:
  - Policies to protect resources on web servers that are deployed behind an IIS web server running ARR.
  - (Optional) CA Arcot components that are installed and configured.
- An IIS web server with the following items installed and configured:
  - Application Request Routing (ARR) configured to forward requests to web servers.
  - A SiteMinder Agent for IIS installed and configured on the server running ARR.
  - An FCC authentication scheme.
- Determine if your agents use central or local configuration.

Set the Parameter Values for ARR and FCC

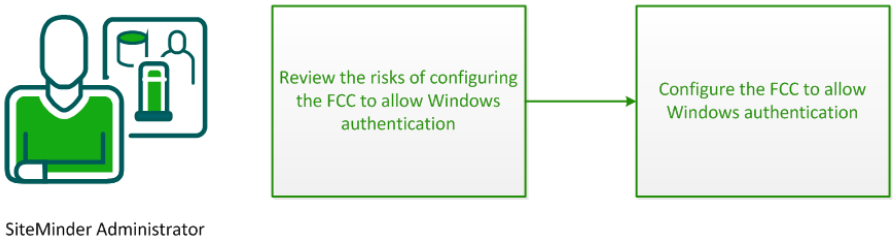
To set the parameter values for ARR and FCC, follow these steps:

1. Perform the task from the following list that corresponds to your agent configuration method:
   - For central configuration, open your agent configuration object (see page 30).
   - For local configuration, open the local configuration file on your web server (see page 34).
2. Locate the FCCCompatMode parameter, and then change its value to yes.
3. Locate the CookieDomain parameter, and then change its value to none (do *not* leave the value blank).

How to Configure the FCC to Allow Windows Authentication

The SiteMinder Forms Credential Collector (FCC) is designed to enable CA Services to trigger custom authentication schemes securely. As such, the FCC can authenticate users against any authentication scheme. However, the FCC does *not* authenticate against Windows authentication schemes by default. This behavior prevents an attacker from exploiting the FCC to generate a SiteMinder session for any valid Windows user in certain configurations.

If your environment requires the FCC to authenticate against the Windows authentication scheme, you can enable it by specifying the EnableFCCWindowsAuth agent configuration parameter. However, before you enable FCC support for Windows authentication, review the risks of doing so and be aware of configurations that expose the vulnerability.



1. Review the risks of enabling the FCC to allow Windows authentication (see page 161).
2. Configure the FCC to allow Windows authentication (see page 162).

Risks of Enabling the FCC to Allow Windows Authentication

By default, the FCC does not authenticate against Windows authentication schemes. You can enable the FCC to allow Windows authentication. However, doing so exposes a vulnerability whereby an attacker could use an FCC to generate a SiteMinder session for any valid Windows user in certain configurations.

The vulnerability is present in configurations in which the same SiteMinder Agent name or Agent group name is used in both an HTML Forms-protected realm and a Windows-protected realm. For example, a configuration in which a single Web Agent is configured to protect different realms that are configured with HTML Forms and Windows authentication.

Consider the following example scenario:

- Resource A is configured in a realm protected using HTML Forms authentication. The FCC challenges users accessing Resource A with an HTML form.
- Resource B is configured in a realm protected using Windows authentication. Users accessing Resource B complete Windows authentication.
- Both resources are hosted on the same IIS Server and are protected by the same Web Agent. Both realms are therefore configured with the same Agent name.

The attack occurs as follows:

1. The attacker modifies the TARGET parameter in the HTML form from "Resource A" to "Resource B."
2. The attacker submits the form with any valid Windows username.
3. The FCC passes the username to the Policy Server for authentication. SiteMinder executes the Windows authentication scheme instead of the HTML Forms authentication scheme and the username is validated.

The result is a SiteMinder session returned to the user which enables single sign-on for all following requests where the new session is considered valid. The attacker is now impersonating the user whose Windows username was submitted to the FCC.

Configure the FCC to Allow Windows Authentication

You configure the FCC to allow Windows authentication by specifying the following agent configuration parameter:

EnableFCCWindowsAuth

Specifies whether an agent, acting as an FCC, can authenticate users against resources that the SiteMinder Windows authentication scheme protects.

This parameter uses the following values:

- Yes—FCCs can authenticate against a Windows authentication scheme.

  Important! When this parameter is set to Yes, an attacker can potentially exploit the FCC to impersonate Windows users without providing required credentials.

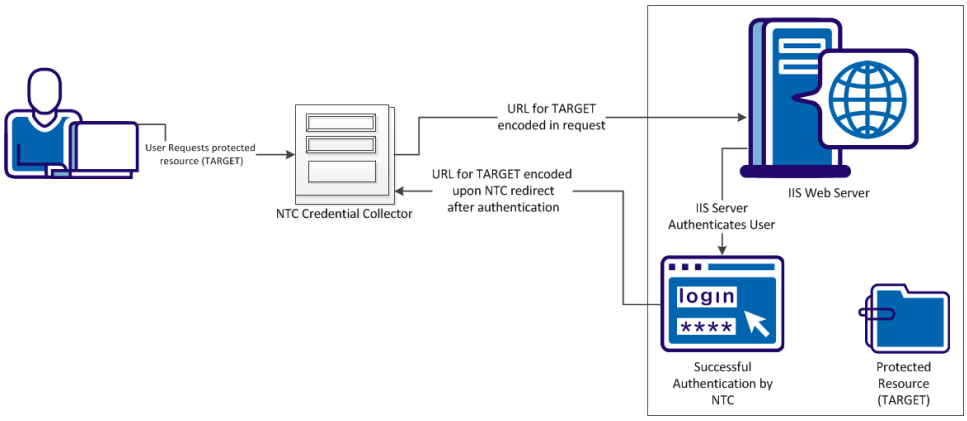
- No—FCCs cannot authenticate against a Windows authentication scheme.

Default: No

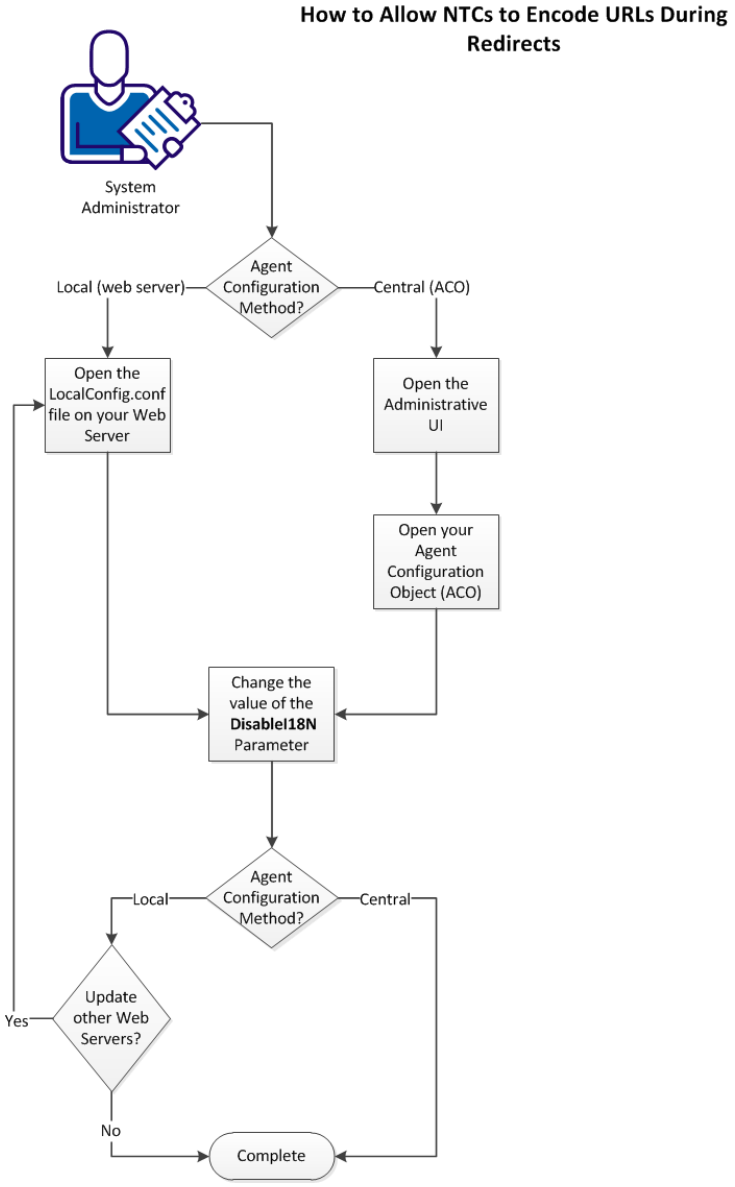
How to Allow the NTC to Encode URLs During Redirects to Protected Resources

SiteMinder can protect resources using Windows credential collectors (NTCs). Users submit their credentials to the NTC, then the NTC logs the user in to the IIS web server. The IIS web server authenticates the user. The NTC redirects the user to the protected (TARGET) resource after authentication.

The NTC normally encodes the characters in the TARGET portion of the URL during the request, but not during the redirect after authentication. You can change your agent configuration so that the TARGET portion of the URL is encoded during the redirect. The following illustration describes this behavior:



The following illustration shows the process of allowing the NTC to encode URLs during requests for protected resources:



To allow the NTC to encode URLs during re–directs to protected resources, follow these steps:

1. Choose the procedure that matches your agent configuration method from the following list:
   - For agents using an Agent Configuration object (ACO) on a Policy Server, follow these steps:
     a. Open the Administrative UI (see page 164).
     b. Open your Agent configuration object (ACO), and then change the value of the DisableI18N parameter (see page 165).
   - For agents using a local configuration file on a web server, follow these steps:
     a. Open the LocalConfig.conf file on your web server with a text editor, and then change the value of the DisableI18N parameter (see page 167).
2. For agents using local configuration, repeat Step 1c for each web server.

   The NTC uses encoded URLs during redirects to protected resources.

Open the Administrative UI to Change Policy Server Objects

Change the objects on your Policy Server by opening the Administrative UI.

Follow these steps:

1. Open the following URL in a browser.

   `https://host_name:8443/iam/siteminder/adminui`

   host_name

   Specifies the fully qualified Administrative UI host system name.
2. Enter your SiteMinder superuser name in the User Name field.
3. Enter the SiteMinder superuser account password in the Password field.

   Note: If your superuser account password contains one or more dollar-sign ($) characters, replace each instance of the dollar-sign character with $DOLLAR$ in the Password field. For example, if the SiteMinder superuser account password is $password, enter $DOLLAR$password in the Password field.
4. Verify that the proper server name or IP address appears in the Server drop-down list.
5. Select Log In.

Change the Value of the DisableI18N parameter in your Agent Configuration Object

You can configure Windows credential collectors to process HTTP encoded characters in target URLs for centrally configured web agents. Centrally–configured web agents use parameter settings stored in an Agent Configuration object on the Policy Server.

Follow these steps:

1. Click the Infrastructure, Agent Configuration Objects.

   A list of Agent Configuration objects appears.

   Click the edit icon in the line Agent Configuration Object you want.

   The Modify Agent Configuration dialog appears.

2. Click the edit icon to the left of the following parameter:

   DisableI18N

   Specifies how the Windows credential collector (NTC) processes the TARGET URL during authentication when the characters of the TARGET URL use HTTP encoding. When the value of this parameter is *no*, any characters in the URL are decoded during authentication. The decoded characters are used in the redirect to the TARGET resource. When the value of this parameter is *yes*, characters in the TARGET URL are not decoded during authentication. Any characters using HTTP encoding remain encoded before and after authentication.

   Default: No.

   The Edit Parameter dialog appears.

3. Change the text in the Value field to yes.

4. Click OK.

   The Edit Parameter dialog closes, and the Modify Agent Configuration dialog appears.

5. Click the edit icon to the left of the following parameter:

   BadUrlChars

   Specifies the character sequences that cannot be used in URL requests. The Web Agent checks the characters in the URL that occur before the "?" character against the list in this parameter. If any of the specified characters are found, the Web Agent rejects the request.

   You can specify the following characters:

   - a backward slash (\)
   - Two forward slashes (//)
   - Period and a forward slash (./)
   - Forward slash and a period (/.)

- Forward slash and an asterisk (/*)
- An asterisk and a period (*.)
- A tilde (~)
- %2d
- %20
- %00-%1f
- %7f-%ff
- %25

Separate multiple characters with commas. Do *not* use spaces.

You can use the bad URL characters in CGI parameters if the question mark (?) precedes the bad URL characters.

Default: //,./,/.,/*,*.,~,\,%00-%1f,%7f-%ff,%25

Limits:

- The default hexadecimal numbers apply to English characters. For other languages, remove any hexadecimal values that correspond to the characters of the language that you want to allow. Examples of such languages include (but are not limited to), Brazilian Portuguese, French, Japanese, and Chinese.
- You can specify characters literally. You can also enter the URL-encoded form of that character. For example, you can enter the letter a, or you can enter the encoded equivalent of %61.
- You can specify a maximum number of 4096 characters (including commas that are used for separating characters).
- You can specify ranges of characters that are separated with hyphens. The syntax is: *starting_character-ending_character*. For example, you can enter a-z as a range of characters.
- Specify any quotation marks (") with the URL-encoded equivalent of %22. Do *not* use ASCII.

The Edit Parameter dialog appears.

6. Remove the following text from the Value field:

   ,%25

7. Click OK.

   The Edit Parameter dialog closes, and the Modify Agent Configuration dialog appears.

8. Click Submit.

   The Modify Agent Configuration dialog closes, and a confirmation message appears.

9. (Optional) Enter any remarks about the change in the Comment field for future reference.

10. Click Yes.

    Your changes will be applied the next time the Web Agent polls the Policy Server.

Change the Value of the DisableI18N parameter in your LocalConfig.conf File

You can configure Windows credential collectors to process HTTP encoded characters in target URLs. Locally–configured web agents use parameter settings stored in a configuration file on each web server.

Follow these steps:

Locate the LocalConfig.conf file on your web server. Use the examples in the following list to locate the file on your type of web server:

IIS web server

web_agent_home\bin\IIS

Oracle iPlanet web server

Oracle_iPlanet_home/https-hostname/config

Apache web server

Apache_home/conf

1. Open your LocalConfig.conf file with a text editor, and then locate the following parameter:

   DisableI18N

   Specifies how the Windows credential collector (NTC) processes the TARGET URL during authentication when the characters of the TARGET URL use HTTP encoding. When the value of this parameter is *no*, any characters in the URL are decoded during authentication. The decoded characters are used in the redirect to the TARGET resource. When the value of this parameter is *yes*, characters in the TARGET URL are not decoded during authentication. Any characters using HTTP encoding remain encoded before and after authentication.

   Default: No.

2. Change the value of the DisableI18n parameter to yes.

3. Locate the following parameter:

   BadUrlChars

   Specifies the character sequences that cannot be used in URL requests. The Web Agent checks the characters in the URL that occur before the "?" character against the list in this parameter. If any of the specified characters are found, the Web Agent rejects the request.

   You can specify the following characters:

   - a backward slash (\)
   - Two forward slashes (//)
   - Period and a forward slash (./)
   - Forward slash and a period (/.)
   - Forward slash and an asterisk (/*)
   - An asterisk and a period (*.)
   - A tilde (~)
   - %2d
   - %20
   - %00-%1f
   - %7f-%ff
   - %25

   Separate multiple characters with commas. Do *not* use spaces.

   You can use the bad URL characters in CGI parameters if the question mark (?) precedes the bad URL characters.

Default: //,./,/.,/*,*.,~,\,%00-%1f,%7f-%ff,%25

Limits:

- The default hexadecimal numbers apply to English characters. For other languages, remove any hexadecimal values that correspond to the characters of the language that you want to allow. Examples of such languages include (but are not limited to), Brazilian Portuguese, French, Japanese, and Chinese.
- You can specify characters literally. You can also enter the URL-encoded form of that character. For example, you can enter the letter a, or you can enter the encoded equivalent of %61.
- You can specify a maximum number of 4096 characters (including commas that are used for separating characters).
- You can specify ranges of characters that are separated with hyphens. The syntax is: *starting_character-ending_character*. For example, you can enter a-z as a range of characters.

Specify any quotation marks (") with the URL-encoded equivalent of %22. Do *not* use ASCII.

4. Remove the following values from the BadURLChars list:

   ,%25

5. Save the changes to your LocalConfig.conf file, and then close the text editor.
6. Repeat Steps 1 through 5 on all web servers which you want to change.

   Windows credential collectors are allowed to process HTTP encoded characters in TARGET URLs.

Chapter 12: Agents and Password Services

This section contains the following topics:

How to Configure FCC Password Services (see page 171)
Password Services Implementations (see page 171)

How to Configure FCC Password Services

To configure password services, follow these steps:

1. Open the Administrative UI
2. Create password policies that are associated with a user directory in your SiteMinder environment. Use the following path in the Redirection URL field:

   ```
   /siteminderagent/forms/smpwservices.fcc
   ```

Note: For more information, see the Policy Server documentation.

Password Services Implementations

SiteMinder uses forms credential collectors (FCCs) to support password services.

Password services help you do the following tasks:

- Encrypt the query strings in password services URLs (see page 172).
- Support password services in multiple languages (see page 173).
- Redirect password services users to a fully qualified URL (see page 174).
- Support SecureID authentication with password services (see page 175).
- Allow users to change their own passwords. Use whichever of the following procedures applies to your situation:
  - Password changes when the SecureURLs parameter is no (see page 176).
  - Password changes when the SecureURLs parameter is yes (see page 178).
  - Password changes when using the Basic Authentication or X.509 Certificate authentication schemes (see page 180).

FCC Password Services and URL Query Encryption

The FCC Password Services application enables query data on the URL to be encrypted, further securing Agent interactions. You can only encrypt query data with FCC Password Services. FCC Password Services files include:

- smpwservices.fcc

  This FCC is installed with the Web Agent and is located at:

  web_agent_home/samples/forms

  If Password Services is invoked and there is no password policy configured, the SiteMinder Administrator at the Policy Server should set the environment variable NETE_PWSERVICES_REDIRECT to a relative path for smpwservices.fcc.

  The path is:

  /siteminderagent/forms/smpwservices.fcc

  The new FCC displays the Password Services form based on the FCC directives authreason and username.

- smpwservices.unauth

  This file handles errors that occur during GET/POST actions of the Password Services forms.

  This file is similar to other FCC unauthorized files that are invoked if there is a failure processing the request during the POST. This FCC handles error conditions, such as an empty TARGET variable. The error reporting is intended to be synchronized with the CGI-based Password Services and for handling any other unknown errors caused by an FCC POST.

- smpwservicesUS-EN.properties

  This properties file is used by smpwservices.fcc to display the user-friendly messages on the Password Services forms.

  This properties file has the user-friendly messages, which an administrator can modify depending on what he wants to display on the Password Services forms. The format for the message is name=value.

How to Localize FCC-based Password Services Change Forms

To localize the user messages for FCC-based Password Services for another locale follow these steps:

1. Create an FCC folder on the web server for a new locale or use an existing folder if appropriate for your locale. The typical naming convention for the folder is forms*locale.*

   Note: The directories and file names that are shown could be case-sensitive, depending on your operating environment and the type of web server in use.

2. Place a copy of the relevant Password Services files in the new folder.
3. Modify the files to accommodate the locale, such as changing the English messages to the language for your locale. Repeat this step with all the files for the locale.
4. In the Administrative UI, change the value of the Redirection URL field in the Password Policy.

   For example, to use FCC Password Services for Japanese users, put a copy of the following files in the folder formsja, which is located in *web_agent_home*/samples:

   - smpwservices.fcc, located in *web_agent_home*/samples/forms
   - smpwservices.unauth, located in *web_agent_home*/samples/forms
   - A new properties file, smpwservicesja.properties

Use a Fully Qualified URL for Password Services Redirects

When you use password services you can instruct a Web Agent to create a fully qualified domain name (FQDN) to where users are redirected. Use the following parameter:

ConstructFullPwsvcUrl

Instructs the agent to add the server name (FQDN) of the system that is hosing the password services before redirecting the user. You define this server name in the password policy on the Policy Server.

For example, suppose that the value of this parameter is yes, and your password policy points to siteminderagent/forms/smpwservices.fcc. the Web Agent redirects to the following URL:

`HTTP://server_name.example.com/siteminderagent/forms/smpwservices.fcc`

The Web Agent uses the value that is defined in your password policy when the value of this parameter is no. For example, if your password policy only points to a subdirectory, the Web Agent redirects users to that subdirectory.

Default: No.

Example: No (redirects to the /siteminderagent/forms/smpwservices.fcc defined in your password policy).

Example: Yes (adds HTTP://*server_name*.example.com to the /siteminderagent/forms/smpwservices.fcc defined in your password policy).

The default URL for password policies in the Administrative UI does *not* contain a server name. The Web Agent redirects users to whatever URL exists in the password policy when the value of the previous parameter is set to yes.

Follow these steps:

1. Use the examples in the following table as a guide for setting the ConstructFullPwsvcURl parameter:

To:	Add this URL to your password policy in the Administrative UI:	Set the value of the ConstructFullPwsvcURl to:
Host the password services on a specific server.	http://server_name.example.com:80/siteminderagent/forms/smpwservices.fcc	No
Host the password services on the same server as the Web Agent using a relative URL.	siteminderagent/forms/smpwservices.fcc	No

To:	Add this URL to your password policy in the Administrative UI:	Set the value of the ConstructFullPwsvcURl to:
Host the password services on the same server as the Web Agent using an FQDN.	siteminderagent/forms/smpwservices.fcc	Yes

Configure SecureID Authentication with FCC Password Services

You must modify the SecureID HTML Form template using the Administrative UI if you are using SecureID as your authentication scheme and both of the following conditions exist in your environment:

- The FCC Password Services feature is configured
- The value of the SecureUrls parameter for the Web Agent is set to yes

SecureID is implemented using Password Services, which is why you must modify the authentication scheme's template.

To configure SecureID Authentication with FCC password services, add the path to the smpwservices.fcc file in the Target field of the SecureID template, as shown in the following example:

```
/siteminderagent/forms/smpwservices.fcc
```

How to Enable User-Initiated Password Changes with FCCs

You can configure the FCC Password services features of SiteMinder to allow users to change their own passwords whenever they want.

Note: Use the following process only if your SiteMinder Web Agent configuration also has the value of the SecureURLs parameter that is set to no.

To enable user-initiated password changes with FCCs, use the following process:

1. Confirm that your user directory contains attributes that support Password Policies.
2. Use the Administrative UI to do the following tasks:
   a. Create an FCC-based password policy and protect the resources that you want.
   b. Configure the password policy to allow authorized users to change their passwords.
3. Create a password change URL that includes the following parts:
   - The FQDN of the logon server (example: http:logonserver.example.com).
   - The URI of the FCC-based Password services (example: siteminderagent/forms/smpwservices.fcc?).
   - The name of the SiteMinder Web Agent (SMAGENTNAME)
   - *One* of the following target URLs:
   – For password-change URLs embedded in FCC pages, use the relative values for the (SMAGENTNAME) and (TARGET) sections, as shown in the following example:

     ```
     <a
     href="http:logonserver.example.com/siteminderagent/forms/smpwservices.fcc
     ?SMAUTHREASON=
     34&SMAGENTNAME=$$smencode(smagentname)$$&TARGET=$$smencode(target)$$">Cha
     nge Password</font></a>
     ```

   – For password-change URLs *not* embedded in FCC pages, hard-code the name of your SiteMinder Agent for the (SMAGENTNAME) section. Then hard-code a fully qualified domain name value for the (TARGET) section, as shown in the following example:

     ```
     <a
     href="http://logonserver.example.com/siteminderagent/forms/smpwservices.f
     cc?SMAUTHREASON=34&SMAGENTNAME=Agent1&TARGET=https://logonserver.example.
     com/protected/myprotectedpage.html">Change Password</font></a>
     ```

4. Embed the password-change URL (from Step 3) as a link in one or more unprotected web pages.
5. Test the password change function with the following steps:
   a. Display a web page that has the password change link you created in Step 3.

b. Click the password change link.

   The password change form appears.

c. Fill out the password change form and submit it.

   If the password change is successful, a confirmation page appears with a link to the protected target resource.

d. Click the link and verify that the resource appears.

e. Close and reopen your browser. Try to access the protected resource using your new password.

   If you can access the resource with your new password, the password change is successful.

How to Enable User-Initiated Password Changes with FCCs (SecureURLs=Yes)

You can configure the FCC Password services features of SiteMinder to allow users to change their own passwords whenever they want.

Note: Use the following process only if your SiteMinder Web Agent configuration also has the value of the SecureURLs parameter that is set to yes.

To enable user-initiated password changes with FCCs, use the following process:

1. Confirm that your user directory contains attributes that support Password Policies.
2. Use the Administrative UI to do the following tasks:
   a. Create an FCC-based password policy and protect the resources that you want.
   b. Configure the password policy to allow authorized users to change their passwords.
   c. Set the value of the ValidTargetDomain parameter to the domain of the target resource you want to protect.
3. Create a password change URL that includes the following parts:
   - The FQDN of the logon server (example: http:logonserver.example.com).
   - The URI of the FCC-based Password services (example: siteminderagent/forms/smpwservices.fcc?).
   - The name of the SiteMinder Web Agent (SMAGENTNAME)
   - *One* of the following target URLs:
   - For password-change URLs embedded in FCC pages, use the relative values for the (SMAGENTNAME) and (TARGET) sections, as shown in the following example:

     ```
     <a
     href="http:logonserver.example.com/siteminderagent/forms/smpwservices.fcc
     ?SMAUTHREASON=
     34&SMAGENTNAME=$$smencode(smagentname)$$&TARGET=$$smencode(target)$$">Cha
     nge Password</font></a>
     ```

   - For password-change URLs *not* embedded in FCC pages, hard-code the name of your SiteMinder Agent for the (SMAGENTNAME) section. Then hard-code a fully qualified domain name value for the (TARGET) section, as shown in the following example:

     ```
     <a
     href="http://logonserver.example.com/siteminderagent/forms/smpwservices.f
     cc?SMAUTHREASON=34&SMAGENTNAME=Agent1&TARGET=https://logonserver.example.
     com/protected/myprotectedpage.html">Change Password</font></a>
     ```

4. Embed the password-change URL (from Step 3) as a link in one or more unprotected web pages.

5. Open the following file on your web server:

   web_agent_home/samples/forms/smpwservices.fcc

   a. Locate the following line:

      ```
      @smpwselfchange=0
      ```

   b. Change the 0 (zero) at the end of the previous line to 1 (one), as shown in the following example:

      ```
      @smpwselfchange=1
      ```

   c. Save and close the smpwservices.fcc file.

6. Embed the URL you created in Step 3 as a link in one or more unprotected web pages.

7. Test the password change function with the following steps:

   a. Display a web page that has the password change link you created in Step 3.

   b. Click the password change link.

      The password change form appears.

   c. Fill out the password change form and submit it.

      If the password change is successful, a confirmation page appears with a link to the protected target resource.

   d. Click the link and verify that the resource appears.

   e. Close and reopen your browser. Try to access the protected resource using your new password.

      If you can access the resource with your new password, the password change is successful.

How to Enable User-Initiated Password Changes when using the SiteMinder X.509 Certificate and Basic Authentication Scheme

You can configure the FCC Password services features of SiteMinder to allow users to change their own passwords. The SiteMinder X.509 Certificate and Basic authentication scheme requires a password-change URL that starts with the HTTPS protocol.

Follow these steps:

1. Confirm that your user directory contains attributes that support Password Policies.
2. Use the Administrative UI to do the following tasks:
   a. Create an FCC-based password policy and protect the resources that you want.
   b. Configure the password policy to allow authorized users to change their passwords.
3. Create a password change URL that includes the following parts:
   - The HTTPS scheme (protocol).
   - The FQDN of the logon server (example: http:logonserver.example.com).
   - The URI of the FCC-based Password services (example: siteminderagent/forms/smpwservices.fcc?).
   - The name of the SiteMinder Web Agent (SMAGENTNAME).
   - *One* of the following target URLs:
   - For password-change URLs embedded in FCC pages, use the relative values for the (SMAGENTNAME) and (TARGET) sections, as shown in the following example:

     ```
     <a
     href="https:logonserver.example.com/siteminderagent/forms/smpwservices.fcc?SMAUTHREASON=
     34&SMAGENTNAME=$$smencode(smagentname)$$&TARGET=$$smencode(target)$$">Change Password</font></a>
     ```

   - For password-change URLs *not* embedded in FCC pages, hard-code the name of your SiteMinder Agent for the (SMAGENTNAME) section. Then hard-code a fully qualified domain name value for the (TARGET) section, as shown in the following example:

     ```
     <a
     href="https://logonserver.example.com/siteminderagent/forms/smpwservices.fcc?SMAUTHREASON=34&SMAGENTNAME=Agent1&TARGET=https://logonserver.example.com/protected/myprotectedpage.html">Change Password</font></a>
     ```

4. Embed the password-change URL (from Step 3) as a link in one or more unprotected web pages.
5. Test the password change function with the following steps:

a. Display a web page that has the password change link you created in Step 3.

b. Click the password change link.

   The password change form appears.

c. Fill out the password change form and submit it.

   A confirmation page appears with a link to the protected target resource.

d. Click the link and verify that the resource appears.

e. Close and reopen your browser. Try to access the protected resource using your new password.

   If you can access the resource with your new password, the password change is successful.

Chapter 13: Single Sign-On (SSO)

This section contains the following topics:

Allow Automatic Access to Resources that use the OPTIONS Method (see page 183)
How Single Sign-on Works in a Single Domain (see page 184)
Single Sign-On Across Multiple Domains (see page 185)
Hardware Load Balancers and Single Sign-On Across Multiple Cookie Domains (see page 186)
Single Sign-On and Authentication Scheme Protection Levels (see page 188)
Single Sign-on and Agent Key Management (see page 188)
How to Configure Single Sign-On (see page 189)

Allow Automatic Access to Resources that use the OPTIONS Method

The SiteMinder Web Agent still challenges authenticated users who attempt to access resources that use the OPTIONS method. Some examples of resources that use the OPTIONS method include (but are not necessarily limited to) the following:

- Microsoft® Word documents
- Microsoft® Excel® spreadsheet documents

This challenge occurs because the application associated with the resource sends a request using the OPTIONS method to the web server. Because this request does *not* include a SiteMinder cookie, the Web Agent issues a challenge.

To prevent users from being challenged for these resources

1. Set the value of the following parameter to yes:

   autoauthorizeoptions

   Automatically authorizes any requests for resources which use the HTTP OPTIONS method.

   If you set the value of this parameter to yes, also set the value of the PersistentCookies parameter to no.

   Limits: yes, no

2. Set the value of the PersistentCookies parameter to no.

How Single Sign-on Works in a Single Domain

SiteMinder provides single sign-on functionality across single and multiple cookie domains. This simplifies using applications across different Web servers and platforms, and improves the user experience because the users do not have to re-authenticate as they move across a single sign-on environment.

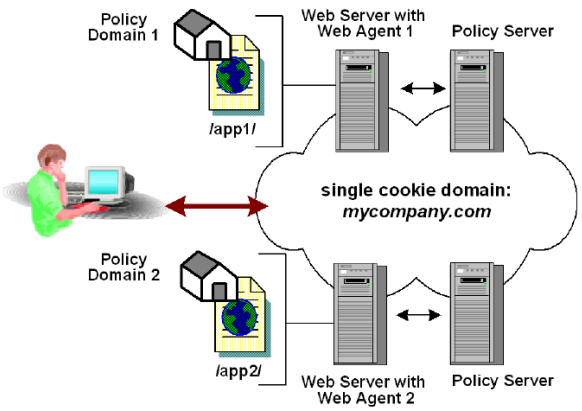
A single domain is an environment where all resources exist in the same cookie domain. Multiple Web Agents in the same cookie domain can be configured for single sign-on if you specify the same cookie domain in each Web Agent's configuration.

If single sign-on is enabled, it uses the following process:

1. The user authenticates once.
2. The Web Agent caches the successful authentication, and then issues a single sign-on cookie to the user's browser.
3. The single sign-on cookie provides the session information, so that users can access the following types of resources without reauthenticating:
   - Protected resources in other realms with an *equal* or *lower* protection level
   - Another web server within this cookie domain

   Users who try to access resources with a *higher* protection level must re-authenticate before they are granted access.

The following illustration shows single sign-on in a single cookie domain:

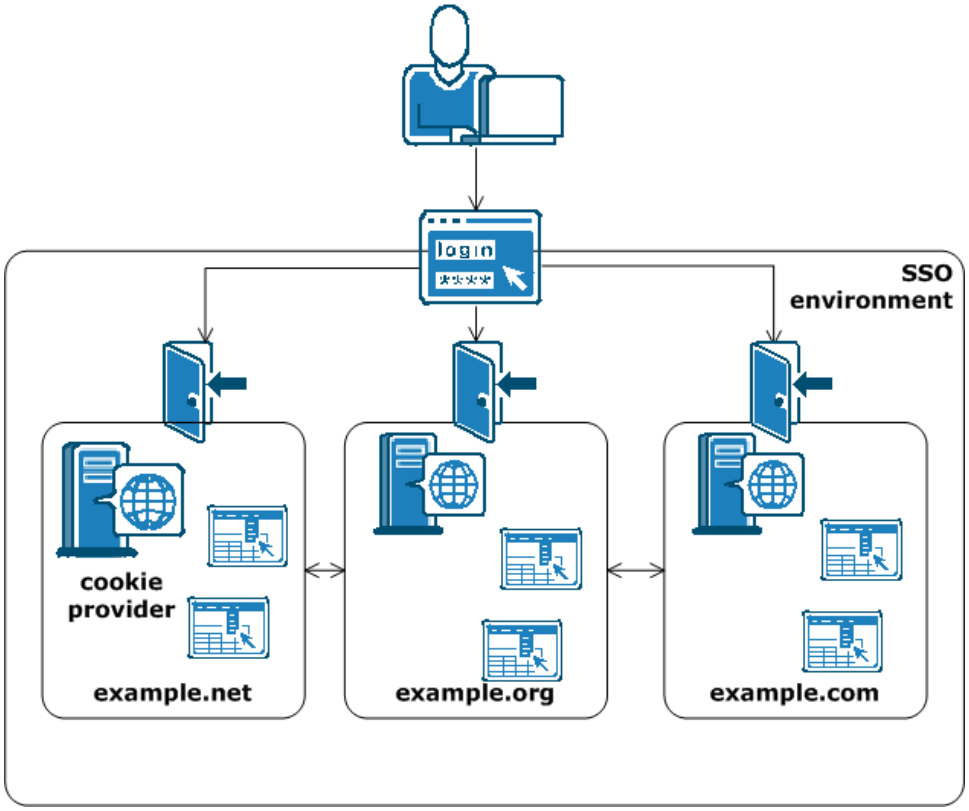


Note: If you are using replicated user directories with non replicated policy stores, the user directory must be named identically for all policy stores. Also, the session ticket key, which encrypts session tickets, must be the same for all key stores in the SSO environment. The session ticket determines the duration of a valid user session.

Single Sign-On Across Multiple Domains

Without single sign-on, users are often required to log on and enter their credentials multiple times as they access different applications and resources on separate servers in different cookie domains. The ability to pass single sign-on information across multiple cookie domains enables a user to authenticate at a site in one cookie domain, and then go to a site in another cookie domain without being rechallenged for credentials. For the user, this seamless navigation makes related sites easier to use.

The following illustration shows single sign-on across multiple cookie domains.



Hardware Load Balancers and Single Sign-On Across Multiple Cookie Domains

SiteMinder implements single sign-on across multiple cookie domains using a SiteMinder Web Agent configured as a cookie provider.

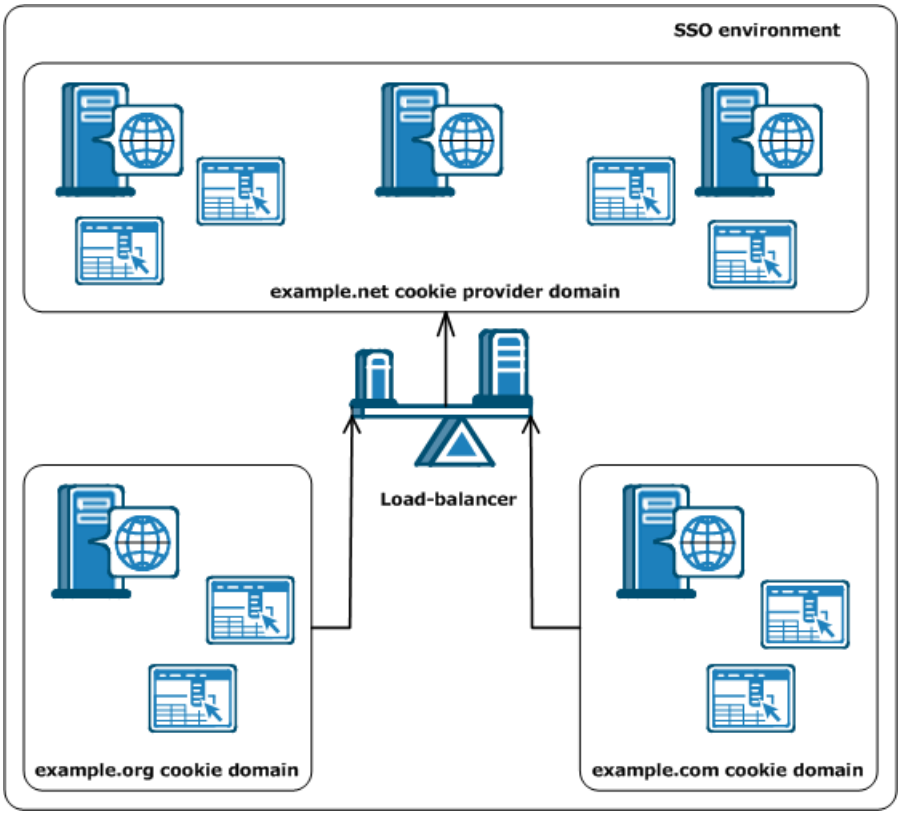
The cookie domain where the cookie provider Web Agent resides is named the cookie provider domain. All the other Web Agents from the other cookie domains within the single sign-on environment, point to one cookie provider.

SiteMinder cookie providers work using the following process:

1. A user requests a protected resource in a domain within the single-sign on environment, and is challenged for credentials.
2. When the user is authenticated, the following cookies are set in the browser of the user:
   - The local cookie for the domain where the user has authenticated.
   - The cookie provider sets the cookie.
3. The user can navigate between the domains in the single-sign on environment without being rechallenged until either of the following events occur:
   - The session of the user times out.
   - The user ends the session (usually by closing the browser).

Will the Agents in your single-sign on environment use load-balancing?

All agents in an SSO environment must refer to a single cookie provider domain. Add a load-balancer between the web servers in your cookie provider domain and the other cookie domains in your SSO environment. The following illustration shows an example:



The Web Agent in the example.org cookie domain points and the Web Agent in the example.com cookie domain both point to the same cookie provider domain of example.net. A load-balancer distributes the traffic evenly between all the web servers in the example.net cookie provider domain.

Note: You do not have to use the same user directory to implement SSO across multiple cookie domains. However, if you are using replicated user directories with nonreplicated policy stores, name the user directory *identically* for all policy stores. Also, the session ticket key, which encrypts session tickets, must be the same for all key stores in the SSO environment. The session ticket determines the duration of a valid user session.

Single Sign-On and Authentication Scheme Protection Levels

With single sign-on, authenticated users of one realm can access a resource in another realm without re-authenticating as long as the second realm is protected by an authentication scheme with an equal or lower protection level. If a user tries to access a resource protected by an authentication scheme with a higher protection level, SiteMinder prompts the user to re-enter their credentials.

SiteMinder lets administrators assign protection levels to authentication schemes with the Administrative UI. Protection levels range from 1 through 20, with 1 being the least secure and 20 being the most secure. These protection levels enable administrators to implement authentication schemes with an additional measure of security and flexibility for a single sign-on environment.

For example, a set of resources that is available to all users has a Basic authentication scheme with a protection level of 1. Another set of resources that should only be available to corporate executives, uses an X.509 certificate scheme with a protection level of 15. If a user authenticates with the Basic theme, then tries to access the resources protected by a certificate scheme, they will be required to re-authenticate.

Note: For more information, see the Policy Server documentation.

Single Sign-on and Agent Key Management

Web Agents use keys to encrypt and decrypt cookies that pass information between Web Agents. When an Agent receives a SiteMinder cookie, the key allows the Agent to decrypt the contents of the cookie. Keys must be set to the same value for all Web Agents communicating with a Policy Server.

To ensure the keys remain secure, the Policy Server can generate these keys, encrypt them, and distribute them to all the Web Agents within a SiteMinder environment. Automated key changes make agent key management easy to implement for large SiteMinder installations that share the same key store, which holds all the key information. Automating key changes also ensures the integrity of the keys.

How to Configure Single Sign-On

To set up your single sign-on environment, follow these steps:

1. Decide which cookie domains you want in your single sign-on environment.
2. Select a cookie domain within your single sign-on environment (from Step 1) to act as the cookie provider domain.
3. If your agents use central configuration, open the Agent Configuration Object (using the Administrative UI). For agents using local configuration, open the Web Agent configuration file (on each web server).
4. For your agent that is the cookie provider, modify the configuration parameters as shown in the following steps:
   a. Restrict the cookie provider functions for better security (see page 190).
   b. Prevent cookie provider replay attacks (see page 191).
   c. Verify that the value of the RequireCookies parameter is yes (see page 192).
   d. (Optional) If you want the cookies to remain valid until the configured session timeout, enable persistent cookies (see page 193). Without persistent cookies, the browser cookies are transient. Transient cookies remain valid for only *one* browser session.
   e. Verify that the value of the CookieDomain parameter specifies the local cookie domain of the system on which the agent is installed (see page 194).
   f. To validate IP addresses, set *one* of the following parameters (see page 195):
      - If you are using persistent cookies, set the PersistentIPCheck parameter.
      - If you are using transient cookies, set the TransientIPCheck parameter.
5. (Optional) Modify any of the following single sign-on parameter settings:
   - Modify the session update period (see page 96).
   - Set secure cookies across multiple domains (see page 196).
   - Ignore the cookie provider for unprotected resources (see page 197).
   - Ignore the cookie provider for POST requests (see page 197).
   - Configure Secure URLs with single sign-on (see page 198).
6. For all other agents in your SSO environment that (all agents that are not a cookie provider) set the configuration parameters as shown in the following steps:
   a. Set the value of the CookieProvider parameter to the name of your cookie provider domain. Use the fully qualified domain name of the web server hosting the agent which is acting as the cookie provider (see page 199).

      Use the syntax shown in the following example:

      http://*server.example.com:port*/siteminderagent/SmMakeCookie.ccc

Note: The cookie provider name requires the .ccc extension, as shown in the previous example.

b. Disable the cookie provider functions for better security (see page 200).

7. If you edited parameters by modifying the agent configuration file, restart the web server, so that the changes take effect.

Restrict Cookie Provider Functions

All agents have cookie-provider functions that are enabled by default. Unauthorized users with stolen SiteMinder SSO cookies could exploit cookie providers and attempt to use a session cookie from one domain to forge session cookies in another cookie domain. These forged session cookies could allow unauthorized access to protected SSO domains.

You can eliminate the potential for stolen SSO cookies to exploit cookie providers and forge session cookies with the following parameter:

LimitCookieProvider

Specifies how the SiteMinder agent acting as a cookie provider handles cookie provider SET requests (.ccc resources). When the value of this parameter is yes, the SET request is ignored unless a cookie exists in the domain of the cookie provider. The cookie provider redirects the user to the TARGET URL without setting a new cookie. When the value of this parameter is no, the SET request is processed and a new cookie is always set during the redirect back to the TARGET URL.

Default: No.

Default: (after using smpolicy-secure.xml to create your Policy Store) Yes.

Agents acting as cookie providers and the other agents operating in your SSO environment could possibly require specific configuration for optimum security.

For example, suppose that your SSO environment contains three domains. A cookie provider in example.com, and two SSO domains named example.org and example.net. The following table describes the agent configuration settings for each domain:

Example.com (Cookie Provider Domain)	Example.org (SSO Cookie Domain)	Example.net (SSO Cookie Domain)
CCCExt = .ccc	**CookieProvider** = http://server1.example.com:80/siteminderagent/SmMakeCookie.ccc	**CookieProvider** = http://server1.example.com:80/siteminderagent/SmMakeCookie.ccc
IgnoreExt = (verify that list of extensions includes .ccc)	**CCCExt** = .ccc	**CCCExt** = .ccc

EnableCookieProvider = yes	**IgnoreExt** = (verify that list of extensions includes .ccc)	**IgnoreExt** = (verify that list of extensions includes .ccc)
LimitCookieProvider = yes	**EnableCookieProvider** = no	**EnableCookieProvider** = no
TracksSessionDomain = yes	**TracksSessionDomain** = yes	**TracksSessionDomain** = yes
TrackCPSessionDomain = yes		

Prevent Cookie Provider Replay Attacks

You can prevent the cookie provider from being vulnerable to replay attacks with the following parameter:

TrackCPSessionDomain

Validates that the cookie domain of the session cookie matches the cookie domain of the cookie provider. Different cookie domains could indicate a possible replay attack.

Default: No (The domain of cookie provider is not validated).

To prevent cookie provider replay attacks, set the value of the TrackCPSessionDomain parameter to yes.

The agent compares cookie domains and rejects requests when the domains do not match.

Set RequireCookies Parameter for Single Sign-On

You can control whether SiteMinder requires cookies with the following parameter:

RequireCookies

Specifies whether SiteMinder requires cookies. SiteMinder requires cookies for the following functions:

- Securing single sign-on environments.
- Enforcing session timeouts.
- Enforcing idle timeouts.

When the value of this parameter is yes, the agent requires one of the following cookies to process HTTP requests:

- SMCHALLENGE
- SMSESSION

When the value of this parameter is no, the following conditions could occur:

- Users are challenged for credentials unexpectedly.
- Timeouts are not strictly enforced.

Important! If the agent requires cookies, instruct your users to accept HTTP cookies in their browsers. Otherwise, the users are denied access to all protected resources.

Default: Yes

To require cookies, set the value of the RequireCookies parameter to yes.

Enabling Persistent Cookies for Single Sign-On

If you want to use single sign-on for multiple browser sessions, use persistent cookies. The following steps describe one possible use for persistent cookies:

1. Users authenticate with SiteMinder, but end their browser sessions before the SiteMinder session expires.
2. Users start new browser sessions later, but the persistent cookie maintains their single-sign on capability.

Persistent cookies remain valid for the configured maximum session time-out *plus* seven days. Many browsers delete the cookie file of the web browser after the cookie expires. Some browsers possibly handle persistent cookies differently.

Follow these steps:

1. Set the PersistentCookies parameter to yes.

   The SMSESSION cookies are persistent.
2. Set the TransientIDCookies parameter to no.

   The SMIDENTITY cookies are persistent.

Specify the Cookie Domain

The CookieDomain parameter defines the cookie domain of the server where you installed the agent. You can modify the domain by setting the following parameter:

CookieDomain

Defines the cookie domain of the agent. Use a fully qualified domain name with at least two periods. For example, the setting .example.com cookie domain matches the following servers:

- w1.example.com
- w2.example.com
- w3.sales.example.com

All web servers in this domain can exchange cookies with a browser. Servers in the same cookie domain use cookies to verify the credentials of a user.

When the parameter value is none, the agent generates cookies only for its own server. For example, myserver.example.com.

If the value is blank (or contains "" in a local configuration file), the agent uses the domain information in the HTTP_HOST header. The agent then bases the value using the setting in the CookieDomainScope parameter.

Default: Empty

Example: .example.com

Limits: This value is case-sensitive. This value requires a fully qualified domain name with at least two periods, as shown in the previous example.

Note: This value is case-sensitive.

Follow these steps:

1. Set the value of the CookieDomain parameter.
2. (Optional) Set the value of the CookieDomainScope parameter.

   CookieDomainScope

   Specifies the number of sections (characters with periods between them) in the domain name.

   When the value is set to 0, the default, the agent chooses the most specific cookie domain for the host without making a server-only cookie. This means that the cookie domain myserver.example.com yields a domain of example.com, and myserver.metals.example.org yields a domain of .metals.example.org.

   If the CookieDomainScope parameter is set to 2, the cookie domain would be .example.com and .example.org respectively.

   Default: 0

   Example: Suppose that your cookie domain is division.example.com. To set the scope of the cookie domain for server.division.example.com, set the value of the CookieDomainScope parameter to 3.

Enable IP Address Validation for Single Sign-On Environments

An unauthorized system can monitor packets, steal a cookie, and use that cookie to gain access to another system. To prevent a breach of security by an unauthorized system, you can enable or disable IP checking with persistent and transient cookies.

The IP checking feature requires agent to compare the IP address stored in a cookie from the last request against the IP address contained in the current request. If the IP addresses do not match, the agent rejects the request.

The two parameters that are used to implement IP checking are PersistentIPCheck and TransientIPCheck. Set them as follows:

- If you enabled PersistentCookies, set PersistentIPCheck to yes.
- If you did not enable PersistentCookies, set TransientIPCheck to yes.

SiteMinder identity cookies are unaffected by IP checking.

Modify the Session Update Period

You can specify how often the Web Agent redirects a request to the Cookie Provider to set a new cookie with the following parameter:

SessionUpdatePeriod

Specifies how often (in seconds) a Web Agent redirects a request to the Cookie Provider to set a new cookie. Refreshing the master cookie decreases the possibility that it will expire due to an idle time-out of the SiteMinder session.

Default: 60

To modify the session update period

1. Make sure the CookieProvider parameter is defined.
2. Change the number of seconds in the SessionUpdatePeriod parameter to reflect the interval you want.

   The session update period is changed.

Set Secure Cookies Across Multiple Domains

Setting the UseSecureCookies parameter configures a Web Agent to only return a *local* cookie to a requesting browser session if the connection between them is secure (HTTPS); if the Web Agent is also configured as a cookie provider, UseSecureCookies does not apply to redirected requests for access to resources in other cookie domains.

To configure a Web Agent acting as a cookie provider to only return cookies to a Web Agent in another cookie domain if that Web Agent is also configured to use secure cookies, you must enable UseSecureCookies and also configure the following parameter:

UseSecureCPCookies

If UseSecureCPCookies is set to Yes, the cookie provider will only send a cookie to a Web Agent in another cookie domain that is also configured to use secure cookies (that is, UseSecureCookies is enabled).

When this setting and UseSecureCookies are both enabled, users in a multiple domain single sign-on environment who move from an SSL web server to a non-SSL web server in another cookie domain will have to reauthenticate. Secure cookies cannot be passed over traditional HTTP connections.

Default: No

To send cookies over SSL connections across multiple domains, set the UseSecureCookies and UseSecureCPCookies to yes on the cookie provider.

More information:

Set Secure Cookies (see page 77)

Ignore the Cookie Provider for Unprotected Resources

Agents forward all requests to the cookie provider by default. If you have unprotected resources, you can reduce network traffic with the following parameter:

IgnoreCPForNotprotected

Prevents the cookie provider from being queried for unprotected resource requests. When this parameter is set to no, all requests are directed to the cookie provider by the Web Agent. For traditional (nonframework) Agents, configure a cookie provider so that value of this parameter appears in the Web Agent log file.

Default: No

To prevent an agent from contacting the cookie provider when unprotected resources are requested, set the value of the IgnoreCPForNotprotected parameter to yes.

Ignore the Cookie Provider for POST Requests

The following parameter enables these behaviors:

- This setting allows traditional agents to act as cookie providers.
- This setting prevents POST requests from being sent to cookie providers (in all environments).

LegacyCookieProvider

Controls whether an agent sends a POST request to a cookie provider. When the agents send POST requests to a traditional agent (operating as a cookie provider), the redirected request becomes a GET. This conversion causes errors. When set to no, the agent sends the POST request to the cookie provider. When set to yes, the agent does *not* send the POST request to the cookie provider.

If you are using central agent configuration, add this parameter to your agent configuration object. This parameter exists in local configuration files.

Default: No (POST requests sent)

Set the value of the LegacyCookieProvider parameter to yes to enable the following behaviors:

- Use traditional agents as cookie providers.
- Prevent any POST requests from being sent to cookie providers.

Configure SecureUrls with Single Sign-on

If your single sign-on network has a Web Agent that supports SecureUrls functionality and another Agent that does not, this could result in internal server error messages when a user requests a protected single sign-on resource.

The log for the Web Agent with SecureUrls support shows the reason for the server error, such as the following:

Error. Unable to process request, SecureUrls is disabled.

Note: All Web Agents in a single sign-on environment must have the SecureUrls parameter set to the same value. SiteMinder does not support interoperability between Web Agents with the SecureUrls parameter set to different values.

Specify the Cookie Provider

To have the other agents in your SSO environment use a cookie provider, specify the location of the agent that is acting as a cookie provider with the following parameter:

CookieProvider

Specifies the URL of the web server where the agent that is acting as the cookie provider resides.

A cookie provider is an agent in a single sign-on environment. The cookie provider sets a browser cookie for the local domain in which it exists. After this cookie is set, users can navigate throughout the single sign-on environment without reauthenticating.

The cookie provider name requires a .ccc extension, as shown in the following examples:

- For IIS, Oracle iPlanet, and Domino web servers, the URL syntax is as follows:

  http://*server.domain:port*/siteminderagent/SmMakeCookie.ccc

- For Apache and Apache-based web servers, the URL syntax is as follows:

  http://*server.domain:port*/SmMakeCookie.ccc

This parameter also affects the following parameters:

- CCCExt
- SessionUpdatePeriod

Default: No default

Example: (IIS, Oracle iPlanet, and Domino web servers)
http://server1.example.com:80/siteminderagent/SmMakeCookie.ccc

Example: (Apache and Apache-based web servers)
http://server1.example.com:80/SmMakeCookie.ccc

Limits: This parameter requires a fully qualified domain name.

Follow these steps:

1. Set the CookieProvider parameter to the URL of the web server that is acting as a cookie provider.
2. Verify that the value of the CCCExt parameter is set to .ccc
3. Add the .ccc extension to the values in the IgnoreExt parameter.
4. (Optional) Modify the session update period.
5. Repeat Steps 1 through 4 on all web agents in your SSO environment that are *not* cookie providers.

   The cookie provider is specified.

Disable Cookie Providers

All SiteMinder agents can act as cookie providers by default. This setting makes configuring an SSO environment easier. For increased security, you can disable the built-in cookie provider functionality with the following parameter:

EnableCookieProvider

Specifies how the agent handles requests from cookie providers (.ccc). When this parameter value is yes, the agent processes the requests. When this parameter value is no, the agent ignores the requests from the cookie provider. The agent denies access to the requested resource. To increase security, set this parameter value to no.

Default: Yes.

Default: (after using smpolicy-secure.xml to create your Policy Store) No.

To prevent an agent from processing requests from cookie providers, set the value of the EnableCookieProvider parameter to no.

If you do not use cookie providers for SSO in your environment, use the agent configuration settings shown in the following table for all of your agents:

Set the configuration parameters of all agents to the following values:
EnableCookieProvider = no

Chapter 14: Comprehensive Log Out

This section contains the following topics:

How Full Logoff Works (see page 201)
Configure Full Logoff (see page 202)
How to Configure Full Logoff for Single Sign-on (see page 203)
Configure Comprehensive Log Out using FCC Forms (see page 205)

How Full Logoff Works

Full logoff support enables a Web developer to make sure that a user is completely logged off from a user session. This protects resources because it gives users a way to end a session without exiting the Web browser and prevents an unauthorized person from assuming control of an open session.

A full logoff uses the following process:

1. A user clicks a button to log off.
2. The Web Agent redirects the user to a customized logoff page that you created.
3. The Web Agent removes the session and authentication cookies from a user's browser.
4. The Web Agent also removes the session cookie from the local cookie domain and the cookie provider domain, which you specify for single sign-on environments.
5. The Web Agent calls the Policy Server and instructs the Policy Server to remove any session information.

   The user is completely logged off.

More information:

Configure Full Logoff Support for Domino Agents (see page 286)

Configure Full Logoff

The full log-out feature uses a custom log-out page that you create with the following parameter:

LogOffUri

Enables the full log-out function by specifying the URI of a custom web page. This custom web page appears to users after they are successfully logged off. Configure this page so that it cannot be stored in a browser cache. Otherwise, a browser could possibly display a log-out page from its cache without logging the user off. If this situation happens, unauthorized users could possibly have an opportunity to assume control of a session.

Note: When the CookiePath parameter is set, the value of the LogOffUri parameter must point to the same cookie path. For example, if the value of your CookiePath parameter is set to example.com, then your LogOffUri must point to example.com/logoff.html

Default: (all agents *except* the CA SiteMinder Agent for SharePoint r12.0.3.0) No default

Limits: Multiple URI values permitted. Do *not* use a fully qualified URL.Use a *relative* URI.

Example:(all agents *except* the CA SiteMinder Agent for SharePoint r12.0.3.0) /Web pages/logoff.html

Follow these steps:

1. Create a custom HTTP application that logs the user off. For example, add an Exit or Sign Off button that redirects the user to a URL you specify.
2. Set up the log-out page so it cannot be cached in web browsers. This setting increases security because the page is always served from the web server, and not the cache of the browser. For example, for HTML pages, you can add the following meta tags to the page:

   ```
   <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
   ```

   ```
   <META HTTP-EQUIV="Expires" CONTENT="-1">
   ```

   Important! Some web browsers do *not* support meta tags. Use a cache-control HTTP header instead.
3. Configure the LogOffUri parameter with the following steps:
   a. Delete the pound sign (#), if necessary.
   b. Enter the URI of the custom HTTP file that will log the user off. Do *not* use a fully qualified URL.

      The full log-out feature is configured.

More information:

Specify the Cookie Path for Agent Cookies (see page 79)

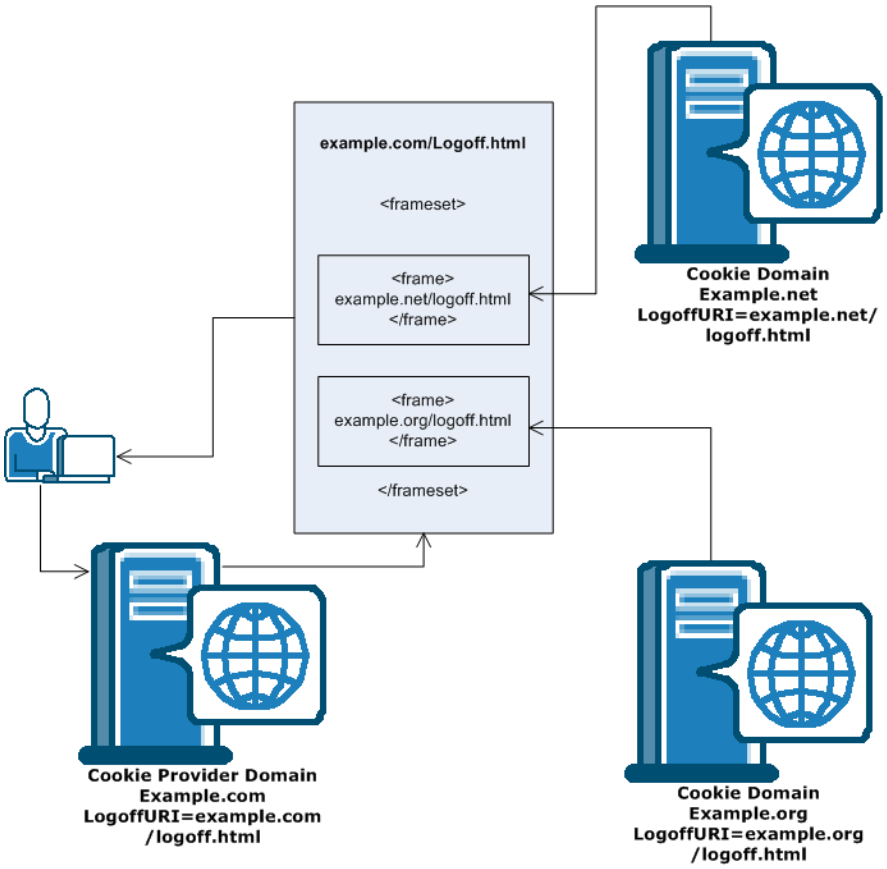
How to Configure Full Logoff for Single Sign-on

In a single sign-on environment, the session cookies are removed only from the local cookie domain and the cookie provider domain associated with the Web Agent. For single sign-on across multiple cookie domains, the full log-off feature of SiteMinder does *not automatically* log a user off across all the cookie domains that the user has visited.

To configure log-offs across multiple cookie domains, use the following process:

1. Create one centralized log-off page that contains separate frames (or iframes) for the other cookie domains in your SSO environment. These frames can be a small size, such as 1x1 pixels.
2. For each frame of the centralized log-off page in Step one, add a hyperlink to the Logoff Uri of the associated cookie domain. For example, if you have two other cookie domains, example.org and example.net, you would do the following steps:
   - Add a hyperlink to the Logoff Uri of example.org to one frame.
   - Add a hyperlink to the Logoff Uri of example.net to the other frame.
3. Configure the LogoffUri of the cookie provider domain to point to the centralized log-off page. When the web server loads this log off page, the frames in the centralized log-off page call the logoff pages from the other cookie domains. The user is logged off from all the cookie domains at once.

The following illustration shows an example of using a centralized log-off page:



Note: You can also place the hyperlinks inside <iframe> tags instead of <frame> tags.

Configure Comprehensive Log Out using FCC Forms

If you use FCC forms to authenticate your users, you can configure a comprehensive log out with your FCC form. This method provides an alternative to the LogoffUri parameter.

Follow these steps:

1. Open the .fcc file that you are using to authenticate your users with a text editor. FCC files are located in the following directory:

   web_agent_home/samples/forms

   web_agent_home

   Indicates the directory where the SiteMinder Agent is installed.

   Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

   Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

   Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

   Default (UNIX/Linux installations): /opt/ca/webagent

2. Add the following text to the top of your FCC page (before the <_html> tag):

   ```
   @smlogout=true
   @target=http://server_name.example.com/directory/your_logout_page.html
   ```

   Note: *your_logout_page* indicates a custom html page you create to inform users that they have logged out.

   Comprehensive logout using FCC forms is configured.

Chapter 15: SSO Security Zones

This section contains the following topics:

Security Zones Overview (see page 207)
Configure Security Zones (see page 215)

Security Zones Overview

Users have the ability to define single sign-on security zones within the same cookie domain, representing a single zone, or across multiple cookie domains, representing different zones. As a result, users have single sign-on within the same zone, but may be re-challenged when entering a different zone, depending on the trust relationship defined between the zones. Zones included in a trusted relationship will not re-challenge a user that has a valid session in any zone in the group.

Single sign-on security zones are implemented entirely by SiteMinder Web Agents. Each zone must reside on a separate Web Agent instance. Multiple zones cannot be created on the same Agent instance.

A security zone is identified by cookies generated by the Web Agent. By default, the Web Agent generates two cookies, a session cookie named SMSESSION and an identity cookie named SMIDENTITY. When you configure security zones, the Web Agent can generate session cookies and identity cookies with unique names so that the zone affiliation is reflected in the cookie names.

Security Zone Definitions

The following terms apply to single sign-on (SSO) security zones:

CAC (Centralized Agent Configuration)

Identifies a mechanism by which a Web Agent borrows its configuration properties from a Web Agent configuration object defined in the policy store.

Cookie Provider

Identifies a mechanism by which single sign-on is implemented in Web Agents across multiple domains. One of the domains is designated as the master domain, and the Web Agents from the other domains are re-directed to a Web Agent in the master domain to provide them with the cookies in that domain.

SSO (Single Sign-On)

Identifies a mechanism by which a user authenticated once will not be rechallenged for credentials.

SSO Zone

Identifies a sub-set of SSO, defined by an arbitrary identifier (zone name) used to segment application SSO within a single cookie domain. All applications in the same SSO zone allow SSO amongst themselves. SSO to and from other SSO zones may or may not be allowed as defined by zone trust relationships.

Trusted SSO Zone

Identifies a foreign zone trusted by a local zone for SSO.

Security Zones Benefits

The SSO Security Zones feature is intended for use in situations where SiteMinder administrators wish to segment their single sign-on environments within the same cookie domain. For example, consider the CA.COM domain. Under standard SiteMinder SSO functionality, all SiteMinder protected applications in CA.COM would use the cookie SMSESSION to manage single sign-on. Consider the following scenario in which SSO Security Zones do not exist:

1. A user accesses an application (APP1). The user is challenged by SiteMinder, logs in to SiteMinder, and creates an SMSESSION cookie.
2. The user accesses a second application (APP2) and is once again challenged by SiteMinder. (Rules prevent SSO from occurring because the user does not have access to APP2 using the APP1 user credentials.) The user logs in and creates a new SMSESSION cookie overwriting the old one with the new logged in session for APP2.
3. The user now returns to APP1 and is challenged yet again, since the user lost the original APP1 session and the APP2 session might not be accepted for APP1. Therefore, SSO does not occur between APP1 and APP2, causing a very frustrating situation.

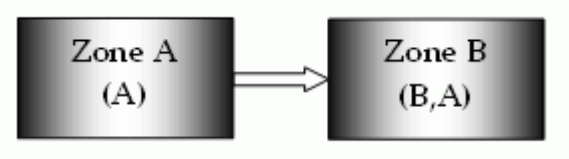
With SSO Security Zones, APP1 can be placed in zone Z1 and APP2 can be placed in zone Z2. Now logging into APP1 creates a Z1SESSION cookie and access to APP2 results in a Z2SESSION cookie. With different names, the cookies no longer overwrite each other so there is only one login per application now, not one for each time the user moves between the applications as in the example above.

Prior to the SSO Security Zones feature, the only way to perform the same grouping of SSO for applications was to create different network domains and therefore different cookie domains (CA1.COM, CA2.COM, and so on), and use various multi-cookie domain configurations with cookie providers. This is not desirable in most enterprises, since using multiple network domains has certain IT maintenance and support consequences.

Security Zone Basic Use Case

Single sign-on can, on a controlled basis, be broken into several security zones that have configurable trust relationships. For example, consider Zone A and Zone B:

- Zone A has only one trusted zone, its own Zone A.
- Zone B has two trusted zones, its own Zone B as well as Zone A.



The trust relationship in the above illustration is indicated by the arrow, meaning that the user sessions established in Zone A can be used for single sign-on in Zone B.

In this example, Zone A might be an administrator-only zone, while Zone B might be a common access zone. An administrator authenticated in Zone A gains access to Zone B without being rechallenged. However, a user authenticated in Zone B is re-challenged when trying to access Zone A.

User sessions in different zones are independent of each other. Suppose a user authenticates in Zone B first, and then authenticates again in Zone B. Two different sessions are created. In fact, the user may have different identities in both sessions. When the user returns to Zone A, the session established in that zone is used.

Consider what would happen if a user is validated using single sign-on in a zone where that user does not yet have a session. If the user authenticates in Zone A and then visits Zone B for the first time, then a user session is created in Zone B, based on the session information in Zone A, possibly updated by the Policy Server. Note that the user session in Zone A is not updated until the user returns to Zone A.

User Sessions Across Security Zones

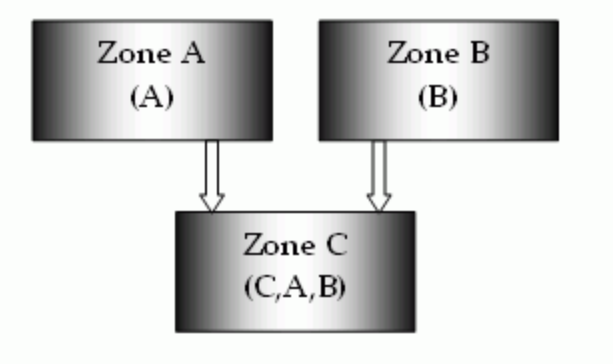
Single sign-on security zones do not need to all belong to the same domain. In fact, zones can be spanned over multiple domains. However, Web Agents only search for trusted zone cookies in their local cookie domain. If no suitable cookies are found, Web Agents continue to redirect to cookie providers only for their own zone.

Trusted Zone Order

A single sign-on security zone is defined by a pair of parameters:

- the security zone name
- an ordered list of trusted zones

The order in which the trusted zones are listed is important. Consider the following example.



In this illustration, Zone C trusts both Zone A and Zone B. Neither Zone A nor Zone B trusts any other zone, but all zones trust themselves.

When a user makes a request in Zone C, the Web Agent looks for a session or identity cookie in the trusted zones, in the order in which the zones are listed. In this example, Zone C has a list of trusted zones that include C, A, and B.

The following is an order of events that might occur:

1. The Web Agent first checks to see if the user has a session in Zone C.
2. If no session is found, the Web Agent checks to see if the user has a session in Zone A.
3. If no session is found, the Web Agent checks to see if the user has a session in Zone B.
4. The session specification from each cookie that is found is used to process authentication requests until a successful login occurs.
5. After a successful authentication, the Web Agent proceeds to authorization.
6. If no cookies are found or no cookies pass authentication, the agent challenges the user for credentials as usual.

Note that the user experience may depend on the order in which the zones are accessed. In this example, if the user accesses Zone B first followed by Zone C, the user's identity in Zone B is also used in Zone C. If the user accesses Zone A first followed by Zone B and Zone C, the user's identity in Zone A is used, despite the fact that the user was re-challenged in Zone B before going to Zone C.

This will also be the case when sessions with different max and idle session timeouts begin to expire. In the current example, a user with valid cookies in Zone A and Zone C will first get access with the Zone C cookie. If the Zone C cookie expires, the Zone A cookie will be used if it has not expired. Therefore, the user's identity could change from a Zone C identity to a Zone A identity without a credential challenge occurring.

Two or more Web Agents may have different lists of trusted zones but still use a common trusted zone name. In this case, the agents implicitly trust each other but will not trust the same foreign zones. This functionality enables applications to be segmented for single sign-on. A Web Agent supports only a single sign-on zone name. All session, identity, and state cookies generated by that agent use the same single sign-on zone name. Therefore, if two applications do not share the same single sign-on trust requirements, they must be hosted on separate web servers each with its own Web Agent and list of trusted zones.

Note: Foreign zones refer to zones other than the one supported by a given Web Agent. For example, if an agent is configured with SSOZoneName="Z1", then any other zone would be foreign to it. This includes the default zone "SM".

The Default Single Sign-On Zone and Trusted Zone List

Web Agents that do not specify a security zone name (such as all pre-SiteMinder 6.x QMR 5 Web Agents) are considered to belong to the default zone. For backwards compatibility, the default zone is implicitly assumed to have a zone name of SM. This allows SiteMinder r12.5 Web Agents to support SMSESSION and SMIDENTITY by default with no configuration changes.

Web Agents that do not specify a list of trusted zones trust only their own single sign-on zone (either a specified zone name or default zone if no zone name has been specified).

A Web Agent can be configured to trust other zones in addition to the default zone. It can also use a specified zone name and list no other trusted zone. Agents always trust their own zone first, regardless of whether or not additional trusted zones are specified. In order for a Web Agent using a non-default zone to trust the default zone as well, it must list "SM" in its trusted zone list.

Request Processing with Multiple User Sessions

Web Agents look up user sessions in the order of trusted zones. If a valid user session is found, the Web Agent uses the session information to process the user request. If no valid session is found, the Web Agent fails over to the next valid user session in the order of trust.

Responses from failed validations are ignored if there is another session to check. Otherwise, they are processed as normal. This means that if the Web Agent finds three trusted sessions to process and the first two fail to validate, only the responses from validating the third and final session are processed.

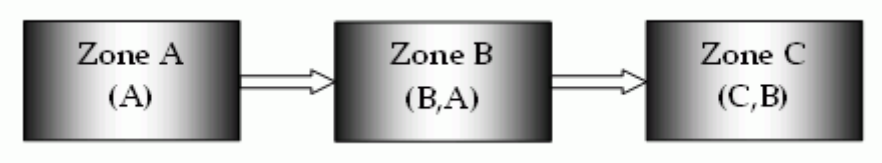
In the case of a successful validation, the Web Agent stops processing sessions and immediately begins processing the responses from the successful validation. If the agent has three sessions to validate and the first session validates successfully, the remaining two are ignored and the agent moves on to process the responses for the first successful validation.

More Information

Trusted Zone Order (see page 210)

Transitive Relationships Across Zones

The trust relationship between single sign-on zones is not fully transitive. If Zone A is trusted by Zone B, and Zone B is trusted by Zone C, Zone A may not necessarily be trusted by Zone C, as illustrated in the following diagram.



Other Cookies Affected by Single Sign-On Zones

SiteMinder uses state cookies to manage the various events surrounding authentication and authorization. All of these cookies by default begin with the default single sign-on security zone prefix SM. If a new single sign-on zone name is specified, then these cookies are also named to reflect the specified non-default zone name. Below is a list of cookies that are affected by defining a new single sign-on zone:

- SMCHALLENGE
- SMDATA
- SMIDENTITY
- SMONDENIEDREDIR
- SMSESSION
- SMTRYNO

If a zone name of Z1 is specified, for example, the Web Agent begins creating Z1CHALLENGE=YES cookies for Basic authentication. This allows administrators to create islands of SiteMinder application single sign-on in a single cookie domain (for example, ca.com) where agents will not interfere with each other. The single sign-on trusted zone list then allows single sign-on to occur between these isolated single sign-on zones in a predictable fashion.

Single Sign-On Zones and Authorization

With single sign-on zones, authorization proceeds normally after a successful authentication without change. Once the validation process identifies a valid session, the session is used for the remainder of request processing and any other sessions identified in the request are ignored. If authorization fails, the user is challenged regardless of whether or not other sessions are available that might authorize successfully.

The first trusted session that passes validation is the session passed to authorization. If this session fails authorization, the user is challenged for credentials.

Configure Security Zones

Two single sign-on parameters have been added to the Web Agent configuration objects in the policy store. These settings may also be used in local configuration files and are added to the sample local configuration files laid down during installation.

Note: All Web Agents configured through the same agent configuration object belong to the same single sign-on zone.

SSOZoneName

Specifies the (case-sensitive) name of the single sign-on security zone a Web Agent supports. The value of this parameter is prepended to the name of the cookie a Web Agent creates. This setting helps you associate cookies with their respective cookie domains. When this parameter is not empty, SiteMinder generates cookies using the following convention:

ZonenameCookiename.

Default: Empty (uses SM as a zone name, which gives the following default names to the cookies):

- SMSESSION
- SMIDENTITY
- SMDATA
- SMTRYNO
- SMCHALLENGE
- SMONDENIEDREDIR

Limits: Single-valued. This parameter supports English-language characters only.

Example: Setting the value to Z1 creates the following cookies:

- Z1SESSION
- Z1IDENTITY
- Z1DATA
- Z1TRYNO
- Z1CHALLENGE
- Z1ONDENIEDREDIR

SSOTrustedZone

Defines an ordered (case-sensitive) list of trusted SSOZoneNames of trust for a single sign-on security zone. Use SM to add the default zone if necessary. Agents always trust their own SSOZoneName above all other trusted single sign-on zones.

Default: Empty (SM or the SSOZoneName if provided)

Limits: Multi-valued

Specify the Single Sign-on Zone for the Agent

SSOZoneName

Specifies the (case-sensitive) name of the single sign-on security zone a Web Agent supports. The value of this parameter is prepended to the name of the cookie a Web Agent creates. This setting helps you associate cookies with their respective cookie domains. When this parameter is not empty, SiteMinder generates cookies using the following convention:

ZonenameCookiename.

Default: Empty (uses SM as a zone name, which gives the following default names to the cookies):

- SMSESSION
- SMIDENTITY
- SMDATA
- SMTRYNO
- SMCHALLENGE
- SMONDENIEDREDIR

Limits: Single-valued. This parameter supports English-language characters only.

Example: Setting the value to Z1 creates the following cookies:

- Z1SESSION
- Z1IDENTITY
- Z1DATA
- Z1TRYNO
- Z1CHALLENGE
- Z1ONDENIEDREDIR

Use the SSOZoneName parameter to enter the name of the single sign-on zone a Web Agent is to support. This parameter is case sensitive. If not specified, it defaults to SM. If the value of the SSOZoneName parameter is non-empty, the Web Agent generates cookies with the naming convention:

zone_namecookie_name

where *zone_name* is the parameter value and *cookie_name* is the general name of the cookie being created.

Cookies affected by this convention include:

- SESSION
- IDENTITY

- DATA
- TRYNO
- CHALLENGE
- ONDENIEDREDIR

If the user is validated in a single sign-on zone in which that user has not yet established a session, the session specification returned by the Policy Server is used to create a new session cookie for that zone.

When a new cookie is created, its zone parameter is set to the zone name, in order to prevent the user from swapping cookies from different zones by simply renaming them. The cookie validation engine verifies if the zone name matches the prefix used in the cookie's name. This applies only to SESSION and IDENTITY cookies.

To specify the name of the single sign on zone you want the Web Agent to support, add the name of the zone to the SSOZoneName parameter.

The Order of Trust and Failover

Use the SSOTrustedZone parameter to specify the single sign-on zone's order of trust. When processing a request, the Web Agent looks for a SESSION or IDENTITY cookie for each zone in the order they appear in the list.

Any cookies found are validated as usual (decrypted, and tested for a valid host name, single sign-on zone name, and timeouts), then stored in an ordered list of trusted sessions if valid. Prior to authentication, the user's active session (and therefore user identity) are considered the first session in the ordered list of valid sessions.

During authentication, the Web Agent will call validate using the first session in the list. If the validation succeeds, the agent moves on and establishes user identity and affirms the active accordingly. If validation fails, the next session is used in a new validation call, and so forth until validation succeeds or the agent runs out of sessions. If no session validates, the agent challenges the user as usual.

Once validated, the agent ignores all other sessions and instead sticks only to the session that validated for the remainder of request processing. This means that should authorization fail, the user is challenged immediately. Any other existing sessions in the request are not used.

Chapter 16: Advanced Configuration Settings

Agents and Proxy Servers

Use any of the following settings to manage your SiteMinder agents running on proxy servers:

- Configure agents that sit behind proxy servers (see page 220).
- Customize cache-control and expireforproxy header settings (see page 222).
- Proxy header usage notes (see page 224).
- Security considerations (see page 225).

Configure Agents that Sit behind Proxy Servers

If a Web Agent will be installed behind a proxy server, you can configure the Web Agent to work with proxy servers using the following parameters:

ProxyTrust

Instructs the agent on a destination server to trust authorizations received from a SiteMinder agent on a proxy server. A destination server is a server that is behind a reverse proxy server. Setting this value to yes increases efficiency because only the agent on the proxy server contacts the Policy Server for authorization. The agent operating on the destination server does *not* contact the Policy Server again reauthorize users.

Default: No

ExpireForProxy

Prevents a client from caching content (pages and potentially headers or cookies). When the value of this parameter is set to yes , the Web Agent inserts one of the following HTTP headers into the HTTP response:

- Expires
- Cache-control

If content is not cached, subsequent requests continue to be forwarded.

When the ExpireForProxy parameter is set to yes, the Web Agent inserts the strings specified in the appropriate ProxyHeaders*suffix_name* parameter into the HTTP response based upon what type of request the Agent performed.

For HTTP/1.1 requests, the Agent inserts the values of the following parameters as headers in the response:

- ProxyHeadersAutoAuth
- ProxyHeadersProtected
- ProxyHeadersUnprotected

For HTTP/1.0 requests, the Agent inserts the values of the following parameters as headers in the response:

- ProxyHeadersAutoAuth10
- ProxyHeadersProtected10
- ProxyHeadersUnprotected10

Default: No

Note: Although this parameter name contains the word 'proxy,' the settings of this parameter also affect the behavior of web browsers, or any other client that connects to a web server on which any SiteMinder Agents using this parameter setting operate.

To tell the proxy not to cache the pages, the Web Agent adds an Expires header for the page. This header is set to a date in the past, which prevents the page from being cached by a proxy, as dictated by the HTTP 1.0 specification. On 302 redirects, a cache-control: no-cache header is set instead. Although this prevents caching of content, this has the negative consequence of affecting the browsing experience for an Internet Explorer (IE) browser, as described by Microsoft Support.

With the use of cache-control: no-cache for 302 redirects, the ActiveX component that manages in-place document viewing in IE relies on the browser's cache to locate the file. Because this header instructs the browser not to cache the file, the ActiveX component cannot locate the file and fails to display the request properly. Further, when you set the Web Agent's ExpireForProxy setting to yes, the back-end server tells the proxy not to cache the resource.

To configure Agents that sit behind proxy servers

1. Set the ProxyTust parameter to yes.
2. Set the ExpireForProxy parameter to yes.
3. (Optional) Customize values the cache-control and ExpireForProxy (HTTP) headers.

   The Agents behind the proxy servers are configured.

More information:

Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)

Customize the Cache-Control and ExpireForProxy Header Settings

You can customize the cache-control and ExpireForProxy headers to secure Web resources without affecting in-place activation of application files (.doc, .pdf, and so on). You can set specific HTTP headers for the following types of content independently to control how that content is cached by a web browser or proxy server:

- Auto-Authorized
- Unprotected
- Protected

Important! We recommend using the default settings unless you are familiar with the ramifications of changing these settings in accordance with RFC 2068. If you plan to change the default settings, note that the SiteMinder session cookie is updated on access of an unprotected page once a user has a session in order to track idle timeout. Therefore, unprotected pages should not be cached on a proxy that caches HTTP headers.

The following characteristics apply to setting headers to prevent caching by proxies:

- All redirects set a Cache-Control: no-cache header, regardless of agent activity.
- The web server sends the appropriate headers back to the proxy/client based on the HTTP protocol used (1.0 or 1.1 and higher).

All parameters should be configured using multi-value strings to suit the use of multiple headers, such as cache-control: private and cache-control: max-age=60.

The following is the new configuration:

1. ProxyHeadersDefaultTime - defaults to 60 seconds
2. ProxyHeadersTimeoutPercentage – defaults to 10 percent
3. The following cache-control headers are available:

   ProxyHeadersAutoAuth

   Specifies the value of an HTTP 1.1 header that the Web Agent inserts into an HTTP response to a client when the ExpireForProxy parameter in the Web Agent Configuration is set to yes. The value of this header determines if *or* for how long the auto-authorized resource is cached.

   Default: Expires: Thu, 01 Dec 1994 16:00:00 GMT

   Example (suggested setting): "Cache-control: max-age=60"

ProxyHeadersAutoAuth10

Specifies the value of an HTTP 1.0 header that the Web Agent inserts into an HTTP response to a client when the ExpireForProxy parameter in the Web Agent Configuration is set to yes. The value of this header determines if or for how long the auto-authorized resource is cached.

Default: Expires: Thu, 01 Dec 1994 16:00:00 GMT

Example (suggested setting): "Expires: Thu, 01 Dec 1994 16:00:00 GMT"

ProxyHeadersProtected

Specifies the value of an HTTP 1.1 header that the Web Agent inserts into an HTTP response to a client when the ExpireForProxy parameter in the Web Agent Configuration is set to yes. The value of this header determines if or for how long the protected resource is cached.

Default: Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

Example (suggested settings): "Cache-Control: private"

ProxyHeadersProtected="Cache-Control: max-age=60"

ProxyHeadersProtected10

Specifies the value of an HTTP 1.0 header that the Web Agent inserts into an HTTP response to a client when the ExpireForProxy parameter in the Web Agent Configuration is set to yes. The value of this header determines if or for how long the protected resource is cached.

Default: Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

Example (suggested settings): "Expires: Thu, 01 Dec 1994 16:00:00 GMT"

ProxyHeadersUnprotected

Specifies the value of an HTTP 1.1 header that the Web Agent inserts into an HTTP response to a client when the ExpireForProxy parameter in the Web Agent Configuration is set to yes. The value of this header determines if or for how long the unprotected resource is cached.

Default: Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

Example (suggested setting): ProxyHeadersUnprotected="Cache-Control: private"

ProxyHeadersUnprotected="Cache-Control: max-age=60"

ProxyHeadersUnprotected10

Specifies the value of an HTTP 1.0 header that the Web Agent inserts into an HTTP response to a client when the ExpireForProxy parameter in the Web Agent Configuration is set to yes. The value of this header determines if or for how long the unprotected resource is cached.

Default: Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

Example (suggested setting): "Expires: Thu, 01 Dec 1994 16:00:00 GMT"

When configuring multiple headers, (for example, the cache-control headers in the suggested setting for unprotected HTTP/1.1 content), note the following:

- You *must* have multiple occurrences of the configuration parameter and you cannot separate these with a comma (,) or the plus-sign (+).
- As the values for these configuration parameters are HTTP response headers, they must comply with RFC 2616 (for HTTP/1.1), RFC 1945 (for HTTP/1.0) and RFC 822. Both HTTP/1.1 and HTTP/1.0 specify the format for an HTTP Header as that of an RFC 822 message, namely "Name: Value" (Name, followed by a colon, white space and then a value).

If you do *not* configure the Web Agent to set the appropriate cache expiration headers when a user accesses unprotected resources, then by default, the Web Agent will not set these headers, thereby allowing a web browser or proxy server to cache an SMSESSION cookie. This cached cookie can be re-used by the web browser or proxy-server after the user has initiated a different session (and therefore a different user context), causing an unauthorized impersonation.

More information:

Configure Agents that Sit behind Proxy Servers (see page 220)

Proxy Header Usage Notes

- To prevent the Web Agent from sending any proxy headers, blank out the ProxyHeadersUnprotected value. For example:

  ProxyHeadersUnprotected=""

  Note: To get a double quote character (“) to appear, use a single quote (‘). The Web Agent automatically converts it to a double quote.

- The value, %% or %d (treated identically) may appear within a ProxyHeaders line. This value is replaced with either the smaller of the IdleTimeout and SessionTimeout multiplied by the ProxyHeadersTimeoutPercentage, or, if the timeouts are not set, the ProxyHeadersDefaultTime is used.

- Ensure that values for the standard (1.1 and higher) and HTTP 1.0 headers are set properly for requests to the back-end server.
- ExpireForProxy="YES" will expire cookie provider redirects carrying the SMSESSION cookie in the query string.

Security Considerations

Browser sessions can persist after logout, so removing the SMSESSION cookie does not prevent a user from using the same browser session to view previously cached files. This problem occurs because the proxy server is not aware of the logout request and retains any protected/unprotected content in cache for the cache-control: private user until it timed out (cache-control: max-age=60). Thus, such a request would result in a page returned with a valid SMSESSION cookie. The only way to ensure security is to disable keep-alives or close the browser.

Further, the local browser cache is affected by the private/max-age combination since it observes local cache across sessions. For this reason, the max-age time for protected resources should be as short as possible.

Employing the if-modified-since and if-none-match request headers when the allowcacheheaders="FALSE" configuration setting is used (default) does not prevent the proxy server from observing these headers. Thus, these observed headers take effect on the request according to the proxy server.

You could work around this issue by installing:

- a Web Agent on the proxy server.
- another filter that removes these headers from the request.

Since HTTP 1.0, HTTP 1.1, or higher use different headers for specifying instructions to caching proxies, these versions should be configured in a way to ensure the most appropriate handling based on the type of connection.

Agents and Reverse Proxy Servers

See any of the following topics to manage your SiteMinder agent that is deployed on reverse proxy servers:

- Learn how reverse proxy servers work with SiteMinder (see page 226).
- Set the SM_PROXYREQUEST HTTP Header for the SiteMinder Secure Proxy Server (see page 228).
- Implement Application Request Routing (ARR) on IIS web servers (see page 228).
- Learn about the reverse proxy deployment factors. (see page 235)
- Configure an Apache-based web server as a reverse proxy (see page 236).
- Configure an Oracle iPlanet 7.0 server as a reverse proxy (see page 240).

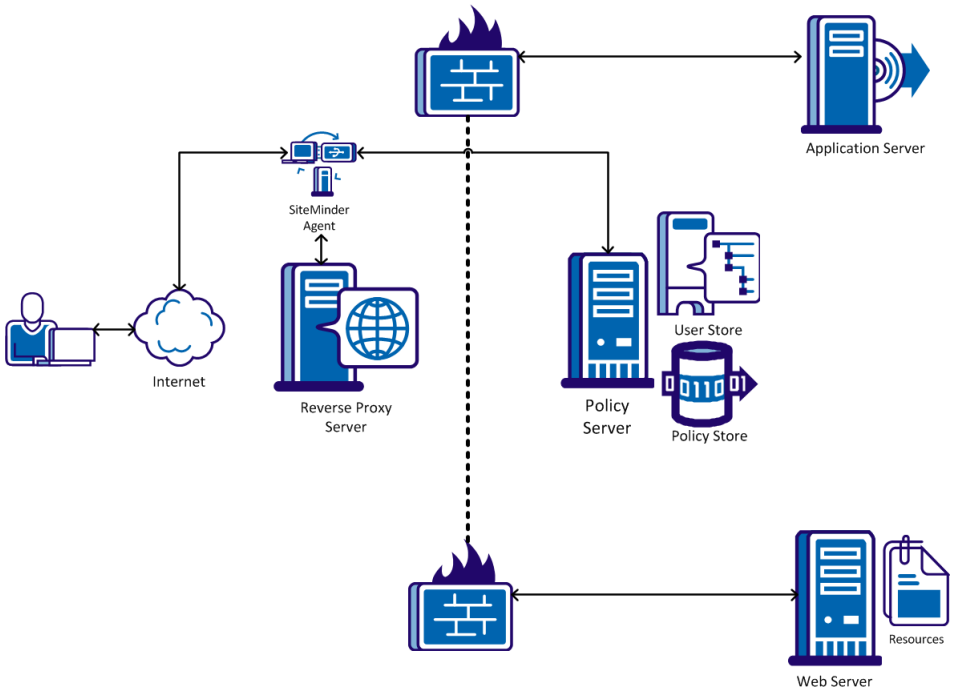
How Reverse Proxy Servers Work with SiteMinder

A reverse proxy server is a proxy server that acts on behalf of an enterprise to forward requests to the internal network of an organization. The reverse proxy server allows clients to access resources on backend servers (those servers behind a firewall).

Reverse proxy servers provide the following advantages:

- Users within a cookie domain can access resources on backend servers without reauthenticating. Users from other domains must authenticate through the reverse proxy server and typically, a firewall before gaining access to those same backend servers.
- Users can access different resources that are hosted on several backend servers using the same domain name.
- Reverse proxy agents support the same features as other SiteMinder agents.
- Protection resources that are on servers for which a SiteMinder agent is not supported. In this situation, deploy a reverse proxy server before the backend server. The supported agent protects the resources hosted on the backend server. The backend server does not require a SiteMinder agent.

SiteMinder agents that are installed on the reverse proxy server can protect resources on backend servers. The following illustration shows a network with a reverse proxy server using a SiteMinder agent:



SiteMinder Secure Proxy Server

For users who require a more sophisticated reverse proxy solution, CA SiteMinder SPS provides the following benefits over the Apache or Oracle iPlanet-based SiteMinder Reverse Proxy Agent:

- An embedded and fully supported web server, including SSL accelerator card support and a GUI tool for managing keys and certificates
- Support for multiple session schemes (cookie-based, and cookie-less)
- Support for flexible proxy rules, such as the following:
  - Support for rules that are based on HTTP headers and SiteMinder responses, in addition to URLs.
  - Ease of use for complex rules.

SM_PROXYREQUEST HTTP Header for SiteMinder Processing with Secure Proxy Server

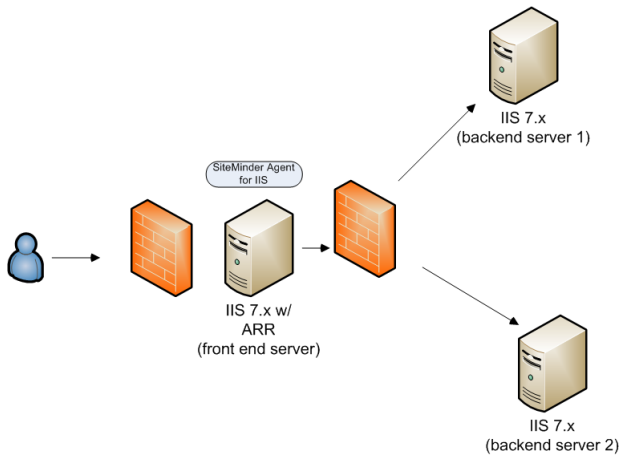
CA SiteMinder SPS introduces a new layer in the traditional SiteMinder architecture. This layer forwards or redirects all requests to destination servers in the enterprise.

When CA SiteMinder SPS processes a request, the URL requested by the user is preserved in an HTTP header variable named SM_PROXYREQUEST. Other applications that require the original URL requested by a user before CA SiteMinder SPS proxied the request can use this header.

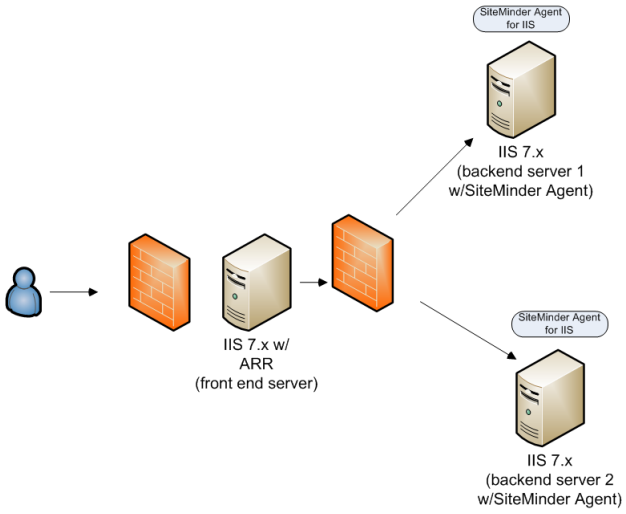
SiteMinder IIS 7.x Web Servers and Application Request Routing (ARR)

The SiteMinder r12.5 Agent for IIS supports the Application Request Routing feature of IIS 7.x. The following configurations are supported:

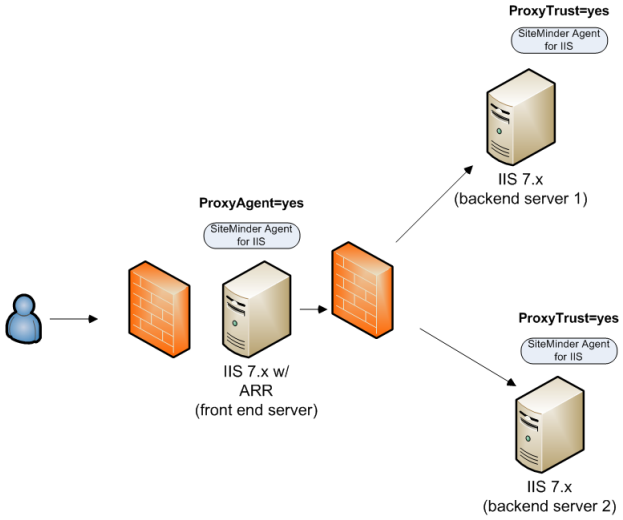
- An IIS 7.x web server running both ARR and the SiteMinder Agent for IIS in a DMZ, as shown in the following illustration (see page 233):



- Multiple IIS 7.x web servers running SiteMinder Web Agents for IIS behind another IIS 7.x server in the DMZ running ARR. This configuration is shown in the following illustration (see page 234):



- An IIS 7.x web server running both ARR and the SiteMinder Agent for IIS in a DMZ, and multiple IIS 7.x web servers running the SiteMinder Agent for IIS behind the the ARR server. This configuration is shown in the following illustration (see page 230):



How to Set up an IIS 7.x Server with ARR and SiteMinder in your DMZ with other SiteMinder Agents for IIS Operating Behind the DMZ

The SiteMinder Agent for IIS protects your entire IIS environment with the following configuration:

- An IIS 7.x web server with Application Request Routing (ARR) and a SiteMinder Agent for IIS in your DMZ (as a front-end server).
- Multiple IIS 7.x web servers *behind* the ARR server in the DMZ, with *each* using the SiteMinder Web Agent *or* Agent for IIS.

  Note: Only certain SiteMinder Web Agents support operating as a reverse-proxy server. However any web server hosting a supported SiteMinder Web Agent or Agent for IIS can accept traffic from a reverse proxy server running SiteMinder. For more information, see the Platform Support Matrix.

To implement the previous configuration, use the following multi-step process:

1. Install and configure ARR on the IIS 7.x web server in your DMZ (front end).

   Note: For more information about Application Request Routing (ARR), go to the IIS website, and search for the phrase, "Application Request Routing."

2. Install and configure a SiteMinder Agent for IIS on your IIS 7.x web server in your DMZ (front-end).

   Note: For more information, see the Web Agent Installation Guide for IIS.

3. Set the Web Agent Configuration parameters for the SiteMinder Agent for IIS in your DMZ (see page 231).

4. Install and configure a SiteMinder Agent for IIS on your first IIS 7.x web server *behind* your DMZ (back-end). For more information, see the Web Agent Installation Guide for IIS.

   Note: In this context, the first server refers to the IIS web server in a farm where the shared configuration information is stored. A node refers to any other IIS web servers in the farm which read the shared configuration from the first server.

5. Install and configure a SiteMinder Agent for IIS on your other IIS 7.x web server nodes *behind* your DMZ (back-ends).

6. Set the Web Agent Configuration Parameters for all of your IIS 7.x Servers using SiteMinder *behind* the DMZ (see page 232). Include the *first* web server and *all* nodes.

Set the SiteMinder Web Agent Configuration Parameters for your IIS 7.x ARR Server in the DMZ

This section describes how to set the Web Agent Configuration parameters running the SiteMinder Agent for IIS in the following situation:

- An IIS 7.x Web Server operates in the DMZ using ARR and the SiteMinder Agent for IIS (front end).
- Other IIS 7.x Web servers behind the DMZ receive requests from the ARR server, but do *not* use the SiteMinder Agent for IIS (back end).

Follow these steps:

1. Verify the following items:
   - ARR 2.0 is installed and configured on the web server in the DMZ.
   - The SiteMinder r12.5 Agent for IIS is installed and configured on the web server in the DMZ.
2. Open the Administrative UI.
3. Open the Agent Configuration Object (ACO) associated with your SiteMinder Agent for IIS (the front–end running in the DMZ).
4. Locate the following parameter:

   ProxyTrust

   Instructs the agent on a destination server to trust authorizations received from a SiteMinder agent on a proxy server. A destination server is a server that is behind a reverse proxy server. Setting this value to yes increases efficiency because only the agent on the proxy server contacts the Policy Server for authorization. The agent operating on the destination server does *not* contact the Policy Server again reauthorize users.

   Default: No
5. Verify that the value set in the ProxyTrust parameter is no.
6. Locate the following parameter:

   ProxyAgent

   Specifies if a Web Agent is acting as a reverse proxy agent.

   When the value of this parameter is yes, the SiteMinder agent on the front-end server preserves the original URL that the user requested in the SM_PROXYREQUEST HTTP header. This header is created whenever protected and unprotected resources are requested. The back-end server can read this header to obtain information about the original URL.

   Default: No
7. Change the value of the ProxyAgent parameter to yes.
8. Submit your changes to the Agent Configuration Object.

   The Web Agent Configuration parameters are set.

Set the Web Agent Configuration Parameters for your IIS 7.x Servers using SiteMinder Behind the DMZ

This section describes how to set the Web Agent Configuration parameters running the SiteMinder Agent for IIS in the following situation:

- An IIS 7.x server operates in the DMZ using ARR (front end).
- Other IIS 7.x servers behind the DMZ receive requests from the ARR server. Those servers also use the SiteMinder Agent for IIS (back end).

Follow these steps:

1. Verify the following items:
   - ARR 2.0 is installed and configured on the web server in the DMZ.
   - The SiteMinder r12.5 Agent for IIS is installed and configured on the first web server and all the nodes *behind* your DMZ.
2. Open the Administrative UI.
3. Open the Agent Configuration Object (ACO) associated with the first IIS server deployed *behind* the DMZ.
4. Locate the following parameter:

   ProxyTrust

   Instructs the agent on a destination server to trust authorizations received from a SiteMinder agent on a proxy server. A destination server is a server that is behind a reverse proxy server. Setting this value to yes increases efficiency because only the agent on the proxy server contacts the Policy Server for authorization. The agent operating on the destination server does *not* contact the Policy Server again reauthorize users.

   Default: No
5. Change the value of the ProxyTrust parameter to yes.
6. Locate the following parameter:

   ProxyAgent

   Specifies if a Web Agent is acting as a reverse proxy agent.

   When the value of this parameter is yes, the SiteMinder agent on the front-end server preserves the original URL that the user requested in the SM_PROXYREQUEST HTTP header. This header is created whenever protected and unprotected resources are requested. The back-end server can read this header to obtain information about the original URL.

   Default: No
7. Verify that the value of the ProxyAgent parameter is set to no.
8. Submit your changes to the Agent Configuration Object.

9. Open the Agent Configuration Object (ACO) associated with an IIS server node deployed *behind* the DMZ.
10. Repeat Steps 5 through 10 on each IIS web server node, until all the nodes behind the DMZ are configured.

    The Web Agent Configuration parameters are set.

How to Set Up an IIS 7.x Server with ARR and SiteMinder in your DMZ

To set up an IIS 7.x web server with Application Request Routing (ARR) and a SiteMinder Agent for IIS in your DMZ (as a front-end server), use the following multi-step process:

1. Install and configure ARR on the IIS 7.x web server in your DMZ (front end).

   Note: For more information about Application Request Routing (ARR), go to the IIS website, and search for the phrase, "Application Request Routing."
2. Install and configure a SiteMinder Agent for IIS on your IIS 7.x web server in your DMZ (front-end).

   Note: For more information, see the Web Agent Installation Guide for IIS.

How to Set up your IIS 7.x Servers with SiteMinder When Operating Behind an ARR Server in a DMZ

The SiteMinder Agent for IIS supports the following configuration using Application Request Routing (ARR):

- Operating several back-end web servers *behind* a DMZ-based IIS 7.x web server running ARR.
- Protecting those back end servers with SiteMinder Web Agents or Agents for IIS.

  Note: Only certain SiteMinder Web Agents support operating as a reverse-proxy server. However any web server hosting a supported SiteMinder Web Agent or Agent for IIS can accept traffic from a reverse proxy server running SiteMinder. For more information, see the Platform Support Matrix.

To implement this configuration, use the following multi-step process:

1. Install and configure ARR on the IIS 7.x web server in your DMZ (front end).

   Note: For more information about Application Request Routing (ARR), go to the IIS website, and search for the phrase, "Application Request Routing."

2. Install and configure a SiteMinder Agent for IIS on your first IIS 7.x web server *behind* your DMZ (back-end). For more information, see the Web Agent Installation Guide for IIS.

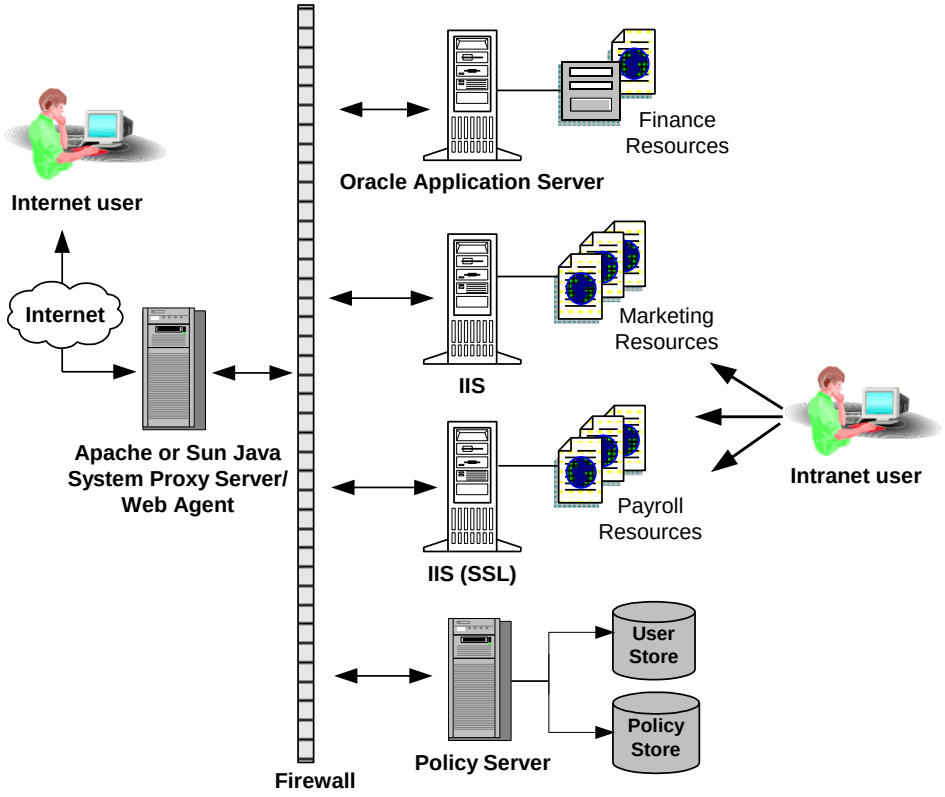
   Note: In this context, the first server refers to the IIS web server in a farm where the shared configuration information is stored. A node refers to any other IIS web servers in the farm which read the shared configuration from the first server.

3. Install and configure a SiteMinder Agent for IIS on your other IIS 7.x web server nodes *behind* your DMZ (back-ends).

SiteMinder Reverse Proxy Deployment Considerations

Typically, when you deploy an Apache or Oracle iPlanet reverse proxy Agent, a firewall is located between the Apache or Oracle iPlanet Web Agent and the servers hosting the protected resources. The Policy Server should also be located behind the firewall.

The following illustration shows a SiteMinder reverse proxy deployment.



When deploying a SiteMinder reverse proxy Agent, consider the following:

- If a policy has been configured to return response attributes, the variables are sent to both the reverse proxy server and the backend web server on which the protected resource resides. When a request is made for a protected resource, the Policy Server first sends response attributes (CGI or HTTP variables) to the Agent on the Apache or Oracle iPlanet server. The Agent then puts the response attributes in the request that is sent to the backend server.
- If any of the backend servers or protected applications provide their own authentication functionality, the authentication must be disabled. Disabling the backend authentication ensures that SiteMinder's authentication takes precedence.

  Important! When configuring the cache for the reverse proxy be aware that all cookies are cached, including the SMSESSION cookie. For assistance see your Apache or Oracle iPlanet web server documentation.

More Information

Define HTTPS Ports (see page 84)

How to Configure an Apache Reverse Proxy Server

You can configure an Apache web server to function as a reverse proxy server with any SiteMinder agent. The following process lists the steps for configuring an Apache reverse proxy server:

1. Update the Apache web server configuration file (see page 237).
2. Update the agent configuration parameters for a SiteMinder agent (see page 239).

Update the Apache Web Server Configuration File

Update the configuration file of Apache web server to make the Apache web server function as a reverse proxy server with a SiteMinder agent.

Follow these steps:

1. Open the httpd.conf file available at the following location:

   ```
   /etc/httpd/conf/httpd.conf
   ```

2. Add the following directives to the httpd.conf file:

   ProxyPass

   Allows mapping of remote servers to the local server. The values in this directive use the following format:

   /local_virtual_path partial_URL_of_remote_server

   Example: ProxyPass /realma/ http://server.example.org/realma/

   ProxyPassReverse

   Allows adjustment of the location header by the Apache server on HTTP redirect responses. The values in this directive use the following format:

   /local_virtual_path partial_URL_of_remote_server

   Example: ProxyPassReverse /realma/ http://server.example.org/realma/

   For the Apache web server, add the following Proxy Pass settings to the configuration file.

   ```
   # SiteMinder Administrative UI
   <Location "/iam/siteminder/">
        <IfModule proxy_module>
                    ProxyPass http://hostname:port/iam/siteminder/
          ProxyPassReverse http://hostname:port/iam/siteminder/
       </IfModule>
     # Alternate unavailable page
     ErrorDocument 503 /siteminderagent/adminui/HTTP_SERVICE_UNAVAILABLE.html
   </Location>
   # CA Styles r5.1.1
   <Location "/castylesr5.1.1/">
       <IfModule proxy_module>
        ProxyPass http://hostname:port/castylesr5.1.1/
       ProxyPassReverse http://hostname:port/castylesr5.1.1/
       </IfModule>
   </Location>
   ```

   Note: *hostname:port* refers to the host and port of the application server running the Administrative UI.

3. Uncomment the following line in the configuration file.

   ```
   LoadModule proxy_module modules/mod_proxy.so
   ```

4. Save and close the configuration file.
5. Restart the Apache web server.

Update the Agent Configuration Parameters for a SiteMinder Agent

For Apache-based servers behind the Apache reverse proxy server, update the following agent configuration parameters.

Follow these steps:

1. Set the value of the following parameter to yes:

   ProxyAgent

   Specifies if a Web Agent is acting as a reverse proxy agent.

   When the value of this parameter is yes, the SiteMinder agent on the front-end server preserves the original URL that the user requested in the SM_PROXYREQUEST HTTP header. This header is created whenever protected and unprotected resources are requested. The back-end server can read this header to obtain information about the original URL.

   Default: No

2. Set the following parameter:

   ProxyTimeout

   Specifies the number of seconds the reverse proxy server waits for the SiteMinder agent that is deployed behind it to respond to a request.

   Default: 120

   Note: This parameter applies to Apache-based agents only.

3. (Optional) Set the following parameter:

   ProxyTrust

   Instructs the agent on a destination server to trust authorizations received from a SiteMinder agent on a proxy server. A destination server is a server that is behind a reverse proxy server. Setting this value to yes increases efficiency because only the agent on the proxy server contacts the Policy Server for authorization. The agent operating on the destination server does *not* contact the Policy Server again reauthorize users.

   Default: No

4. Edit the BadURLChars parameter by removing all occurrences of the following value from the list:

   %

5. Set the httpsports parameter to indicate to the Apache server which port is set up for SSL.

6. Restart the Apache web server.

   Note: For more information about modifying agent configuration parameters, see the *Policy Server Configuration Guide*.

Configure an Oracle iPlanet 7.0 Reverse Proxy Server

You can use a Oracle iPlanet 7.0 web server as a reverse proxy with SiteMinder.

Note: The SiteMinder Agent Configuration wizard *only* modifies the *default* obj.conf file on the Oracle iPlanet (formerly Sun Java System) web server. To protect other instances or reverse proxy deployments with SiteMinder, copy the SiteMinder settings from the default obj.conf file to any respective *instance_name*-obj.conf files. For example, your web server created an obj.conf file when you installed it, but you later added a server instance named my_server.example.com. To protect resources on my_server.example.com with SiteMinder, copy the SiteMinder settings the wizard added from the obj.conf file to the my_server.example.com-obj.conf file.

Follow these steps:

1. Add the following directive to the *instance_name*-obj.conf file:

   NameTrans

   Specifies the local and remote virtual paths using the following format:

   NameTrans fn="map" from="*local_virtual_path*" name="reverse-proxy-/local_virtual_path" to="*remote_virtual_path*"

   Example: NameTrans fn="map" from="/realma" name="reverse-proxy-/reamla" to="http://server.example.org/realma/"

2. Add the following directives at the *end* of the obj.conf file:

   Object name

   Specifies the name of the local virtual path and the URL of the remote virtual path used in the NameTrans directive, using the following format:

   <Object name="reverse-proxy-/*local_virtual_path*">

   Route fn="set-origin-server" server="http://*remote_server_URL*:*port*"

   </Object>

   Example: <Object name="reverse-proxy-/realma">

   Route fn="set-origin-server" server="http://server.example.org:*port*"

   </Object>

   Object ppath

   Specifies the partial path that is given to the server by the client.

   Example: <Object ppath="http:*">

   Service fn="proxy-retrieve" method="*"

   </Object>

3. Restart the web server.

   The reverse proxy is configured.

HTTP Header Settings

Use any of the following settings to control URL processing:

- Remove the Server HTTP header to accommodate the URLScan utility (see page 241).

Remove the Server HTTP Header if Using the URLScan Utility

If you want to use the URLScan utility from Microsoft to remove the Server HTTP Header from the responses your IIS Web server sends, you also need to set the following parameter for your IIS Web Agent:

SuppressServerHeader

Prevents an IIS Web Agent from returning the Server HTTP Header in its responses. When the value of this parameter is set to no, the Web Agent sends the Server header with its responses and the IIS Web server passes it along to the client. When the value of this parameter is set to yes, the web agent does not send the Server header in its responses.

Default: No

The URLScan utiltiy removes the header from the IIS server's responses, while hte SuppressServerHeader parameter removes the header from the Web Agent's responses. Both the utility and the parameter must be set to prevent the Server header from being sent to the client in all responses.

To keep the Web Agent from sending the Server header in responses, set the value of the SuppressServerHeader parameter to yes.

Ensure Custom Responses Comply with X-Frame Options

If you use the X-Frame-Options response header in your web applications, you can ensure that any customized responses from your agent return this header properly. The setting in the X-frame options header determines if the browser renders a page with content between a <frame> or an <iframe> tag.

You can determine whether the custom responses from your agent comply with X-frame-options with the following parameter:

XFrameOptions

Specifies whether custom responses comply with the x-frame-options response headers. Setting this parameter sets any custom responses with the correct x-frame-options header.

Default: No (responses do not comply with x-frame options response headers).

Range: Yes, No

To ensure that your custom responses comply with x-frame options, set the value of the XFrameOptions parameter to yes.

URL Settings

Use any of the following settings to control how URLs are processed:

- Specify the protocol using lowercase characters (see page 153).
- Decode query data in a URL (see page 84).
- Set a maximum URL size (see page 244).

Specify Redirect URL Protocols with Lowercase Characters

If you protect legacy applications that do not confirm to RFC 2396 with a forms-based authentication scheme, and you need the protocol portions of URLs to be lowercase, then set the following parameter:

LowerCaseProtocolSpecifier

Specifies whether the scheme (protocol) portion of a redirect URL uses only lowercase characters. This configuration parameter accommodates legacy applications that do not confirm to RFC 2396. This RFC states that applications must handle the protocol portion of a URL in both uppercase and lowercase. Change this parameter in any of the following situations:

- You use legacy applications that do not confirm to RFC 2396.
- Your redirect URLS contain query data.
- You use an HTML-forms (FCC) authentication scheme.

Default: No (uppercase characters are used HTTP, HTTPS).

Example: Yes (lowercase characters are used http, https).

To specify lowercase protocols for the URLs in your environment, set the value of the LowerCaseProtocolSpecifier parameter to yes.

Decode Query Data in a URL

To have the Web Agent's Base64 algorithm decode a URL's query data before calling the Policy Server (so the Policy Server sees the proper resource), use the following parameter:

DecodeQueryData

Specifies whether the Web Agent decodes the query data in a URL before calling the Policy Server. Set this parameter to yes if you need do any of the following tasks in your environment:

- If you need to ensure the rules filer acts against the proper string.
- If you need to or write rules against the data in a query string.

Default: No

To have the Web Agent decode the query data in a URL before calling the Policy Server, set the value of the DecodeQueryData parameter to yes.

Set a Maximum URL Size

You can increase the maximum URL size that a Web Agent can handle with the following parameter:

MaxUrlSize

Specifies the maximum size (in bytes) of a URL that a Web Agent can handle. Because different web servers have different limitations on URL length, check the documentation from your web server vendor before setting this parameter.

Default: 4096 B

To change the maximum URL size, change the number of bytes specified in the MaxUrlSize parameter.

IIS Web Server Settings

Use any of the following settings to manage your Agent for IIS:

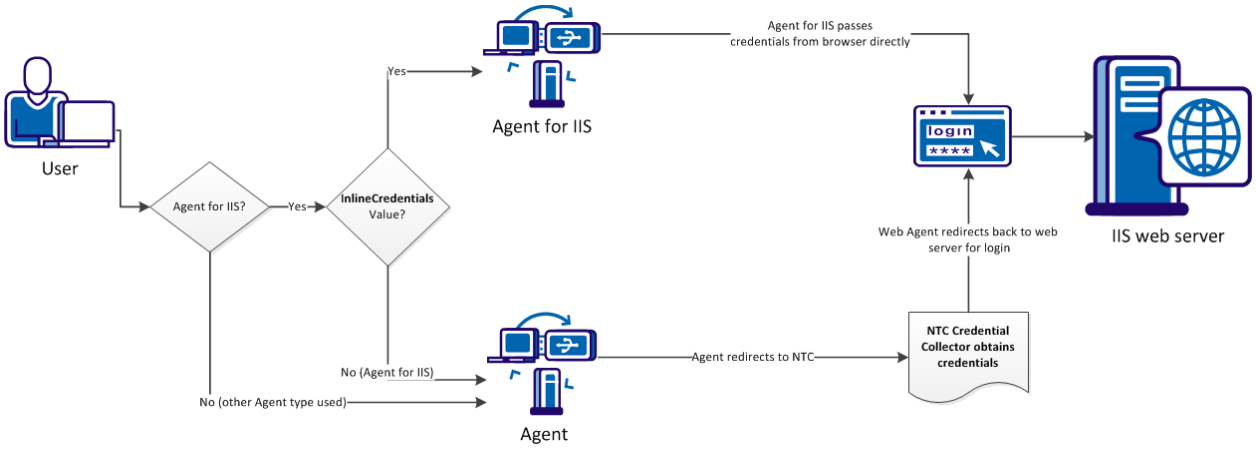
- Remove integrated windows authentication (IWA) redirection with the InlineCredentials parameter (see page 244).
- Record the user name and transaction ID in IIS server logs (see page 246).
- Use the NetBIOS name or UPN for IIS authentication (see page 248).
- Configure NT Challenge/Response authentication (see page 248).
- Implement an information card authentication scheme (see page 254).
- Configure an FCC template for an information card authentication scheme (see page 255).
- Control IIS 7.x module execution order (see page 256).
- Use an IIS proxy user account (see page 258).
- Enable anonymous user access (see page 259).
- Disable Windows security context on an Agent for IIS (see page 259).
-

Configure Agents for IIS to Obtain User Credentials Without Redirecting to an NTLM Credential Collector (NTC)

By default, SiteMinder Agents redirect requests for resources protected by the Windows authentication scheme to an NTLM credential collector (NTC) to retrieve their Windows credentials.

You can configure SiteMinder Agents for IIS to obtain the credentials of the user from the HTTP request inline (that is, without redirecting to an NTC).

The following illustration describes the differences between the two credential collection methods:



To configure an agent to obtain credentials of the user from the HTTP request without redirecting to an NTC, set the InlineCredentials configuration parameter as follows:

InlineCredentials

Specifies how the Agent for IIS handles user credentials. When the value of this parameter is yes, the Agent for IIS reads the credentials directly from the HTTP request. When the value of this parameter is no, the Agent redirects to an NTC credential collector.

Default: No

Note: If any SiteMinder Agents in your environment are configured to use NTC redirects, configure NT challenge/response authentication.

More information:

Configure Agents for IIS to Support NT Challenge/Response Authentication (see page 248)

Record the User Name and Transaction ID in IIS Server Logs

The Web Agent generates a unique transaction ID for each successful user authorization request. The Agent adds the ID to the HTTP header. The ID is also recorded in the following logs:

- Audit log
- Web server log (if the server is configured to log query strings)
- Policy Server log

You can track user activities for a given application using the transaction ID.

Note: For more information, see the Policy Server documentation.

The transaction ID appears in the log as a mock query parameter in the log that is appended to the end of an existing query string. The following example shows transaction ID (in bold) appended to a query string (which ends with STATE=MA):

```
172.24.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realm/index.html,
STATE=MA&SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1
```

If no query parameters are in the URL, the Agent adds the transaction ID at the end of the web server log entry. For example:

```
172.24.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realma/index.html,
SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1.
```

Note: Web Agents log user names and access information in native web server log files when users access resources.

Agents protecting resources on IIS servers do *not* record the SiteMinder transaction ID and authenticated user name in the IIS server logs by default. Because these agents can operate as ISAPI extensions (in Classic Pipeline mode), the server would have already logged a transaction. You can force the agent to add this information with the following parameters:

AppendIISServerLog

Instructs the agent to add the authenticated user name and SiteMinder transaction ID to the cs-uri-query field of the IIS server log. For URLs containing query strings, commas separate the query string, SiteMinder transaction ID, and username in the cs-uri-query field of the IIS server log.

Default: No

SetRemoteUser

Specifies a value for the *REMOTE_USER* HTTP header variable which legacy applications could possibly require.

Default: No

To record the transaction ID and user name in the IIS server log.

1. Set the value of the AppendIISServerLog parameter to yes.
2. Set the value of the SetRemoteUser parameter to yes.

   The user name and transaction ID appears in the IIS server logs.

Use the NetBIOS Name or UPN for IIS Authentication

In an IIS network, you may have a NetBIOS name that is different than the domain name for the location of a requested resource. When a user tries to access a protected resource and there are multiple domain controllers, user authentication fails and the web server log shows an "IIS logon failure." You can control whether the UPN or NetBIOS name is sent to the IIS web server with the following parameter:

UseNetBIOSforIISAuth

Specifies whether the IIS 6.0 Web Agent sends the user principal name (UPN) or the NetBIOS name to the IIS 6.0 web server for IIS user authentication.

Note: This parameter is valid only if an Active Directory user store is associated with the Policy Server.

If you enable this parameter, the Policy Server extracts the UserDN, the UPN, and the NetBIOS name from the Active Directory during SiteMinder authentication, and sends this data back to the IIS 6.0 Web Agent.

Depending on whether or not you selected the Run in Authenticated User's Security Context option for the user directory with the Administrative UI and how you set the UseNetBIOSforIIAuth parameter, a user's logon credentials are sent as follows:

- When the UseNetBIOSforIISAuth parameter is set to no, the IIS 6.0 Web Agent sends the UPN name.
- When the UseNetBIOSforIISAuth parameter is set to yes, the Web Agent sends the NetBIOS name.

The IIS web server authenticates the user with the credentials it receives from the Web Agent.

Default: No

To have the Web Agent use the NetBIOS name for IIS authentication, set the UseNetBIOSAuth parameter to yes.

Configure Agents for IIS to Support NT Challenge/Response Authentication

If any SiteMinder Agents in your environment are configured to use NTC redirects, configure NT challenge/response authentication.

With NT challenge/response authentication, the IIS web server challenges the Internet Explorer browser of a user when that user requests access to a resource.

Note: NT challenge/response authentication only works with Internet Explorer browsers.

You can implement NT challenge/response authentication in either of the following ways:

- Challenge users when they try to access protected resources. Users in single-sign on environments are only challenged the first time that they request a resource.
- Have your users configure the automatic logon feature of their Internet Explorer browser.

  The automatic logon feature allows users to access a resource without being challenged. The authentication process still takes place, but the NT challenge/response process between the browser and the server is transparent to the user. Automatic logon is typically used for Intranets where security is less strict and you want users to have seamless access to resources. We do not recommend using the Automatic logon feature for communication across the Internet.

SiteMinder Agents use credential collectors to gather the Windows credentials of users for the NT challenge/response authentication scheme. The agent supports the NTC extension for collecting NTLM credentials.

Note: Set the NTCEXT only if you want to change this default behavior.

To make SiteMinder operate with NT challenge/response authentication, use the following process:

1. Set up the NT Challenge response authentication for the IIS web server with the following tasks:
   a. Map the .ntc file extension (see page 250).
   b. Create and configure the virtual directory, and then verify that it requires the NT challenge and response credentials (see page 250).
2. Configure the Windows authentication scheme for NT challenge/response authentication in the Administrative UI (see page 251).
3. Specify an NTLM credential collector (see page 142).
4. Configure policies for NT Challenge/Response authentication using the Administrative UI.

   Note: For more information, see the *Policy Server Configuration Guide*.
5. (Optional) Have your users configure the automatic logon feature of their Internet Explorer browser (see page 252).

   The NT Challenge Response Authentication for IIS is configured.

More Information

Configure Agents for IIS to Obtain User Credentials Without Redirecting to an NTLM Credential Collector (NTC) (see page 244)

Map the .NTC File Extension

Map the .NTC file extension to the ISAPIWebAgent.dll application to configure NT Challenge/Response Authentication on the IIS Web Server.

To map the .NTC file extension

1. Open the Internet Services Manager.
2. Right-click Web Sites in the left pane, and then right-click Default Web Site in the right pane and select Properties.

   The Default Web Site Properties dialog appears.
3. Click the Home Directory tab.
4. In the Application Settings section, click Configuration.

   The Application Configuration dialog appears.
5. Click Add.

   The Add/Edit Application Extension Mapping dialog opens.

   a. In the Executable field, click Browse and locate the following file: *web_agent_home*/bin/ISAPIWebAgent.dll.

   b. Click Open.

   c. In the Extension field, enter .ntc.
6. Click OK three times.

   The Add/Edit Application Extension Mapping dialog, the Application Configuration dialog and the Default Web Site Properties dialog close. The .ntc file extension is mapped.

Create and Configure the Virtual Directory for Windows Authentication Schemes (IIS 7.5)

To use the SiteMinder Windows authentication scheme, configure a virtual directory on the IIS 7.x web server. The virtual directory requires Windows challenge and response for credentials.

Follow these steps:

1. Open the Internet Information Services (IIS) Manager.
2. In the left pane, expand the following items:
   - The web server icon
   - The Sites folder
   - The Default Web Site icon
3. Right-click the siteminderagent virtual directory, and then select Add Virtual Directory.

   The Add Virtual Directory dialog appears.
4. In the Alias field, type the following value:

   `ntlm`
5. Click the Browse button (next to the Physical Path field), and then locate the following directory:

   web_agent_home`\samples`

   The virtual directory is created.
6. Configure the virtual directory with *one* of the following steps:
   - To protect all the resources on the entire website with the SiteMinder Windows authentication scheme, click the Default Web Site icon.
   - If you do *not* want to protect the entire website, with the SiteMinder Windows authentication scheme, click the ntlm virtual directory (you created in Step 4).
7. Double-click the Authentication icon.

   The Authentication dialog appears.
8. Do the following steps:
   a. Right-click Anonymous Authentication, and then select Disable.
   b. Right-click Windows Authentication, and then select Enable.

      The virtual directory for Windows authentication schemes is configured.

   Note: Reboot the web server for these changes to take effect.

Configure the Windows Authentication Scheme for Challenge/Response Authentication

To implement NT Challenge/Response authentication, provide the policy administrator responsible for configuring the Windows authentication scheme with the following values:

Server Name

The fully qualified domain name of the IIS web server, for example:

server1.myorg.com

Target

/siteminderagent/ntlm/smntlm.ntc

Note: The directory must correspond to the virtual directory already configured by the installation. The target file, smntlm.ntc, does not need to exist and can be any name that ends in .ntc or the custom MIME type that you use in place of the default.

Library

smauthntlm

More Information

MIME Types for Credential Collectors (see page 135)

Specify an NTLM Credential Collector

The NTLM credential collector (NTC) is an application within the Web Agent. The NTC collects NT credentials for resources that the Windows authentication scheme protects. This scheme applies to resources on an IIS web server that are accessed by Internet Explorer browsers.

Each credential collector has an associated MIME type. For IIS, the NTC MIME TYPE is defined in the following parameter:

NTCExt

Specifies the MIME type that is associated with the NTLM credential collector. This collector gathers NT credentials for resources that the Windows authentication scheme protects. This scheme applies to resources on IIS web servers that only Internet Explorer browser users access.

You can have multiple extensions in this parameter. If you are using an Agent Configuration Object, select the multivalue option. If you are using a local configuration file, separate each extension with a comma.

Default: .ntc

If your environment already uses the default extension that the NTCExt parameter specifies, you can specify a different MIME type.

To change the extension that triggers the credential collector, add a different file extension to the NTCExt parameter.

Configure Automatic Logon for Internet Explorer

To authenticate users *without* the agent challenging them for their credentials, Internet Explorer browser users must configure the Automatic Logon browser security setting.

Follow these steps:

1. Start the Internet Explorer browser.
2. Open the Internet Options dialog. (Refer to the Internet Explorer online help to find out how to open the dialog for your version of the browser).
3. Click the Security tab.
4. Click the correct security zone.
5. Click Custom Level.
6. Scroll down to the User Authentication section. Under the Logon option, click the Automatic Logon with current username and password option.
7. Apply the changes.

   The Security Settings dialog and the Internet Options dialog close. Your settings are saved, and automatic login is configured.

How to Implement an Information Card Authentication Scheme

CA SiteMinder supports an Information Card Authentication Scheme (ICAS) that implements Windows CardSpace. Users who request access to protected resources can select an authentication card. SiteMinder uses the information contained in the card to verify the identity of the user.

Implementing an ICAS requires configuration changes on the following SiteMinder components:

- The server hosting the SiteMinder Web Agent
- The SiteMinder Policy Server
- The smkey database

Follow these steps:

1. Do the following tasks on the web server:
   a. Enable SSL communication on the IIS web server.

      Note: For more information, see your Microsoft documentation, or go to http://support.microsoft.com/

   b. Export the web server certificate as a .pfx file.
   c. Customize the SiteMinder InfoCard.fcc template.
2. Do the following tasks on the Policy Server:
   a. Install the JCE on the Policy Server.
   b. Update the java.security file on the Policy Server.
   c. Update the config.properties file on the Policy Server.
   d. If you do not already have an smkey database, Create one with the Policy Server Configuration wizard.
   e. Add the .pfx file certificate from the web server to the smkey database.
   f. Configure the user directory in the Policy Server.
   g. Create a custom authentication scheme for CardSpace using the Administrative UI.
   h. (Optional) Store the claims in the session store to use in responses.
   i. (Optional) Enable personalization by allowing the retrieval of claim values from the session store.
   j. (Optional) Configure an active response to retrieve a stored claim value.

Configure an FCC Template for an Information Card Authentication Scheme

The SiteMinder Web Agent includes a Forms Credential Collector (FCC) template that you can use to implement an ICAS in SiteMinder.

Follow these steps:

1. Open the following default FCC file with a text editor:

   web_agent_home\samples_default\forms\InfoCard.fcc

2. Save a *copy* of the file to the following directory (creating a copy preserves the default FCC settings in case you need them later):

   web_agent_home\samples\forms\

3. Record the following information from your copy of the FCC file:

   Important! The Policy Server needs this information for its configuration.

   - The fully qualified domain name of the IIS web server that hosts the Web Agent.
   - The name of the FCC file you saved in Step 2.
   - The value (*without* quotation marks) of the requiredClaims parameter tag in the FCC file from Step 2. See the following example:

     ```
     http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier
     ```

   - (Optional) The value of Level of Assurance (LOA) requiredClaims parameter tag when White List processing is done. See the following example:

     ```
     < param name="requiredClaims" value="
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel1 ”/>
     ```

     The URIs for the various LOA’s are

     ```
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel1
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel2
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel3
     ```

4. (Optional) Use the text editor to make any of the following changes in copy of the FCC file.

   - To use a custom logo, replace the netegrity_logo.gif file with your own graphic and update the following link in the FCC file accordingly:

     ```
     <img alt="Logo" src="/siteminderagent/dmspages/netegrity_logo.gif">
     ```

Control IIS 7.x Module Execution Order when using the SiteMinder Agent for IIS

When you install and configure the SiteMinder Agent for IIS on an IIS web server, the Agent for IIS executes before any other modules. If your IIS environment requires another module to execute first, you can change the number set the following location in the Windows Registry:

```
HKLM\Software\Netegrity\SiteMinder Web Agent\Microsoft IIS\RequestPriority
```

For example, suppose another module in your IIS 7.x web server (like UrlScan) is assigned the same execution priority as the SiteMinder Agent for IIS. Use this setting to control when the SiteMinder module executes.

Follow these steps:

1. Open the Windows Registry Editor on your IIS web server.
2. Expand the following keys:

   ```
   HKLM\Software\Netegrity\SiteMinder Web Agent\Microsoft IIS
   ```

3. Locate the following value:

   ```
   RequestPriority
   ```

4. Change the value of RequestPriority to the number which corresponds to the following value you want:

   PRIORITY_ALIAS_FIRST

   Executes the SiteMinder Agent for IIS before any other modules on your IIS web server. This setting is the default.

   Example: 0 (First)

   Default: 0

   PRIORITY_ALIAS_HIGH

   Executes the SiteMinder Agent for IIS module after any modules set to execute first, but before any modules set to execute with medium, low or last priority.

   Example: 1 (High)

   PRIORITY_ALIAS_MEDIUM

   Executes the SiteMinder Agent for IIS module after modules set to execute first and high, but before modules set to execute with low or last priority.

   Example: 2 (Medium)

   PRIORITY_ALIAS_LOW

   Executes the SiteMinder Agent for IIS module after modules set to execute first, high, and medium, but before modules set to execute with last priority.

   Example: 3 (Low)

   PRIORITY_ALIAS_LAST

Executes the module for the SiteMinder Agent for IIS *after* all other modules.

Example: 4 (Last)

5. Save your changes and close the registry editor.
6. Test your settings and verify that the module you want executes before the Agent for IIS module executes.

Use an IIS Proxy User Account (IIS Only)

If users try to access resources on an IIS web server protected by SiteMinder, the Web Agent may deny access if those users lack sufficient IIS privileges for those resources. For example, if users are stored in an LDAP user directory on a UNIX system, those users may not have access to the Windows system with the IIS web server.

The IIS web server has a default proxy account that has sufficient privileges for users who are granted access by SiteMinder. The Web Agent uses the values of the DefaultUserName and DefaultPassword parameters as credentials even if the user has a valid Windows security context.

Follow these steps:

1. Set the value of the ForceIISProxyUser parameter to *one* of the following values:
   - If access to the applications on the IIS server is based on the users' credentials themselves, set the value of the ForceIISProxyUser parameter to yes.
   - If access to the applications on the IIS server is based on a specific account (such as a proxy) which acts on behalf of the users, set the value of the ForceIISProxyUser parameter to no.

   Default: No
2. If you are *not* using either of the following Windows features, continue with Step 3:
   - The Windows authentication scheme
   - The Windows User Security Context
3. Enter the user name for the proxy user account in the DefaultUserName parameter. If you are using a domain account, and the local machine is not a part of that domain, use the syntax shown in the following example:

   DefaultUserName=*Windows_domain\acct_with_admin_privilege*

   Otherwise, specify just the user name.
4. Enter the password associated with the existing Windows user account in the DefaultPassword parameter.

   Important! We recommend setting this parameter in your Agent Configuration Object because you can encrypt it. If you set it in a local configuration file, the value is stored unencrypted in plain text.

   The IIS Proxy account is configured.

Enable Anonymous User Access

If you do not want users to have access as the proxy user, you can set the following parameter:

UseAnonAccess

Instructs the IIS Web Agent to execute the web application as an anonymous user, instead of using credentials of the proxy user.

Default: No

Note: This parameter applies to IIS Web Agents only.

To enable anonymous user access, set the UseAnonAccess parameter to yes.

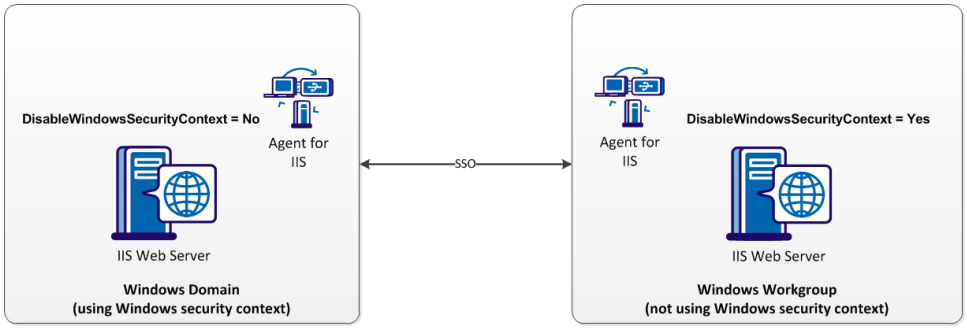
Disable Windows Security Context on Agents for IIS

The SiteMinder Policy Server obtains the Windows security context from the session of the user. In most situations, this environment is acceptable for single-sign on because the session information is available to all agents.

The following situation provides an example of a situation where different settings are required for single-sign on:

- One SiteMinder agent uses the Windows security context.
- Another SiteMinder agent does not use the Windows security context.

This situation is shown in the following illustration:



To permit SSO between a Windows domain using Windows security context and a Windows workgroup not using Windowssecurity context, set the following parameter:

DisableWindowsSecurityContext

Disables the Windows security context for the agent. When the value of this parameter is yes, the agent ignores the Windows security context of the user. When the value of this parameter is false or no, the agent uses the Windows security context contained in the session of the user. This parameter allows single-sign on between Windows environments which use the security context Windows environments that do not.

Default: False

Limits: Yes, No

Apache Web Server Settings

Use any of the following settings to manage your SiteMinder Agent for Apache-based servers:

- Set the HttpsPorts parameter (see page 260).
- Use legacy applications (see page 261).
- Use the HTTPHostRequest parameter for the port number (see page 262).
- Record the transaction ID in Apache web server logs (see page 262).
- Choose how content types are transferred in POST requests (see page 263).
- Restrict IPC semaphore-related message output to the Apache error log (see page 263).
- Delete certificates from Stronghold servers (see page 264).

Use the HttpsPorts Parameter on Apache 2.x Servers

If you use an SSL accelerator or any intermediate device that changes the value of the HTTP_HOST header with an Apache 2.x Web server and use the HttpsPorts parameter, additional web server configuration changes are required.

Follow these steps:

1. Open the httpd.conf file of your Apache Web server, and then make the following changes:
   - Change the value of the UseCanonicalName parameter to on.
   - Change the value of the ServerName parameter to the following:

     server_name:*port_number*

     server_name

     Specifies the host name of the SSL accelerator.
2. Modify the following configuration parameter for your Web Agent:
   - Change the value of the GetPortFromHeaders parameter to yes.

Use Legacy Applications with an Apache Web Agent

If you have legacy applications (that do not support HTTP 1.1), and you want to run them on an Apache Web Server, you can set the following parameter:

LegacyTransferEncodingBehavior

Specifies the type of message encoding used by the Web Agent. When the value of this parameter is set to no, transfer-encoding is supported.

When the value of this parameter is set to yes, content encoding is used. The transfer-encoding header is ignored and only the content-length header is supported.

Default: No

To use legacy applications with an Apache Web Server, set the value of the LegacyTransferEncodingBehavior parameter to yes.

Important! If you set the value of this parameter to yes, these features will not work: Federation; preservation of POST data longer than 4 KB; and large certificates may not be recognized.

Use the HTTP HOST Request for the Port Number

If you have applications that perform load balancing by redirecting traffic to specific web servers *without* modifying the actual HTTP headers, you should configure the Web Agent to redirect users back to the proper external port (instead of the port used by the load balancer) with the following parameter:

GetPortFromHeaders

Directs the Web Agent to obtain the port number from the HTTP HOST request header instead of obtaining it from the web server service structures.

Default: No

Note: This parameter is required for Apache Web Agents.

To use the port number in the HTTP HOST request header, set the GetPortFromHeaders parameter to yes.

Record the Transaction ID in Apache Web Server Logs

The Web Agent generates a unique transaction ID for each successful user authorization request. The Agent adds the ID to the HTTP header. The ID is also recorded in the following logs:

- Audit log
- Web server log (if the server is configured to log query strings)
- Policy Server log

You can track user activities for a given application using the transaction ID.

Note: For more information, see the Policy Server documentation.

The transaction ID appears in the log as a mock query parameter in the log that is appended to the end of an existing query string. The following example shows transaction ID (in bold) appended to a query string (which ends with STATE=MA):

```
172.24.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realm/index.html,
STATE=MA&SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1
```

If no query parameters are in the URL, the Agent adds the transaction ID at the end of the web server log entry. For example:

```
172.24.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realma/index.html,
SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1.
```

Note: Web Agents log user names and access information in native web server log files when users access resources.

You can record the SiteMinder transaction ID in the Apache web server logs SMTRANSACTIONID header variable.

Follow these steps:

1. Open the httpd.conf file.
2. Add the SM_TRANSACTIONID header variable to the LogFormat directive.

   For example:

   ```
   LogFormat "%h %l %u %t \"%r\" %>s %b \"%{SM_TRANSACTIONID}i\"" common
   ```

   Note: For more information about the httpd.conf file and the LogFormat directive, see your Apache web server documentation.
3. Restart the server to apply the change.

   The transaction ID is recorded in the Apache web server logs.

Choose How Content Types are Transferred in POST Requests

If you are using an Apache web server, you can control how content is transferred to the server during POST requests with the following parameter:

LegacyStreamingBehavior

Specifies how content will be transferred to the server during POST requests. When the value of this parameter is set to yes, all content types are streamed, *except* for the following:

- text/xml
- application/x-www-form-urlencoded

When the value of this parameter is set to no, all content types are spooled.

Default: No

To stream most types of content in POST requests, change the value of the LegacyStreamingBehavior parameter to yes.

Restrict IPC Semaphore-Related Message Output to the Apache Error Log

By default the Apache Web Agent logs all levels (informational and error) of IPC semaphore-related messages to the Apache error log, regardless of the configured Apache logging level.

To restrict the verbosity of Web Agent IPC semaphore-related output to the Apache error log, add the following parameter in the trace.conf file located in *web_agent_home*/config:

nete.stderr.loglevel

Specifies the level of IPC semaphore-related messages the Web Agent logs to the Apache error log. Accepts the following values:

off

The Web Agent logs no IPC semaphore-related messages to the Apache error log.

error

The Web Agent logs only IPC semaphore-related error messages to the Apache error log.

info

(Default) The Web Agent logs IPC semaphore-related error and informational messages to the Apache error log.

Example: Define the nete.stderr.loglevel parameter in trace.conf

In the following snippet from trace.conf, the nete.stderr.loglevel parameter is configured to restrict the Web Agent to log only IPC semaphore-related *error* messages to the Apache error log:

```
# CA Web Agent IPC logging levels
# nete.stderr.loglevel=error
```

Delete Certificates from Stronghold (Apache Agent Only)

Stronghold web servers write client certificates to a local, temporary file, which the Web Agent uses for certificate-based authentication. The Stronghold server uses this file to make information in the client certificate available for authentication. As users visit a website, these certificate files increase, taking up space on your server. You can configure the Web Agent to delete a certificate file after the Agent has finished using it.

To delete certificate files, set the DeleteCerts parameter to yes.

Oracle iPlanet Web Server Settings

Use any of the following settings to manage your SiteMinder Agent Oracle iPlanet servers:

- Restrict directory browsing (see page 265).
- Handle multiple AuthTrans functions (see page 266).
- Record the transaction ID in the Oracle iPlanet web server log (see page 266).

Restrict Directory Browsing on an Oracle iPlanet Web Server

To help ensure that users who try to browse the directories of a Oracle iPlanet web server are challenged by SiteMinder, you can set the following parameter:

DisableDirectoryList

Specifies whether the Web Agent allows a user to view or browse the contents of a directory without challenging them first. This occurs when *all* of the following conditions are true:

- The realm is set to protect a root resource (/)
- The default web page in the directory (such as index.html) is renamed or deleted.

Default: No

To restrict directory browsing on a Oracle iPlanet server

1. Add the DisableDirectoryList parameter to your Agent Configuration object or your local configuration file.
2. Set the value of the DisableDirectoryList parameter to yes.

   Directory browsing is restricted. SiteMinder challenges users who try to browse directories.

Handle Multiple AuthTrans Functions for Oracle iPlanet Web Servers

AuthTrans functions are directives that initialize the Oracle iPlanet web server. The Oracle iPlanet web server executes AuthTrans functions in the order that they are listed in the obj.conf file. The Oracle iPlanet server reads through the AuthTrans functions until it finds a function that returns a REQ_PROCEED command. Once a REQ_PROCEED command executes, no other AuthTrans functions are executed.

By default, SiteMinder is the first AuthTrans function and it returns a REQ_PROCEED. To allow other AuthTrans functions to execute, you need to add the EnableOtherAuthTrans parameter and set the value to yes.

The default value for this parameter is no. To enable multiple AuthTrans functions set the EnableOtherAuthTrans parameter to yes.

By adding this parameter, you permit the SiteMinder Web Agent to exist with other functions.

Be sure the SiteMinder Agent function is the first entry in the obj.conf file for the AuthTrans directive. The entry should read:

```
AuthTrans fn="SiteMinderAgent"
```

Record the Transaction ID in Oracle iPlanet Web Server Logs

Valid on Solaris

The Web Agent generates a unique transaction ID for each successful user authorization request. The Agent adds the ID to the HTTP header. The ID is also recorded in the following logs:

- Audit log
- Web server log (if the server is configured to log query strings)
- Policy Server log

You can track user activities for a given application using the transaction ID.

Note: For more information, see the Policy Server documentation.

The transaction ID appears in the log as a mock query parameter in the log that is appended to the end of an existing query string. The following example shows transaction ID (in bold) appended to a query string (which ends with STATE=MA):

```
172.24.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realm/index.html,
STATE=MA&SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1
```

If no query parameters are in the URL, the Agent adds the transaction ID at the end of the web server log entry. For example:

```
172.24.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realma/index.html,
SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1.
```

Note: Web Agents log user names and access information in native web server log files when users access resources.

You can record the SiteMinder transaction ID in the Oracle iPlanet web server logs.

Follow these steps:

1. Open the magnus.conf file.
2. Add the following header variable to the existing list of HTTP server variables that you want to log when the web server initializes:

   ```
   %Req->headers.SM_TRANSACTIONID%"
   ```

   Note: Enter the header variable in uppercase unless the value of the LowerCaseHTTP parameter is set to yes in your Agent Configuration Object or local configuration file.

   The following example shows the SMTRANSACTIONID header variable in bold at the end of an existing entry. However, you can place it anywhere in the list of variables.

   ```
   Init fn="flex-init" access="D:/iPlanet/server4/https-orion/logs/access"
   format.access="%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%] \"
   %Req->srvhdrs.clf-status% %Req-srvhdrs.content-length% %Req->headers.-
   SM_TRANSACTIONID%"
   ```

3. Restart the Oracle iPlanet Server to apply the change.

   The transaction ID appears in the Oracle iPlanet web server logs. The following example shows a web server log entry with the transaction ID in bold:

   ```
   11.22.33.44 - user1 [21/Nov/2003:16:12:24 -0500] "GET /Anon/index.html HTTP/1.0"
   200 748 3890b4b9-58f8-4a74df53-07f6-0002df88
   ```

More information:

Use Lower Case HTTP in Headers (for Oracle iPlanet, Apache, and Domino web servers) (see page 119)

Domino Web Server Settings

Domino servers sometimes require special SiteMinder agent parameters. These parameters are used only for Domino servers unless otherwise indicated. Use the topics in the following categories to protect your Domino resources:

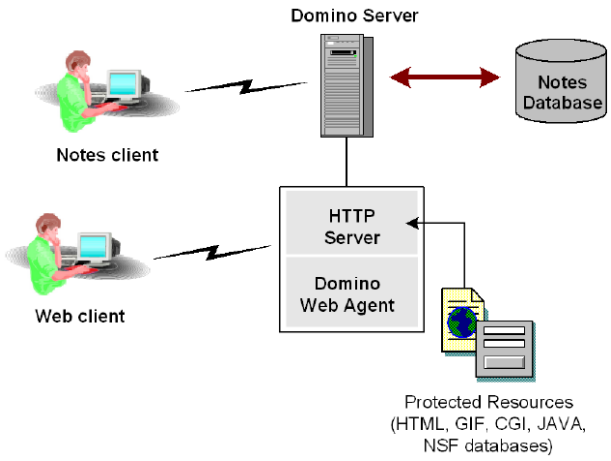
- For general information, see the following topics:
  - Domino agents overview (see page 269).
  - Domino URL syntax (see page 270).
  - Domino aliases (see page 271).
- For basic configuration information, see the following topics:
  - Configure the Domino agent (see page 272).
  - Configure Domino-specific agent functions (see page 272).
  - Specify user directories for Domino (see page 272).
  - Guidelines for creating policies on Domino servers (see page 273).
  - Configure policies for Domino (see page 273).
  - Create rules for Domino server resources (see page 274).
- For information about SiteMinder authentication, see the following topics:
  - Authenticate users with the Domino server (see page 276).
  - Authenticate as the Domino superuser (see page 277).
  - Authenticate as the actual user or default user (see page 278).
  - Modify the Domino default user and Domino superuser (see page 278).
  - Use the Encryptkey tool to set the Domino default user or superuser (see page 279).
  - Coordinate SiteMinder and Domino authentication.
  - Force SiteMinder to authenticate users (see page 280).
  - Use a SiteMinder header for authentication (see page 281).
  - Disable Domino session authentication (see page 281).
  - Use an anonymous SiteMinder authentication scheme with Domino (see page 282).
- For information about using SiteMinder forms credential collectors (FCCs), see the following topics:
  - Enable a Domino agent to collect credentials for authentication (see page 282).
  - Map URLs for FCC redirects with a Domino agent (see page 142).
  - Disable URL normalization (see page 283).

- For information about managing access to Lotus Notes documents, see the following topics:
  - Control access to Lotus Notes documents (see page 284).
  - Convert Lotus Notes document names (see page 285).
- For information about subjects not covered in the previous lists, see the following topics:
  - Configure full log out support for Domino agents (see page 286).
  - Use a Domino agent with a WebSphere application server. (see page 287)

Domino Agents Overview

The Domino Application Server is a messaging and Web application platform that offers secure access for Lotus Notes clients. The Domino Web Agent protects only the HTTP interface of the Domino Application Server, controlling access to HTML, JAVA, CGI, and other Web resources, such as Notes served over the web. It does not protect the Notes server.

The following illustration shows how the Domino Web Agent integrates with the Domino server.



Domino stores data in groups of Notes databases. Resources in a Notes database can be a variety of objects, such as documents, views, forms, and navigators. These objects can include text, video, graphics, and audio content.

Notes objects are opened using a URL. To make Notes objects available for the Web, Domino dynamically creates Web pages from the objects in the Notes database. In the case of database views, Domino also creates URL links to the documents in a view. The dynamic creation of pages from the Notes database provides users with the most current information.

Domino URL Syntax

Access to resources on a Domino server is based on the URL. Domino servers use a specific URL syntax.

Domino servers can interpret standard URLs, such as one shown in the following example:

```
http://www.example.com/index.html
```

Domino URL commands can use the following syntax:

`http://`*`host/database`*`.nsf/`*`Domino_object?Action_Argument`*

Host

Indicates the DNS entry or IP address of the server.

Database

Specifies the database file name with the path relative to the notes \data directory or the database Replica ID.

Domino_object

Specifies the object in the database, for example, a view, document, form, or navigator.

Action

Identifies the operation that performed on the Notes object. For example: ?OpenDatabase, ?OpenView, ?OpenDocument, ?OpenForm, ?ReadForm, ?EditDocument. If no action is specified in the URL, the default is used.

Default: ?Open.

Argument

Defines how the Domino server delivers an object. For example, if the action and argument is?OpenView&Expand=5, this argument specifies the number of rows displayed in an expanded format.

The following example shows a URL to access a view in a Notes database named financials.nsf:

```
http://www.example.com/financials.nsf/reports?OpenView
```

Domino Aliases

One of the Notes database conventions is to create aliases for objects. For example, the alias might identify a resource by its Notes ID or Replica ID instead of the object name. Using aliases makes programming easier for developers because the names of the Notes resources can change without requiring code changes.

The following Domino URLs access the same resource though the resource is identified by its aliases:

- http://www.domino.com/85255e01001356a8852554c20756?OpenView
- http://www.domino.com/85267E00075A80C/people?OpenView
- http://www.domino.com/__852567E00075A80C.nsf/people?OpenView

Regardless of how a resource is identified, the Domino Web Agent converts all Domino naming conventions into a standard URL based on the name of the database resource. This simplifies data entry into the SiteMinder policy store.

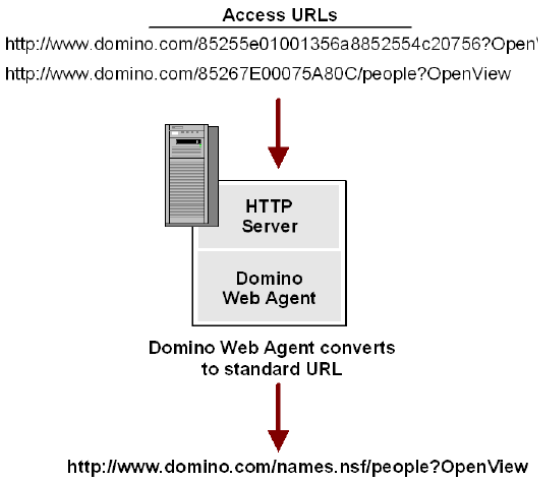
For example, the following Domino URLs are pointing to the people view in the names.nsf database. The database and view are referred to by Replica ID and Notes ID:

- http://www.domino.com/85255e01001356a8852554c20756?OpenView
- http://www.domino.com/85267E00075A80C/people?OpenView

The Domino Web Agent converts these URLs to a standard URL, as follows:

- http://www.domino.com/names.nsf/people?OpenView

The following illustration shows the conversion of aliases to a named object.



Configure the Domino Web Agent

The Domino Web Agent uses all the standard Web Agent settings to do the following:

- Configure the Web Agent to communicate with the Policy Server
- Add and remove Agent identities for virtual servers
- Modify Web Agent settings
- Configure single sign-on
- Configure error message logging

You can configure these centrally at the Policy Server or locally in the Agent configuration file.

In addition to the standard functions, there are Domino-specific parameters you can set.

More Information

Configure Domino-Specific Agent Functions (see page 272)

Configure Domino-Specific Agent Functions

In addition to the standard Web Agent settings, there are specific Domino configuration parameters that you can set only for the Domino Web Agent. These settings determine how Domino authenticates and authorizes a user with SiteMinder. You can configure these settings centrally in the Agent Configuration Object on the Policy Server or locally in the Agent configuration file on the web server.

Note: The Domino Web Agent does not support the auditing feature used to track user activity.

Specify User Directories for Domino

The Domino Directory is integrated with every Domino server. You can enable LDAP service for the Domino server so that Policy Server can use the Domino Directory to authenticate and authorize users. If you enable Domino's LDAP service, you do not need to configure a separate user directory for authentication.

To enable LDAP service, see your Domino Server documentation.

More information:

Contact CA Technologies (see page 3)

Guidelines for Creating Policies on Domino Servers

Use the following guidelines when creating SiteMinder policies for the Domino server:

- A user can open a form with a parent document to view default values for the form. The parent document is the original form that is used to create the document. To prevent unauthorized users from viewing default values for forms on which they do not have access, set the SkipDominoAuth parameter to no.
- If you replicate databases on the same computer, create a duplicate set of rules to protect each database.
- If the Domino Agent cannot associate an alias for a Notes document with a form, then each document requires its own rule for protection.
- The Domino server uses special identifiers in URL commands for certain database documents, for example, $DefaultView, $DefaultForm, $DefaultNav, and $SearchForm. The Domino agent converts these identifiers to a standard URL to access the document.

  For $defaultNav, the Domino Agent performs an action of ?OpenDatabase. You do not need to create additional rules for these types of identifiers.
- Aliases in the Notes database protect their associated resources. If no alias exists, the resource name or comment protects the associated resource.
- The Lotus Notes software allows multiple objects of different types to have the same name and alias. If you create a rule that uses a wildcard with the ?Open action, such as, ?Open*, be aware that this rule protects different types of resources that share an alias or name.
- Forms protect the documents that they create. The action that is used with the form is ?ReadForm.
- The Domino Agent protects files with the .nsf extension. Do not add this extension to the IgnoreExt parameter.

Configure Policies for Domino

The Domino server can represent the same Notes object in different ways. An object can be identified using the name, ReplicaID, UniversalID, and alias.

For the Domino Web Agent to communicate effectively with the Domino server, the Domino Agent processes access requests to Notes resources using only the object name. This enables the SiteMinder policy store to understand the entry.

Expressed as a URL, the access method to any resource would be:

http://host/database.nsf/resource_name?Open

Create Rules for Domino Server Resources

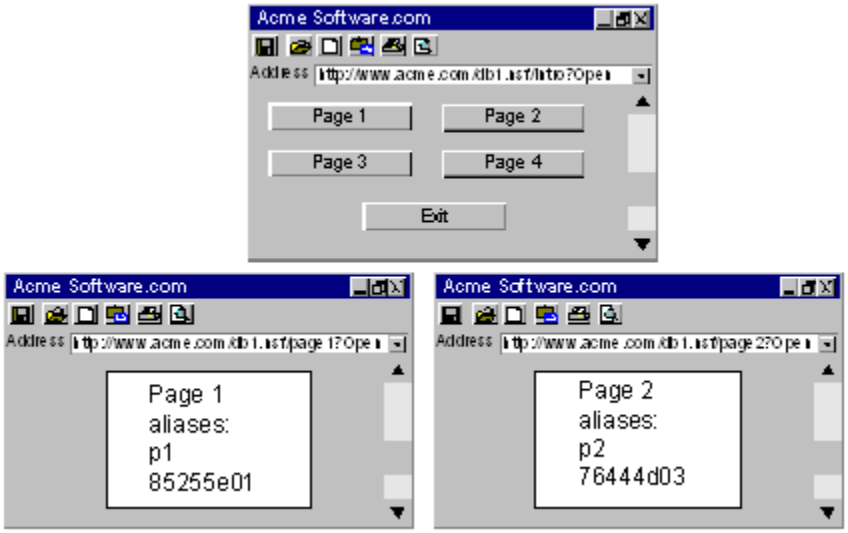
Actions for the Notes database resources should be considered when you create rules. Any resource not specified with an action will default to the action ?Open. The rules that are included in a SiteMinder policy must account for the default action, ?Open, and equivalent actions for ?Open, such as ?OpenDatabase, ?OpenView, ?OpenDocument, ?OpenFrameset.

The Domino Web Agent enables a policy administrator to create one rule for many aliases that point to the same resource. You only need one rule because the Domino Agent converts Domino's multiple representations of a resource into one URL. This function of the Domino Agent is important to consider when creating rules for SiteMinder policies.

You create realms and rules using the Administrative UI.

Note: For more information, see the Policy Server documentation.

In the following illustration, the URL is a link to Acme's Domino server, with a Notes database called db1.nsf. This database contains two files: page1 and page2.



Example 1: Protecting one document and all its aliases.

For access to page1 and all its aliases, you create only one rule for the realm db1.nsf. The Domino Agent is able to interpret all the different naming conventions and convert them to a one standard URL format.

For your realms and rules, do the following:

- When creating a realm you would specify a resource filter for the database where page1 resides. For example, to protect all files in the database you would configure the following:

  Resource filter: /db1.nsf/

  To protect not only page1 but all its aliases, you would configure the following:

  Resource filter: /db1.nsf/page1

- To create a rule that protects any action on page1, enter an asterisk (*) in the Resource field of the Rule Properties dialog box. For example:

  Resource: *

  This * wildcard indicates that any action, such as ?Open, ?EditDocument can be performed on page1 by the users that are bound to the policy.

Example 2: Protecting different documents in the same database.

To protect page2 in the db1.nsf database in addition to page1, you need to create a second rule.

Resource Filter: /db1.nsf/page2

Resource: *

Example 3: Protecting different actions on a single resource

To protect individual actions on a resource, for example, if you wanted only some users to perform the action ?EditDocument and all users to perform the action ?ReadForm, each action would require its own rule for each resource, as follows:

- Rule 1

  Resource Filter: /db1.nsf/page1

  Resource: ?OpenView

- Rule 2

  Resource Filter: /db1.nsf/page1

  Resource: ?EditDocument

You could also use one rule as follows:

Resource Filter: /db1.nsf/page

Resource: ?Open*

Note: In the Resource field, there is no forward slash (/) before ?Open.

Even if there are aliases for this resource, the one rule would protect the original page and all its aliases.

Instead of creating several rules for different actions, you could specify a single rule and use wildcards to cover all actions, for example:

Resource filter: /db1.nsf/page

Resource: ?Open*

With the rule, you are then protecting the resource:

http://www.acme.com/db1.nsf/page*?Open*

Note: If you want a rule to be literal, write a regular expression.

Authenticate Users with the Domino Server

The Domino server must authenticate and authorize users even if SiteMinder has already gone through this process. SiteMinder works with Domino's authentication process by providing the Domino server with a user identity that is also configured in the Domino Directory, which is the list of users and their privileges. The Domino server uses this identity to authenticate and authorize the user for access to database resources.

Note: A user name must be resolved unambiguously, or else the Domino Agent denies the authentication request. This may require some adjustments in your user directory.

The Domino Web Agent identifies the user to the Domino server as one of the following:

- Super user
- Actual user
- Default user

To determine which identity the Domino Web Agent uses when communicating with the Domino server, you configure the following parameters:

SkipDominoAuth

Determines which name to pass to the domino server for server authentication.

DominoSuperUser

Identifies a user who has access to all resources on the Domino server.

DominoDefaultUser

Identifies a user with default access to the Notes database, which means this person has general access privileges.

Note: You can configure the DominoSuperUser and DominoDefaultUser locally, in the Agent configuration file, or centrally, in the Agent Configuration Object. In the Agent configuration file, these settings have encrypted values. In the Agent Configuration Object, you have the choice of encrypting these values or leaving them in plain text.

More Information

Force SiteMinder to Authenticate Users (see page 280)
Authenticate as the Domino Super User (see page 277)
Authenticate as the Actual User or the Default User (see page 278)

Authenticate as the Domino Super User

A Domino Super User is a user who has access to all resources on the Domino server. If your Web site or portal is designed with SiteMinder in mind, you are securing resources and applications by implementing SiteMinder policies. As a result, the Domino server does not have to restrict user access based on its own security. In this case, users can be identified as the Super User for Domino's authentication purposes.

To identify the user as the Super User, you enable the SkipDominoAuth parameter and specify a value for the DominoSuperUser parameter. This action makes sure that SiteMinder and not Domino authenticates users. The user that you specify must also be in the Domino Directory.

Authenticate as the Actual User or the Default User

If a user is defined in the Domino Directory, Domino authenticates that user with their user name. However, if the user is not in the Domino Directory, and they have been authenticated by SiteMinder against another user directory, then the Domino Web Agent identifies that user to the Domino server as the DominoDefaultUser.

The default user has default access to the Notes database, which means this person should have general access privileges such as Domino's depositor, reader, or author level of access, configured in ACLs.

For the Domino Agent to use this value, set the SkipDominoAuth parameter to no.

There may be some Notes databases that do not require protection from SiteMinder. Resources that are not protected by SiteMinder are not authenticated as the default Domino user. Instead, the Domino server prompts users for their credentials (if anonymous access is disabled).

Modify the Domino Default User and the Domino Super User

To modify the DominoDefaultUser and DominoSuperUser parameters, do one of the following:

- Change it in the Agent Configuration Object, if configuring centrally

  You can modify the DominoDefaultUser and DominoSuperUser settings in the Agent Configuration Object. You can choose whether the values are encrypted or in plain text.

  Note: For more information, see the Policy Server documentation.

- Modify the parameters in the Agent configuration file using the encryptkey tool.

  In the Agent configuration file, the DominoDefaultUser and DominoSuperUser values must be encrypted. Consequently, you have to modify these values using the encryptkey tool.

  Important! Do not edit these settings directly in the Agent configuration file.

Use Encryptkey to Set the Domino Default or Super User

To set or change the value of DominoSuperUser or DominoDefaultUser in the Agent configuration file

1. Do one of the following:
   - UNIX: Navigate to the Domino Agent's bin directory. For example:

     /$HOME/ca/SiteMinder/Web Agent/bin
   - Windows: Open a command prompt window and navigate to the Domino Agent's Bin directory. For example:

     C:\Program Files\ca\SiteMinder Web Agent\Bin
2. Run the encryptkey tool, using the following arguments:
   - For DominoSuperUser:

     encryptkey -path *path_to_Agent_config_file*

     -dominoSuperUser *new_value*
   - For DominoDefaultUser:

     encryptkey -path *path_to_Agent_config_file*

     -dominoDefaultUser *new_value*

   For example:

   encryptkey -path "c:\program files\ca\SiteMinder Web Agent\Bin\Lotus Domino5\webagent.conf"

   -dominoSuperUser admin

   Note: The path to the Agent configuration file must contain the file name, such as, webagent.conf. Also, if any value in the path contains spaces, the entire path must be surrounded by quotation marks.

   Note: The encryptkey tool is not provided as a part of the SiteMinder Web Agent kit. However, the tool remains useful to Domino users who can manipulate it to generate encrypted DominoSuperUser settings for local configuration. You can contact Support to download a copy of this tool.

Force SiteMinder to Authenticate Users

To have SiteMinder (and not Domino) authenticate users, set the SkipDominoAuth parameter to yes.

With SkipDominoAuth set to yes and a Super User defined, SiteMinder first identifies and authorizes the user. The Domino Web Agent then identifies that user to the Domino Server as the Super User. As a Super User, the user has access to any resource on the Domino server, assuming the user has the appropriate ACLs.

You should also set SkipDominoAuth parameter to yes when users are not stored in the Domino Directory because Domino will not have an identity to use for authorization privileges.

If you set SkipDominoAuth to no, Domino authenticates users on its own using the actual user name or the default user name.

The following table shows how the setting of the SkipDominoAuth parameter affects how the user is identified.

SkipDominoAuth Value	Identified to the Domino Server As	Notes
yes	Super User	Super User must be defined in the Domino Directory
no	Actual User	User must be in the Domino Directory
no	Default User	User must be in the Domino Directory
no	Super User	The requested resource is automatically authorized, meaning that no authentication challenge will be presented to the user

More Information

Authenticate as the Actual User or the Default User (see page 278)

Use a SiteMinder Header for Authentication

The DominoUseHeaderForLogin and DominoLookUpHeaderForLogin parameters can be used to identify a Domino user for authentication.

DominoUseHeaderForLogin

Instructs the Domino Web Agent to pass the SiteMinder header value to the Domino Web Server. The Domino server uses the header data to identify a user in its user directory.

Set this parameter to a header name. For example, if you specify DominoUseHeaderForLogin="HTTP_SM_USER", the Web Agent passes the user's login name to the Domino server.

DominoLookUpHeaderForLogin

Instructs the Domino Web Agent to ask the Domino Web Server if the user requesting access to a resource is unique or ambiguous within the Domino user directory. This check is useful if a user named Jones tries accessing a resource and there are several users named Jones in the user directory. If this parameter is set to no, the Domino Web Agent does no checking with the Domino Web Server.

Default: Yes

Disable Domino Session Authentication

SiteMinder provides authentication and authorization functionality; therefore, the Domino session authentication feature is not needed. It should be disabled if the Web Agent is installed.

Under some conditions, having Domino session authentication enabled causes the user session to behave differently. This change in behavior does not affect security on a SiteMinder-enabled site. It reflects the intersection of SiteMinder and Domino session management rules.

Use an Anonymous SiteMinder Authentication Scheme with Domino

To use an anonymous SiteMinder authentication scheme with a Domino agent, set the following parameter:

DominoUserForAnonAuth

Specifies a value for anonymous users. This value is sent to the Domino server when users access Domino resources that are protected with an anonymous SiteMinder authentication scheme.

Default: No (anonymous authentication scheme *not* used)

Example: Anonymous (use with anonymous authentication schemes)

The previous parameter applies only when using an anonymous SiteMinder authentication scheme with Domino. Do *not* change its value for other authentication schemes or server types.

Enable a Domino Agent to Collect Credentials for Authentication

A credential collector is an application within the Web Agent, which gathers user credentials for forms, SSL, and Windows authentication schemes, and for single sign-on across multiple cookie domains. The credentials gathered by the credential collector are based on the type of authentication scheme configured for a particular group of protected resources.

For a Domino Web Agent to act as a credential collector, you have to configure various MIME types, represented as file extensions in the Agent configuration file.

Credential collectors are generally auto-authorized, that is, when you add a file extension to these parameters, they are, by default, included in the IgnoreExt parameter. Domino Server cannot correctly process URLs that include files with these extensions, so the Domino Agent has to ignore these files.

Note: For more information, see the Policy Server documentation.

Map URLs for FCC Redirects with a Domino Web Agent

To protect Domino view (.nsf) resources with a forms authentication scheme, map the URLs before they are redirected to the forms credential collector.

Follow these steps:

1. Set the value of the DominoNormalizeUrls parameter to yes.
2. Set the value of the DominoMapUrlForRedirect parameter to yes.

   Domino URLS are mapped before redirection to the FCC.

Disable URL Normailization

The process of URL normalization modifies URLs from a Domino representation to a URL format used by a typical web browser. The Domino Web Agent relies on the Domino web server APIs to normalize a Domino URL.

During the normalization process, the Domino Server APIs periodically return a URL with a carriage return (0x0D in hex) and/or a line feed character (0x0A in hex) added to the normalized URL. The addition of these characters appears to be related to specific Notes database (.nsf) files and access patterns within these files.

The following example shows a normalized URL with an added carriage return:

- URL:
  http://server.ca.com:80/agentrunner.nsf/be68f4545348400461332?OpenView
- URL is mapped to:
  http://server.ca.com:80/agentrunner.nsf/AgentContext?OpenView
- URL is normalized to:
  http://xxxxx.ca.com:port/agentrunner.nsf/0x0d/AgentContext?OpenView

If necessary, you can ensure that URLs with Domino resource IDs are not normalized with the following parameter:

DominoNormalizeUrls

Specifies if the SiteMinder Web Agent converts Domino URLs to a URL-friendly name before redirecting them to a Forms Credential Collector.

The MapUrlsForRedirect parameter must also be set to yes for the Domino URLs to be converted.

If the DominoNormalizeUrls parameter is set to no, URLs will *not* be normalized, even if the MapUrlsForRedirect parameter is set to yes.

Important! If you set the DominoNormalizeUrls parameter to no, you cannot protect individual documents within a Notes database; you can only protect the entire database or subdirectories of the Domino Web server.

Default: Yes

To turn off normalization and ensure that URLs are not altered, set the DominoNormalizeUrls parameter to no.

Control Access to Lotus Notes Documents

The Web Agent offers a finer level of granularity for protecting Lotus Notes documents on Domino. The folloiwng parameter controls this protection:

DominoLegacyDocumentSupport

Specifies how a Web Agent handles user requests for protected Lotus Notes documents in a Domino environment. Setting this parameter to yes grants users ReadForm permission only for the requested document.

Default: No

Use the DominoLegacyDocumentSupport parameter to configure the Web Agent to process user-requested actions when accessing Notes documents. This offers a finer granularity of protection on Domino.

Notes documents do not have names. They are saved to the database with a reference to the form used to create them. When a user requests a Notes document, the Domino Web Agent finds the form for that document by converting the request into a URL. This URL includes the original Domino action. If no form is found, then nothing is used.

For example:

"http://server.domain.com/db.nsf?OpenDocument"

in the URL To ensure that the Web Agent performs the user-requested Domino action on the document that is specified in the URL, such as ?OpenDocument or ?EditDocument, set the DominoLegacyDocumentSupport parameter to no.

For example, if the URL request is:

http://www.dominoserver.com/names.nsf/9348730948938987785784395880982039857983 49?EditDocument

The Domino Agent converts the preceding URL to:

http://www.dominoserver.com/names.nsf/Person?EditDocument

where Person is the name of the form used to create the document identified by the NotesID in the original URL.

To force the Domino Web Agent revert back to its pre-4.6 operation for accessing Notes documents, which means that only the action ?ReadForm is permitted, set this parameter to yes. With the legacy document support enabled, the Domino Agent would convert the URL in the previous example to:

http://www.dominoserver.com/names.nsf/Person?ReadForm

Convert Notes Document Names

Unlike views and forms, Notes documents do not have names; they are saved to the database with a reference to the form that was used to create the document. If a user is trying to access a document and the Domino Web Agent cannot convert it to a readable name, the Agent uses the name of the form that generated the document to create a URL. This applies only to documents. If there is no original form, the Agent uses the embedded form. If neither apply, the document is protected using the Domino identifier $defaultForm.

For example, if the incoming URL is:

http://www.domino.com/names.nsf/8567489d60034we50938450098?OpenDocument

The Agent uses:

http://www.domino.com/names.nsf/Person?ReadForm

In this example, Person is the name of the document.

Configure Full Logoff Support for Domino Agents

The full log-out feature uses a custom log-out page that you create with the following parameter:

LogOffUri

Enables the full log-out function by specifying the URI of a custom web page. This custom web page appears to users after they are successfully logged off. Configure this page so that it cannot be stored in a browser cache. Otherwise, a browser could possibly display a log-out page from its cache without logging the user off. If this situation happens, unauthorized users could possibly have an opportunity to assume control of a session.

Note: When the CookiePath parameter is set, the value of the LogOffUri parameter must point to the same cookie path. For example, if the value of your CookiePath parameter is set to example.com, then your LogOffUri must point to example.com/logoff.html

Default: (all agents *except* the CA SiteMinder Agent for SharePoint r12.0.3.0) No default

Limits: Multiple URI values permitted. Do *not* use a fully qualified URL.Use a *relative* URI.

Example:(all agents *except* the CA SiteMinder Agent for SharePoint r12.0.3.0) /Web pages/logoff.html

Follow these steps:

1. Create a custom HTTP application that logs the user off. For example, add an Exit or Sign Off button that redirects the user to a URL you specify.
2. Set up the log-out page so it cannot be cached in web browsers. This setting increases security because the page is always served from the web server, and not the cache of the browser. For example, for HTML pages, you can add the following meta tags to the page:

   ```
   <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
   ```

   ```
   <META HTTP-EQUIV="Expires" CONTENT="-1">
   ```

   Important! Some web browsers do *not* support meta tags. Use a cache-control HTTP header instead.
3. Configure the LogOffUri parameter with the following steps:
   a. Delete the pound sign (#), if necessary.
   b. Enter the URI of the custom HTTP file that will log the user off. Do *not* use a fully qualified URL.

      The full log-out feature is configured.

More Information

How Full Logoff Works (see page 201)

Use a Domino Agent with a WebSphere Application Server

A Domino web server acts as the front end to a WebSphere Application Server by providing a filter plug-in that intercepts requests before forwarding them to the WebSphere server.

Force Domino Server to Authenticate Unprotected SiteMinder Resources

Suppose you have resources on your Domino server that you not want to protect with SiteMinder. You can still protect those resources with your Domino server instead. To protect these resources, set the following parameter:

UseDominoUserForUnprotected

Specifies if the Domino server authenticates requests with a Domino user for resources that *only* the Domino server (*not* SiteMinder) protects.

If the value of this parameter is yes, the agent passes the Domino user to the Domino server. The Domino server authenticates the user. If the value of this parameter is no (or the parameter is disabled), the agent does *not* pass the Domino user to the Domino server. The Domino server does not authenticate the user.

Default: Disabled

Follow these steps:

1. Locate the previous parameter.
2. Remove the # (comment) character in front of the parameter.
3. Change the value of the parameter to yes.

Backward Compatibility Settings

Use any of the following settings to manage the backward comparability of your SiteMinder Agent:

- Accommodate legacy URL encoding (see page 288).
- Determine how content types are transferred in POST requests (see page 263).
- Accommodate testing tools that do not send HOST headers (see page 289).

Accommodate Legacy URL Encoding

The legacy URL encoding used by CA uses dollar sign ($) characters. If the dollar signs cause problems, you can make the Web Agent use hyphen (-) characters instead of dollar signs with the following parameter:

LegacyEncoding

Forces the Web Agent to replace any dollar sign ($) characters in legacy URLs with a hyphen (-). This also ensures backwards comparability with MSR, Password Services, and DMS. When this parameter is set to no, a Web Agent converts the string SM to -SM-. When this parameter is set to yes, the Web Agent does *not* convert the dollar sign ($) character.

Default: (Framework Agents) No

Default: (Traditional Agents) Yes

To encode legacy URLs using hyphens instead of dollar signs, set the value of the LegacyEncoding parameter to no.

Choose How Content Types are Transferred in POST Requests

If you are using an Apache web server, you can control how content is transferred to the server during POST requests with the following parameter:

LegacyStreamingBehavior

Specifies how content will be transferred to the server during POST requests. When the value of this parameter is set to yes, all content types are streamed, *except* for the following:

- text/xml
- application/x-www-form-urlencoded

When the value of this parameter is set to no, all content types are spooled.

Default: No

To stream most types of content in POST requests, change the value of the LegacyStreamingBehavior parameter to yes.

Accommodate Testing Tools that do not send HOST Headers

The SiteMinder Web Agent uses the value of the HOST header in an HTTP request to determine the following settings:

- Agent name
- Server name
- Server IP address

SiteMinder Web Agents only accept HTTP version 1.1 requests, because HTTP versions 0.9 and 1.0 do not use HOST headers. This poses problems for some testing tools that do not send HOST headers, because the Web Agent rejects those requests.

SiteMinder r12.5 supports a new Agent Configuration parameter that lets you define a HOST header value. The Web Agent uses this value in any request that does *not* contain a HOST header.

To accommodate testing tools that do not send HOST headers

1. Open one of the following items:
   - If you are using Central Configuration, open your Agent Configuration Object.
   - If you are using Local Configuration, open your LocalConfig.conf file.
2. Add the following parameter:

   DefaultHostName

   Defines a value for the HOST header. Add this parameter to your Agent Configuration Object or LocalConfig.conf file to use a testing or performance tool that sends HTTP version 0.9 or version 1.0 requests (without HOST headers). If this parameter is not set, the Web Agent *only* accepts HTTP 1.1 requests.

   Default: None (blank)

   Example: webserver.example.com
3. Set the value of the previous parameter to the host name you want. See the previous example.
4. Save and close *one* of the following items:
   - If you are using Central Configuration, save and close your Agent Configuration Object.
   - If you are using Local Configuration, save and close your LocalConfig.conf file.

   The Web Agent substitutes the DefaultHostName value for any HTTP request without a HOST header.

Agent Setting for Federation Domains

If SiteMinder is acting as a legacy federation SP, you can configure the Identity Provider Discovery (IPD) profile for SAML 2.0 transactions. IPD enables a user to select which IdP generates an assertion for an authentication request.

During the discovery process, you can prevent a user from being redirected to a malicious web site. Configure the Web Agent to validate the domain of the IdP that satisfies the authentication request.

To enable the validation process, set the value of the following parameter:

ValidFedTargetDomain

(Federation only–SAML 2.0). Lists all valid domains for your federated environment when implementing Identity Provider Discovery.

When the SiteMinder Identity Provider Discovery (IPD) Service receives a request, it examines the IPDTarget query parameter in the request. This query parameter lists a URL where the Discovery Service must redirect to after it processes the request. For an IdP, the IPDTarget is the SAML 2.0 Single Sign-on service. For an SP, the target is the requesting application that wants to use the common domain cookie.

Federation Web Services compares the domain of the IPDTarget URL to the list of domains specified for the ValidFedTargetDomain parameter. If the URL domain matches one of the configured domains in the ValidFedTargetDomain, the IPD Service redirects the user to the designated URL in the IPDTarget parameter. This redirect is to a URL at the SP.

If there is no domain match, the IPD Service denies the user request and they receive a 403 Forbidden in the browser. Additionally, errors are reported in the FWS trace log and the affwebservices log. These messages indicate that the domain of the IPDTarget is not defined as a valid federation target domain.

If you do not configure the ValidFedTargetDomain setting, no validation is done and the user is redirected to the target URL.

Limits: Valid domains within the federated network

Default: No default

Specify a valid domain in the ValidFedTargetDomain parameter. This setting is a multi-value parameter, so you can enter multiple domains.

If you are modifying a local configuration file, list the domains separately, for example:

validfedtargetdomain=".examplesite.com"

validfedtargetdomain=".abccompany.com"

For more information about the Identity Provider Discovery profile, see the *Federation Security Services Guide*.

Chapter 17: Performance

Set a Time-out for Saved Credentials

When a user chooses to have credentials saved, the Policy Server instructs the Web Agent to create a persistent cookie containing the user's credentials. The cookie allows Web Agents to authenticate a user based on the credentials saved in the cookie, instead of challenging the user to authenticate. You can control how long the persistent cookie remains with the following parameter:

SaveCredsTimeout

Specifies the number of hours that a persistent cookie containing the user credentials will be saved. During this time interval, the Web Agent authenticates the user with the data stored in the cookie. After this time interval expires, the cookie is removed and the Web Agent challenges the user again.

Default: 720 (30 days)

To set a timeout for saved credentials, enter the number of hours you want in the SaveCredsTimeout parameter.

Note: For more information, see the Policy Server documentation.

Web Agent Caches

The Web Agent stores user session and resource information in cache memory. This technique improves the Web Agent efficiency because the Web Agent does not have to retrieve information from the Policy Server each time a user requests access.

By configuring the cache settings, you can manage how this information is stored. The number of entries in the cache determines the size of the cache. The total number of entries in each cache cannot exceed the maximum cache size specified.

Note: Restart the Web Server for changes in the Web Agent cache settings to take effect.

The following guidelines apply to cache management:

- When a cache is full, new entries replace the least recently used entries.
- For the resource cache, entries are removed when the value of the ResourceCacheTimeout parameter is reached.
- For the user session cache, entries are removed based on the session timeout values that you set for each realm.

SiteMinder empties cached resource information when you modify a policy. You can also empty the user and resource caches manually from the Administrative UI.

Note: For more information, see the Policy Server documentation.

Use the following parameters to manage the caches of your agent:

- Cache anonymous users (see page 293).
- Set the size of the maximum resource cache (see page 294).
- Set the size of the maximum user session cache (see page 295).
- Control how long resource entries remain cached (see page 296).
- Disable the resource cache (see page 296).

Cache Anonymous Users

You can configure the Web Agent to store anonymous user information in a cache with the following parameter:

CacheAnonymous

Specifies if the Web Agent caches anonymous user information. You may want to set this parameter in any of the following situations:

- If your web site gets mostly anonymous users and you want to store their session information.
- If your web site gets a mix of registered and anonymous users.

  You may want to disable this parameter to keep the anonymous user information from filling the cache and leaving no room for registered users.

Default: No

To store anonymous user information in cache, set the value of the CacheAnonymous parameter to yes.

Set the Maximum Resource Cache Size

You can set a maximum on the number of resource cache entries, such as Web pages, that the Web Agent tracks with the following parameter:

MaxResourceCacheSize

Specifies the maximum number of entries that the Web Agent keeps in its resource cache. An entry contains the following information:

- A Policy Server response about whether a resource is protected
- Any additional attributes returned with the response

When the maximum is reached, new resource records replace the oldest resource records.

If you set this value to a high number, be sure that sufficient system memory is available.

If you are viewing Web Agent statistics using the OneView Monitor, you may notice that the value shown for the ResourceCacheCount is greater than the value you specified for the MaxResourceCacheSize parameter. This is not an error. The Web Agent uses the MaxResourceCacheSize parameter as a guideline and the values may at times differ because the MaxResourceCacheSize parameter represents the maximum number of average-sized entries in the resource cache. The actual cache entries are most likely larger or smaller than the pre-determined average size; therefore, the effective maximum number of entries may be more or less than the value specified.

Note: For Web Agents that use shared memory, such as the framework Agents, the cache is pre-allocated to a constant size based on the MaxResourceCacheSize value and will not grow.

Default: (Domino web servers) 1000

Default: (IIS and Sun Java System web servers) 700

Default: (Apache web servers) 750

To set the maximum resource cache size

1. Set the value of the MaxResourceCacheSize parameter to the maximum number of resources you want.
2. For framework agents, you must restart the web server to apply the change.

   The maximum resource cache size is changed.

Set the Maximum User Session Cache Size

You can set a maximum for the number of users the Agent maintains in the session cache with the following parameter:

MaxSessionCacheSize

Specifies the maximum number of users the Agent maintains in its session cache. The session cache stores the session IDs of users who authenticate successfully. Authenticated users accessing other resources within the realm during a session, are authenticated using the session cache instead of the Policy Server. When the maximum number is reached, the Agent replaces the oldest user records with new user records.

Base the value of this parameter on the number of users that you expect to access and use resources for a sustained period. If you set this value to a high number, verify that sufficient system memory is available.

Note: Regardless of the cache size, all entries in the session cache of the Web Agent *expire automatically* after one hour.

Default: (Domino web servers) 1000

Default: (IIS and Oracle iPlanet web servers) 700

Default: (Apache web servers) 750

To set a maximum size of the user session cache

1. Set the value of the MaxSessionCacheSize parameter to the maximum number of users you want.
2. For framework agents, you must restart the web server to apply the change.

   The maximum user session cache size is changed.

Control How Long Resource Enteries Remain Cached

You can change the amount of time that resource entries will remain in the cache with the following parameter:

ResourceCacheTimeout

Specifies the number of seconds that resource entries remain in the cache. If a user tries to access a protected resource after the time interval has been exceeded, the Web Agent removes the cached entries and contacts the Policy server.

Default: 600 (10 minutes)

Note: If you change the value of this parameter, you must restart the web server to apply the change.

To change how long the resource entries remain cached, set the ResourceCacheTimeout parameter to the number of seconds you want.

Disable the Resource Cache

If you are protecting an application that uses dynamic, unique URLs, you may want to disable the resource cache. Since the URLs used by the application are unique, then they will not be read from the cache.

To disable the resource cache, change the value of the MaxResourceCacheSize to zero.

Monitoring Web Agents

Use any of the following methods to monitor the performance of your agents:

- Monitor agents with the OneView monitor (see page 297).
- Use CA Wily Introscope to monitor agents (see page 297).

More information:

How to Manage Web Agent and Policy Server Communication (see page 53)

Monitor Web Agents with the OneView Monitor

The SiteMinder OneView monitor reports cache statistics and other information to the Policy Server, which administrators can use to analyze and fine-tune the Web Agent. You control the SiteMinder OneView monitor with the following parameter:

EnableMonitoring

Specifies whether the agent sends monitoring information to the Policy Server.

Default: Yes.

To have the Web Agent use the SiteMinder OneView Monitor, set the EnableMonitoring parameter to yes.

Note: For more information, see the Policy Server documentation.

Use CA Wily Introscope to Monitor Web Agents

If you are already using CA Wily Introscope in your organization, you can monitor the health of your SiteMinder Web Agents with the following parameter:

EnableIntroscopeApiSupport

Collects information about the agent and sends it to CA Introscope® using a plug-in. This parameter uses the following settings:

- When set to yes, the plug-in calls an API to collect the data.
- When set to no, the plug-in creates an HTTP header with the data.
- When set to both, the plug-in calls the API to collect the data *and* creates an HTTP header with the data.
- When set to none, data not collected.

Default: No.

Limits: Yes, Both, No, None.

Example: (HTTP header) sm-wa-perf-counters = *server_name*.example.com:6180,86117203,86118343,1,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1125,0,15,1,1,750,750,

To use CA Wily Introscope to monitor the health of your Web Agents, set the value of the EnableIntroscopeApiSupport parameter to *one* of the following:

- Yes
- Both
- No

Ignore Unprotected Resources

You can improve the performance of SiteMinder by ignoring requests for resources that you do *not* want to protect. The following parameters are available:

- Reduce overhead by ignoring certain file extensions (see page 299).
- Specify which virtual servers the agent ignores (see page 130).
- Ignore query data in URLs (see page 302).
- Allow unrestricted access to URIs (see page 303).

Reduce Overhead by Ignoring File Extensions of Unprotected Resources

You can reduce SiteMinder overhead by instructing the Web Agent to ignore requests for certain types of resources with the following parameter:

IgnoreExt

Specifies the types of resources for which the Web Agent passes requests to the web server without checking SiteMinder policies.The Web Agent allows access to the items specified by this parameter even if they exist in a realm that is protected by a SiteMinder policy.

Requests for resources that meet either of the following conditions may be ignored:

- The resource ends in one of the extensions that you configure the Web Agent to ignore.
- The URI of the protected resource contains a single period (.).

  For example, if a URI for a requested resource is /my.dir/ the Web Agent passes the request directly to the web server.

Default: .class, .gif, .jpg, .jpeg, .png, .fcc, .scc, .sfcc, .ccc, .ntc

Important! Use caution when setting the IgnoreExt parameter. There are some security issues that you may want to consider.

By default, the Agent does *not* ignore requests for resources that contain two or more periods separated by a slash (/). Web Agents handle requests for resources using the process shown in the following example:

1. The .gif extension is added to the IgnoreExt parameter. Requests for resources with the .gif extension are be ignored by the Web Agent.
2. A request is made for the following URI:

   /dir1/app.pl/file1.gif,
3. The Web Agent checks /dir1/app.pl/file1.gif against the policy server because some web servers will execute /dir1/app.pl as an application instead of serving the file1.gif resource.

   Granting access to /dir1/app.pl/file1.gif without consulting the web server may have caused a security breach.

To reduce overhead by ignoring the file extensions of unprotected resources, add the extensions of the resources you want to ignore to the value IgnoreExt parameter.

Specify Virtual Servers for the Web Agent to Ignore

If a web server at your site supports several virtual servers, there may be resources on these virtual servers that you do not want to protect with the Web Agent. To simplify how the Web Agent distinguishes which portions of a web server's content it protects, use the following parameter:

IgnoreHost

Specifies the fully qualified domain names of any virtual servers that you want the web Agent to ignore. Resources on such virtual servers will be auto-authorized, and the Web Agent always grants access to them regardless of which client makes the request. The authorization decision is based on the configuration of the Web Agent instead of being based on a policy.

The list of ignored hosts is checked first before any other auto-authorization checks, such as the IgnoreExt and IgnoreURL settings. Therefore, the double-dot rule will not trigger an authorization call to the Policy Server for resources on an ignored host but would not be ignored by extension.

The host portion of the URL entries for the IgnoreHost parameter must exactly match what the Web Agent reads for the host header of the requested resource.

Note: This value is case-sensitive.

If the URL uses a specific port, then the port must specified.

For centrally-managed agents, use a multi-value parameter in the Agent Configuration Object to represent several servers. For agents configured with a local configuration file, list each host on a separate line in the file.

Example: (URL shown with port specified)

IgnoreHost="myserver.example.org:8080"

Example: (local configuration file)

IgnoreHost="my.host.com"

IgnoreHost="your.host.com"

Default: No default

To specify virtual servers for the Web Agent to Ignore, do either of the following tasks:

- For central configuration, add the servers you want to ignore to your agent configuration object. For more than one server, use the multi-value setting for the parameter.
- For local configuration, add a separate line for each server in the local configuration file.

Resources using the specified URLs are ignored by the Web Agent and access to those resources is granted automatically.

More Information

Handle Complex URIs (see page 86)

Ignore Query Data in a URL

The IgnoreQueryData parameter affects the way Web Agents treat URLs. If you do not want the Web Agent cache the entire URL and send the URIs with their query strings to the Policy Server for rule processing, you improve performance with the following parameter:

IgnoreQueryData

Specifies whether the Web Agent will cache the entire URL (including the query strings) and send the entire URI to the Policy Server for rule processing. A full URL string contains a URI, a hook (?), and some query data, as shown in the following example:

URI?*query_data*

URLs that have been the subjects of requests are cached by default. Subsequent requests search the cache for a match. If requests for the same URI contain different query data, the match fails. Ignoring the query data improves performance.

When the IgnoreQueryData parameter is set to yes, the following occurs:

- The URL is truncated at the hook. Only the URI is cached and sent to the Policy Server. The query data is maintained elsewhere, for the purpose of maintaining the proper state for redirects.
- Only the part before the hook is sent to the Policy Server for rule processing.
- Both URIs in the following example are handled as the same resource:

  /myapp?data=1

  /myapp?data=2

When the IgnoreQueryData parameter is set to no, the following occurs:

- The entire URL is cached.
- The entire URI is sent to the Policy Server for rule processing.
- The URIs in the following example are handled as different resources:

  /myapp?data=1

  /myapp?data=2

Default: No

To have the Web Agent send only URIs to the Policy Server for processing, set the value of the IgnoreQueryData parameter to yes.

Important! Do not enable this setting if you have policies which depend on URL query data.

Allow Un-restricted Access to URIs

If you have URIs that you do not want to protect with SiteMinder, you can direct the Web Agent to ignore and allow un-restricted access to those URIs by setting the following parameter:

IgnoreUrl

Specifies a URI within a URL that will not be protected. Users attempting to access the resource associated with the URI will not be challenged. The Web Agent ignores the URI portion of the string after three forward slashes. For example, if you set this parameter to the following value:

```
http://www.example.com/directory
```

The Web Agent ignores the following URI:

```
directory
```

The Web Agent ignores the specified URI wherever it occurs, even if it is under a different domain. For example, the Web Agent ignores the URI shown previously in all of the following URLs:

```
http://www.example.com/directory
http://www.example.net/directory
http://www.example.org/directory
```

Note: This value is case-sensitive.

Default: No default.

Example: (multiple URIs in local configuration file)

IgnoreUrl="http://www.example.com/directory"

IgnoreUrl="http://www.example.com/directory2"

Example: (using a URI only, without specifying a domain)

IgnoreUrl="/*resource*/"

To allow un-restricted access to URIs, do either of the following tasks:

- For central configuration, add the fully qualified domain names with the URIs that you want to ignore to your agent configuration object. For more than one URI, use the multi-value setting for the parameter.
- For local configuration, add a separate line for each fully qualified domain name and URI in the local configuration file.

Resources using the specified URIs are ignored by the Web Agent and access to those resources is granted automatically.

Chapter 18: Logging and Tracing

This section contains the following topics:

Logs of Start-up Events (see page 305)
Error Logs and Trace Logs (see page 305)
How to Set Up Trace Logging (see page 311)

Logs of Start-up Events

To assist in debugging, startup events are recorded in a log. Each message may provide clues about the problem. These logs are stored in the following locations:

- On Windows systems, these events are recorded in the Windows Application Event log.
- On UNIX systems, these events are sent to STDERR. Apache servers map STDERR to the Apache error_log file, so these events are also recorded in that log.

Error Logs and Trace Logs

You can use the Web Agent logging function to monitor the performance of the Web Agent and its communication with the Policy Server. The logging feature provides accurate and comprehensive information about the operation of SiteMinder processes to analyze performance and troubleshoot issues.

A log is a record of events that occur during program execution. A log consists of a series of log messages, each one describing some event that occurred during program execution. Log messages are written to log files.

Note: IIS Agents create log files only after the first user request is submitted. Apache 2.0 Web Agents create log files when the Apache server starts.

The Web Agent uses the following log files:

Error log

Contains program and operational-level errors. One example is when the Web Agent cannot communicate with Policy Server. The level of detail output in this log cannot be customized. Error logs contain the following types of messages:

Error messages

Contain program-level errors, which indicate incorrect or abnormal program behavior, or an inability to function as expected due to some external problem, such as a network failure. There are also operational-level errors. This type of error is a failure that prevents the operation from succeeding, such as opening a file or authenticating a user.

Informational messages

Contain messages for the user or administrator that some event has occurred; that is, that a server has started or stopped, or that some action has been taken.

Warning messages

Contain warnings for the user or administrator of some condition or event that is unusual or indicative of a potential problem. This does not necessarily mean there is anything wrong.

Trace log

Contains detailed warning and informational messages, which you can configure. Examples include trace messages and flow state messages. This file also includes data such as header details and cookie variables. Trace logs contain the following messages:

Trace messages

Provide detailed information about program operation for tracing and/or debugging purposes. Trace messages are ordinarily turned off during normal operation. In contrast to informational, warning, and error messages, trace messages are embedded in the source code and can not easily be localized. Moreover, trace messages may include significant data in addition to the message itself; for example, the name of the current user or realm.

You specify the location of both the error and trace log files when you configure the Web Agent. Use the error and trace logs to help solve any issues that may prevent the Web Agent from operating properly.

Note: For Agents on Windows platforms, set the EnableWebAgent parameter to yes to ensure that the Web Agent log gets created. If you leave EnableWebAgent set to no (the default) and set the logging parameters, the Agent log gets created only for Agents on UNIX platforms.

More Information

Set Up and Enable Error Logging (see page 308)
Configure Trace Logging (see page 312)

Parameter Values Shown in Log Files

Web Agents list configuration parameters and their values in the Web Agent error log file, but there are differences between the ways that Traditional and Framework agents do this.

Framework agents record the configuration parameters and their values in the log file exactly as you entered them in the Agent Configuration Object or the local configuration file. All of the parameters, including those which may contain an incorrect value, are recorded in the log file.

Traditional agents process the parameter values before recording them. If the parameter has a proper value, the parameter and its value are recorded in the log file. Parameters with incorrect values are *not* recorded in the log file.

Set Up and Enable Error Logging

Error logs require the following settings:

- Logging is enabled.
- A location for the log file is specified.

The parameters that enable error logging and determine options such as appending log data are defined in a local configuration file or an Agent Configuration Object at the Policy Server.

Agents that are installed on an IIS or Apache web servers do not support dynamic configuration of log parameters that are set locally in a local configuration file. The changes take effect when the Agent is restarts. However, these log settings can be stored and updated dynamically in an agent configuration object at the Policy Server.

Note: IIS Agents create log files only after the first user request is submitted. Apache 2.0 Web Agents create log files when the Apache server starts.

Follow these steps:

1. If you do not have a log file already, create a log file and any related directories.
2. Set the value of the LogFile parameter to yes.

   Note: Setting the value of this parameter to yes in a local configuration file of a web server overrides any of the logging settings that are defined on the Policy Server. For example, suppose that the value of this parameter is set to yes in a LocalConfig.conf file. The agent creates log files even though the value of the AllowLocalConfig parameter in the corresponding agent configuration object is set to no. You can also set the related logging parameters in the LocalConfig.conf file also to override any other settings in the agent configuration object.
3. Specify the full path to the error file, including the file name, in any of the following parameters:

   LogFileName

   Specifies the full path (including the file name) of the log file.

   Default: No

   Example: (Windows) *web_agent_home*\log\WebAgent.log

   Example: (UNIX/LInux) /export/iPlanet/servers/https-jsmith/logs/WebAgent.log

   LogFileName32

   Specifies the full path of a log file for a SiteMinder Web Agent for IIS (on 64-bit Windows operating environments protecting 32-bit applications). The 32-bit applications run in Wow64 mode on the 64-bit Windows operating environment. If logging is enabled but this parameter is not set, the Web Agent for IIS appends _32 to the log file name.

Default: No

Limits: For Windows 64-bit operating environments only. Specify the file name at the end of the path.

Example: (Windows 64-bit operating environments using Wow64 mode) *web_agent_home*\log\WebAgent32.log.

4. (Optional) Set the following parameters (in the Agent Configuration Object on the Policy Server or in the local configuration file):

LogAppend

Adds new log information to the end of an existing log file. When this parameter is set to no, the entire log file is rewritten each time logging is invoked.

Default: No

LogFileSize

Specifies the size limit of the log file in megabytes. When the current log file reaches this limit, a new log file is created. The new log file uses one of the following naming conventions:

- For framework agents, the new log file has a sequence number that is appended to the original name. For example, a log file named myfile.log is renamed to myfile.log.1 when the size limit is reached.
- For traditional agents, the new log files are named by appending the date and timestamp to the original name. For example, a log file named myfile.log, is renamed to myfile.log.09-18-2003-16-07-07 when the size limit is reached.

Archive or remove the old files manually.

Default: 0 (no rollover)

Example: 80

LogLocalTime

Specifies whether the logs use Greenwich Mean Time (GMT) or local time. To use GMT, change this setting to no. If this parameter does not exist, the default setting is used.

Default: Yes

If you use a local configuration file, your settings resemble the following example:

```
LogFile="yes"
LogFileName="/export/iPlanet/servers/https-myserver/logs/errors.log"
LogAppend="no"
LogFileSize="80"
LogLocalTime="yes"
```

Error logging is enabled.

Enable Transport Layer Interface (TLI) Logging

When you want to examine the connections between the agent and the Policy Server, enable transport layer interface logging.

To enable TLI logging

1. Add the following environment variable to your web server.

   `SM_TLI_LOG_FILE`

2. Specify a directory and log file name for the value of the variable, as shown in the following example:

   directory_name/log_file_name`.log`

3. Verify that your agent is enabled.

4. Restart your web server.

   TLI logging is enabled.

Limit the Number of Log Files Saved

You can limit the number of log files that an agent keeps. For example, if you want to save disk space on the system that stores your agent logs, you can limit the number of log files using the following parameter:

LogFilesToKeep

Specifies the number of agent log files that are kept. New log files are created in the following situations:

- When the agent starts.
- When the size limit of the log file (specified by the value of the LogFileSize parameter) is reached.

Changing the value of this parameter does *not* automatically delete any existing logs files which exceed the number that you want to keep. For example, If your system has 500 log files stored, and you decide to keep only 50 of those files, the agent does *not* delete the other 450 files.

Setting the value of this parameter to zero retains all the log files.

Default: 0

Follow these steps:

1. Archive or delete any existing log files from your system.
2. Set the value of the LogAppend parameter to no.
3. Change the value of the LogFilesToKeep parameter to the number of log files that you want to keep.

How to Set Up Trace Logging

To set up trace logging, use the following process:

1. Set up and Enable Trace logging.
2. Determine what you want to record in the trace log by reviewing the following lists:
   - Trace Log Components and Subcomponents
   - Trace Message Data Fields
   - Data Field Filters
3. Duplicate the default Trace Configuration File.
4. Modify the duplicate file to include the items you want to record.
5. Restart the agent.

Configure Trace Logging

Before you can use trace logging, you must configure it by specifying a name, location, and parameters for the trace log file. These settings control the size and format of the file itself. After trace logging is configured, you determine the content of the trace log file separately. This lets you change the types of information contained in your trace log at any time, without changing the parameters of the trace log file itself.

To configure trace logging

1. Locate the WebAgentTrace.conf file on your web server. Duplicate the file.

   Note: If you are running the SiteMinder Agent for IIS and protecting 32-bit applications on a 64-bit system (WoW64 mode), create two duplicates. There are separate directories for 32 and 64-bit applications on 64-bit Windows operating environments.

2. Open your Agent Configuration Object or local configuration file.

3. Set the TraceFile parameter to yes.

   Note: Setting the value of this parameter to yes in a local configuration file of a web server overrides any of the logging settings that are defined on the Policy Server. For example, suppose that the value of this parameter is set to yes in a LocalConfig.conf file. The agent creates log files even though the value of the AllowLocalConfig parameter in the corresponding agent configuration object is set to no. You can also set the related logging parameters in the LocalConfig.conf file also to override any other settings in the agent configuration object.

4. Specify the full path to the trace log files in following parameters:

   TraceFileName

   Specifies the full path to the trace log file.

   Default: No default

   Limits: Specify the file name in this parameter.
   Example: *web_agent_home*\log\trace.log

   TraceFileName32

   Specifies the full path to the trace file for the SiteMinder Agent for IIS is running on a 64-bit Windows operating environment and protecting 32-bit applications. Set this parameter if you have a SiteMinder Agent for IIS installed on a 64-bit Windows operating environment and protecting a 32-bit Windows application. The 32-bit applications run in Wow64 mode on the 64-bit Windows operating environment. If trace logging is enabled but this parameter is not set, the Web Agent for IIS appends _32 to the file name.

   Default: No default.

   Limits: For Windows 64-bit operating environments only. Specify the trace file name at the end of the path.

Example: (Windows 64-bit operating environments using Wow64 mode) web_agent_home\log\WebAgentTrace32.log.

5. Specify the full path to the duplicate copies of WebAgentTrace.conf file (you created in Step 1) in the following parameters:

   TraceConfigFile

   Specifies the location of the WebAgentTrace.conf configuration file that determines which components and events to monitor.

   Default: No default

   Example: *web_agent_home*\config\WebAgentTrace.conf

   TraceConfigFile32

   Specifies the location of the WebAgentTrace.conf configuration file that determines which components and events to monitor. Set this parameter if you have a SiteMinder Agent for IIS installed on a 64-bit Windows operating environment and protecting a 32-bit Windows application. The 32-bit applications run in Wow64 mode on the 64-bit Windows operating environment. If logging is enabled but this parameter is not set, the Web Agent for IIS appends _32 to the file name.

   Default: No default.

   Limits: For Windows 64-bit operating environments only. Specify the configuration file name at the end of the path.

   Example: (Windows 64-bit operating environments using Wow64 mode) *web_agent_home*\config\WebAgentTrace32.conf.

   Note: This file is not used until the web server is restarted.

6. Define the format of the information in your trace log file by setting the following parameters in your Agent Configuration Object or local configuration file:

   TraceAppend

   Adds new logging information to the end of an existing log file instead of rewriting the entire file each time logging is invoked.

   Default: No

   TraceFormat

   Specifies how the trace file displays the messages. Choose *one* of the following options:

   - default—uses square brackets [] to enclose the fields.
   - fixed—uses fields with a fixed width.
   - delim—uses a character of your choice to delimit the fields.
   - xml—uses XML-like tags. A DTD or style sheet is *not* provided with the Web Agent.

   Default: default (square brackets)

TraceDelimiter

Specifies a custom character that separates the fields in the trace file.

Default: No default

Example: |

TraceFileSize

Specifies (in megabytes) the maximum size of a trace file. The Web Agent creates a new file when this limit is reached.

Default: 0 (a new log file is not created)

Example: 20 (MB)

LogLocalTime

Specifies whether the logs use Greenwich Mean Time (GMT) or local time. To use GMT, change this setting to no. If this parameter does not exist, the default setting is used.

Default: Yes

7. Edit the WebAgentTrace.conf file to have Web Agent monitor the activities you want.

   Framework Web Agents do not support dynamic configuration of log parameters set locally in the Agent configuration file. Consequently, when you modify a parameter, the change does not take effect until you restart the web server. However, these log settings can be stored and updated dynamically if you configure them in an Agent configuration object on the Policy Server.

   Note: IIS Agents create log files only after the first user request is submitted. Apache 2.0 Web Agents create log files when the Apache server starts.

8. Restart the web server so the Web Agent uses the new trace configuration file.

Trace Log Components and Subcomponents

The SiteMinder Agent can monitor specific SiteMinder components. When you monitor a component, all of the events for that component are recorded in the trace log. Each component has one or more subcomponents that the agent can also monitor. If you do not want the agent to record all of the events for a component, you can specify only those subcomponents you want to monitor instead.

For example, if you want to record only the single sign-on messages for an agent on a web server, you would specify the WebAgent component and the SSO subcomponent.

The following components and subcomponents are available:

AgentFramework

Records all Agent framework messages. (Applies only to framework agents.) The following subcomponents are available:

- Administration
- Filter
- HighLevelAgent
- LowLevelAgent
- LowLevelAgentWP

AffiliateAgent

Records web Agent messages related to the 4.x Affiliate Agent, which is part of Federation Security Services, a separately-purchased product. (Applies only to framework agents.) The following subcomponent is available:

- RequestProcessing

SAMLAgent

Web Agent messages related to the SAML Affiliate Agent. (Applies only to framework agents.) The following subcomponent is available:

- RequestProcessing

WebAgent

Records all Web Agent log messages. Applies to all Agents *except* IIS 6.0 or Apache 2.0 Agents. The following subcomponents are available:

- AgentCore
- Cache
- authentication
- Responses
- Management
- SSO
- Filter

Agent_Functions

Records all Agent API messages. The following subcomponents are available:

- Init
- UnInit
- IsProtected
- Login
- ChangePassword
- Validate
- Logout
- Authorize
- Audit
- FreeAttributes
- UpdateAttributes
- GetSessionVariables
- SetSessionVariables
- DeleteSessionVariables
- Tunnel
- GetConfig
- DoManagement

Agent_Con_Manager

Records messages related to internal processing of the Agent API. The following subcomponents are available:

- RequestHandler
- Cluster
- Server
- WaitQueue
- Management
- Statistics

For an explanation of each subcomponent, see the WebAgentTrace.conf file.

Trace Message Data Fields

You can define what each trace message for a specific component contains by specifying which data fields to include in the message.

Data fields use the following syntax:

```
data:data_field1,data_field2,data_field3
```

Some data fields are shown in the following example:

```
data:message,date,time,user,agentname,IPAddr
```

There may not be data for fields in each message, so blank fields my occur. For example, if you select RealmOID as a data field, some trace messages will display the realm's OID while others will not.

The following data fields are available:

Message

Includes the actual trace message

SrcFile

Includes the source file and line number of the trace message

Pid

Includes the process ID

Tid

Includes the thread ID

Date

Includes the date

Time

Includes the time

PreciseTime

Includes the time, including milliseconds

Function

Includes the function in the code containing the trace message

User

Includes the name of the user

Domain

Includes the SiteMinder domain

Realm

Includes the SiteMinder realm

AgentName

Includes the Agent name being used

TransactionID

Includes the transaction ID

DomainOID

Includes the SiteMinder domain OID

IPAddr

Includes the client IP address

RequestIPAddr

Includes the trace file displays the IP of the server where Agent is present

IPPort

Includes the client IP port

CertSerial

Includes the certificate serial number

SubjectDN

Includes the subject DN of the certificate

IssuerDN

Includes the Issuer DN of the certificate

SessionSpec

Includes the SiteMinder session spec

SessionID

Includes the SiteMinder session ID

UserDN

Includes the User DN

Resource

Includes the requested resource

Action

Includes the requested action

RealmOID

Includes the realm OID

ResponseTime

Includes the average response time in milliseconds of the Policy Servers associated with a CA Web Agent or SDK Agent and API application

Note: To output the ResponseTime to a trace log, include the component Agent_Con_Manager along with the data field ResponseTime in the WebAgentTrace.conf file or other file specified in the Policy Server Configuration Object (ACO) and restart the Policy Server. The Agent_Con_Manager component, or Agent API Connection Manager, calculates the ResponseTime each time a response is received from a Policy Server and keeps a running average. To locate the ResponseTime in the trace log, search for [PrintStats].

Trace Message Data Field Filters

To focus on a specific problem, you can narrow the output of the trace log by specifying a filter based on the value of a data field. For example, if you are having problems with an index.html page, you can filter on resources with an html suffix by specifying Resource:==/html in the trace configuration file. Each filter should be on a separate line in the file.

Filters use the following syntax:

data_field:*filter*

The following types of filters are available:

- == (exact match)
- != (does not equal)

The filters use boolean logic as shown in the following examples:

Action:!=get (all actions except get)

Resource:==/html (all resources ending in /html)

Determine the Content of the Trace Log

The WebAgentTrace.conf file determines the content of the trace log. You can control which components and data items appear in your trace log by modifying the settings of the WebAgentTrace.conf file on your web server. The following factors apply when editing the file:

- Entries are case-sensitive.

  When you specify a component, data field, or filter, the values must match exactly the options in the WebAgentTrace.conf file instructions.

- Uncomment the configuration settings lines.
- If you modify the WebAgentTrace.conf file before installing a new agent over an existing agent, the file is overwritten. Rename or back up the file first. After the installation, you can integrate your changes into the new file.

Follow these steps:

1. Open the WebAgentTrace.conf file.

   Note: We recommend duplicating the original file and changing the copy. Modifying the copy preserves the default settings.

2. Add components and subcomponents using the following steps:

   a. Find the section that matches your type of agent. For example, if you have an Apache 2.0 Agent that is installed on your server, look for a line resembling the following example:

      ```
      # For Apache 2.0, Apache 2.2, IIS 7.0 and SunOne Web Agents
      ```

      Note: For more information, see the *SiteMinder Web Agent Installation Guide*.

   b. Locate the following line in that section:

      ```
      #components:
      ```

   c. Uncomment the line. Then add the component names that you want after the colon. Separate multiple components commas as shown in the following example:

      ```
      components:  AgentFramework, HTTPAgent
      ```

   d. (Optional) Follow the component name with the name of a subcomponent you want. Separate the subcomponent name with a slash as shown in the following example:

      ```
      components:  AgentFramework/Administration
      ```

3. Add data fields and filters using the following steps:
   a. Locate the following line in the appropriate section:

      ```
      #data:
      ```

   b. Uncomment the line. Then add the data fields that you want after the colon. Separate multiple data fields with commas as shown in the following example:

      ```
      data: Date, Time, Pid, Tid, TransactionID, Function, Message, IPAddr
      ```

   c. (Optional) Add filters to your data fields by following the data field with a colon, the Boolean operator and the value you want. The values you specify for the filters must match exactly. The following example shows a filter which logs activities for a specific IP address:

      ```
      data: Date, Time, Pid, Tid, TransactionID, Function, Message,
      IPAddr:==127.0.0.1
      ```

      Note: Each filter must be on a separate line in the file.
4. Save your changes and close the file.
5. Restart the web server to apply your changes.

   The content of the trace log has been determined.

Limit the Number of Trace Log Files Saved

You can limit the number of trace logs that a SiteMinder agent keeps. For example, if you want to save disk space on the system that stores your agent logs, you can limit the number of trace logs using the following parameter:

TraceFilesToKeep

Specifies the number of SiteMinder agent trace log files that are kept. New trace logs are created in the following situations:

- When the agent starts.
- When the size limit of the trace log (specified by the value of the TraceFileSize parameter) is reached.

Changing the value of this parameter does *not* automatically delete any existing trace logs which exceed the number that you want to keep. For example, If your system has 500 trace logs stored, and you decide to keep only 50 of those files, the agent does *not* delete the other 450 trace logs.

Setting the value of this parameter to zero retains all the trace logs.

Default: 0

Follow these steps:

1. Archive or delete any existing trace logs from your system.
2. Set the value of the TraceAppend parameter to no.
3. Change the value of the TraceFilesToKeep parameter to the number of trace logs that you want to keep.

Collect Detailed Agent Connection Data with an Agent Connection Manager Trace Log

To collect detailed information about the connections between a Web Agent and Policy Server, you create a Trace Log file that contains information gathered by the Agent Collection Manager.

To collect detailed web agent connection data

1. Open your Agent Configuration object or local configuration file.
2. Set the value of the TraceFile parameter to yes.

   Note: Setting the value of this parameter to yes in a local configuration file of a web server overrides any of the logging settings defined on the Policy Server. For example, when the value of this parameter is set to yes in a LocalConfig.conf file log files are generated even if the value of the AllowLocalConfig parameter in the corresponding Agent Configuration object on the Policy Server is set to no. Additionally, set the related trace logging parameters (that define the file name, size, and so on) in the LocalConfig.conf file to override any Policy Server trace log settings.
3. Specify the full path to the trace log file for your Agent Connection Data in the TraceFileName parameter. This is the file that contains the trace log output.
4. Set the value of the TraceConfigFile parameter to the full path of the following file:

   web_agent_home/config/AgentConMgr.conf

   web_agent_home

   Indicates the directory where the SiteMinder Agent is installed.

   Default (Windows 32-bit installations of SiteMinder Web Agents only): C:\Program Files\CA\webagent

   Default (Windows 64-bit installations [SiteMinder Web Agents for IIS only]): C:\Program Files\CA\webagent\win64

   Default (Windows 32-bit applications operating on 64-bit systems [Wow64 with SiteMinder Web Agents for IIS only]): C:\Program Files (x86)\webagent\win32

   Default (UNIX/Linux installations): /opt/ca/webagent

5. Define the format the trace log file for your Agent Connection Data by setting the following parameters:

   TraceAppend

   Adds new logging information to the end of an existing log file instead of rewriting the entire file each time logging is invoked.

   Default: No

TraceDelimiter

Specifies a custom character that separates the fields in the trace file.

Default: No default

Example: |

TraceFileSize

Specifies (in megabytes) the maximum size of a trace file. The Web Agent creates a new file when this limit is reached.

Default: 0 (a new log file is not created)

Example: 20 (MB)

TraceFormat

Specifies how the trace file displays the messages. Choose *one* of the following options:

- default—uses square brackets [] to enclose the fields.
- fixed—uses fields with a fixed width.
- delim—uses a character of your choice to delimit the fields.
- xml—uses XML-like tags. A DTD or style sheet is *not* provided with the Web Agent.

Default: default (square brackets)

LogLocalTime

Specifies whether the logs use Greenwich Mean Time (GMT) or local time. To use GMT, change this setting to no. If this parameter does not exist, the default setting is used.

Default: Yes

6. Restart your web server so the new settings take effect.

   Detailed information about the Web Agent connections will be collected.

   Note: For SiteMinder r12.5, the BusyHandleCount and FreeHandleCount attributes are not used.

Chapter 19: Troubleshooting

Chapter 20: Agent Error Codes

Agent for IIS Troubleshooting Log

Symptom:

Is there an IIS 7.x log file that can help me troubleshoot the SiteMinder Agent for IIS?

Solution:

Open the following log file:

web_agent_home\log\IIS70Trace.log

This log file contains the following types of information:

- Application pool id
- Application pool pipeline mode
- Filter type (native HTTP module as opposed to ISAPI filter)
- 32-bit or 64-bit

Duplicate LLAWP Error Appears in Log File

Symptom:

My log file shows the following error:

```
Duplicate LLAWP
```

Solution:

This error occurs when an application pool recycles. The application pool tries to start new LLAWP processes before the current LLAWP processes are fully shut down.

To prevent this error, configure the application pools in your IIS web server to disallow overlapping rotation. This setting forces the application pool to wait for the current LLAWP processes to stop before any new ones are started.

Note: For more information, go the IIS website and search for the phrase, "DisallowOverlappingRotation"

Custom Error Pages not Appearing

Valid on Oracle Directory Enterprise Edition (formerly Oracle iPlanet Directory Server Enterprise Edition)

Symptom:

I set either of the following configuration parameters, but users receive a generic error message from the server instead:

CSSErrorFile

Specifies the location of a custom-error message file or URL that you want to display to the users if they try to open a URL that contains possible cross-site scripting characters.

Default: No default

ServerErrorFile

Instructs the Web Agent to display a custom error page to users who encounter server errors. Specify a file path or URL for this parameter.

Default: No default

Solution:

Do the following steps:

1. Open the *instance_name*-obj.conf file on your web server.
2. Locate the following line:

   ```
   AuthTrans fn="SiteMinderAgent"
   ```

3. Add UseOutputStreamSize="0"to the end of the previous line, as shown in the following example:

   ```
   AuthTrans fn="SiteMinderAgent" UseOutputStreamSize="0"
   ```

4. Save the file, and then restart the web server.

Unable to initialize tracing message

Valid on IIS 7.x

Symptom:

I am running 32-bit and 64-bit agents on the same IIS web server. I want to record trace logs for the 32-bit agent, but I'm seeing the following message instead:

```
[INFO] LLAWP: Unable to initialize tracing.
```

The 64-bit agent trace log is not affected.

Solution:

Verify that the following configuration parameters are defined for your agent:

TraceConfigFile32

Specifies the location of the WebAgentTrace.conf configuration file that determines which components and events to monitor. Set this parameter if you have a SiteMinder Agent for IIS installed on a 64-bit Windows operating environment and protecting a 32-bit Windows application. The 32-bit applications run in Wow64 mode on the 64-bit Windows operating environment. If logging is enabled but this parameter is not set, the Web Agent for IIS appends _32 to the file name.

Default: No default.

Limits: For Windows 64-bit operating environments only. Specify the configuration file name at the end of the path.

Example: (Windows 64-bit operating environments using Wow64 mode) *web_agent_home*\config\WebAgentTrace32.conf.

LogFileName32

Specifies the full path of a log file for a SiteMinder Web Agent for IIS (on 64-bit Windows operating environments protecting 32-bit applications). The 32-bit applications run in Wow64 mode on the 64-bit Windows operating environment. If logging is enabled but this parameter is not set, the Web Agent for IIS appends _32 to the log file name.

Default: No

Limits: For Windows 64-bit operating environments only. Specify the file name at the end of the path.

Example: (Windows 64-bit operating environments using Wow64 mode) *web_agent_home*\log\WebAgent32.log.

TraceFileName32

Specifies the full path to the trace file for the SiteMinder Agent for IIS is running on a 64-bit Windows operating environment and protecting 32-bit applications. Set this parameter if you have a SiteMinder Agent for IIS installed on a 64-bit Windows operating environment and protecting a 32-bit Windows application. The 32-bit applications run in Wow64 mode on the 64-bit Windows operating environment. If trace logging is enabled but this parameter is not set, the Web Agent for IIS appends _32 to the file name.

Default: No default.

Limits: For Windows 64-bit operating environments only. Specify the trace file name at the end of the path.

Example: (Windows 64-bit operating environments using Wow64 mode) web_agent_home\log\WebAgentTrace32.log.

Japanese Pages Rendered Improperly (153202, 153609)

Symptom:

When a user is redirected to another page for one of the following reasons, content in the resulting page is not rendered properly.

- New registration
- Change the password
- Password expired

Solution:

Add the following line to the http.conf file of your Apache web server:

```
“BrowserMatch ".*" suppress-error-charset”
```

Save the http.conf file and restart your web server for this setting to take effect.

Chapter 21: Agent Error Codes

This section contains the following topics:

00-0001 (see page 331)
00-0002 (see page 332)
00-0004 (see page 332)
00-0005 (see page 332)
00-0006 (see page 333)
00-0007 (see page 333)
00-0008 (see page 333)
00-0009 (see page 334)
00-0010 (see page 334)
00-0011 (see page 334)
00-0012 (see page 335)
00-0013 (see page 335)
00-0014 (see page 336)
00-0015 (see page 336)
00-0016 (see page 336)
00-0017 (see page 337)
10-0001 (see page 337)
10-0002 (see page 337)
10-0003 (see page 337)
10-0004 (see page 338)
10-0005 (see page 338)
10-0007 (see page 338)
20-0001 (see page 339)
20-0002 (see page 339)
20-0003 (see page 340)
30-0026 (see page 340)

00-0001

Reason:

Unable to resolve the agent name from an IP address

Action:

Check the agent configuration and ensure that each HOST address served by the web server has a corresponding AgentName mapped to it or that DefaultAgentName is set properly.

00-0002

Reason:

Illegal Characters exist in a URL or characters defined in the BadUrlChars parameter have been detected in a URL.

Action:

Do one of the following:

- Remove the offending characters from the URL
- Remove the characters from the list in the BadUrlChars parameter so they will no longer be blocked.

00-0004

Reason:

An SSLCRED cookie contains a status of error.

Action:

Investigate the Web Agent acting as the secure credential collector (SCC) and verify its configuration.

Typically, this error occurs only when the SCC agent cannot acquire credentials from its environment. This indicates a possible configuration error.

00-0005

Reason:

A FORMCRED cookie contains a status of error.

Action:

Investigate the Web Agent acting as the forms credential collector (FCC) and verify its configuration.

Typically, this error occurs only when the FCC agent cannot acquire credentials from its environment. This indicates a possible configuration error.

00-0006

Reason:

An NTLM Protected Resource was not found in the resource cache as expected.

Action:

Investigate the Windows authentication scheme setup to verify the configuration.

00-0007

Reason:

An ASCII encoding error exists. This is an internal Web Agent error.

Action:

Investigate the web server and Web Agent to diagnose possible service instability.

Contact Customer Support with the Web Agent log and configuration files available for review.

More information:

Contact CA Technologies (see page 3)

00-0008

Reason:

SSL Authentication failed. This error indicates a bad certificate or that the user is not authenticated.

Action:

Try a different certificate or investigate the SSL authentication scheme configuration for possible issues.

00-0009

Reason:

Bad or Missing SSL credentials.

Action:

Try a different certificate or username and password pair. Investigate the SSL authentication scheme configuration for possible issues.

00-0010

Reason:

Access Denied. This error indicates a general failure that resulted in blocked access.

Action:

Investigate the Web Agent and Policy Server logs to determine the root cause of the failure.

00-0011

Reason:

Credential Collector Error.This indicates that a general failure in Forms or SSL-based advanced authentication resulted in blocked access.

Action:

Do the following:

- Check the Web Agent and Policy Server logs to determine the root cause of the failure.
- Investigate the advanced authentication scheme setup for issues.

00-0012

Reason:

Encryption Error. This indicates an internal Web Agent error.

Action:

Do the following:

- Investigate the web server and Web Agent to diagnose a possible service instability.
- Review Key Store setup to verify that proper Agent Keys are in use.
- Contact Customer Support and send the Web Agent log and configuration files for review.

More information:

Contact CA Technologies (see page 3)

00-0013

Reason:

Agent Configuration Error. One or more errors occurred during startup preventing valid configuration of the Web Agent.

Action:

Do the following:

- On Windows, check the Application Event Log for more information.
- For Apache agents, check the Apache error log for more information.
- For Oracle iPlanet UNIX agents, start Oracle iPlanet from a shell prompt and look for possible errors displayed there through STDERR.
- Check that SmHost.conf file exists (host is registered properly) and contains valid entries.
- Check that Agent Configuration file contains a valid HostConfigFile entry that points to a valid SmHost.conf file.
- Check that AgentConfigObject contains a valid value.

00-0014

Reason:

Could not Log the user out.

Action:

Check the following files for more information:

- Web Agent log file
- Web Agent trace file
- Policy Server log file
- Policy Server trace file

00-0015

Reason:

SiteMinder accounting server answered SM_AGENTAPI_NO to auditing request.

Action:

Check the following files for more information:

- Policy Server log file
- Policy Server trace file

00-0016

Reason:

Unable to resolve the FQ host name.

Action:

Check the Web Agent logs to determine the host name that the Agent is trying to resolve. If the host name is correct, check the DNS settings of the web server on which the agent runs.

00-0017

Reason:

Invalid redirect target found.

Action:

Examine the log file of the Web Agent which is reporting this message to locate the URL being processed (usually an FCC or other advanced authentication URL) and determine if the value of the TARGET CGI parameter appears valid.

10-0001

Reason:

Unable to read the 'SERVER_NAME' HTTP variable.

Action:

Check that the web browser and web server are HTTP 1.0-compliant.

10-0002

Reason:

Unable to read the 'URL' HTTP variable.

Action:

Check that the web browser and web server are HTTP 1.0-compliant.

10-0003

Reason:

Unable to read the 'method' HTTP variable.

Action:

Check that the web browser and web server are HTTP 1.0-compliant.

10-0004

Reason:

Unable to read the 'host' HTTP variable.

Action:

Check that the web browser and web server are HTTP 1.0-compliant.

10-0005

Reason:

Unable to read the 'URI' HTTP variable.

Action:

Check that the web browser and web server are HTTP 1.0-compliant.

10-0007

Reason:

The URL is too long.

Action:

Increase setting of the MaxUrlSize parameter; the default setting is 4096 bytes.

More information:

Set a Maximum URL Size (see page 244)

20-0001

Reason:

Unable to reach SiteMinder accounting server or an unexpected Policy Server error occurred.

Action:

Do the following:

- Check the Policy Server logs for more detailed information on the error.
- Check connectivity between the Web Agent and the Policy Server by pinging the Policy Server. If a firewall is configured between the Agent and the Policy Server, check that it is not blocking the following service ports:
  - 44441 (accounting)
  - 44442 (authentication)
  - 44443 (authorization)

20-0002

Reason:

Unable to reach SiteMinder authentication server or an unexpected Policy Server error occurred.

Action:

Do the following:

- Check Policy Server logs for more detailed information on the error.
- Check connectivity between the Web Agent and the Policy Server by pinging the Policy Server. If a firewall is configured between the Agent and the Policy Server, check that it is not blocking the following service ports:
  - 44441 (accounting)
  - 44442 (authentication)
  - 44443 (authorization)

20-0003

Reason:

Unable to reach SiteMinder authorization server or an unexpected Policy Server error occurred.

Action:

Do the following:

- Check Policy Server logs for more detailed information on the error.
- Check connectivity between the Web Agent and the Policy Server by pinging the Policy Server. If a firewall is configured between the Agent and the Policy Server, check that it is not blocking the following service ports:
  - 44441 (accounting)
  - 44442 (authentication)
  - 44443 (authorization)

30-0026

Reason:

The Password Services Redirect URL is not available.

Action:

Check that you have configured the redirection URL for password services.

Appendix A: Agent Parameters

This section contains the following topics:

List of Agent Configuration Parameters (see page 341)

List of Agent Configuration Parameters

The following table lists the agent configuration parameters:

To set this parameter:	See this procedure:
AcceptTPCookie	Configure Support for SDK Third-Party Cookies (see page 83)
AgentConfigObject	Parameters Found Only in Local Configuration Files (see page 35)
AgentName	Set the AgentName and DefaultAgentName Values (see page 50)
AgentNamesAreFQHostNames	Configure Credential Collectors in a Mixed Environment (see page 147)
AgentWaitTime	Accommodate Network Latency (see page 54)
AllowCacheHeaders	Control How HTTP Header Resources are Cached (see page 117)
AllowLocalConfig	Implement Local Configuration (see page 36) Restrict Changes to Local Configuration Parameters (see page 39)
AppendIISServerLog	Record the User Name and Transaction ID in IIS Server Logs (see page 246)
autoauthorizeoptions	Allow Automatic Access to Resources that use the OPTIONS Method (see page 183)
BadCSSChars	Protect Web Sites Against Cross-Site Scripting (see page 65)
BadFormChars	Specify Bad Form Characters
BadQueryChars	Specify Bad Query Characters (see page 67)
BadUrlChars	Specify Bad URL Characters (see page 69)
CacheAnonymous	Cache Anonymous Users (see page 293)
CCCExt	Specify the Cookie Provider (see page 199)
ConformToRFC2047	Disable Conformance to RFC 2047 (see page 174)
ConstructFullPwsvcUrl	Use a Fully Qualified URL for Password Services Redirects (see page 174)
CookieDomain	How to Configure Single Sign-On (see page 189)
CookieDomainScope	Implement Cookie Domain Resolution (see page 82)

To set this parameter:	See this procedure:
CookiePath	Specify the Cookie Path for Agent Cookies (see page 79)
CookiePathScope	Specify the Cookie Path for Agent Cookies (see page 79)
CookieProvider	How to Configure Single Sign-On (see page 189)
CookieValidationPeriod	Protect Session Cookies from Misuse with Validation Periods and Expired Cookie URLs (see page 97)
CSSChecking	Configure the Web Agent to Check For Cross Site-Scripting (see page 65)
CSSErrorFile	How to Set Up Error Handling (see page 123)
Custom401ErrorFile	How to Set Up Error Handling (see page 123)
CustomIpHeader	Configure IP Address Validation (see page 115)
DecodeQueryData	Decode Query Data (see page 84)
DefaultAgentName	Set the AgentName and DefaultAgentName Values (see page 50)
DefaultHostName	Accommodate Testing Tools that do not send HOST Headers (see page 289)
DefaultPassword	Use an IIS Proxy User Account (see page 258)
DefaultUsername	Use an IIS Proxy User Account (see page 258)
DeleteCerts	Delete Certificates from Stronghold Servers (see page 264)
DisableAuthSrcVars	Disable Default HTTP Header Variables (see page 121)
DisableDirectoryList	Restrict Directory Browsing on a Sun Java System Server (see page 265)
DisableDNSLookups	Help Prevent DNS DOS Attacks (see page 72)
DisableDotDotRule	Handle Complex URIs (see page 86)
DisableSessionVars	Disable Default HTTP Header Variables (see page 121)
DisableUserNameVars	Disable Default HTTP Header Variables (see page 121)
DisableWindowsSecurityContext	Disable Windows Security Context on Agents for IIS (see page 259).
DisallowUTF8NonCanonical	Protect J2EE Applications against Cross-Site Scripting Attacks (see page 66)
DLPExclusionList	Exclude Resources from the DLP Content Classifications **Note:** For more information, see the *SiteMinder Implementation Guide*.
DLPSupportEnabled	Modify the SharePoint Agent Configuration Object **Note:** For more information, see the *SiteMinder Implementation Guide*.
DominoDefaultUser	Authenticate Users with the Domino Server (see page 276)
DominoLegacyDocumentSupport	Handle User-Requested Actions on Lotus Notes Documents (see page 284)

To set this parameter:	See this procedure:
DominoLookUpHeaderForLogin	Use a SiteMinder Header for Authentication (see page 281)
DominoMapUrlForRedirect	Map URLs for FCC Redirects with a Domino Web Agent (see page 142)
DominoNormalizeUrls	Map URLs for FCC Redirects with a Domino Web Agent (see page 142)
DominoSuperUser	Authenticate as the Domino Super User (see page 277)
DominoUseHeaderForLogin	Use a SiteMinder Header for Authentication (see page 281)
DominoUserForAnonAuth	Use an Anonymous SiteMinder Authentication Scheme with Domino (see page 282)
EnableAuditing	Configure Auditing to Track User Activity (see page 63)
EnableCookieProvider	Disable Cookie Providers (see page 200)
EnableFCCWindowsAuth	Configure the FCC to allow Windows authentication (see page 160)
EnableFormCache	Configure the Form Cache (see page 141)
EnableIntroscopeApiSupport	Use CA Technologies Wily Introscope to Monitor Web Agents (see page 297)
EnableMonitoring	Monitor Web Agents with the OneView Monitor (see page 297)
EnableOtherAuthTrans	Handle Multiple AuthTrans Functions (see page 266)
EnableWebAgent	Enable a Web Agent (see page 59)
EncryptAgentName	Encrypt the Agent Name (see page 53)
EnforceRealmTimeouts	How to Enforce Timeouts across Multiple Realms (see page 103)
ExpiredCookieURL	Protect Session Cookies from Misuse with Validation Periods and Expired Cookie URLs (see page 97)
ExpireForProxy	Configure Agents that Sit behind Proxy Servers (see page 149)
FCCCompatMode	Use FCCs and NTCs in a Mixed Environment (see page 149)
FCCExt	Set Up Credential Collectors for IIS and Domino web servers (see page 137)
FCCForceIsProtected	Force an FCC to Establish Realm Context for Forms Authentication (see page 140)
ForceCookieDomain	Force the Cookie Domain (see page 81)
ForceFQHost	Force the Cookie Domain (see page 81)
ForceIISProxyUser	Use an IIS Proxy User Account (see page 258)
FormCacheTimeout	Configure the Form Cache (see page 141)
GetPortFromHeaders	Use the HTTP HOST Request for the Port Number (see page 262)
HostConfigFile	Parameters Found Only in Local Configuration Files (see page 35)

To set this parameter:	See this procedure:
HTTPHeaderEncodingSpec	Set the HTTP Header Encoding Spec (see page 118)
HttpsPorts	Define HTTPS Ports (see page 84)
IdleTimeoutURL	Redirect a User after a Session Time-out (see page 102)
IgnoreCPForNotprotected	Ignore the Cookie Provider for Unprotected Resources (see page 197)
IgnoreExt	Reduce Overhead by Ignoring File Extensions of Unprotected Resources (see page 299)
IgnoreHost	Specify Virtual Servers to be Ignored by the Web Agent (see page 130)
IgnoreQueryData	Ignore Query Data (see page 302)
IgnoreUrl	Allow Unrestricted Access to URIs (see page 303)
LegacyCookieProvider	Disable FCC Realm Context Confirmation to Improve Performance (see page 140)
LegacyEncoding	Accommodate Legacy URL Encoding (see page 288)
LegacyStreamingBehavior	Choose How Content Types are Transferred in POST Requests (see page 263)
LegacyTransferEncodingBehavior	Use Legacy Applications with an Apache Web Agent (see page 261)
LegacyVariables	Enable Legacy Variables for HTTP Headers (see page 120)
LimitCookieProvider	Restrict Cookie Provider Functions (see page 190)
LoadPlugin	WebAgent.conf file for Framework Agents (see page 33)
localconfigfile	WebAgent.conf file for Framework Agents (see page 33)
LogAppend	Set Up and Enable Error Logging (see page 308)
LogFile	Set Up and Enable Error Logging (see page 308)
LogFileName	Set Up and Enable Error Logging (see page 308)
LogFileSize	Set Up and Enable Error Logging (see page 308)
LogFilesToKeep	Limit the Number of Log Files Saved (see page 310)
LogLocalTime	Set Up and Enable Error Logging (see page 308)
LogoffUri	How to Configure Full Logoff for Single Sign-on (see page 203)
LowerCaseHTTP	Use Lower Case HTTP in Headers (see page 119)
LowerCaseProtocolSpecifier	Specify URL Protocols with Lowercase Characters (see page 153)
MasterCookiePath	Specify the Cookie Path for Agent Cookies (see page 79)
MaxResourceCacheSize	Set the Maximum Resource Cache Size (see page 294)
MaxSessionCacheSize	Set the Maximum User Session Cache Size (see page 295)

To set this parameter:	See this procedure:
MaxTimeoutURL	Redirect a User after a Session Time-out (see page 102)
MaxUrlSize	Set a Maximum URL Size (see page 244)
NTCExt	Specify an NTLM Credential Collector (see page 142)
OverlookSessionForMethods	Prevent Session Cookie Creation or Updates (see page 98)
OverlookSessionForMethodUri	Prevent Session Cookie Creation or Updates Based on Method and URI (see page 99)
OverlookSessionForUrls	Prevent Session Cookie Creation or Updates (see page 98)
OverrideIgnoreExtFilter	Protect Resources Without Extensions (see page 72)
P3PCompactPolicy	Configure your Web Agent to Accommodate P3P Compact Policies (see page 88)
PersistentCookies	Set Persistent Cookies (see page 78)
PersistentIPCheck	Compare IP Addresses to Prevent Security Breaches (see page 75)
PostPreservationFile	Enable Post Preservation between Framework and Traditional Agents (see page 144)
PreserveHeaders	Preserve HTTP Headers (see page 116)
PreservePostData	Enable or Disable POST Preservation (see page 73)
ProxyAgent	SiteMinder Reverse Proxy Deployment Considerations (see page 235)
ProxyDefinition	SiteMinder Reverse Proxy Deployment Considerations (see page 235)
ProxyHeadersAutoAuth	Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)
ProxyHeadersAutoAuth10	Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)
ProxyHeadersProtected	Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)
ProxyHeadersProtected10	Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)
ProxyHeadersUnprotected	Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)
ProxyHeadersUnprotected10	Customize the Cache-Control and ExpireForProxy Header Settings (see page 222)
ProxyTimeout	SiteMinder Reverse Proxy Deployment Considerations (see page 235)
ProxyTrust	Configure Agents that Sit behind Proxy Servers (see page 220)
PSPollInterval	Change How Often an Agent Checks for Policy or Key Updates (see page 61)

To set this parameter:	See this procedure:
RemoteUserVar	Configure the Web Agent to set the REMOTE_USER Variable (see page 106)
ReqCookieErrorFile	Set Up Error Handling (see page 123)
RequireCookies	Require Cookies for Basic Authentication (see page 76)
ResourceCacheTimeout	Control How Long Resource Enteries Remain Cached (see page 296)
SaveCredsTimeout	Set a Timeout for Saved Credentials (see page 291)
SCCExt	Set Up Credential Collectors for IIS and Domino Web Servers (see page 137)
SecureApps	Secure Applications (see page 73)
SecureURLs	Configure SecureUrls with Single Sign-on (see page 198)
ServerErrorFile	Set Up Error Handling (see page 123)
SessionGracePeriod	Modify the Session Grace Period (see page 95)
SessionUpdatePeriod	Modify the Session Update Period (see page 96)
SetRemoteUser	Configure the Web Agent to set the REMOTE_USER Variable (see page 106)
SFCCExt	Set Up Credential Collectors for IIS and Domino Web Servers (see page 137)
SkipDominoAuth	Authenticate Users with the Domino Server (see page 276)
SSOTrustedZone	The Order of Trust and Failover (see page 218)
SSOZoneName	Configure Security Zones (see page 215)
StoreSessioninServer	Enable Single Use Session Cookies (see page 100)
SuppressServerHeader	Remove the Server HTTP Header if Using the URLScan Utility (see page 241)
TargetAsRelativeURI	Use a Relative Target for Credential Collector Redirects (see page 138)
TraceAppend	Configure Trace Logging (see page 312)
TraceConfigFile	Configure Trace Logging (see page 312)
TraceDelimiter	Configure Trace Logging (see page 312)
TraceFile	Configure Trace Logging (see page 312)
TraceFileName	Configure Trace Logging (see page 312)
TraceFileSize	Configure Trace Logging (see page 312)
TraceFilesToKeep	Limit the Number of Trace Log Files Saved (see page 322)
TraceFormat	Configure Trace Logging (see page 312)

To set this parameter:	See this procedure:
TrackCPSessionDomain	Prevent Cookie Provider Replay Attacks (see page 191)
TrackSessionDomain	Validate a Session Cookie Domain (see page 101)
TransientIDCookies	Control Identity Cookies (see page 77)
TransientIPCheck	Compare IP Addresses to Prevent Security Breaches (see page 75)
UseAnonAccess	Enable Anonymous User Access (see page 259)
UseDominoUserForUnprotected	Force the Domino server to authenticate unprotected resources (see page 287)
UseHTTPOnlyCookies	Safeguard Information in Cookies with HTTP-Only Attribute (see page 76)
UseNetBIOSforIISAuth	Use the NetBIOS Name or UPN for IIS Authentication (see page 248)
UseSecureCookies	Set Secure Cookies (see page 77)
UseSecureCPCookies	Set Secure Cookies Across Multiple Domains (see page 196)
UseServerRequestIp	Resolve Agent Identity by IP Address (see page 74)
ValidFedTargetDomain	Define Valid Federation Target Domains
ValidTargetDomain	Define Valid Target Domains (see page 139)
WebAppClientResponse	Apply SiteMinder Behavior to a Web Application Client (see page 89)
XFrameOptions	Ensure Custom Responses Comply with X-Frame Options (see page 242)

Chapter 22: SiteMinder Support Matrix

Locate the Platform Support Matrix

Use the Platform Support Matrix to verify that the operating environment and other required third-party components are supported.

Follow these steps:

1. Log in to the CA Support site.
2. Locate the Technical Support section.
3. Enter SiteMinder in the Product Finder field.

The SiteMinder product page appears.

4. Click Product Status, SiteMinder Family of Products Platform Support Matrices.

Note: You can download the latest JDK and JRE versions at the Oracle Developer Network.

Index

P

Parameters
- AcceptTPCookie • 83
- AgentConfigObject • 23, 35
- AgentNamesAreFQHostNames • 137
- AgentWaitTime • 54
- AllowCacheHeaders • 117, 225
- AllowLocalConfig • 30, 36, 39, 40, 323
- AppendIISServerLog • 246
- BadQueryChars • 67
- BadUrlChars • 69, 236
- CacheAnonymous • 23, 293
- ConformToRFC2047 • 119
- ConstructFullPwsvcUrl • 174
- CookieDomain • 194
- CookieDomainScope • 194
- CookiePath • 79
- CookiePathScope • 79
- CookieProvider • 199
- CookieValidationPeriod • 97
- CSSChecking • 65
- Custom401ErrorFile • 122, 123, 124
- CustomIpHeader • 115
- DecodeQueryData • 84
- DefaultAgentName • 50, 53, 100, 129, 137, 147, 149, 152
- DefaultHostName • 289
- DefaultPassword • 107, 258
- DefaultUsername • 107, 258
- DeleteCerts • 264
- DisableAuthSrcVars • 121
- DisableDirectoryList • 265
- DisableDNSLookups • 72
- DisableDotDotRule • 86
- DisableSessionVars • 121
- DisableUserNameVars • 121
- DominoLegacyDocumentSupport • 284
- DominoLookUpHeaderForLogin • 281
- DominoMapUrlForRedirect • 142
- DominoNormalizeUrls • 283
- DominoUseHeaderForLogin • 281
- DominoUserForAnonAuth • 282
- EnableAuditing • 63
- EnableFormCache • 141
- EnableIntroscopeApiSupport • 297
- EnableMonitoring • 297
- EnableOtherAuthTrans • 266
- EncryptAgentName • 53
- EnforceRealmTimeouts • 103
- ExpiredCookieURL • 97
- ExpireForProxy • 220
- FCCExt • 137
- FCCForceIsProtected • 140
- ForceCookieDomain • 81
- ForceFQHost • 81
- ForceIISProxyUser • 107, 258
- FormCacheTimeout • 141
- GetPortFromHeaders • 262
- HTTPHeaderEncodingSpec • 118
- HttpsPorts • 84, 235
- IdleTimeoutURL • 102
- IgnoreCPForNotprotected • 197
- IgnoreExt • 299
- IgnoreHost • 130
- IgnoreQueryData • 302
- IgnoreUrl • 303
- LegacyCookieProvider • 197
- LegacyEncoding • 288
- LegacyStreamingBehavior • 263
- LegacyTransferEncodingBehavior • 261
- LegacyVariables • 120
- LogAppend • 308
- LogFile • 308
- LogFileName • 308
- LogFileSize • 308
- LogFilesToKeep • 310
- LogLocalTime • 308, 312, 323
- LogOffUri • 202
- LowerCaseHTTP • 119
- LowerCaseProtocolSpecifier • 153
- MasterCookiePath • 79
- MaxResourceCacheSize • 23, 294
- MaxSessionCacheSize • 23, 295
- MaxTimeoutURL • 102
- MaxUrlSize • 244
- NTCExt • 142
- OverlookSessionForMethods • 98, 99
- OverlookSessionForMethodUri • 99
- OverlookSessionForUrls • 98

OverrideIgnoreExtFilter • 72
P3PCompactPolicy • 88
PersistentCookies • 75, 78
PersistentIPCheck • 75, 115
PostPreservationFile • 23, 144
PreserveHeaders • 116
PreservePostData • 73
ProxyAgent • 231, 232, 236
ProxyDefinition • 115
ProxyTimeout • 236
ProxyTrust • 220, 231, 232, 236
PSPollInterval • 61
RemoteUserVar • 106
ReqCookieErrorFile • 122, 123
RequireCookies • 76
ResourceCacheTimeout • 23, 296
SaveCredsTimeout • 291
SecureApps • 73
SecureURLs • 156
ServerErrorFile • 328
ServerPath • 55, 56
SessionGracePeriod • 95
SessionUpdatePeriod • 96
SetRemoteUser • 246
SkipDominoAuth • 273, 276, 277, 278, 280
SSOTrustedZone • 215
SSOZoneName • 215, 217
StoreSessioninServer • 100
SuppressServerHeader • 241
TargetAsRelativeURI • 138
TraceAppend • 312, 323
TraceConfigFile • 312
TraceDelimiter • 312, 323
TraceFile • 312
TraceFileName • 312
TraceFileSize • 312, 323
TraceFilesToKeep • 322
TraceFormat • 312, 323
TrackSessionDomain • 101
TransientIDCookies • 77, 78
TransientIPCheck • 75
UseAnonAccess • 259
UseHTTPOnlyCookies • 76
UseNetBIOSforIISAuth • 248
UseSecureCookies • 77
UseServerRequestIp • 74
ValidTargetDomain • 139