Blue Coat® Systems SG™ Appliance

Configuration and Management Guide

Volume 5: Securing the Blue Coat SG Appliance

SGOS Version 5.1.x

Blue Coat®

Contact Information

Blue Coat Systems Inc.
420 North Mary Ave
Sunnyvale, CA 94085-4121

http://www.bluecoat.com/support/contact.html

bcs.info@bluecoat.com
http://www.bluecoat.com

For concerns or feedback about the documentation: documentation@bluecoat.com

Document Number: 231-02841
Document Revision: SGOS 5.1.x—03/2007

Contents

Chapter 1: About Security

Enterprise-wide security begins with security on the SG appliance, and continues with controlling user access to the Intranet and Internet.

SSH and HTTPS are the recommended (and default) methods for managing access to the SG appliance. SSL is the recommended protocol for communication between the appliance and a realm's off-box authentication server.

Controlling SG Appliance Access

You can control access to the SG appliance several ways: by limiting physical access to the system, by using passwords, restricting the use of console account, through per-user RSA public key authentication, and through Blue Coat Content Policy Language (CPL). How secure the system needs to be depends upon the environment.

You can limit access to the SG appliance by:

- Restricting physical access to the system and by requiring a PIN to access the front panel.
- Restricting the IP addresses that are permitted to connect to the SG appliance CLI.
- Requiring a password to secure the Setup Console.

These methods are in addition to the restrictions placed on the console account (a console account user password) and the Enable password. For information on using the console account, refer to *Volume 2: Getting Started*.

By using every possible method (physically limiting access, limiting workstation IP addresses, and using passwords), the SG appliance is very secure.

Once the SG appliance is secure, you can limit access to the Internet and intranet. It is possible to control access to the network without using authentication. You only need to use authentication if you want to use identity-based access controls.

Controlling User Access with Identity-based Access Controls

The SG appliance provides a flexible authentication architecture that supports multiple services with multiple backend servers (for example, LDAP directory servers together with NT domains with no trust relationship) within each authentication scheme with the introduction of the *realm*.

A *realm* authenticates and authorizes users for access to SG services using either explicit proxy or transparent proxy mode, discussed in *Volume 3: Proxies and Proxy Services*.

Multiple authentication realms can be used on a single SG appliance. Multiple realms are essential if the enterprise is a managed provider or the company has merged with or acquired another company. Even for companies using only one protocol, multiple realms might be necessary, such as the case of a company using an LDAP server with multiple authentication boundaries. You can use realm sequencing to search the multiple realms all at once.

A realm configuration includes:

- Realm name.

- Authentication service—(IWA, LDAP, RADIUS, Local, Certificate, Sequences, Netegrity SiteMinder®, Oracle COREid™, Policy Substitution).
- External server configuration—Backend server configuration information, such as host, port, and other relevant information based on the selected service.
- Authentication schema—The definition used to authenticate users.
- Authorization schema—The definition used to authorize users for membership in defined groups and check for attributes that trigger evaluation against any defined policy rules.
- One-time passwords are supported for RADIUS realms only.

SSL Between the SG Appliance and the Authentication Server

SSL communication between the SG appliance and LDAP and IWA authentication servers is supported. In addition, you can also use SSL between the client and the SG appliance. For more information on using SSL between the client and the appliance, see "Using SSL with Authentication and Authorization Services" on page 28.

Configuring a realm to use SSL between the SG appliance and the authentication server is performed on a per-realm basis. Part of the SSL configuration is specifying whether to verify the server's certificate. If the server certificate is to be verified, then the server's certificate must be signed by a Certificate Authority that the SG appliance trusts, and the common name in the server certificate must match the server host as specified in the realm configuration.

The realms use the default SSL client defined on the SG appliance for SSL communications to the authentication servers.

Note: If the browser is configured for on-line checking of certificate revocation, the status check must be configured to bypass authentication.

About This Book

The first few chapters of *Volume 5: Securing the Blue Coat SG Appliance* deal with limiting access to the SG appliance . The remainder of the book discusses the various realms:

- Chapter 16: "Managing the Credential Cache"
- Appendix A: "Glossary"
- Appendix B: "Using the Authentication/Authorization Agent"

Document Conventions

The following section lists the typographical and Command Line Interface (CLI) syntax conventions used in this manual.

Table 1-1. Document Conventions

Conventions	Definition
Italics	The first use of a new or Blue Coat-proprietary term.
`Courier font`	Command line text that appears on your administrator workstation.
`Courier Italics`	A command line variable that is to be substituted with a literal name or value pertaining to the appropriate facet of your network system.
`Courier Boldface`	A Blue Coat literal to be entered as shown.
{ }	One of the parameters enclosed within the braces must be supplied
[ ]	An optional parameter or parameters.
|	Either the parameter before or after the pipe character can or must be selected, but not both.

Chapter 2: Controlling Access to the SG Appliance

You can control access to the SG appliance several ways: by limiting physical access to the system, by using passwords, restricting the use of console account, through per-user RSA public key authentication, and through Blue Coat Content Policy Language (CPL). How secure the system needs to be depends upon the environment.

This section contains:

- "Limiting Access to the SG Appliance"
- "About Password Security" on page 14
- "Limiting User Access to the SG Appliance—Overview" on page 15
- "Moderate Security: Restricting Management Console Access Through the Console Access Control List (ACL)" on page 17
- "Maximum Security: Administrative Authentication and Authorization Policy" on page 18

Limiting Access to the SG Appliance

You can limit access to the SG appliance by:

- Restricting physical access to the system and by requiring a PIN to access the front panel.
- Restricting the IP addresses that are permitted to connect to the SG appliance CLI.
- Requiring a password to secure the Setup Console.

These methods are in addition to the restrictions placed on the console account (a console account user password) and the Enable password. For information on using the console account, refer to *Volume 2: Getting Started*.

By using every possible method (physically limiting access, limiting workstation IP addresses, and using passwords), the SG appliance is very secure.

This section discusses:

- "Requiring a PIN for the Front Panel"
- "Limiting Workstation Access" on page 14
- "Securing the Serial Port" on page 14

Requiring a PIN for the Front Panel

On systems that have a front panel display, you can create a four-digit PIN to protect the system from unauthorized use. The PIN is hashed and stored. You can only create a PIN from the command line.

To create a front panel PIN, after initial configuration is complete:

From the (config) prompt:

```
SGOS#(config) security front-panel-pin PIN
```

where *PIN* is a four-digit number.

To clear the front-panel PIN, enter:

```
SGOS#(config) security front-panel-pin 0000
```

Limiting Workstation Access

During initial configuration, you have the option of preventing workstations with unauthorized IP addresses from accessing the CLI. If this option is not enabled, all workstations are allowed to access the CLI. You can also add allowed workstations later to the access control list (ACL). (For more information on limiting workstation access, see "Moderate Security: Restricting Management Console Access Through the Console Access Control List (ACL)" on page 17.)

Securing the Serial Port

If you choose to secure the serial sort, you must provide a Setup Console password that is required to access the Setup Console in the future.

Once the secure serial port is enabled:

- The Setup Console password is required to access the Setup Console.
- An authentication challenge (username and password) is issued to access the CLI through the serial port.

To recover from a lost Setup Console password, you can:

- Use the Front Panel display to either disable the secure serial port or enter a new Setup Console password.
- Use the CLI `restore-defaults factory-defaults` command to delete all system settings. For information on using the `restore-defaults factory-defaults` command, refer to *Volume 10: Managing the Blue Coat SG Appliance*.
- Use the reset button (if the appliance has a reset button) to delete all system settings.

To enable the secure serial port, refer to the *Installation Guide* for your platform.

About Password Security

In the SG appliance, the console administrator password, the Setup Console password, and Enable (privileged-mode) password are hashed and stored. It is not possible to reverse the hash to recover the plaintext passwords.

In addition, the `show config` and `show security` CLI commands display these passwords in their hashed form. The length of the hashed password depends on the hash algorithm used so it is not a fixed length across the board.

Passwords that the SG appliance uses to authenticate itself to outside services are encrypted using triple-DES on the appliance, and using RSA public key encryption for output with the `show config` CLI command. You can use a third-party encryption application to create encrypted passwords and copy them into the SG appliance using an encrypted-password command (which is available in several modes and described in those modes). If you use a third-party encryption application, verify it supports RSA encryption, OAEP padding, and Base64 encoded with no new lines.

These passwords, set up during configuration of the external service, include:

- Access log FTP client passwords (primary, alternate)—For configuration information, refer to *Volume 9: Access Logging*.
- Archive configuration FTP password—For configuration information, refer to the archive configuration information in *Volume 2: Getting Started*.

- RADIUS primary and alternate secret—For configuration information, see Chapter 13: "RADIUS Realm Authentication and Authorization".
- LDAP search password—For configuration information, see "LDAP Search & Groups Tab (Authorization and Group Information)" on page 96.
- Content filter download passwords—For configuration information, refer to the content filtering information in *Volume 8: Managing Content*.

Limiting User Access to the SG Appliance—Overview

When deciding how to give other users read-only or read-write access to the SG appliance, sharing the basic console account settings is only one option. The following summarizes all available options:

Note: If Telnet Console access is configured, Telnet can be used to manage the SG appliance with behavior similar to SSH with password authentication.

SSL configuration is not allowed through Telnet, but is permissible through SSH.

Behavior in the following sections that applies to SSH with password authentication also applies to Telnet. Use of Telnet is not recommended because it is not a secure protocol.

- Console account—minimum security

  The console account username and password are evaluated when the SG appliance is accessed from the Management Console through a browser and from the CLI through SSH with password authentication. The Enable (privileged-mode) password is evaluated when the console account is used through SSH with password authentication and when the CLI is accessed through the serial console and through SSH with RSA authentication. The simplest way to give access to others is sharing this basic console account information, but it is the least secure and is not recommended.

  To give read-only access to the CLI, do not give out the Enable (privileged-mode) password.

- Console access control list—moderate security

  Using the access control list (ACL) allows you to further restrict use of the console account and SSH with RSA authentication to workstations identified by their IP address and subnet mask. When the ACL is enforced, the console account can only be used by workstations defined in the console ACL. Also, SSH with RSA authentication connections are only valid from workstations specified in the console ACL (provided it is enabled).

  After setting the console account username, password, and Enable (privileged-mode) password, use the CLI or the Management Console to create a console ACL. See "Moderate Security: Restricting Management Console Access Through the Console Access Control List (ACL)" on page 17.

- Per-user RSA public key authentication—moderate security

  Each administrator's public keys are stored on the appliance. When connecting through SSH, the administrator logs in with no password exchange. Authentication occurs by verifying knowledge of the corresponding private key. This is secure because the passwords never go over the network.

  This is a less flexible option than CPL because you cannot control level of access with policy, but it is a better choice than sharing the console credentials.

- Blue Coat Content Policy Language (CPL)—maximum security

  CPL allows you to control administrative access to the SG appliance through policy. If the credentials supplied are not the console account username and password, policy is evaluated when the SG appliance is accessed through SSH with password authentication or the Management Console. Policy is never evaluated on direct serial console connections or SSH connections using RSA authentication.

  - Using the CLI or the Management Console GUI, create an authentication realm to be used for authorizing administrative access. For administrative access, the realm must support BASIC credentials—for example, LDAP, RADIUS, Local, or IWA with BASIC credentials enabled.
  - Using the Visual Policy Manager, or by adding CPL rules to the Local or Central policy file, specify policy rules that: (1) require administrators to log in using credentials from the previously-created administrative realm, and (2) specify the conditions under which administrators are either denied all access, given read-only access, or given read-write access. Authorization can be based on IP address, group membership, time of day, and many other conditions. For more information, refer to *Volume 7: VPM and Advanced Policy*.
  - To prevent anyone from using the console credentials to manage the SG appliance, set the console ACL to deny all access (unless you plan to use SSH with RSA authentication). For more information, see "Moderate Security: Restricting Management Console Access Through the Console Access Control List (ACL)" on page 17. You can also restrict access to a single IP address that can be used as the emergency recovery workstation.

The following chart details the various ways administrators can access the SG console and the authentication and authorization methods that apply to each.

Table 2-1. SG Console Access Methods/Available Security Measures

Security Measures Available	Serial Console	SSH with Password Authentication	SSH with RSA Authentication	Management Console
Username and password evaluated (console-level credentials)		✓		✓
Console Access List evaluated		✓ (if console credentials are offered)	✓	✓ (if console credentials are offered)
CPL <Admin> Layer evaluated		✓ (see Note 1 below)		✓ (see Note 2 below)
Enable password required to enter privileged mode (see Note 2 below)	✓	✓	✓	
CLI `line-vty timeout` command applies.	✓	✓	✓	
Management Console Login/Logout				✓

Note 1: When using SSH (with a password) and credentials other than the console account, the enable password is actually the same as the login password. The privileged mode password set during configuration is used only in the serial console, SSH with RSA authentication, or when logging in with the console account.

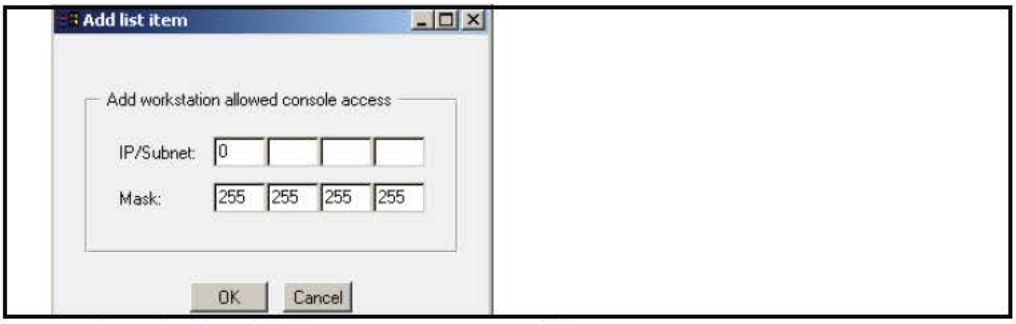
Note 2: In this case, user credentials are evaluated against the policy before executing each CLI command. If you log in using the console account, user credentials are not evaluated against the policy.

Moderate Security: Restricting Management Console Access Through the Console Access Control List (ACL)

The SG appliance allows you to limit access to the Management Console and CLI through the console ACL. An ACL, once set up, is enforced only when console credentials are used to access either the CLI or the Management Console, or when an SSH with RSA authentication connection is attempted. The following procedure specifies an ACL that lists the IP addresses permitted access.

To create an ACL:

1. Select **Configuration > Authentication > Console Access > Console Access**.
2. (Optional) To add a new address to the ACL, click **New**.



   a. In the IP/Subnet fields, enter a static IP address.
   b. In the **Mask** fields, enter the subnet mask. To restrict access to an individual workstation, enter `255.255.255.255`.
   c. Click **OK** to add the workstation to the ACL and return to the Console Access page.
   d. Repeat 2 to add other IP addresses.
3. (Optional) To remove a source address from the ACL, select the address to remove from the Console Access page and click **Delete**.
4. (Optional) To change a source IP address, select the IP address to revise and click **Edit**. See 2, above, for details.
5. To impose the ACL defined in the list box, select **Enforce ACL for built-in administration**. To allow access to the CLI or Management Console using console account credentials from any workstation, deselect the checkbox. The ACL is ignored.

Important: Before you enforce the ACL, verify the IP address for the workstation you are using is included in the list. If you forget, or you find that you mistyped the IP address, you must correct the problem using the serial console.

---

6. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Create an ACL

```
SGOS#(config) security allowed-access add ip_address [subnet_mask]
SGOS#(config) security enforce-acl enable | disable
SGOS#(config) security allowed-access remove ip_address [subnet_mask]
```

Maximum Security: Administrative Authentication and Authorization Policy

The SG appliance permits you to define a rule-based administrative access policy. This policy is enforced when accessing:

- the Management Console through http or https
- the CLI through SSH when using password authentication
- the CLI through telnet
- the CLI through the serial port if the secure serial port is enabled

These policy rules can be specified either by using the VPM or by editing the Local policy file. Using policy rules, you can deny access, allow access without providing credentials, or require administrators to identify themselves by entering a username and password. If access is allowed, you can specify whether read-only or read-write access is given. You can make this policy contingent on IP address, time of day, group membership (if credentials were required), and many other conditions.

Serial-console access is not controlled by policy rules. For maximum security to the serial console, physical access must be limited.

SSH with RSA authentication also is not controlled by policy rules. You can configure several settings that control access: the enable password, the console ACL, and per-user keys configured through the **Configuration > Services > SSH > SSH Client page.** (If you use the CLI, SSH commands are under `config > services > ssh-console`.)

Defining Administrator Authentication and Authorization Policies

The SG appliance uses CPL to define policies, including administrator, authentication, and authorization policies. CPL also allows you to give administrator privileges to users in any external authentication service.

The following summarizes the steps required to define Administrator Authentication and Authorization policies on the SG appliance:

- (Optional) If you need to give administrative access to existing users or groups, create and configure the authentication realm.
- Define the policies in the appropriate policy file where you keep the `<Admin>` Layer layers and rules.
- Load the policy file on the SG appliance.

When you define such policies, make sure you define them in the appropriate policy file(s). For more information on policy files and how they are used, refer to *Volume 7: VPM and Advanced Policy*.

Defining Policies Using the Visual Policy Manager

To define policies through the Management Console, use the Visual Policy Manager. When you use the VPM, policies are configured in CPL and saved in the VPM policy file. For examples of Administrator authentication or authorization policy CPL, continue with the next section. The VPM is described in detail in *Volume 7: VPM and Advanced Policy*.

Defining Policies Directly in Policy Files

To define policies manually, type CPL *rules* directly in one of the two policy files, Central or Local.

> **Important:** For specific information on creating policies within the policy files, refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide*.

Following are the CPL elements that can be used to define administrator policies for the SG appliance.

To define administrator policies by editing a policy file:

1. Open the policy file in a text editor.
2. Define the policies, using the correct CPL syntax.
3. Save the file.
4. Load the policy file (refer to *Volume 7: VPM and Advanced Policy*).

Admin Transactions and <Admin> Layers

Admin transactions execute `<Admin>` layers. Only a restricted set of conditions, properties, and actions are permitted in `<Admin>` layers. The table below lists the conditions permitted in the `<Admin>` layer.

Table 2-2. Network Connection Conditions

<Admin> Network Connection Conditions	
`client_address=`*`ip_address`* `[.subnetmask]`	Tests for a match between *`ip_address`* and the IP address of the client transaction source.
`proxy.port=`*`number`*	Tests for a match between *`number`* and the port number for which the request is destined.
`proxy.address=`*`ip_address`*	Tests for a match between *`ip_address`* and the IP address of the network interface card for which the request is destined.
`proxy.card=`*`number`*	Tests for a match between *`number`* and the ordinal number associated with the network interface card for which the request is destined.
<Admin> General Conditions	
`condition=`*`condition.label`*	Tests if the specified defined condition is true.
`release.id=`	Tests the SG release id.
<Admin> Date/Time Conditions	
`date[.utc]=[`*`date`* | *`date…date`*`]`	Tests for a match between *`date`* and the date timestamp associated with the source of the transaction. *`date`* specifies a single date of the form `YYYY-MM-DD` or an inclusive range, as in `YYYY-MM-DD…YYYY-MM-DD`. By default, date is calculated based on local time. To calculate year based on the Coordinated Universal Time, include the `.utc` qualifier

Table 2-2. Network Connection Conditions (Continued)

`year[.utc]=[year | year…year]`	Tests for a match between *year* and the year timestamp associated with the source of the transaction. *year* specifies a single Gregorian calendar year of the form YYYY or an inclusive range of years, as in YYYY...YYYY. By default, year is calculated based on local time. To calculate year based on the Coordinated Universal Time, include the .utc qualifier.
`month[.utc]=[month | month…month]`	Tests for a match between *month* and the month timestamp associated with the source of the transaction. *month* specifies a single Gregorian calendar month of the form MM or an inclusive range of months, as in MM...MM. By default, month is calculated based on local time. To calculate month based on the Coordinated Universal Time, include the .utc qualifier.
`weekday[.utc]=[number | number…number]`	Tests for a match between *weekday* and the weekday timestamp associated with the source of the transaction. *weekday* specifies a single day of the week (where Monday=1, Tuesday=2, and Sunday=7) or an inclusive range of weekdays, as in *number...number*. By default, weekday is calculated based on local time. To calculate weekday based on the Coordinated Universal Time, include the .utc qualifier.
`day[.utc]=[day | day…day]`	Tests for a match between *day* and the day timestamp associated with the source of the transaction. *day* specifies a single Gregorian calendar day of the month of the form DD or an inclusive range of days, as in *DD...DD*. By default, day is calculated based on local time. To calculate day based on the Coordinated Universal Time, include the .utc qualifier.
`hour[.utc]=[hour | hour…hour]`	Tests for a match between *hour* and the hour timestamp associated with the source of the transaction. *hour* specifies a single Gregorian hour of the form HH (00, 01, and so forth, through 23) or an inclusive range of hours, as in HH…HH. By default, hour is calculated based on local time. To calculate hour based on the Coordinated Universal Time, include the .utc qualifier.
`minute[.utc]=[minute | minute…minute]`	Tests for a match between *minute* and the minute timestamp associated with the source of the transaction. *minute* specifies a single Gregorian minute of the form MM (00, 01, and so forth, through 59) or an inclusive range of minutes, as in MM...MM. By default, minute is calculated based on local time. To calculate minute based on the Coordinated Universal Time, include the .utc qualifier.
`time[.utc]=[time | time…time]`	Tests for a match between *time* and the time timestamp associated with the source of the transaction. *time* specifies military time of the form TTTT (0000 through 2359) or an inclusive range of times, as in TTTT...TTTT. By default, time is calculated based on local time. To calculate time based on the Coordinated Universal Time, include the .utc qualifier.
<Admin> Authorization Conditions	
`attribute.name =value`	Tests if the current transaction is authorized in a RADIUS or LDAP realm, and if the authenticated user has the specified attribute with the specified value. This trigger is unavailable if the current transaction is not authenticated

Table 2-2. Network Connection Conditions (Continued)

`authenticated={yes | no}`	Tests if authentication was requested and the credentials could be verified.
`group=group_name`	If `authenticate=yes`, the group condition tests the source of the transaction for membership in the specified groupname.
`has_attribute.name=boolean`	Tests if the current transaction is authorized in an LDAP realm and if the authenticated user has the specified LDAP attribute.
`realm=realm_name`	If `authenticate=yes`, the realm condition tests the source of the transaction for membership in the specified realm name.
`user=username`	If `authenticate=yes`, the user condition tests the source of the transaction for the expected username.
`user.domain=windows_domain_name`	(This condition is IWA-realm specific.) If `authenticate=yes`, the user_domain condition tests whether the realm type is IWA and whether the domain component of the username is the expected domain name.
<Admin> Read-only or Read-write Conditions	
`admin_access=read | write`	`read` tests whether the source of the transaction has read-only permission for the SG console. `write` tests whether the source has read-write permission. When an Administrator logs into the CLI, the SG appliance executes an `<Admin>` transaction that includes the condition `admin_access=read`. If the transaction is ultimately allowed (all conditions have been met), the user will have read-only access to configuration information through the CLI. Further, when that user executes the CLI `enable` command, or logs into the Management Console, the SG appliance executes an `<Admin>` transaction with `admin_access=write`. If the transaction is allowed, the user will have read-write access within the CLI or the Management Console.

The table below lists the properties permitted in the `<Admin>` layer:

Table 2-3. Properties in the <Admin> Layer

<Admin> Properties	
`deny`	Refuse service to the source of the transaction.
`authenticate(realm_name)`	Requests authentication of the transaction source for the specified realm.
`authenticate.force( )`	If `yes` is specified then forces authentication even if the transaction is denied. This results in the user information being available for logging. If `no`, then early denial without authentication is possible.
`allow`	Permit further service to the source of the transaction.
`log.suppress.field-id ( )`	Controls suppression of the specified *`field-id`* in all facilities
`log.suppress.field-id[log_list]( )`	Controls suppression of the specified *`field-id`* in the specified facilities.

Table 2-3. Properties in the <Admin> Layer (Continued)

`log.rewrite.`*`field-id`*`( )`	Controls rewrites of a specific log field in all facilities.
`log.rewrite.`*`field-id`*`[`*`log_list`*`] ( )`	Controls rewrites of a specific log field in a specified list of log facilities.

The table below lists the actions permitted in the `<Admin>` layer:

Table 2-4. Actions permitted in the <Admin> Layer

<Admin> Actions	
`notify_email( )`	Sends an e-mail notification to the list of recipients specified in the Event Log mail configuration when the transaction terminates.
`notify_snmp( )`	The SNMP trap is sent when the transaction terminates.

Example Policy Using CPL Syntax

To authenticate users against an LDAP realm, use the following syntax in the Local Policy file:

```
<admin>
  authenticate(LDAP_Realm)

<admin>
  group="cn=Administrators,cn=Groups,dc=bluecoat,dc=com" allow
```

This authenticates users against the specified LDAP realm. If the users are successfully authenticated and belong to group `Administrators`, they are allowed to administer the SG appliance.

Chapter 3: Controlling Access to the Internet and Intranet

After the SG appliance is secure, you can limit access to the Internet and intranet. It is possible to control access to the network without using authentication. You only need to use authentication if you want to use identity-based access controls.

This section contains:

Using Authentication and Proxies

Authentication means that the SG appliance requires proof of user identity in order to make decisions based on that identity. This proof is obtained by sending the client (a browser, for example) a *challenge*—a request to provide credentials. Browsers can respond to different kinds of credential challenges:

- Proxy-style challenges—Sent from proxy servers to clients that are explicitly proxied. In HTTP, the response code is 407.

  An authenticating explicit proxy server sends a proxy-style challenge (407/Proxy-Authenticate) to the browser. The browser knows it is talking to a proxy and that the proxy wants proxy credentials. The browser responds to a proxy challenge with proxy credentials (Proxy-Authorization: header). The browser must be configured for explicit proxy in order for it to respond to a proxy challenge.

- Origin-style challenges—Sent from origin content servers (OCS), or from proxy servers impersonating a OCS. In HTTP, the response code is 401 Unauthorized.

  In transparent proxy mode, the SG appliance uses the OCS authentication challenge (HTTP 401 and WWW-Authenticate)—acting as though it is the location from which the user initially requested a page. A transparent proxy, including a reverse proxy, must not use a proxy challenge, because the client might not be expecting it.

Once the browser supplies the credentials, the SG appliance authenticates them.

Understanding Authentication Modes

You can control the way the SG appliance interacts with the client for authentication by controlling the authentication mode. The mode specifies the challenge type and the accepted surrogate credential.

Note: *Challenge type* is the kind of challenge (for example, proxy or origin-ip-redirect) issued.

Surrogate credentials are credentials accepted in place of the user's real credentials.

- **Auto**: The default; the mode is automatically selected, based on the request. Auto can choose any of **proxy**, **origin**, **origin-ip**, or **origin-cookie-redirect**, depending on the kind of connection (explicit or transparent) and the transparent authentication cookie configuration.
- **Proxy**: The SG appliance uses an explicit proxy challenge. No surrogate credentials are used. This is the typical mode for an authenticating explicit proxy. In some situations proxy challenges do not work; origin challenges are then issued.

  If you have many requests consulting the back-end authentication authority (such as LDAP, RADIUS, or the BCAAA service), you can configure the SG appliance (and possibly the client) to use persistent connections. This dramatically reduces load on the back-end authentication authority and improves the all-around performance of the network.

Important: Windows supports Kerberos authentication only to origin servers; proxy servers cannot participate. Therefore, explicit authentication modes are not compatible with Kerberos. However, because Internet Explorer automatically selects NTLM for an explicit challenge (where the browser is configured with the proxy as a proxy server), no special processing is required for explicit authentication. An origin redirect authentication mode, such as `authenticate.mode (origin-cookie-redirect)`, can be used to obtain Kerberos authentication when using an explicit proxy if the browser is configured to bypass the proxy for the virtual URL.

- **Proxy-IP**: The SG appliance uses an explicit proxy challenge and the client's IP address as a surrogate credential. Proxy-IP specifies an insecure forward proxy, possibly suitable for LANs of single-user workstations. In some situations proxy challenges do not work; origin challenges are then issued.
- **Origin**: The SG appliance acts like an OCS and issues OCS challenges. The authenticated connection serves as the surrogate credential.
- **Origin-IP**: The SG appliance acts like an OCS and issues OCS challenges. The client IP address is used as a surrogate credential. **Origin-IP** is used to support IWA authentication to the upstream device when the client cannot handle cookie credentials. This mode is primarily used for automatic downgrading, but it can be selected for specific situations.
- **Origin-cookie**: The SG appliance acts like an origin server and issues origin server challenges. A cookie is used as the surrogate credential. **Origin-cookie** is used in forward proxies to support pass-through authentication more securely than **origin-ip** if the client understands cookies. Only the HTTP and HTTPS protocols support cookies; other protocols are automatically downgraded to **origin-ip**.

  This mode could also be used in reverse proxy situations if impersonation is not possible and the origin server requires authentication.
- **Origin-cookie-redirect**: The client is redirected to a virtual URL to be authenticated, and cookies are used as the surrogate credential. The SG appliance does not support origin-redirects with the CONNECT method. For forward proxies, only origin-*-redirect modes are supported for Kerberos/IWA authentication. (Any other mode uses NTLM authentication.)

Note: During cookie-based authentication, the redirect to strip the authentication cookie from the URL is logged as a 307 (or 302) TCP_DENIED.

- **Origin-IP-redirect**: The client is redirected to a virtual URL to be authenticated, and the client IP address is used as a surrogate credential. The SG appliance does not support origin-redirects with the CONNECT method. For forward proxies, only origin-*-redirect modes are supported for Kerberos/IWA authentication. (Any other mode uses NTLM authentication.)
- **SG2**: The mode is selected automatically, based on the request, and uses the SGOS 2.x-defined rules.
- *Form-IP*: A form is presented to collect the user's credentials. The form is presented whenever the user's credential cache entry expires.
- **Form-Cookie**: A form is presented to collect the user's credentials. The cookies are set on the OCS domain only, and the user is presented with the form for each new domain. This mode is most useful in reverse proxy scenarios where there are a limited number of domains.
- **Form-Cookie-Redirect**: A form is presented to collect the user's credentials. The user is redirected to the authentication virtual URL before the form is presented. The authentication cookie is set on both the virtual URL and the OCS domain. The user is only challenged when the credential cache entry expires.
- **Form-IP-redirect**: This is similar to **form-ip** except that the user is redirected to the authentication virtual URL before the form is presented.

Important: Modes that use an IP surrogate credential are insecure: After a user has authenticated from an IP address, all further requests from that IP address are treated as from that user. If the client is behind a NAT, or on a multi-user system, this can present a serious security problem.

The default value is `auto`.

For more information on using authentication modes, refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide*.

Setting the Default Authenticate Mode Property

Setting the `authentication.mode` property selects a challenge type and surrogate credential combination. In `auto` mode, explicit IWA uses connection surrogate credentials. In `sg2` mode, explicit IWA uses IP surrogate credentials.

To configure the IWA default authenticate mode settings:

```
SGOS#(config) security default-authenticate-mode {auto | sg2}
```

Understanding Origin-Style Redirection

Some authentication modes redirect the browser to a *virtual authentication site* before issuing the origin-style challenge. This gives the user feedback as to which credentials are required, and makes it possible to (but does not require) send the credentials over a secure connection.

Since browser requests are transparently redirected to the SG appliance, the appliance intercepts the request for the virtual authentication site and issues the appropriate credential challenge. Thus, the challenge appears to come from the virtual site, which is usually named to make it clear to the user that SG credentials are requested.

If authentication is successful, the SG appliance establishes a surrogate credential and redirects the browser back to the original request, possibly with an encoded surrogate credential attached. This allows the SG appliance to see that the request has been authenticated, and so the request proceeds. The response to that request can also carry a surrogate credential.

To provide maximum flexibility, the virtual site is defined by a URL. Requests to that URL (only) are intercepted and cause authentication challenges; other URLs on the same host are treated normally. Thus, the challenge appears to come from a host that in all other respects behaves normally.

Note: Sharing the virtual URL with other content on a real host requires additional configuration if the credential exchange is over SSL.

You can configure the virtual site to something that is meaningful for your company. The default, which requires no configuration, is `www.cfauth.com`. See "Configuring Transparent Proxy Authentication" on page 26 to set up a virtual URL for transparent proxy.

Tip: Using CONNECT and Origin-Style Redirection

You cannot use the CONNECT method with origin-style redirection or form redirect modes. An error message similar to the following is displayed:

```
Cannot use origin-redirect for CONNECT method (explicit proxy of https
URL)
```

Instead, you can add policy to either bypass authentication on the CONNECT method, or use proxy authentication. For example:

```
<proxy>
  allow http.method=CONNECT authenticate.mode(proxy)
authenticate(ldap)
  allow authenticate(cert) authenticate.mode(origin-cookie-redirect)
```

Selecting an Appropriate Surrogate Credential

IP surrogate credentials are less secure than cookie surrogate credentials and should be avoided if possible. If multiple clients share an IP address (such as when they are behind a NAT firewall or on a multi-user system), the IP surrogate mechanism cannot distinguish between those users

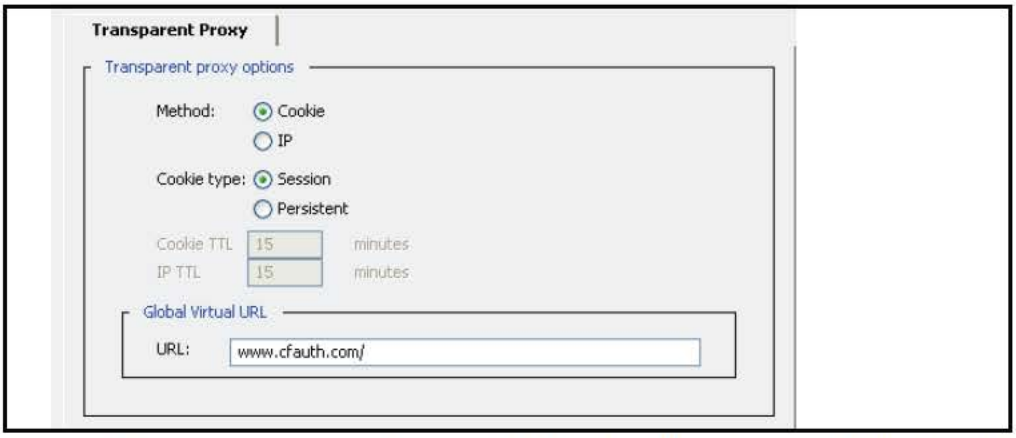
Configuring Transparent Proxy Authentication

The following sections provide general instructions on configuring for transparent proxy authentication.

In addition to configuring transparent proxy authentication, you must also enable a transparent proxy port before the transparent proxy is functional. To enable a transparent proxy port, refer to *Volume 3: Proxies and Proxy Services*.

To set transparent proxy options:

1. Select **Configuration > Authentication > Transparent Proxy**.



2. Select the transparent proxy method—Cookie-based or IP address-based. The default is **Cookie**.

   If you select **Cookie**, the **Cookie Type** radio buttons are available. Click either: **Session**, for cookies that are deleted at the end of a session, or **Persistent**, for cookies that remain on a client machine until the cookie TTL (Time To Live) is reached or the credentials cache is flushed. The default is Session.

   If you select **Persistent Cookies**, enter the Cookie TTL. If you choose **IP** address-based, enter the IP address TTL. The default for each is 15 minutes.

   Note: A value of 0 (zero) for the IP address TTL re-prompts the user for credentials once the specified cache duration for the particular realm has expired.

   For authentication modes that make use of IP surrogate credentials, once the IP address TTL expires the proxy re-challenges all client requests that do not contain credentials for which an IP surrogate credential cache entry previously existed.

   If at this point the client supplied a different set of credentials than previously used to authenticate—for which an entry in the user credential cache still exists—the proxy fails authentication. This is to prevent any another client to potentially gain network access by impersonating another user by supplying his or her credentials. However, once the user credential cache entry's TTL has expired, you can supply a different set of credentials than previously used for authentication.

   ---

3. Select the Virtual URL. The default is `www.cfauth.com`. Blue Coat recommends you change the virtual hostname to something meaningful to you, preferably the IP address of the SG appliance, unless you are doing secure credentials over SSL. Using the IP address of the SG appliance enables you to be sure that the correct SG appliance is addressed in a cluster configuration.

   Note: To later delete or change the virtual URL, enter quote marks (“”) in the virtual URL window and click **Apply**. This removes the current URL.

   ---

4. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Set Transparent Proxy Options

```
SGOS#(config) security transparent-proxy-auth method {cookie | ip}
SGOS#(config) security transparent-proxy-auth cookie {persistent |
session}
SGOS#(config) security transparent-proxy-auth time-to-live persistent-
cookie minutes
SGOS#(config) security transparent-proxy-auth time-to-live ip minute
SGOS#(config) security transparent-proxy-auth cookie virtual-url url
```

Using SSL with Authentication and Authorization Services

Blue Coat recommends that you use SSL during authentication to secure your user credentials. Blue Coat now supports SSL between the client and the SG appliance and between the SG appliance to LDAP and IWA authentication servers.

Using SSL Between the Client and the SG Appliance

To configure SSL for to use origin-cookie-redirect or origin-ip-redirect challenges, you must:

- Specify a virtual URL with the HTTPS protocol (for example, `https://virtual_address`.
- Create a keyring and certificate on the SG appliance.
- Create an HTTPS service to run on the port specified in the virtual URL and to use the keyring you just created.

Note: You can use SSL between the client and the SG appliance for origin-style challenges on transparent and explicit connections (SSL for explicit proxy authentication is not supported).

In addition, if you use a forward proxy, the challenge type must use redirection; it cannot be an origin or origin-ip challenge type.

When redirected to the virtual URL, the user is prompted to accept the certificate offered by the SG appliance (unless the certificate is signed by a trusted certificate authority). If accepted, the authentication conversation between the SG appliance and the user is encrypted using the certificate.

Note: If the hostname does not resolve to the IP address of the SG appliance, then the network configuration must redirect traffic for that port to the appliance. Also, if you use the IP address as the virtual hostname, you might have trouble getting a certificate signed by a CA-Certificate authority (which might not be important).

For information on creating a keyring and a certificate, refer to *Volume 3: Proxies and Proxy Services*.

You can use SSL between the SG appliance and IWA and LDAP authentication servers. For more information, see " SSL Between the SG Appliance and the Authentication Server" .

Creating a Proxy Layer to Manage Proxy Operations

Once hardware configuration is complete and the system configured to use transparent or explicit proxies, use CPL or VPM to provide on-going management of proxy operations.

Using CPL

Below is a table of all commands available for use in proxy layers of a policy. If a condition, property, or action does not specify otherwise, it can be used only in <Proxy> layers. For information on creating effective CPL, refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide*.

Table 3-1. CPL Commands Available in the <Proxy> Layer

<Proxy> Layer Conditions	Meaning
admin.access=	Tests the administrative access requested by the current transaction. Can also be used in <Admin> layers.
attribute.name=	Tests if the current transaction is authenticated in a RADIUS or LDAP realm, and if the authenticated user has the specified attribute with the specified value. Can also be used in <Admin> layers.
authenticated=	Tests if authentication was requested and the credentials could be verified; otherwise, false. Can also be used in <Admin> layers.
bitrate=	Tests if a streaming transaction requests bandwidth within the specified range or an exact match. Can also be used in <Cache> layers.
category=	Tests if the content categories of the requested URL match the specified category, or if the URL has not been categorized. Can also be used in <Cache> layers.
client_address=	Tests the IP address of the client. Can also be used in <Admin> layers.
client.connection. negotiated_cipher=	Test the cipher suite negotiated with a securely connected client. Can also be used in <Exception> layers.
client.connection. negotiated_cipher. strength=	Test the cipher strength negotiated with a securely connected client. Can also be used in <Exception> layers.
client.host=	Test the hostname of the client (obtained through RDNS). Can also be used in <Admin>, <Forward>, and <Exception> layers.
client.host.has_name=	Test the status of the RDNS performed to determine 'client.host'. Can also be used in <Admin>, <Forward>, and <Exception> layers.
client_protocol=	Tests true if the client transport protocol matches the specification. Can also be used in <Exception> layers.
condition=	Tests if the specified defined condition is true. Can be used in all layers.
console_access=	(This trigger was formerly admin=yes|no.) Tests if the current request is destined for the admin layer. Can also be used in <Cache> and <Exception> layers.
content_management=	(This trigger was formerly content_admin=yes|no.) Tests if the current request is a content-management transaction. Can also be used in <Exception> and <Forward> layers.
date[.utc]=	Tests true if the current time is within the startdate..enddate range, inclusive. Can be used in all layers.
day=	Tests if the day of the month is in the specified range or an exact match. Can be used in all layers.

Table 3-1. CPL Commands Available in the <Proxy> Layer (Continued)

exception.id=	Indicates that the requested object was not served, providing this specific exception page. Can also be used in <Exception> layers.
ftp.method=	Tests ftp request methods against any of a well-known set of FTP methods. Can also be used in <Cache> and <Exception> layers.
group=	Tests if the authenticated condition is set to yes, the client is authenticated, and the client belongs to the specified group. Can also be used in <Admin> layers.
has_attribute.*name*=	Tests if the current transaction is authenticated in an LDAP realm and if the authenticated user has the specified LDAP attribute. Can also be used in <Admin> layers.
hour=	Tests if the time of day is in the specified range or an exact match. Can be used in all layers.
http.method=	Tests HTTP request methods against any of a well known set of HTTP methods. Can also be used in <Cache> and <Exception> layers.
http.method.regex=	Test the HTTP method using a regular expression. Can also be used in <Exception> layers.
http.request_line.regex=	Test the HTTP protocol request line. Can also be used in <Exception> layers.
http.request.version=	Tests the version of HTTP used by the client in making the request to the SG appliance. Can also be used in <Cache> and <Exception> layers.
http.response_code=	Tests true if the current transaction is an HTTP transaction and the response code received from the origin server is as specified. Can also be used in <Cache> and <Exception> layers.
http.response.version=	Tests the version of HTTP used by the origin server to deliver the response to the SG appliance. Can also be used in <Cache> and <Exception> layers.
http.transparent_ authentication=	This trigger evaluates to true if HTTP uses transparent proxy authentication for this request. Can also be used in <Cache> and <Exception> layers.
im.buddy_id=	Tests the buddy_id associated with the IM transaction. Can also be used in <Exception> layers.
im.chat_room.conference=	Tests whether the chat room associated with the transaction has the conference attribute set. Can also be used in <Exception> layers.
im.chat_room.id=	Tests the chat room ID associated with the transaction. Can also be used in <Exception> layers.
im.chat_room.invite_ only=	Tests whether the chat room associated with the transaction has the invite_only attribute set. Can also be used in <Exception> layers.
im.chat_room.type=	Tests whether the chat room associated with the transaction is public or private. Can also be used in <Exception> layers.
im.chat_room.member=	Tests whether the chat room associated with the transaction has a member matching the specified criterion. Can also be used in <Exception> layers.
im.chat_room.voice_ enabled=	Tests whether the chat room associated with the transaction is voice enabled. Can also be used in <Exception> layers.
im.client=	Test the type of IM client in use. Can also be used in <Exception>, <Forward>, and <Cache> layers.

Table 3-1. CPL Commands Available in the <Proxy> Layer (Continued)

`im.file.extension=`	Tests the file extension. Can also be used in `<Exception>` layers.
`im.file.`*`name`*`=`	Tests the file name (the last component of the path), including the extension. Can also be used in `<Exception>` layers.
`im.file.path=`	Tests the file path against the specified criterion. Can also be used in `<Exception>` layers.
`im.file.size=`	Performs a signed 64-bit range test. Can also be used in `<Exception>` layers.
`im.message.reflected`	Test whether IM reflection occurred. Can also be used in `<Exception>` and `<Forward>` layers.
`im.message.route=`	Tests how the IM message reaches its recipients. Can also be used in `<Exception>` layers.
`im.message.size=`	Performs a signed 64-bit range test. Can also be used in `<Exception>` layers.
`im.message.text.substring=`	Performs a signed 64-bit range test. Can also be used in `<Exception>` layers.
`im.message.opcode=`	Tests the value of an opcode associated with an `im.method` of `send_unknown` or `receive_unknown`.
`im.message.type=`	Tests the message type. Can also be used in `<Exception>` layers.
`im.method=`	Tests the method associated with the IM transaction. Can also be used in `<Cache>` and `<Exception>` layers.
`im.user_id=`	Tests the user_id associated with the IM transaction. Can also be used in `<Exception>` layers.
`live=`	Tests if the streaming content is a live stream. Can also be used in `<Cache>` layers.
`minute=`	Tests if the minute of the hour is in the specified range or an exact match. Can be used in all layers.
`month=`	Tests if the month is in the specified range or an exact match. Can be used in all layers.
`proxy.address=`	Tests the IP address of the network interface card (NIC) on which the request arrives. Can also be used in `<Admin>` layers.
`proxy.card=`	Tests the ordinal number of the network interface card (NIC) used by a request. Can also be used in `<Admin>` layers.
`proxy.port=`	Tests if the IP port used by a request is within the specified range or an exact match. Can also be used in `<Admin>` layers.
`raw_url`	Test the value of the raw request URL. Can also be used in `<Exception>` layers.
`raw_url.host`	Test the value of the 'host' component of the raw request URL. Can also be used in `<Exception>` layers.
`raw_url.path`	Test the value of the 'path' component of the raw request URL. Can also be used in `<Exception>` layers.
`raw_url.pathquery`	Test the value of the 'path and query' component of the raw request URL. Can also be used in `<Exception>` layers.

Table 3-1. CPL Commands Available in the <Proxy> Layer (Continued)

raw_url.port	Test the value of the 'port' component of the raw request URL. Can also be used in <Exception> layers.
raw_url.query	Test the value of the 'query' component of the raw request URL. Can also be used in <Exception> layers.
realm=	Tests if the authenticated condition is set to yes, the client is authenticated, and the client has logged into the specified realm. an also be used in <Admin> layers.
release.id=	Tests the SG release ID. Can be used in all layers.
request.header_address.*header_name*=	Tests if the specified request header can be parsed as an IP address. Can also be used in <Cache> layers.
request.header.*header_name*=	Tests the specified request header (*header_name*) against a regular expression. Can also be used in <Cache> layers.
request.header.*header_name*.count	Test the number of header values in the request for the given header_name. Can also be used in <Exception> layers.
request.header.*header_name*.length	Test the total length of the header values for the given header_name. Can also be used in <Exception> layers.
request.header.Referer.url.host.has_name=	Test whether the Referer URL has a resolved DNS hostname. Can also be used in <Exception> layers.
request.header.Referer.url.is_absolute	Test whether the Referer URL is expressed in absolute form. Can also be used in <Exception> layers.
request.raw_headers.count	Test the total number of HTTP request headers. Can also be used in <Exception> layers.
request.raw_headers.length	Test the total length of all HTTP request headers. Can also be used in <Exception> layers.
request.raw_headers.regex	Test the value of all HTTP request headers with a regular expression. Can also be used in <Exception> layers.
request.x_header.*header_name*.count	Test the number of header values in the request for the given *header_name*. Can also be used in <Exception> layers.
request.x_header.*header_name*.length	Test the total length of the header values for the given *header_name*. Can also be used in <Exception> layers.
response.header.*header_name*=	Tests the specified response header (*header_name*) against a regular expression. Can also be used in <Cache> layers.
response.x_header.*header_name*=	Tests the specified response header (*header_name*) against a regular expression. Can also be used in <Cache> layers.
server_url[.case_sensitive|.no_lookup]=	Tests if a portion of the requested URL exactly matches the specified pattern. Can also be used in <Forward> layers.
socks.accelerated=	Controls the SOCKS proxy handoff to other protocol agents.
socks.method=	Tests the protocol method name associated with the transaction. Can also be used in <Cache> and <Exception> layers.
socks.version=	Switches between SOCKS 4/4a and 5. Can also be used in <Exception> and <Forward> layers.

Table 3-1. CPL Commands Available in the <Proxy> Layer (Continued)

streaming.content=	(This trigger has been renamed from streaming.) Can also be used in <Cache>, <Exception>, and <Forward> layers.
time=	Tests if the time of day is in the specified range or an exact match. Can be used in all layers.
tunneled=	
url.domain=	Tests if the requested URL, including the domain-suffix portion, matches the specified pattern. Can also be used in <Forward> layers.
url.extension=	Tests if the filename extension at the end of the path matches the specified string. Can also be used in <Forward> layers.
url.host=	Tests if the host component of the requested URL matches the IP address or domain name. Can also be used in <Forward> layers.
url.host.has_name	Test whether the request URL has a resolved DNS hostname. Can also be used in <Exception> layers
url.is_absolute	Test whether the request URL is expressed in absolute form. Can also be used in <Exception> layers
url.host.is_numeric=	This is true if the URL host was specified as an IP address. Can also be used in <Forward> layers.
url.host.no_name=	This is true if no domain name can be found for the URL host. Can also be used in <Forward> layers.
url.host.regex=	Tests if the specified regular expression matches a substring of the domain name component of the request URL. Can also be used in <Forward> layers.
url.host.suffix=	Can also be used in <Forward> layers.
url.path=	Tests if a prefix of the complete path component of the requested URL, as well as any query component, matches the specified string. Can also be used in <Forward> layers.
url.path.regex=	Tests if the regex matches a substring of the path component of the request URL. Can also be used in <Forward> layers.
url.port=	Tests if the port number of the requested URL is within the specified range or an exact match. Can also be used in <Forward> layers.
url.query.regex=	Tests if the regex matches a substring of the query string component of the request URL. Can also be used in <Forward> layers.
url.regex=	Tests if the requested URL matches the specified pattern. Can also be used in <Forward> layers.
url.scheme=	Tests if the scheme of the requested URL matches the specified string. Can also be used in <Forward> layers.
user=	Tests the authenticated user name of the transaction. Can also be used in <Admin> layers.
user.domain=	Tests if the authenticated condition is set to yes, the client is authenticated, the logged-into realm is an IWA realm, and the domain component of the user name is the specified domain. Can also be used in <Admin> layers.
weekday=	Tests if the day of the week is in the specified range or an exact match. Can be used in all layers.

Table 3-1. CPL Commands Available in the <Proxy> Layer (Continued)

`year=`	Tests if the year is in the specified range or an exact match. Can be used in all layers.

Table 3-2. Properties Available in the <Proxy> Layer

<Proxy> Layer Properties	Meaning
`action.`*`action_label`*`( )`	Selectively enables or disables a specified define action block. Can also be used in `<Cache>` layers.
`allow`	Allows the transaction to be served. Can be used in all layers except `<Exception>` and `<Forward>` layers.
`always_verify( )`	Determines whether each request for the objects at a particular URL must be verified with the origin server.
`authenticate( )`	Identifies a realm that must be authenticated against. Can also be used in <Admin> layers.
`authenticate.force( )`	Either disables proxy authentication for the current transaction (using the value `no`) or requests proxy authentication using the specified authentication realm. Can also be used in `<Admin>` layers.
`authenticate.form( )`	When forms-based authentication is in use, authenticate.form ( ) selects the form used to challenge the user.
`authenticate.mode(auto)` `authenticate.mode(sg2)`	Setting the authentication.mode property selects a challenge type and surrogate credential combination. In `auto` mode, explicit IWA uses connection surrogate credentials. In `sg2.mode`, explicit IWA uses IP surrogate credentials.
`authenticate.redirect_stored_requests`	Sets whether requests stored during forms-based authentication can be redirected if the upstream host issues a redirecting response.
`bypass_cache( )`	Determines whether the cache is bypassed for a request.
`check_authorization( )`	In connection with CAD (Caching Authenticated Data) and CPAD (Caching Proxy Authenticated Data) support, `check_authorization( )` is used when you know that the upstream device will sometimes (not always or never) require the user to authenticate and be authorized for this object. Can also be used in `<Cache>` layers.
`delete_on_abandonment( )`	If set to yes, then if all clients requesting an object close their connections prior to the object being delivered, the object fetch from the origin server is abandoned. Can also be used in `<Cache>` layers.
`deny`	Denies service. Can be used in all layers except `<Exception>` and `<Forward>` layers.
`dynamic_bypass( )`	Used to indicate that a particular transparent request should not be handled by the proxy, but instead be subjected to our dynamic bypass methodology.
`exception( )`	Indicates not to serve the requested object, but instead serve this specific exception page. Can be used in all layers except `<Exception>` layers.
`ftp.server_connection( )`	Determines when the control connection to the server is established.
`ftp.welcome_banner( )`	Sets the welcome banner for a proxied FTP transaction.
`http.client.recv.timeout`	Sets the socket timeout for receiving bytes from the client.

Table 3-2. Properties Available in the <Proxy> Layer (Continued)

`http.request.version( )`	The `http.request.version( )` property sets the version of the HTTP protocol to be used in the request to the origin content server or upstream proxy. Can also be used in `<Cache>` layers.
`http.response.parse_meta_tag. Cache-Control( )`	Controls whether the 'Cache-Control' META Tag is parsed in an HTML response body. Can also be used in `<Cache>` layers.
`http.response.parse_meta_tag. Expires`	Controls whether the 'Expires' META Tag is parsed in an HTML response body. Can also be used in `<Cache>` layers.
`http.response.parse_meta_tag. Pragma.no-cache`	Controls whether the 'Pragma: no-cache' META Tag is parsed in an HTML response body. Can also be used in `<Cache>` layers.
`http.response.version( )`	The `http.response.version( )` property sets the version of the HTTP protocol to be used in the response to the client's user agent.
`http.server.recv. timeout( )`	Sets the socket timeout for receiving bytes from the upstream host. Can also be used in `<Forward>` layers.
`im.block_encryption`	Prevents the encryption of AOL IM messages by modifying messages during IM login time.
`im.reflect`	Sets whether IM reflection should be attempted.
`im.strip_attachments( )`	Determines whether attachments are stripped from IM messages.
`im.transport`	Sets the type of upstream connection to make for IM traffic.
`log.suppress.`*`field-id`*`( )`	The `log.suppress.`*`field-id`*`( )` controls suppression of the specified field-id in all facilities (individual logs that contain all properties for that specific log in one format). Can be used in all layers.
`log.suppress.`*`field-id [log_list]( )`	The `log.suppress.`*`field-id [log_list]( )` property controls suppression of the specified field-id in the specified facilities. Can be used in all layers.
`log.rewrite.`*`field-id`*`( )`	The `log.rewrite.`*`field-id`*`( )` property controls rewrites of a specific log field in all facilities. Can be used in all layers.
`log.rewrite.`*`field-id [log_list]( )`	The `log.rewrite.`*`field-id [log_list]( )` property controls rewrites of a specific log field in a specified list of log facilities. Can be used in all layers.
`reflect_ip( )`	Determines how the client IP address is presented to the origin server for explicitly proxied requests. Can also be used in `<Forward>` layers.
`request.filter_service( )`	Websense is the built in service name for the off-box content filtering service. Can also be used in `<Cache>` layers.
`request.icap_service( )`	Determines whether a request from a client should be processed by an external ICAP service before going out.
`shell.prompt`	Sets the prompt for a proxied Shell transaction.
`shell.realm_banner`	Sets the realm banner for a proxied Shell transaction.
`shell.welcome_banner`	Sets the welcome banner for a proxied Shell transaction.
`socks.accelerate( )`	The `socks.accelerate` property controls the SOCKS proxy handoff to other protocol agents.

Table 3-2. Properties Available in the <Proxy> Layer (Continued)

`socks.authenticate( )`	The same realms can be used for SOCKS proxy authentication as can be used for regular proxy authentication.
`socks.authenticate. force( )`	The `socks.authenticate.force( )` property forces the realm to be authenticated through SOCKS.

Table 3-3. Actions Available in the <Proxy> Layer

<Proxy> Layer Actions	Meaning
`log_message( )`	Writes the specified string to the SG event log. Can be used in all layers except `<Admin>`.
`notify_email( )`	Sends an e-mail notification to the list of recipients specified in the Event Log mail configuration. Can be used in all layers.
`notify_snmp( )`	The SNMP trap is sent when the transaction terminates. Can be used in all layers.
`redirect( )`	Ends the current HTTP transaction and returns an HTTP redirect response to the client.
`transform`	Invokes the active content or URL rewrite transformer.

Chapter 4: Understanding and Managing X.509 Certificates

Blue Coat uses certificates for various applications, including:

- authenticating the identity of a server
- authenticating an SG appliance
- securing an intranet
- encrypting data

The certificates Blue Coat uses are X.509 certificates. X.509 is a cryptographic standard for public key infrastructure (PKI) that specifies standard formats for public key certificates. Several RFCs and books exist on the public key cryptographic system (PKCS). This discussion of the elements of PKCS is relevant to their implementation in SGOS.

Section A: Concepts

This section discusses concepts surrounding certificates and SGOS.

Public Keys and Private Keys

In PKCS systems, the intended recipient of encrypted data generates a private/public keypair, and publishes the public key, keeping the private key secret. The sender encrypts the data with the recipient's public key, and sends the encrypted data to the recipient. The recipient uses the corresponding private key to decrypt the data.

For two-way encrypted communication, the endpoints can exchange public keys, or one endpoint can choose a symmetric encryption key, encrypt it with the other endpoint's public key, and send it.

Certificates

The SGOS software uses:

- SSL Certificates.
- CA Certificates.
- External Certificates.

You can also use wildcard certificates during HTTPS termination. Microsoft's implementation of wildcard certificates is as described in RFC 2595, allowing an * (asterisk) in the leftmost-element of the server's common name only. For information on wildcards supported by Internet Explorer, refer to the Microsoft knowledge base, article: 258858. Any SSL certificate can contain a common name with wildcard characters.

SSL Certificates

SSL certificates are used to authenticate the identity of a server or a client. A certificate is confirmation of the association between an identity (expressed as a string of characters) and a public key. If a party can prove they hold the corresponding private key, you can conclude that the party is who the certificate says it is. The certificate contains other information, such as its expiration date.

The association between a public key and a particular server is done by generating a certificate signing request using the server's or client's public key. A certificate signing authority (CA) verifies the identity of the server or client and generates a signed certificate. The resulting certificate can then be offered by the server to clients (or from clients to servers) who can recognize the CA's signature. Such use of certificates issued by CAs has become the primary infrastructure for authentication of communications over the Internet.

The SG trusts all root CA certificates trusted by Internet Explorer and Firefox. The list is updated periodically to be in sync with the latest versions of IE and Firefox.

CA certificates installed on the SG are used to verify the certificates presented by HTTPS servers and the client certificates presented by browsers. Browsers offer a certificate if the server is configured to ask for one and an appropriate certificate is available to the browser.

Section A: Concepts

CA Certificates

CA certificates are certificates that belong to certificate authorities. CA certificates are used by SGdevices to verify X.509 certificates presented by a client or a server during secure communication. SG appliances are pre-installed with the most common CA certificates.

SG appliances come with many popular CA certificates already installed. You can review these certificates using the Management Console or the CLI. You can also add certificates for your own internal certificate authorities.

External Certificates

An external certificate is any X509 certificate for which the SG appliance does not have the private key. The certificate can be used to encrypt data, such as access logs, with a public key so that it can only be decrypted by someone who has the corresponding private key. Refer to *Volume 9: Access Logging* for information about encrypting access logs.

Keyrings

A keyring contains a public/private keypair. It can also contain a certificate signing request or a signed certificate. Keyrings are named, can be created, deleted and viewed; there are built-in keyrings for specified purposes. For information on managing keyrings, see Section B: "Using Keyrings and SSL Certificates" on page 41.

Cipher Suites Supported by SGOS Software

A cipher suite specifies the algorithms used to secure an SSL connection. When a client makes an SSL connection to a server, it sends a list of the cipher suites that it supports.

The server compares this list with its own supported cipher suites and chooses the first cipher suite proposed by the client that they both support. Both the client and server then use this cipher suite to secure the connection.

Note: You can delete cipher suites that you do not trust. However, SGOS does not provide any mechanism to change the ordering of the ciphers used.

All cipher suites supported by the SG appliance use the RSA key exchange algorithm, which uses the public key encoded in the server's certificate to encrypt a piece of secret data for transfer from the client to server. This secret is then used at both endpoints to compute encryption keys.

By default, the SG appliance is configured to allow SSLv2 and v3 as well as TLSv1 traffic. The cipher suites available for use differ depending on whether you configure SSL for version 2, version 3, TLS, or a combination of these.

Table 4-1. Cipher Suites Shipped with the SG Appliance

SGOS Cipher #	Cipher Name	Strength	Exportable	Description
1	RC4-MD5	Medium	No	128-bit key size.
2	RC4-SHA	Medium	No	128-bit key size.
3	DES-CBC3-SHA	High	No	168-bit key size.

Table 4-1. Cipher Suites Shipped with the SG Appliance (Continued)

SGOS Cipher #	Cipher Name	Strength	Exportable	Description
4	DES-CBC3-MD5	High	No	168-bit key size.
5	RC2-CBC-MD5	Medium	No	128-bit key size.
6	RC4-64-MD5	Low	No	64-bit key size.
7	DES-CBC-SHA	Low	No	56-bit key size.
8	DES-CBC-MD5	Low	No	56-bit key size.
9	EXP1024-RC4-MD5	Export	Yes	56-bit key size.
10	EXP1024-RC4-SHA	Export	Yes	56-bit key size.
11	EXP1024-RC2-CBC-MD5	Export	Yes	56-bit key size.
12	EXP1024-DES-CBC-SHA	Export	Yes	56-bit key size.
13	EXP-RC4-MD5	Export	Yes	40-bit key size.
14	EXP-RC2-CBC-MD5	Export	Yes	40-bit key size.
15	EXP-DES-CBC-SHA	Export	Yes	40-bit key size.
16	AES128-SHA	Medium	No	128-bit key size.
17	AES256-SHA	High	No	256-bit key size.

Cipher Suite configuration is discussed in "Changing the Cipher Suites of the SSL Client" on page 174.

Server-Gated Cryptography and International Step-Up

Due to US export restrictions, international access to a secure site requires that the site negotiates export-only ciphers. These are relatively weak ciphers ranging from 40-bit to 56-bit key lengths, and are vulnerable to attack.

Server Gated Cryptography (SGC) is a Microsoft extension to the certificate that allows the client receiving the certificate to first negotiate export strength ciphers, followed by a re-negotiation with strong ciphers. Netscape has a similar extension called International Step-up.

SGOS supports both SGC and International Step-up in its SSL implementation. There are, however, known anomalies in Internet Explorer's implementation that can cause SSL negotiation to fail. Refer to the following two documents for more detail and check for recent updates on the Microsoft support site.

```
http://support.microsoft.com/support/kb/articles/Q249/8/63.ASP
http://support.microsoft.com/support/kb/articles/Q244/3/02.ASP
```

To take advantage of this technology, SGOS supports VeriSign's Global ID Certificate product. The Global ID certificate contains the extra information necessary to implement SGC and International Step-up.

Section B: Using Keyrings and SSL Certificates

Keyrings are virtual containers, holding a public/private keypair with a customized keylength and a certificate or certificate signing request.

Certificates can be meant for internal use (self-signed) or they can be meant for external use.

In general, SSL certificates involve three parties:

- The subject of the certificate.
- The Certificate Authority (CA), which signs the certificate, attesting to the binding between the public key in the certificate and the subject.
- The "relying party," which is the entity that trusts the CA and relies on the certificate to authenticate the subject.

Keyrings and certificates are used in:

- Encrypting data.
- Digitally Signing Access Logs.
- Authenticating end users.
- Authenticating an SG appliance.

The steps in creating keyrings and certificates include:

- Create a keyring. A default keyring is shipped with the system and is used for accessing the Management Console, although you can use others. You can also use the default keyring for other purposes. You can create other keyrings for each SSL service. (See "Creating a Keyring" on page 42.)

  Note: You can also import keyrings. For information on importing keyrings, see "Importing an Existing Keypair and Certificate" on page 53.

- (Optional) Create Certificate Signing Requests (CSRs) to be sent to Certificate Signing Authorities (CAs).
- Import X.509 certificates issued by trusted CA authorities for external use and associate them with the keyring. (See "Managing SSL Certificates" on page 46.)

  -or-

  Create certificates and associate them with the keyring. (See "Creating Self-Signed SSL Certificates" on page 47.)
- (Optional, if using SSL Certificates from CAs) Import Certificate Revocation Lists (CRLs) so the SG appliance can verify that certificates are still valid.
- Creating an HTTP Reverse Proxy Service and associating the keyring with the service. (Refer to *Volume 3: Proxies and Proxy Services*.)

Note: These steps must be done using a secure connection such as HTTPS, SSH, or a serial console.

Creating a Keyring

The SG appliance ships with three keyrings already created:

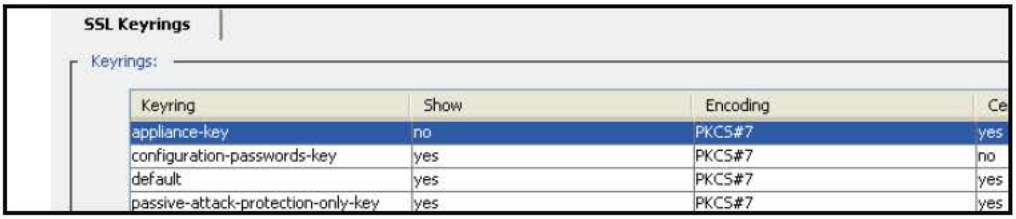
- default: The default keyring contains a certificate and an automatically-generated keypair. The default keyring is intended for securely accessing the SG appliance Management Console. Create an additional keyring for each HTTPS service defined.
- configuration-passwords-key: The `configuration-passwords-key` keyring contains a keypair but does not contain a certificate. This keyring is used to encrypt passwords in the `show config` command and should not be used for other purposes.
- appliance-key: The `appliance-key` keyring contains an internally-generated keypair. If the SG appliance is authenticated (has obtained a certificate from the Blue Coat CA appliance-certificate server), that certificate is associated with this keyring, which is used to authenticate the device. (For more information on authenticating the SG appliance, refer to *Volume 6: Advanced Networking*.)

Note: The `appliance-key` keyring is used by the system. It is not available for other purposes.

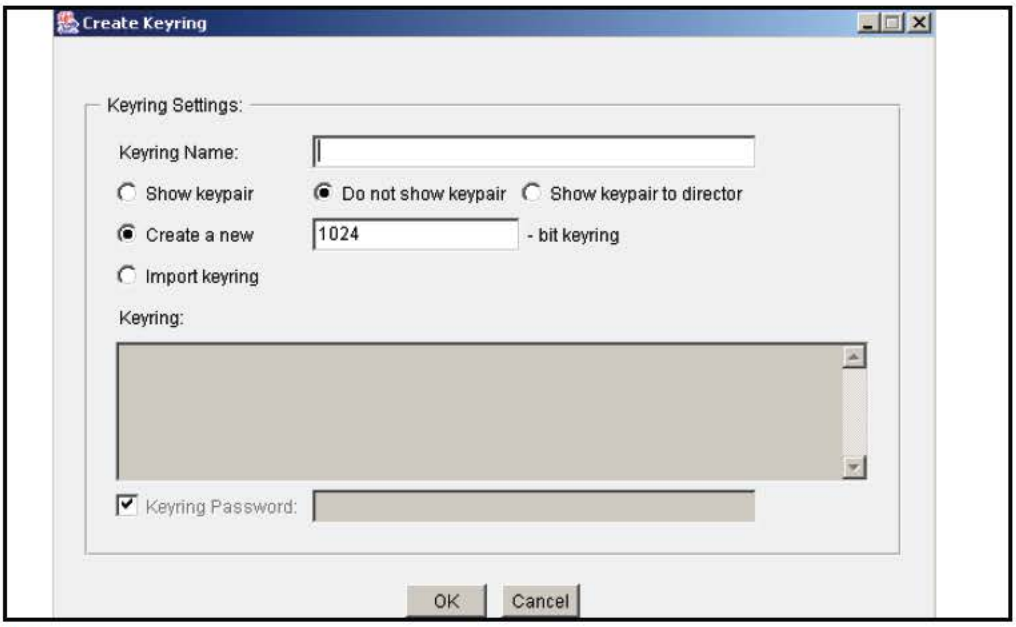
If an origin content server requires a client certificate and no keyring is associated with the SG appliance SSL client, the HTTPS connections fails. For information on using the SSL client, see Appendix C: "Managing the SSL Client" on page 173.

To create a keyring:

1. Select **Configuration > SSL > Keyrings > SSL Keyrings**.



2. Click **Create**; the **Create Keyring** dialog appears.



3. Fill in the pane:

- **Keyring Name:** Give the keyring a meaningful name.

  Note: Spaces in keyring names are not supported. Including a space can cause unexpected errors while using such keyrings.

- Select the show option you need:
  - **Show keypair** allows the keys to be viewed and exported.
  - **Do not show keypair** prevents the keypair from being viewed or exported.
  - **Show keypair to director** is a keyring viewable only if Director is issuing the command using a SSH-RSA connection.

  Note: The choice among **show, do not show keypair,** and **show keypair to director** has implications for whether keyrings are included in profiles and backups created by Director. For more information, refer to the *Blue Coat Director User Guide*.

- Select the key length in the **Create a new** _____ **-bit keyring** field. A length of 1024 bits is the maximum (and default). For deployments reaching outside the U.S., determine the maximum key length allowed for export.

  Click **OK**. The keyring is created with the name you chose. It does not have a certificate associated with it yet. To associate a certificate, see "Importing a Server Certificate" on page 48.

  -or-

- Select the **Import keyring** radio button.

The grayed-out **Keyring** field becomes enabled, allowing you to paste in an already existing private key. Any certificate or certificate request associated with this private key must be imported separately. For information on importing a certificate, see "Importing a Server Certificate" on page 48.

If the private key that is being imported has been encrypted with a password, select **Keyring Password** and enter the password into the field.

Note: The only way to retrieve a keyring's private key from the SG appliance is by using Director or the command line —it cannot be exported through the Management Console.

4. Click **OK**.

To view or edit a keyring:

1. Select **Configuration > SSL > Keyrings > SSL Keyrings**.
2. Click **View/Edit**.

Related CLI Syntax to Create an SSL Keyring

```
SGOS#(config) ssl
SGOS#(config ssl) create keyring {show | show-director | no-show}
keyring_id [key_length]
```

Notes

- To view the keypair in an encrypted format, you can optionally specify `des` or `des3` before the *`keyring_id`*, along with an optional password. If the optional password is provided on the command line, the CLI does not prompt for a password.
- If the optional password is not provided on the command line, the CLI asks for the password (interactive). If you specify either `des` or `des3`, you are prompted.
- To view the keypair in unencrypted format, select either the optional *`keyring_id`* or use the `unencrypted` command option.
- You cannot view a keypair over a Telnet connection because of the risk that it could be intercepted.

Deleting an Existing Keyring and Certificate

To delete a keyring and the associated certificate:

1. Select **Configuration > SSL > Keyrings > SSL Keyrings**.
2. Highlight the name of the keyring to delete.
3. Click **Delete**.

   The Confirm delete dialog appears.
4. Click **OK** in the Confirm delete dialog.

Related CLI Syntax to Delete a Keyring and the Associated Certificate

```
SGOS#(config) ssl
SGOS#(config ssl) delete keyring keyring_id
```

Section C: Managing Certificates

This section discusses how to manage certificates, from obtaining certificate signing requests to using certificate revocation lists.

In this section are:

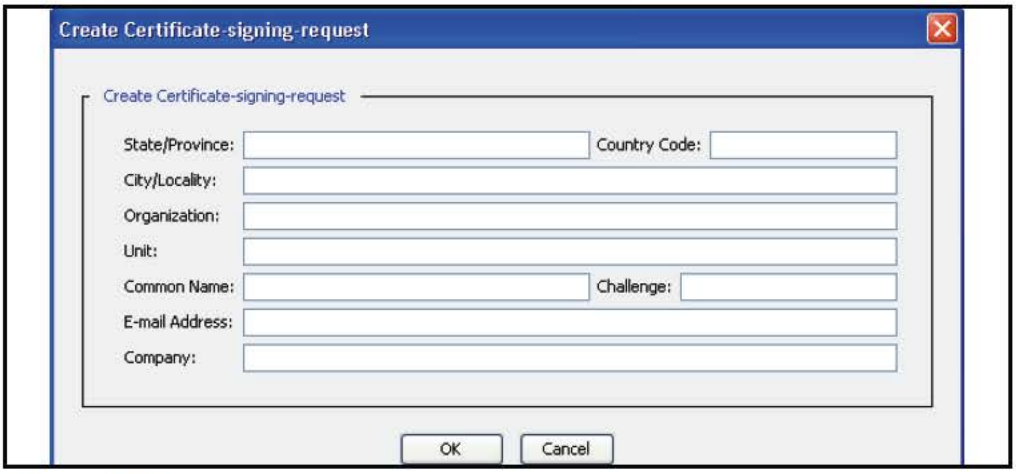
Managing Certificate Signing Requests

Certificate signing requests (CSRs) are used to obtain a certificate signed by a Certificate Authority. You can also create CSRs off box.

Creating a CSR

To create a CSR:

1. Select **Configuration > SSL > SSL Keyrings;** click **Edit/View**.
2. From the drop-down list, select the keyring for which you need a signed certificate.
3. From the **Certificate Signing Request** tab, click the **Create** button.



4. Fill in the fields:
   - **State/Province**—Enter the state or province where the machine is located.
   - **Country Code**—Enter the two-character ISO code of the country.
   - **City/Locality**—Enter the city.
   - **Organization**—Enter the name of the company.
   - **Unit**—Enter the name of the group that is managing the machine.
   - **Common Name**—Enter the URL of the company.
   - **Challenge**—Enter a 4-16 character alphanumeric challenge.

- **E-mail Address**—The e-mail address you enter must be 40 characters or less. A longer e-mail address generates an error.
- **Company**—Enter the name of the company.

5. The **Create** tab displays the message: **Creating....**
6. Click **OK**.

Related CLI Syntax to Create a CSR

```
SGOS#(config) ssl
SGOS#(config ssl) create signing-request keyring_id
SGOS#(config ssl) create signing-request keyring_id [attribute_value]
[attribute_value]
```

Viewing a Certificate Signing Request

Once a CSR is created, you must submit it to a CA in the format the CA requires. You can view the output of a certificate signing request either through the Management Console or the CLI.

To view the output of a certificate signing request:

1. Select **Configuration > SSL > SSL Keyrings**.
2. Click **Edit/View**.
3. From the drop-down list, select the keyring for which you have created a certificate signing request.

   The certificate signing request displays in the Certificate Signing Request window and can be copied for submission to a CA.

Managing SSL Certificates

SSL certificates can be obtained two ways:

- Created on the SG appliance as a self-signed certificate

  To create a SSL self-signed certificate on the SG appliance using a Certificate Signing Request, continue with the next section.

- Imported after receiving the certificate from the signing authority

  If you plan to use SSL certificates issued by Certificate Authorities, the procedure is:

  - Obtain the keypair and Certificate Signing Requests (CSRs), either off box or on box, and send them to the Certificate Authority for signing.
  - After the signed request is returned to you from the CA, you can import the certificate into the SG appliance. To import a certificate, see "Importing a Server Certificate" on page 48.

Note: If a Website presents a certificate that is signed by a CA not on Blue Coat default CA list, you might see the following message:

Network Error (ssl_failed)
A secure SSL session could not be established with the Web Site:

You must import the CA Certificate onto the SG appliance before the device can trust the site.

To import an SSL Certificate, skip to "Importing a Server Certificate" on page 48.

Creating Self-Signed SSL Certificates

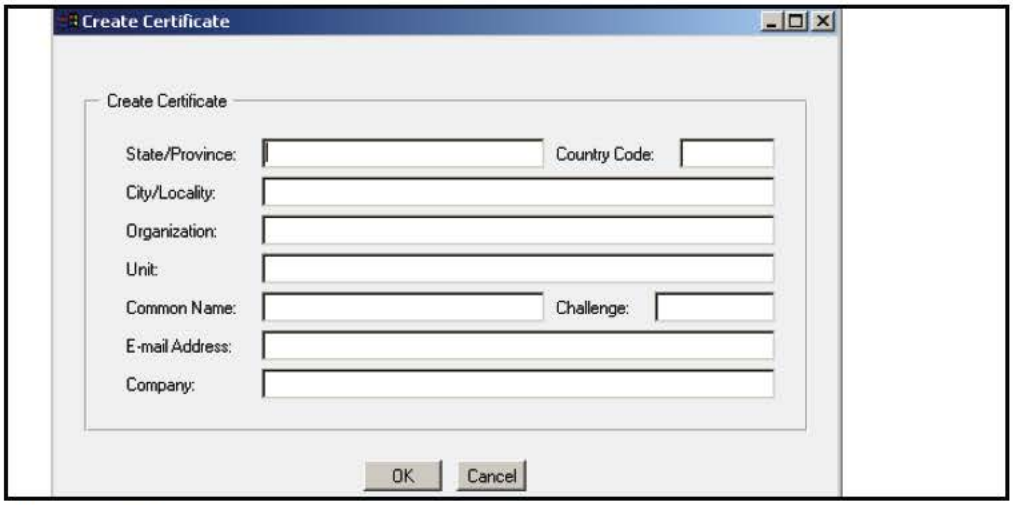
The SG appliance ships with a self-signed certificate, associated with the default keyring. Only one certificate can be associated with a keyring. If you have multiple uses, use a different keyring and associated certificate for each one.

Adding a Self-Signed SSL Certificate

Self-signed certificates are generally meant for intranet use, not Internet.

To create a self-signed certificate:

1. Select **Configuration > SSL > Keyrings > SSL Keyrings**.
2. Highlight the keyring for which you want to add a certificate.
3. Click **Edit/View** in the **Keyring** tab.
4. Click **Create**.



5. Fill in the fields:
   - **State/Province**—Enter the state or province where the machine is located.
   - **Country Code**—Enter the two-character ISO code of the country.
   - **City/Locality**—Enter the city.
   - **Organization**—Enter the name of the company.

- **Unit**—Enter the name of the group that is managing the machine.
- **Common Name**—A common name should be the one that contains the URL with client access to that particular origin server.
- **Challenge**—Enter a 4-16 character alphanumeric challenge.
- **E-mail Address**—The e-mail address you enter must be 40 characters or less. A longer e-mail address generates an error.
- **Company**—Enter the name of the company.

6. The **Create** tab displays the message: **Creating.....**

Related CLI Syntax to Create a Self-Signed SSL Certificate

```
SGOS#(config ssl) create certificate keyring_id
SGOS#(config ssl) create certificate keyring-id [attribute_value]
[attribute_value]
```

Example:

```
SGOS#(config ssl) create certificate keyring-id cn bluecoat challenge
test c US state CA company bluecoat
```

Importing a Server Certificate

After the CA signs the server certificate and returns it to you, you can import the certificate onto the SG appliance.

To import a server certificate:

1. Copy the certificate to your clipboard. Be sure to include the "Begin Certificate" and "End Certificate" statements.
2. Select **Configuration > SSL > Keyrings**.
3. Highlight the keyring for which you want to import a certificate.
4. Click **Edit/View** in the **Keyrings** tab.
5. In the **Certificate** panel, click **Import**.
6. Paste the certificate you copied into the dialog box. Click **OK**.

   The certificate should display in the SSL Certificates Pane, associated with the keyring you selected earlier.

Using Certificate Revocation Lists

Certificate Revocation Lists (CRLs) enable checking server and client certificates against lists provided and maintained by CAs that show certificates that are no longer valid. Only CRLs that are issued by a trusted issuer can be successfully verified by the SG appliance. The CRL can be imported only when the CRL issuer certificate exists as a CA certificate on the SG appliance.

You can determine if the SG appliance SSL certificates are still valid by checking *Certificate Revocation Lists* (CRLs) that are created and issued by trusted Certificate Signing Authorities. A certificate on the list is no longer valid.

Section C: Managing Certificates

Only CRLs that are issued by a trusted issuer can be verified by the SG appliance successfully. The CRL can be imported only when the CRL issuer certificate exists as a CA certificate on the SG appliance.

SGOS allows:

- One local CRL list per certificate issuing authority.
- An import of a CRL that is expired; a warning is displayed in the log.
- An import of a CRL that is effective in the future; a warning is displayed in the log.

CRLs can be used for the following purposes:

- Checking revocation status of client or server certificates with HTTPS Reverse Proxy.
- Checking revocation status of client or server certificates with SSL proxy. (For more information on using CRLS with the SSL proxy, refer to *Volume 3: Proxies and Proxy Services*.)
- SG appliance-originated HTTPS downloads (secure image download, content filter database download, and the like).
- PEM-encoded CRLs, if cut and pasted through the inline command.
- DER-format (binary) CRLs, if downloaded from a URL.

To import a CRL:

You can choose from among four methods to install a CRL on the SG appliance:

- Use the Text Editor, which allows you to enter the installable list (or copy and paste the contents of an already-created file) directly onto the SG appliance.
- Create a local file on your local system.
- Enter a remote URL, where you placed an already-created file on an FTP or HTTP server to be downloaded to the SG appliance.
- Use the CLI `inline` command.

To update a CRL:

1. Select **Configuration > SSL > CRLs**.
2. Click New or highlight an existing CRL and click **Edit**.
3. Give the CRL a name.
4. From the drop-down list, select the method to use to install the CRL; click **Install**.
   - Remote URL:

     Enter the fully-qualified URL, including the filename, where the CRL is located. To view the file before installing it, click **View**. Click **Install**.

     The **Install CRL** dialog displays. Examine the installation status that displays; click **OK**.
   - Local File:

     Click **Browse** to display the Local File Browse window. Browse for the CRL file on the local system. Open it and click **Install**. When the installation is complete, a results window opens. View the results, close the window, click **Close**.

- **Text Editor:**

  Copy a new CRL file into the window, and click Install.

  When the installation is complete, a results window opens. View the results, close the window, click Close.

  Note: The Management Console text editor can be used to enter a CRL file. You cannot use it to enter CLI commands.

5. Click OK; click **Apply**

Related CLI Syntax to Create a CRL

At the `(config)` command prompt, enter the following commands:

```
SGOS#(config) ssl
SGOS#(config ssl) create crl list_name
or
SGOS#(config) ssl
SGOS#(config ssl) inline crl CRL_list_name eof
Paste CRL here
eof
```

Troubleshooting Certificate Problems

Two common certificate problems are discussed below.

- If the client does not trust the Certificate Signing Authority that has signed the appliance's certificate, an error message similar to the following appears in the event log:

  ```
  2004-02-13 07:29:28-05:00EST "CFSSL:SSL_accept error:14094416:SSL
  routines:SSL3_READ_BYTES:sslv3 alert certificate unknown" 0
  310000:1
  ../cf_ssl.cpp:1398
  ```

  This commonly occurs when you use the HTTPS-Console service on port 8082, which uses a self-signed certificate by default. When you access the Management Console over HTTPS, the browser displays a pop-up that says that the security certificate is not trusted and asks if you want to proceed. If you select **No** instead of proceeding, the browser sends an *unknown CA alert* to the SG appliance.

  You can eliminate the error message one of two ways:

  - If this was caused by the Blue Coat self-signed certificate (the certificate associated with the default keyring), import the certificate as a trusted Certificate Signing Authority certificate . See "Importing a Server Certificate" on page 48 for more information.
  - Import a certificate on the SG appliance for use with HTTPS-Console that is signed by a CA that a browser already trusts.

- If the SG appliance's certificate is not accepted because of a *host name mismatch* or it is an *invalid certificate*, you can correct the problem by creating a new certificate and editing the HTTPS-Console service to use it. For information on editing the HTTPS-Console service, refer to *Volume 3: Proxies and Proxy Services*.

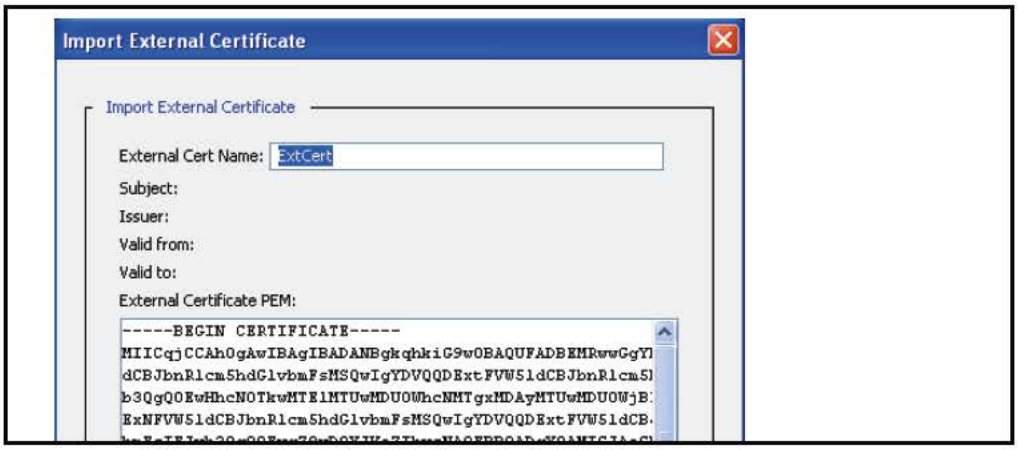
Section D: Using External Certificates

External certificates are certificates for which Blue Coat does not have the private key. The first step in using external certificates is to import the certificates onto the SG appliance.

Importing and Deleting External Certificates

To Import an external certificate:

1. Copy the certificate onto the clipboard.
2. Select **Configuration > SSL > External Certificates**.
3. Click **Import**.



4. Enter the name of the external certificate into the **External Cert Name** field and paste the certificate into the **External Certificate** field. Be sure to include the `----BEGIN CERTIFICATE----` and `-----END CERTIFICATE----` statements.
5. Click **OK**.

Deleting an External Certificate

To delete an external certificate:

1. Select **Configuration>SSL>External Certificates**.
2. Highlight the name of the external certificate to be deleted.
3. Click **Delete**.
4. Click **OK** in the Confirm delete dialog that appears;
5. Select **Apply** to commit the changes to the SG appliance.

Digitally Signing Access Logs

You can digitally sign access logs to certify that a particular SG appliance wrote and uploaded a specific log file. Signing is supported for both content types—text and gzip—and for both upload types—continuous and periodic. Each log file has a signature file

associated with it that contains the certificate and the digital signature used for verifying the log file. When you create a signing keyring (which must be done before you enable digital signing), keep in mind the following:

- The keyring must include a certificate. .
- The certificate purpose must be set for **smime** signing**.** If the certificate purpose is set to anything else, you cannot use the certificate for signing.
- Add the `%c` parameter in the filenames format string to identify the keyring used for signing. If encryption is enabled along with signing, the `%c` parameter expands to *`keyringName_Certname`*.

For more information about digitally signing access logs, refer to *Volume 9: Access Logging*.

Section E: Advanced Configuration

This section includes the following topics:

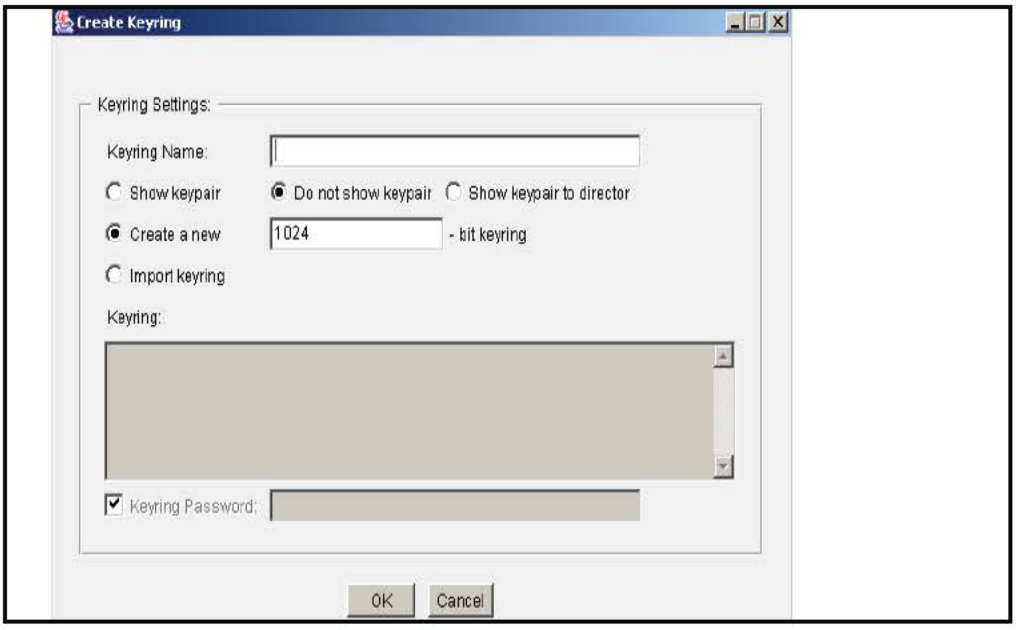
Importing an Existing Keypair and Certificate

If you have a keypair and certificate used on one system, you can import the keypair and certificate for use on a different system. You can also import a certificate chain containing multiple certificates. Use the `inline certificate` command to import multiple certificates through the CLI.

If you are importing a keyring and one or more certificates onto an SG appliance, first import the keyring, followed by the related certificates. The certificates contain the public key from the keyring, and the keyring and certificates are related.

To Import a keyring:

1. Copy the already-created keypair onto the clipboard.
2. Select **Configuration > SSL > Keyrings > SSL Keyrings**.
3. Click **Create**.



4. Fill in the dialog window as follows:
   a. **Show keypair** allows the keys to be exported.
   b. **Do not show keypair** prevents the keypair from being exported.
   c. **Show keypair to director** is a keyring viewable only if Director is issuing the command using a SSH-RSA connection.

   Note: The choice among **show, do not show and show keypair to director** has implications for whether keyrings are included in profiles and backups created by Director. For more information, refer to the *Blue Coat Director Configuration and Management Guide*.

   d. Select the **Import keyring** radio button.

      The grayed-out **Keyring** field becomes enabled, allowing you to paste in the already existing keypair. The certificate associated with this keypair must be imported separately.

      If the keypair that is being imported has been encrypted with a password, select **Keyring Password** and enter the password into the field.
5. Click **OK**.

To import a certificate and associate it with a keyring:

1. Copy the certificate onto the clipboard.
2. Select **Configuration > SSL > Keyrings** and click **Edit/View**.
3. From the drop-down list, select the keyring that you just imported.
4. Click **Import** in the **Certificate** field.
5. Paste the certificate into the Import Certificate dialog that appears. Be sure to include the `----BEGIN CERTIFICATE----` and `-----END CERTIFICATE----` statements.
6. Click **OK**.

Related CLI Syntax to Import a Keyring

```
SGOS#(config ssl) inline {keyring show | show-director | no-show}
keyring_id eof
Paste keypair here
eof
```

Related CLI Syntax to Import a Certificate and Associate it with a Keyring

```
SGOS#(config) ssl
SGOS#(config ssl) inline certificate keyring_id eof
Paste certificate here
eof
```

Section E: Advanced Configuration

About Certificate Chains

A certificate chain is one that requires that the certificates form a chain where the next certificate in the chain validates the previous certificate, going up the chain to the root, which is signed by a trusted CA. Expiration is done at the single certificate level and is checked independently of the chain verification. Each certificate in the chain must be valid for the entire chain to be valid. You can import a certificate chain containing multiple certificates.

The valid certificate chain can be presented to a browser. To get the SG appliance to present a valid certificate chain, the keyring for the HTTPS service must be updated.

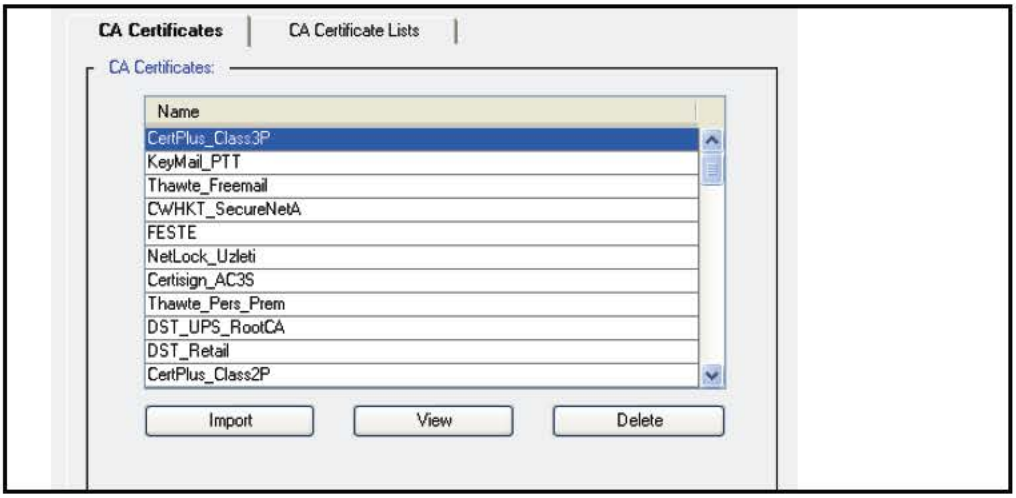
The appliance's CA-certificate list must also be updated if the SG appliance uses HTTPS to communicate with the origin server *and* if the SG appliance is configured, through the `ssl-verify-server` option, to verify the certificate (chain) presented by HTTPS server. If the SG appliance uses HTTP to communicate with the origin server, updating the CA-certificate list has no effect.

Importing a CA Certificate

A CA Certificate is a certificate that verifies the identity of a Certificate Authority. The certificate is used by the SG appliance to verify server and client certificates.

To import an approved CA certificate:

1. Copy the certificate to the clipboard.
2. Select **Configuration > SSL > CA Certificates > CA Certificates**.



3. Click **Import**.
4. Give the certificate a name.

Note: Spaces in CA Certificate names are not supported. Including a space can cause unexpected errors while using such certificates.

5. Paste the signed CA Certificate into the **Import CA Certificate** field.
6. Click **OK**.

To view a CA certificate:

1. Select **Configuration > SSL > CA Certificates > CA Certificates**.
2. Select the certificate you want to view.
3. Click **View**.Examine the contents and click **Close**.

To delete a CA certificate:

1. Select **Configuration > SSL > CA Certificates > CA Certificates**.
2. Select the certificate to delete.
3. Click **Delete**.
4. Click **OK**.

Related CLI Syntax to Import a CA Certificate

```
SGOS#(config) ssl
SGOS#(config ssl) inline ca-certificate ca_certificate_name eof
Paste certificate here
eof
```

Creating CA Certificate Lists

A CA certificate list can refer to any subset of the available CA Certificates on the SG appliance. When configuring an HTTPS service to do HTTPS Reverse Proxy, this list can be specified to restrict the set of certificate authorities that are trusted to validate client certificates presented to that service.

The default is that no list is configured; all certificates are used in authentication.

To create a CA-Certificate list:

1. Select **Configuration > SSL > CA Certificates > CA Certificate Lists**.

2. Click **New** to create a new list.
3. Enter a meaningful name for the list in the **CA-Certificate List Name** field.
4. To add CA Certificates to the list, highlight the certificate and click **Add**. You cannot add a certificate to a certificate list if it is not already present.

5. To remove CA Certificates from the list, highlight the certificate in the **Add** list and click **Remove**.
6. Click **OK**

Related CLI Syntax to Manage CA-Certificate Lists

- To enter configuration mode:

```
SGOS#(config ssl) create ccl list_name
SGOS#(config ssl) edit ccl list_name
```

- The following subcommands are available:

```
SGOS#(config ssl ccl list_name) add ca_cert_name
SGOS#(config ssl) delete ca-certificate ca_certificate_name
```

To import a CA certificate:

1. Copy the certificate to your clipboard. Be sure to include the "Begin Certificate" and "End Certificate" statements.
2. Select **Configuration > SSL > Keyrings**.
3. Highlight the keyring for which you want to import a certificate.
4. Click **Edit/View** in the **Keyrings** tab.
5. In the **Certificate** panel, click **Import**.
6. Paste the certificate you copied into the dialog box. Click **OK**.

   The certificate should display in the SSL Certificates Pane, associated with the keyring you selected earlier.
7. Select **Apply** to commit the changes to the SG appliance.

Section E: Advanced Configuration

Chapter 5: Certificate Realm Authentication

Certificate realms are useful for companies that have a Public Key Infrastructure (PKI) in place and would like to have the SG appliance authenticate their end-users using the client's X.509 certificates. If the users are members of an LDAP or Local group, the Certificate Realm can also forward the user credentials to the specified authorization realm, which determines the user's authorization (permissions).

This section discusses the following topics:

How Certificate Realm Works

Once an SSL session has been established, the user is asked to select the certificate to send to the SG appliance. If the certificate was signed by a Certificate Signing Authority that the SG appliance trusts, including itself, then the user is considered authenticated. The username for the user is the one extracted from the certificate during authentication.

At this point the user is authenticated. If an authorization realm has been specified, such as LDAP or Local, the certificate realm then passes the username to the specified authorization realm, which figures out which groups the user belongs to.

Note: If you authenticate with a certificate realm, you cannot also challenge for a password.

Certificate realms do not require an authorization realm. If no authorization realm is configured, the user cannot be a member of any group.

You do not need to specify an authorization realm if:

- The policy does not make any decisions based on groups
- The policy works as desired when all certificate realm-authenticated users are not in any group

To use a Certificate Realm, you must:

- Configure SSL between the client and SG appliance (for more information, see "Using SSL with Authentication and Authorization Services" on page 28).
- Enable **verify-client** on the HTTPS service to be used (for more information, refer to *Volume 3: Proxies and Proxy Services*).
- Verify that the certificate authority that signed the client's certificates is in the SG *trusted* list.

Creating a Certificate Realm

To create a certificate realm:

1. Select **Configuration > Authentication > Certificate > Certificate Realms**.
2. Click **New**.

3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

Defining a Certificate Realm

To define certificate authentication properties:

1. Select **Configuration > Authentication > Certificate > Certificate Main**.

2. From the **Realm Name** drop-down list, select the Certificate realm for which you want to change realm properties.
3. (Optional) From the **Authorization Realm Name** drop-down list, select the LDAP or Local realm you want to use to authorize users.
4. From the **username attribute** field, enter the attribute that specifies the common name in the subject of the certificate. **CN** is the default.
5. (Optional, if you are configuring a Certificate realm with LDAP authorization) Enter the list of attributes (the container attribute field) that should be used to construct the user's distinguished name.

   For example, **$(OU) $(O)** substitutes the OU and O fields from the certificate.

6. (Optional, if you are configuring a Certificate realm with LDAP authorization) Select or deselect **Append Base DN**.
7. (Optional, if you are configuring a Certificate realm with LDAP authorization) Enter the Base DN where the search starts. If no BASE DN is specified and Append Base DN is enabled, the first Base DN defined in the LDAP realm used for authorization is appended.
8. **Cache credentials**: Specify the length of time, in seconds, that user and administrator credentials received from the Local password file are cached. Credentials can be cached for up to 3932100 seconds. The default is **900** seconds (15 minutes).

Defining Certificate Realm General Properties

The Certificate General tab allows you to specify the display name and a virtual URL.

To configure certificate realm general settings:

1. Select **Configuration > Authentication > Certificate > Certificate General**.

2. From the **Realm name** drop-down list, select the Certificate realm for which to change properties.
3. If needed, change the Certificate realm display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
4. You can specify a virtual URL based on the individual realm. For more information on the virtual URL, see Chapter 3: "Controlling Access to the Internet and Intranet".
5. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure a Certificate Realm

- To enter configuration mode:

```
SGOS#(config) security certificate create-realm realm_name
SGOS#(config) security certificate edit-realm realm_name
```

- The following commands are available:

```
#(config certificate_realm) authorization append-base-dn {disable | dn dn_to_append | enable}
#(config certificate_realm) authorization container-attr-list list_of_attribute_names
#(config certificate_realm) authorization no {container-attr-list | realm-name}
#(config certificate_realm) authorization realm-name authorization_realm_name
#(config certificate_realm) authorization username-attribute username_attribute
```

```
#(config certificate_realm) cache-duration seconds
#(config certificate_realm) display-name display_name
#(config certificate_realm) exit
#(config certificate_realm) rename new_realm_name
#(config certificate_realm) view
#(config certificate_realm) virtual-url url
```

Revoking User Certificates

Using policy, you can revoke certain certificates by writing policy that denies access to users who have authenticated with a certificate you want to revoke. You must maintain this list on the SG appliance; it is not updated automatically.

Note: This method of revoking user certificates is meant for those with a small number of certificates to manage.

For information on using automatically updated lists, refer to *Volume 3: Proxies and Proxy Services*.

A certificate is identified by its issuer (the Certificate Signing Authority that signed it) and its serial number, which is unique to that CA.

Using that information, you can use the following strings to create a policy to revoke user certificates:

- `user.x509.serialNumber`—This is a string representation of the certificate's serial number in HEX. The string is always an even number of characters long, so if the number needs an odd number of characters to represent in hex, there is a leading zero. Comparisons are case insensitive.
- `user.x509.issuer`—This is an RFC2253 LDAP DN. Comparisons are case sensitive.
- (optional) `user.x509.subject`: This is an RFC2253 LDAP DN. Comparisons are case sensitive.

Example

If you have only one Certificate Signing Authority signing user certificates, you do not need to test the issuer. In the `<Proxy>` layer of the Local Policy file:

```
<proxy>
  deny user.x509.serialnumber=11
  deny user.x509.serialNumber=0F
```

If you have multiple Certificate Signing Authorities, test both the issuer and the serial number. In the `<Proxy>` layer of the Local Policy file:

```
<proxy>
  deny
user.x509.issuer="Email=name,CN=name,OU=name,O=company,L=city,ST=state
or province,C=country" user.x509.serialnumber=11\
deny user.x509.issuer="CN=name,OU=name,O=company, L=city,ST=state or
province,C=country" \
deny user.x509.serialnumber=2CB06E9F00000000000B
```

Creating the Certificate Authorization Policy

When you complete Certificate realm configuration, you can create CPL policies. Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file `<Proxy>` and other layers.

Be aware that the default policy condition for these examples is *allow*. On new SGOS 5.x systems, the default policy condition is *deny*.

- Every Certificate realm authenticated user is allowed access the SG appliance.

```
<Proxy>
   authenticate(CertificateRealm)
```

- A subnet definition determines the members of a group, in this case, members of the Human Resources department. (They are allowed access to the two URLs listed. Everyone else is denied permission.)

```
<Proxy>
   authenticate(CertificateRealm)
<Proxy>
   Define subnet HRSubnet
      192.168.0.0/16
      10.0.0.0/24
   End subnet HRSubnet
   [Rule] client_address=HRSubnet
     url.domain=monster.com
     url.domain=hotjobs.com
     deny
.
.
.
[Rule]
  deny
```

Tips

If you use a certificate realm and see an error message similar to the following

```
Realm configuration error for realm "cert": connection is not SSL.
```

This means that certificate authentication was requested for a transaction, but the transaction was not done on an SSL connection, so no certificate was available.

This can happen in three ways:

- The authenticate mode is either `origin-IP-redirect/origin-cookie-redirect` or `origin-IP/origin-cookie`, but the virtual URL does not have an `https:` scheme. This is likely if authentication through a certificate realm is selected with no other configuration, because the default configuration does not use SSL for the virtual URL.
- In a server accelerator deployment, the authenticate mode is origin and the transaction is on a non-SSL port.

- The authenticate mode is `origin-IP-redirect/origin-cookie-redirect`, the user has authenticated, the credential cache entry has expired, and the next operation is a POST or PUT from a browser that does not handle 307 redirects (that is, from a browser other than Internet Explorer). The workaround is to visit another URL to refresh the credential cache entry and then try the POST again.
- Forms authentication modes cannot be used with a Certificate realm. If a form mode is in use and the authentication realm is a Certificate realm, a Policy Substitution realm, or an IWA realm, you receive a configuration error.

Chapter 6: Oracle COREid Authentication

The SG appliance can be configured to consult an Oracle COREid (formerly known as Oracle NetPoint) Access Server for authentication and session management decisions. This requires that a COREid realm be configured on the SG appliance and policy written to use that realm for authentication.

The SG appliance supports authentication with Oracle COREid v6.5 and v7.0.

Access to the COREid Access System is done through the Blue Coat Authentication and Authorization Agent (BCAAA), which must be installed on a Windows 2000 system or higher with access to the COREid Access Servers.

Understanding COREid Interaction with Blue Coat

Within the COREid Access System, BCAAA acts as a custom AccessGate. It communicates with the COREid Access Servers to authenticate the user and to obtain a COREid session token, authorization actions, and group membership information.

HTTP header variables and cookies specified as authorization actions are returned to BCAAA and forwarded to the SG appliance. They can (as an option) be included in requests forwarded by the appliance.

Within the SG system, BCAAA acts as its agent to communicate with the COREid Access Servers. The SG appliance provides the user information to be validated to BCAAA, and receives the session token and other information from BCAAA.

Each SG COREid realm used causes the creation of a BCAAA process on the Windows host computer running BCAAA. When a process is created, a temporary working directory containing the Oracle COREid files needed for configuration is created for that process. A single host computer can support multiple SG realms (from the same or different SG appliances); the number depends on the capacity of the BCAAA host computer and the amount of activity in the realms.

Configuration of the SG COREid realm must be coordinated with configuration of the Access System. Each must be aware of the AccessGate. In addition, certain authorization actions must be configured in the Access System so that BCAAA gets the information the SG appliance needs.

Configuring the COREid Access System

Note: Blue Coat assumes you are familiar with the configuration of the COREid Access System and WebGates.

Since BCAAA is an AccessGate in the COREid Access System, it must be configured in the Access System just like any other AccessGate. BCAAA obtains its configuration from the SG appliance so configuration of BCAAA on the host computer is not required. If the Cert Transport Security Mode is used by the Access System, then the certificate files for the BCAAA AccessGate must reside on BCAAA's host computer.

COREid protects resources identified by URLs in policy domains. A SG COREid realm is associated with a single protected resource. This could be an already existing resource in the Access System, (typical for a reverse proxy arrangement) or it could be a resource created specifically to protect access to SG services (typical for a forward proxy).

Important: The request URL is not sent to the Access System as the requested resource; the requested resource is the entire SG realm. Access control of individual URLs is done on the SG appliance using policy.

The COREid policy domain that controls the protected resource must use one of the challenge methods supported by the SG appliance.

Supported challenge methods are Basic, X.509 Certificates and Forms. Acquiring the credentials over SSL is supported as well as challenge redirects to another server.

The SG appliance requires information about the authenticated user to be returned as COREid authorization actions for the associated protected resource. Since authentication actions are not returned when a session token is simply validated, the actions must be authorization and not authentication actions.

The following authorization actions should be set for all three authorization types (Success, Failure, and Inconclusive):

- A HeaderVar action with the name `BCSI_USERNAME` and with the value corresponding to the simple username of the authenticated user. For example, with an LDAP directory this might be the value of the cn attribute or the uid attribute.
- A HeaderVar action with the name `BCSI_GROUPS` and the value corresponding to the list of groups to which the authenticated user belongs. For example, with an LDAP directory this might be the value of the `memberOf` attribute.

Once the COREid AccessGate, authentication scheme, policy domain, rules, and actions have been defined, the SG appliance can be configured.

Additional COREid Configuration Notes

The SG appliance's credential cache only caches the user's authentication information for the lesser of the two values of the time-to-live (TTL) configured on the SG appliance and the session TTL configured in the Access System for the AccessGate.

Configuring the SG Realm

The SG realm must be configured so that it can:

- Communicate with the Blue Coat agent(s) that act on its behalf (hostname or IP address, port, SSL options, and the like).
- Provide BCAAA with the information necessary to allow it to identify itself as an AccessGate (AccessGate id, shared secret).
- Provide BCAAA with the information that allows it to contact the primary COREid Access Server (IP address, port, connection information).
- Provide BCAAA with the information that it needs to do authentication and collect authorization information (protected resource name), and general options (off-box redirection).

For more information on configuring the SG COREid realm, see "Creating a COREid Realm" on page 67.

Note: All SG appliance and agent configuration is done on the appliance. The appliance sends the necessary information to BCAAA when it establishes communication.

Participating in a Single Sign-On (SSO) Scheme

The SG appliance can participate in SSO using the encrypted `ObSSOCookie` cookie. This cookie is set in the browser by the first system in the domain that authenticates the user; other systems in the domain obtain authentication information from the cookie and so do not have to challenge the user for credentials. The SG appliance sets the `ObSSOCookie` cookie if it is the first system to authenticate a user, and authenticates the user based on the cookie if the cookie is present.

Since the SSO information is carried in a cookie, the SG appliance must be in the same cookie domain as the servers participating in SSO. This imposes restrictions on the `authenticate.mode()` used on the SG appliance.

- A reverse proxy can use any `origin` mode.
- A forward proxy must use one of the `origin-redirect` modes (such as `origin-cookie-redirect`). When using `origin-*-redirect` modes, the virtual URL's hostname must be in the same cookie domain as the other systems. It cannot be an IP address; the default `www.cfauth.com` does not work either.

When using `origin-*-redirect`, the `SSO` cookie is automatically set in an appropriate response after the SG appliance authenticates the user. When using `origin` mode (in a reverse proxy), setting this cookie must be explicitly specified by the administrator using the policy substitution variable `$(x-agent-sso-cookie)`. The variable `$(x-agent-sso-cookie)` expands to the appropriate value of the set-cookie: header.

Avoiding SG Appliance Challenges

In some COREid deployments all credential challenges are issued by a central authentication service. Protected services do not challenge and process request credentials; instead, they work entirely with the `SSO` token. If the request does not include an `SSO` token, or if the `SSO` token is not acceptable, the request is redirected to the central service, where authentication occurs. Once authentication is complete, the request is redirected to the original resource with a response that sets the `SSO` token.

If the COREid authentication scheme is configured to use a forms-based authentication, the SG appliance redirects authentication requests to the form URL automatically. If the authentication scheme is not using forms authentication but has specified a challenge redirect URL, the SG appliance only redirects the request to the central service if `always-redirect-offbox` is enabled for the realm on the SG. If the `always-redirect-offbox` option is enabled, the authentication scheme must use forms authentication or have a challenge redirect URL specified.

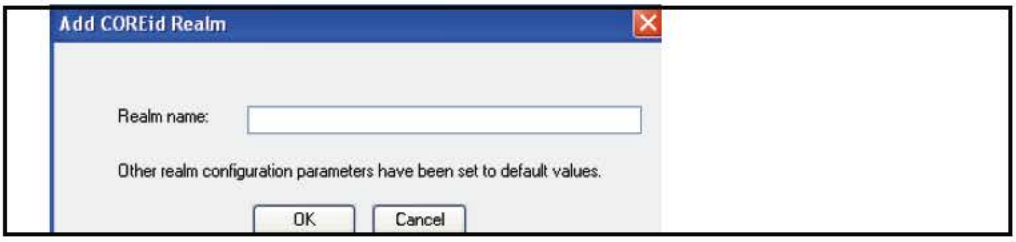
Note: The SG appliance must not attempt to authenticate a request for the off-box authentication URL. If necessary, `authenticate(no)` can be used in policy to prevent this.

Creating a COREid Realm

To create a COREid realm:

1. Select **Configuration > Authentication > Oracle COREid > COREid Realms**.

2. Click **New**.



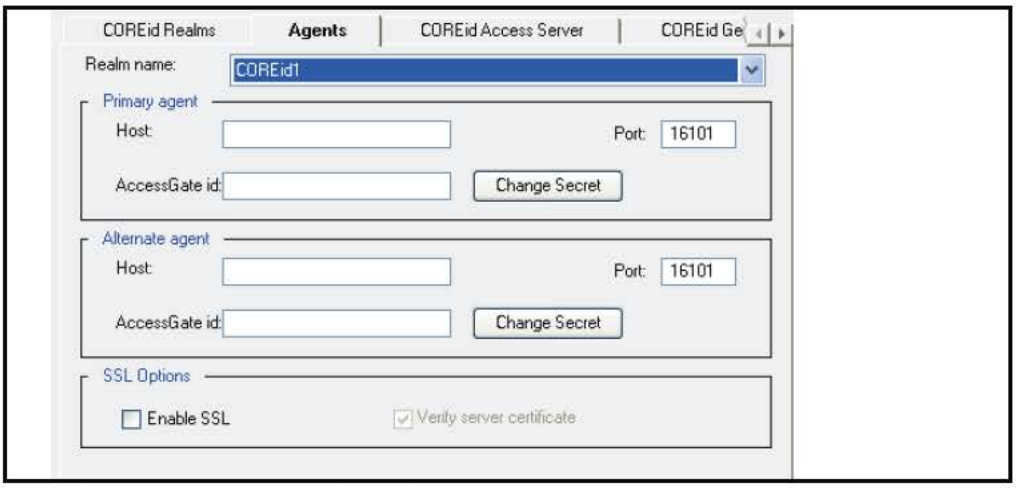
3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter. The name should be meaningful to you, but it does not have to be the name of the COREid AccessGate.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

Configuring Agents

You must configure the COREid realm so that it can find the Blue Coat Authentication and Authorization Agent (BCAAA).

To configure the BCAAA agent:

1. Select **Configuration > Authentication > Oracle COREid > Agents**.



2. Select the realm name to edit from the drop-down list.
3. In the **Primary agent** section, enter the hostname or IP address where the agent resides.
4. Change the port from the default of 16101 if necessary.
5. Enter the AccessGate ID in the **AccessGate id** field. The AccessGate ID is the ID of the AccessGate as configured in the Access System.
6. If an AccessGate password has been configured in the Access System, you must specify the password on the SG appliance. Click **Change Secret** and enter the password. The passwords can be up to 64 characters long and are always case sensitive.

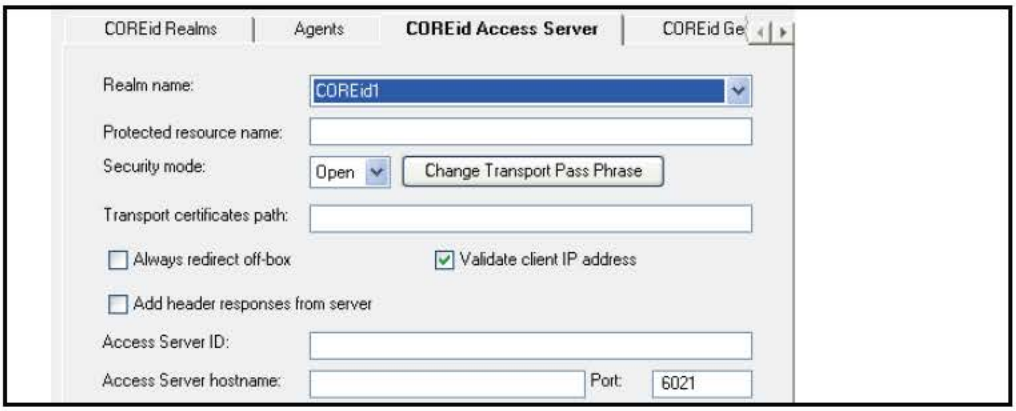
7. (Optional) Enter an alternate agent host and AccessGate ID in the **Alternate agent** section.
8. (Optional) Select **Enable SSL** to enable SSL between the SG appliance and the BCAAA agent.
9. (Optional) By default, if SSL is enabled, the COREid BCAAA certificate is verified. If you do not want to verify the agent certificate, disable this setting.

Configuring the COREid Access Server

Once you create a COREid realm, use the COREid Access Server page to specify the primary Access Server information.

To configure the COREid Access Server:

1. Select **Configuration > Authentication > Oracle COREid > COREid Access Server.**



2. Select the realm name to edit from the drop-down list.
3. Enter the protected resource name. The protected resource name is the same as the resource name defined in the Access System policy domain.
4. Select the Security Transport Mode for the AccessGate to use when communicating with the Access System.
5. If Simple or Cert mode is used, specify the Transport Pass Phrase configured in the Access System. Click **Change Transport Pass Phrase** to set the pass phrase.
6. If Cert mode is used, specify the location on the BCAAA host machine where the key, server and CA chain certificates reside. The certificate files must be named `aaa_key.pem`, `aaa_cert.pem`, and `aaa_chain.pem`, respectively.
7. To force authentication challenges to always be redirected to an off-box URL, select **Always redirect off-box.**
8. To enable validation of the client IP address in SSO cookies, select **Validate client IP address**. If the client IP address in the `SSO` cookie can be valid yet different from the current request client IP address because of downstream proxies or other devices, then deselect the **Validate client IP address** in the realm. Also modify the WebGates participating in SSO with the SG appliance. Modify the `WebGateStatic.lst` file to either set the **ipvalidation** parameter to false or to add the downstream proxy/device to the IPValidationExceptions lists.

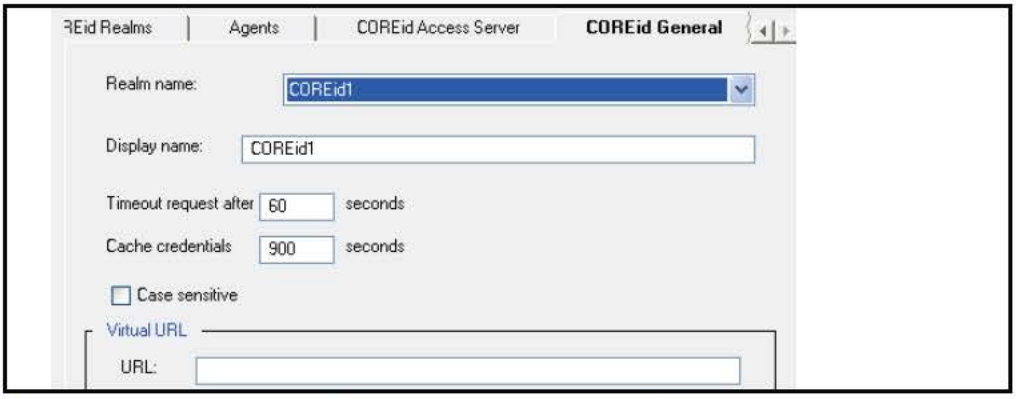
9. If your Web applications need information from the Authorization Actions, select **Add Header Responses**. Authorization actions from the policy domain obtained during authentication are added to each request forwarded by the SG appliance. Header responses replace any existing header of the same name; if no such header exists, the header is added. Cookie responses replace a cookie header with the same cookie name, if no such cookie header exists, one is added.
10. Specify the ID of the AccessGate's primary Access Server.
11. Specify the hostname of the AccessGate's primary Access Server.
12. Specify the port of the AccessGate's primary Access Server.
13. Select **Apply** to commit the changes to the SG appliance.

Configuring the General COREid Settings

The COREid General tab allows you to set a display name, cache credentials timeout, request timeout value, and case-sensitivity and create a virtual URL.

To manage general settings for the COREid realm:

1. Select **Authentication > Oracle COREid > COREid General**.



2. From the **Realm Name** drop-down list, select the COREid realm for which you want to change properties.
3. If needed, change the COREid realm display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
4. Specify the length of time, in seconds, to elapse before timeout if a response from BCAAA is not received.
5. Specify the length of time, in seconds, that user and administrator credentials are cached. Credentials can be cached for up to 3932100 seconds. The default cache-duration is **900** seconds (15 minutes).
6. If you want username and group comparisons on the SG appliance to be case sensitive, select **Case sensitive**.
7. Specify the virtual URL to redirect the user to when they need to be challenged by the SG appliance. If the appliance is participating in SSO, the virtual hostname must be in the same cookie domain as the other servers participating in the SSO. It cannot be an IP address or the default, `www.cfauth.com`.
8. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure a COREid Realm

- To enter configuration mode:

```
SGOS#(config) security coreid create-realm realm_name
SGOS#(config) security coreid edit-realm realm_name
```

- The following subcommands are available:

```
#(config) security coreid edit-realm realm_name
   #(config coreid realm_name) access-server-hostname hostname
   #(config coreid realm_name) access-server-id id
   #(config coreid realm_name) access-server-port port
   #(config coreid realm_name) add-header-responses {disable | enable}
   #(config coreid realm_name) alternate-agent accessgate-id name
   #(config coreid realm_name) alternate-agent encrypted-secret
   encrypted_shared_secret
   #(config coreid realm_name) alternate-agent host hostname
   #(config coreid realm_name) alternate-agent port port
   #(config coreid realm_name) alternate-agent secret shared_secret
   #(config coreid realm_name) always-redirect-offbox {disable |
   enable{
   #(config coreid realm_name) cache-duration seconds
   #(config coreid realm_name) case-sensitive {disable | enable}
   #(config coreid realm_name) certificate-path certificate_path
   #(config coreid realm_name) display-name display_name
   #(config coreid realm_name) encrypted-transport-pass-phrase
   encrypted_pass_phrase
   #(config coreid realm_name) exit
   #(config coreid realm_name) no alternate-agent | certificate-path
   #(config coreid realm_name) primary-agent accessgate-id name
   #(config coreid realm_name) primary-agent encrypted-secret
   encrypted_shared_secret
   #(config coreid realm_name) primary-agent host hostname
   #(config coreid realm_name) primary-agent port port
   #(config coreid realm_name) primary-agent secret shared_secret
   #(config coreid realm_name) protected-resource-name resource_name
   #(config coreid realm_name) rename new_realm_name
   #(config coreid realm_name) security-mode {cert | open | simple}
   #(config coreid realm_name) ssl {disable | enable}
   #(config coreid realm_name) ssl-verify-agent {disable | enable}
   #(config coreid realm_name) timeout seconds
   #(config coreid realm_name) transport-pass-phrase pass_phrase
   #(config coreid realm_name) validate-client-IP {disable | enable}
   #(config coreid realm_name) view
   #(config coreid realm_name) virtual-url url
```

Creating the CPL

You can create CPL policies now that you have completed COREid realm configuration. Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate for your purposes.

The examples below assume the default policy condition is *allow*. On new SGOS 5.x systems, the default policy condition is *deny*.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file `<Proxy>` and other layers.

- Every COREid-authenticated user is allowed access the SG appliance.

```
<Proxy>
  authenticate(COREidRealm)
```

- Group membership is the determining factor in granting access to the SG appliance.

```
<Proxy>
  authenticate(COREidRealm)
<Proxy>
  group=”cn=proxyusers, ou=groups, o=myco”
  deny
```

Chapter 7: Forms-Based Authentication

You can use forms-based authentication exceptions to control what your users see during authentication. You can:

- Specify the realm the user is to authenticate against.
- Specify that the credentials requested are for the SG appliance. This avoids confusion with other authentication challenges.
- Make the form comply with company standards and provide other information, such as a help link.

The authentication form (an HTML document) is served when the user makes a request and requires forms-based authentication. If the user successfully authenticates to the SG appliance, the appliance redirects the user back to the original request.

If the user does not successfully authenticate against the SG appliance and the error is user-correctable, the user is presented with the authentication form again.

Note: You can configure and install an authentication form and several properties through the Management Console and the CLI, but you must use policy to dictate the authentication form's use.

With forms-based authenticating, you can set limits on the maximum request size to store and define the request object expiry time. You can also specify whether to verify the client's IP address against the original request and whether to allow redirects to the original request.

To create and put into use forms-based authentication, you must complete the following steps:

- Create a new form or edit one of the existing authentication form exceptions
- Set storage options
- Set policies

Section A: Understanding Authentication Forms

Three authentication forms are created initially:

- authentication_form: Enter Proxy Credentials for Realm $(cs-realm). This is the standard authentication form that is used for authentication with the SG appliance.
- new_pin_form: Create New PIN for Realm $(cs-realm). This form is used if you created a RADIUS realm using RSA SecurID tokens. This form prompts the user to enter a new PIN. The user must enter the PIN twice in order to verify that it was entered correctly.
- query_form: Query for Realm $(cs-realm). This form is used if you created a RADIUS realm using RSA SecurID tokens. The form is used to display the series of yes/no questions asked by the SecurID new PIN process.

You can customize any of the three initial authentication form exceptions or you can create other authentication forms. (You can create as many authentication form exceptions as needed. The form must be a valid HTML document that contains valid form syntax.)

Each authentication form can contain the following:

- **Title** and sentence instructing the user to enter SG credentials for the appropriate realm.
- **Domain**: Text input with maximum length of 64 characters The name of the input must be `PROXY_SG_DOMAIN`, and you can specify a default value of `$(x-cs-auth-domain)` so that the user's domain is prepopulated on subsequent attempts (after a failure).

  The input field is optional, used only if the authentication realm is an IWA realm. If it is used, the value is prepended to the username value with a backslash.
- **Username**: Text input with maximum length of 64 characters. The name of the input must be `PROXY_SG_USERNAME`, and you can specify a default value of `$(cs-username)` so the username is prepopulated on subsequent attempts (after a failure).
- **Password**: The password should be of type `PASSWORD` with a maximum length of 64 characters. The name of the input must be `PROXY_SG_PASSWORD`.
- **Request ID**: If the request contains a body, then the request is stored on the SG appliance until the user is successfully authenticated.

  The request ID should be of type `HIDDEN`. The input name must be `PROXY_SG_REQUEST_ID`, and the value must be `$(x-cs-auth-request-id)`. The information to identify the stored request is saved in the request id variable.
- **Challenge State:** The challenge state should be of type `HIDDEN`. If a RADIUS realm is using a response/challenge, this field is used to cache identification information needed to correctly respond to the challenge.

  The input name must be `PROXY_SG_PRIVATE_CHALLENGE_STATE`, and the value must be `$(x-auth-private-challenge-state)`.
- **Submit button**. The submit button is required to submit the form to the SG appliance.
- **Clear form button**.The clear button is optional and resets all form values to their original values.

- **Form action URI:** The value is the authentication virtual URL plus the query string containing the base64 encoded original URL `$(x-cs-auth-form-action-url)`.
- Form METHOD of POST. The form method must be POST. The SG appliance does not process forms submitted with GET.

The SG appliance only parses the following input fields during form submission:

- `PROXY_SG_USERNAME` (required)
- `PROXY_SG_PASSWORD` (required)
- `PROXY_SG_REQUEST_ID` (required)
- `PROXY_SG_PRIVATE_CHALLENGE_STATE` (required)
- `PROXY_SG_DOMAIN` (optional) If specified, its value is prepended to the username and separated with a backslash.

Authentication_form

The initial form, authentication_form, looks similar to the following:

```
<HTML>
<HEAD>
<TITLE>Enter Proxy Credentials for Realm $(cs-realm)</TITLE>
</HEAD>
<BODY>
<H1>Enter Proxy Credentials for Realm $(cs-realm)</H1>
<P>Reason for challenge: $(exception.last_error)
<P>$(x-auth-challenge-string)
<FORM METHOD="POST" ACTION=$(x-cs-auth-form-action-url)>
$(x-cs-auth-form-domain-field)
<P>Username: <INPUT NAME="PROXY_SG_USERNAME" MAXLENGTH="64"
VALUE=$(cs-username)></P>
<P>Password: <INPUT TYPE=PASSWORD NAME="PROXY_SG_PASSWORD"
MAXLENGTH="64"></P>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_REQUEST_ID" VALUE=$(x-cs-auth-
request-id)>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_PRIVATE_CHALLENGE_STATE"
VALUE=$(x-auth-private-challenge-state)>
<P><INPUT TYPE=SUBMIT VALUE="Submit"> <INPUT TYPE=RESET></P>
</FORM>
<P>$(exception.contact)
</BODY>
</HTML>
```

If the realm is an IWA realm, the `$(x-cs-auth-form-domain-field)` substitution expands to:

```
<P>Domain: <INPUT NAME=PROXY_SG_DOMAIN MAXLENGTH=64 VALUE=$(x-cs-auth-
domain)>
```

If you specify `$(x-cs-auth-form-domain-field)`, you do not need to explicitly add the domain input field.

For comparison, the new_pin_form and query_form look similar to the following:

New_pin_form

```
<HTML>
<HEAD>
<TITLE>Create New PIN for Realm $(cs-realm)</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
function validatePin() {
var info;
var pin = document.pin_form.PROXY_SG_PASSWORD;
if (pin.value != document.pin_form.PROXY_SG_RETYPE_PIN.value) {
         info = "The PINs did not match. Please enter them again.";
} else {
                    // Edit this regular expression to match local PIN
definition
                    var re=/^[A-Za-z0-9]{4,16}$/
                    var match=re.exec(pin.value);
                 if (match == null) {
                 info = "The PIN must be 4 to 16 alphanumeric
characters";
         } else {
                return true;
         }
}
alert(info);
pin.select();
pin.focus();
return false;
}// -->
</script>
</HEAD>
<BODY>
<H1>Create New PIN for Realm $(cs-realm)</H1>
<P>$(x-auth-challenge-string)
<FORM NAME="pin_form" METHOD="POST" ACTION=$(x-cs-auth-form-action-
url)ONSUBMIT="return validatePin()">
$(x-cs-auth-form-domain-field)
<P> Enter New Pin: <INPUT TYPE=PASSWORD NAME="PROXY_SG_PASSWORD"
MAXLENGTH="64"></P>
<P>Retype New Pin: <INPUT TYPE=PASSWORD NAME="PROXY_SG_RETYPE_PIN"
MAXLENGTH="64"></P>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_USERNAME" VALUE=$(cs-username)>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_REQUEST_ID" VALUE=$(x-cs-auth-
request-id)>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_PRIVATE_CHALLENGE_STATE" VALUE=$(x-
auth-private-challenge-state)>
<P><INPUT TYPE=SUBMIT VALUE="Submit"></P>
</FORM>
<P>$(exception.contact)
</BODY>
</HTML>
```

Query_form

```
<HTML>
<HEAD>
<TITLE>Query for Realm $(cs-realm)</TITLE>
</HEAD>
<BODY>
<H1>Query for Realm $(cs-realm)</H1>
<P>$(x-auth-challenge-string)
<FORM METHOD="POST" ACTION=$(x-cs-auth-form-action-url)>
$(x-cs-auth-form-domain-field)
<INPUT TYPE=HIDDEN NAME="PROXY_SG_USERNAME" VALUE=$(cs-username)>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_REQUEST_ID" VALUE=$(x-cs-auth-
request-id)>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_PRIVATE_CHALLENGE_STATE" VALUE=$(x-
auth-private-challenge-state)>
<INPUT TYPE=HIDDEN NAME="PROXY_SG_PASSWORD"">
<P><INPUT TYPE=SUBMIT VALUE="Yes"
ONCLICK="PROXY_SG_PASSWORD.value='Y'"">
<INPUT TYPE=SUBMIT VALUE="No" ONCLICK="PROXY_SG_PASSWORD.value='N'""></
P>
</FORM>
<P>$(exception.contact)
```

</BODY>
</HTML>

User/Realm CPL Substitutions for Authentication Forms

CPL user/realm substitutions that are common in authentication form exceptions are listed below. The syntax for a CPL substitution is:

$(CPL_substitution)

group	user-name	x-cs-auth-request-id
groups	user.x509.issuer	x-cs-auth-domain
realm	user.x509.serialNumber	x-cs-auth-form-domain-field
user	user.x509.subject	x-cs-auth-form-action-url
cs-realm	x-cs-auth-request-id	x-auth-challenge-string
	x-auth-private-challenge-state	

Note: Any substitutions that are valid in CPL and in other exceptions are valid in authentication form exceptions.

For a discussion of CPL and a complete list of CPL substitutions, as well as a description of each substitution, refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide*.

Tip

There is no realm restriction on the number of authentication form exceptions you can create. You can have an unlimited number of forms, although you might want to make them as generic as possible to cut down on maintenance.

Section B: Creating and Editing a Form

You can create a new form or you can edit one of the existing ones. If you create a new form, you need to define its type (authentication_form, new_pin_form, or query_form). The form is created from the default definition for that type. Editing the initial forms does not affect how future forms are created.

To create or edit an authentication form:

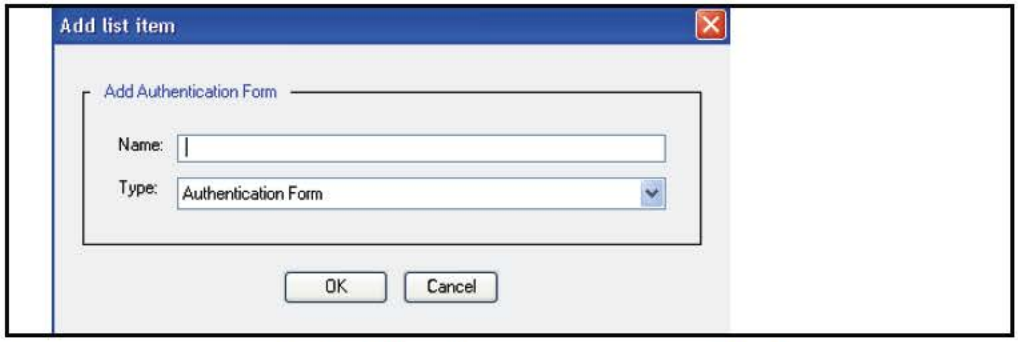
1. Select **Configuration > Authentication > Forms**.
2. Select one of the buttons below the authentication forms:
   - Highlight the form you want to edit, delete, or view.

     Note: **View** in the Authentication Forms panel and **View** in the Default Definitions panel have different functions. **View** in the Authentication Forms panel allows you to view the form you highlighted; **View** in the Default Definitions panel allows you view the original, default settings for each form. This is important in an upgrade scenario; any forms already installed will not be changed. You can compare existing forms to the default version and decide if your forms need to be modified.

   - Click **New** to create a new form.

To create a new form:

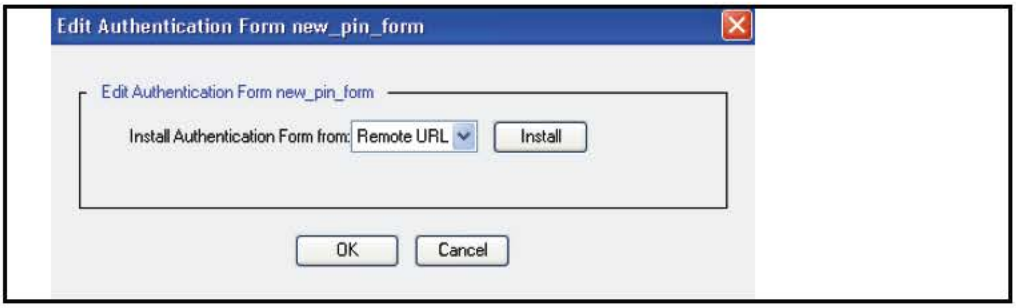
The **New** button works independently of the highlighted form. The template used for the new form is chosen from the **Add Authentication Form** dialog.



- Enter the form name and select the authentication type from the dropdown menu.
- Click **OK**.

To edit a form:

If you highlight the form you want to edit and click **Edit.**



- From the drop-down list, select the method to use to install the authentication form; click **Install**.

  - Remote URL:

    Enter the fully-qualified URL, including the filename, where the authentication form is located. To view the file before installing it, click **View**. Click **Install**. To view the results, click **Results**; to close the dialog when through, click **OK**.

  - Local File:

    Click **Browse** to bring up the Local File Browse window. Browse for the file on the local system. Open it and click **Install**. When the installation is complete, a results window opens. View the results; to close the window, click **Close**.

  - **Text Editor:**

    The current authentication form is displayed in the text editor. You can edit the form in place. Click **Install** to install the form. When the installation is complete, a results window opens. View the results; to close the window, click **Close**.

Related CLI Syntax to Create a Form

```
#(config) security authentication-forms copy [source_form_name
target_form_name
#(config) security authentication-forms create {authentication-form |
new-pin-form | query-form} form_name
#(config) security authentication-forms delete form_name
#(config) security authentication-forms inline form_name eof_marker
#(config) security authentication-forms load form_name
#(config) security authentication-forms no path [form_name]
#(config) security authentication-forms path [form_name] path
#(config) security authentication-forms view
```

Section C: Setting Storage Options

When a request requiring the user to be challenged with a form contains a body, the request is stored on the SG appliance while the user is being authenticated. Storage options include:

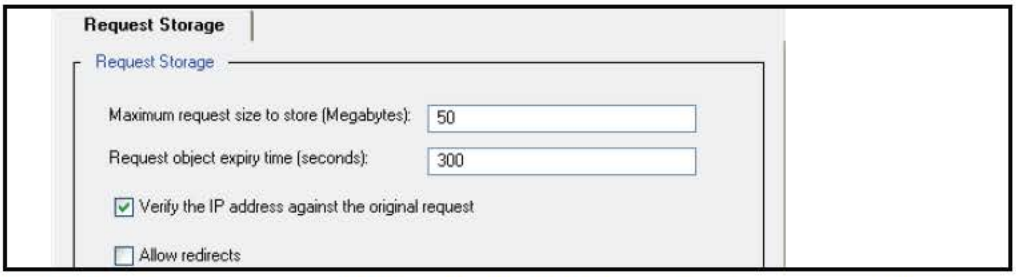
- the maximum request size.
- the expiration of the request.
- whether to verify the IP address of the client requesting against the original request.
- whether to allow redirects from the origin server

The storage options are global, applying to all form exceptions you use.

The global allow redirects configuration option can be overridden on a finer granularity in policy using the `authenticate.redirect_stored_requests(yes|no)` action.

To set storage options:

1. Select **Configuration > Authentication > Request Storage**.



2. In the **Maximum request size to store (Megabytes)** field, enter the maximum POST request size allowed during authentication. The default is 50 megabytes.
3. In the **Request object expiry time (seconds)** field, enter the amount of time before the stored request expires. The default is 300 seconds (five minutes). The expiry time should be long enough for the user to fill out and submit the authentication form.
4. If you do not want the SG appliance to **Verify the IP address against the original request**, deselect that option. The default is to verify the IP address.
5. To **Allow redirects** from the origin servers, select the checkbox. The default is to not allow redirects from origin servers.

Note: During authentication, the user's POST is redirected to a GET request. The client therefore automatically follows redirects from the origin server. Because the SG appliance is converting the GET to a POST and adding the post data to the request before contacting the origin server, the administrator must explicitly specify that redirects to these POSTs requests can be automatically followed.

---

6. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Set Storage Options

```
SGOS#(config) security request-storage max-size megabytes
SGOS#(config) security request-storage expiry-time seconds
SGOS#(config) security request-storage verify-ip enable | disable
```

Section C: Setting Storage Options

```
SGOS#(config) security request-storage allow-redirects enable |
disable
```

Section D: Using CPL with Forms-Based Authentication

To use forms-based authentication, you must create policies that enable it and also control which form is used in which situations. A form must exist before it can be referenced in policy.

- Which form to use during authentication is specified in policy using one of the CPL conditions `authenticate.form(form_name)`, `authenticate.new_pin_form(form_name)`, or `authenticate.query_form (form_name)`.

  These conditions override the use of the initial forms for the cases where a new pin form needs to be displayed or a query form needs to be displayed. All three of the conditions verify that the form name has the correct type.

  Note: Each of these conditions can be used with the form authentication modes only. If no form is specified, the form defaults to the CPL condition for that form. That is, if no name is specified for `authenticate.form(form_name)`, the default is `authentication_form`; if no name is specified for `authenticate.new_pin_form(form_name)`, the default is `authenticate.new_pin_form`, and if no name is specified for `authenticate.query_form(form_name)`, the default is `authenticate.query_form`.

- Using the `authentication.mode( )` property selects a combination of challenge type and surrogate credentials. The `authentication.mode( )` property offers several options specifically for forms-based authentication:

  - **Form-IP**—The user's IP address is used as a surrogate credential. The form is presented whenever the user's credential cache entry expires.
  - **Form-Cookie**—Cookies are used as surrogate credentials. The cookies are set on the OCS domain only, and the user is presented with the form for each new domain. This mode is most useful in reverse proxy scenarios where there are a limited number of domains.
  - **Form-Cookie-Redirect**—The user is redirected to the authentication virtual URL before the form is presented. The authentication cookie is set on both the virtual URL and the OCS domain. The user is only challenged when the credential cache entry expires.
  - **Form-IP-redirect** —This is similar to **Form-IP** except that the user is redirected to the authentication virtual URL before the form is presented.

- If you authenticate users who have third-party cookies explicitly disabled, you can use the `authenticate.use_url_cookie( )` property.
- Since the `authentication.mode( )` property is defined as a form mode (above) in policy, you do not need to adjust the default authenticate mode through the CLI.
- Using the `authenticate.redirect_stored_requests(yes|no)` action allows granularity in policy over the global allow redirect config option.

For information on using these CPL conditions and properties, refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide*.

Tips

- If the user is supposed to be challenged with a form on a request for an image or video, the SG appliance returns a 403 error page instead of the form. If the reason for the challenge is that the user's credentials have expired and the object is from the same domain as the container page, then reloading the container page results in the user receiving the authentication form and being able to authenticate. However, if the client browser loads the container page using an existing authenticated connection, the user might still not receive the authentication form.

  Closing and reopening the browser should fix the issue. Requesting a different site might also cause the browser to open a new connection and the user is returned the authentication form.

  If the container page and embedded objects have a different domain though and the authentication mode is **form-cookie**, reloading or closing and reopening the browser might not fix the issue, as the user is never returned a cookie for the domain the object belongs to. In these scenarios, Blue Coat recommends that policy be written to either bypass authentication for that domain or to use a different authentication mode such as **form-cookie-redirect** for that domain.

- Forms-based authentication works with HTTP browsers only.
- Because forms only support Basic authentication, authentication-form exceptions cannot be used with a Certificate realm. If a form is in use and the authentication realm is or a Certificate realm, you receive a configuration error.
- User credentials are sent in plain text. However, they can be sent securely using SSL if the virtual URL is HTTPS.
- Because not all user requests support forms (such as WebDAV and streaming), create policy to bypass authentication or use a different authentication mode with the same realm for those requests.

Chapter 8: IWA Realm Authentication and Authorization

Integrated Windows Authentication (IWA) is an authentication mechanism available on Windows networks. (The name of the realm has been changed from NTLM to IWA.)

IWA is a Microsoft-proprietary authentication suite that allows Windows clients (running on Windows 2000 and higher) to automatically choose between using Kerberos and NTLM authentication challenge/response, as appropriate. When an IWA realm is used and a resource is requested by the client from the SG appliance, the appliance contacts the client's domain account to verify the client's identity and request an access token. The access token is generated by the domain controller (in case of NTLM authentication) or a Kerberos server (in the case of Kerberos authentication) and passed to (and if valid, accepted by) the SG appliance.

Refer to the Microsoft Web site for detailed information about the IWA protocol.

This section discusses the following topics:

How Blue Coat Works with IWA

The server side of the Kerberos or NTLM authentication exchange is handled by the Blue Coat Authentication and Authorization Agent (BCAAA).

A single BCAAA service can support multiple SG appliances; however, the service starts a processor agent for each realm that only handles authentication requests coming from that particular realm.

BCAAA must be installed on a domain controller or member server. If the server where the BCAAA service is installed and its domain have a trust relationship with other domains, the user is authenticated automatically by the other domains.

For a server to participate in an IWA Kerberos authentication exchange, it must share a secret with the Kerberos server (called a KDC) and have registered an appropriate Service Principal Name.

For instructions on installing the BCAAA service and configuring a Service Principal Name, see Appendix B: "Using the Authentication/Authorization Agent" on page 157.

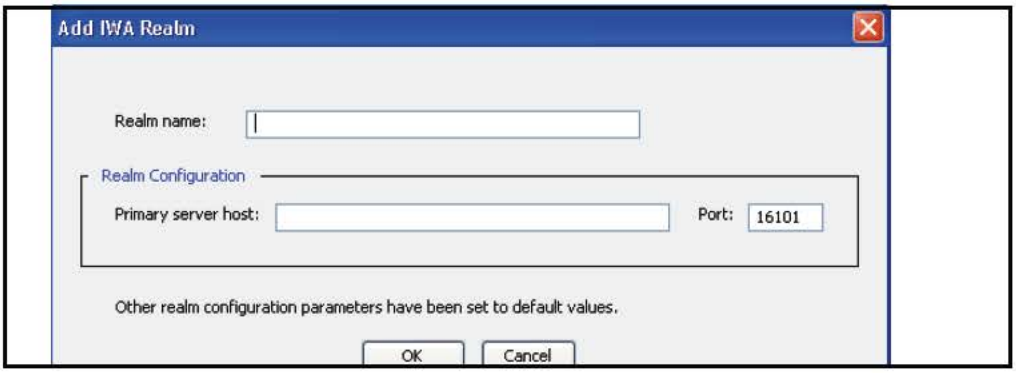
Creating an IWA Realm

To create an IWA realm, you must provide at least the primary host of the IWA server for that realm.

To create an IWA realm:

1. Select **Configuration > Authentication > IWA > IWA Realms**.

2. Click **New**.

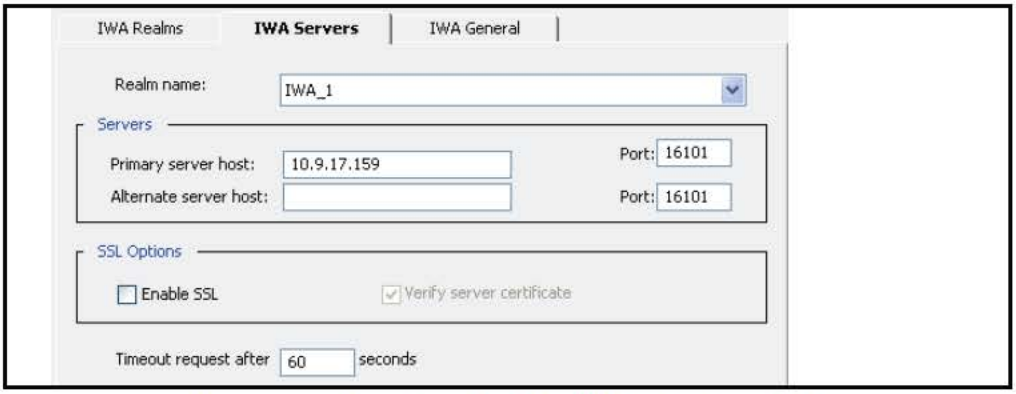


3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter.
4. Identify the primary server host for the machine running BCAAA. You must enter a valid host or an error message is generated.
5. (Optional) The default port is 16101. You can change the port number if the primary server is listening on a different port.
6. Click **OK**.
7. Select **Apply** to commit the changes to the SG appliance.

IWA Servers

Once you create an IWA realm, you can use the IWA Servers page to change the current default settings.

1. Select **Configuration > Authentication > IWA > IWA Servers**.



2. From the **Realm Name** drop-down list, select the IWA realm for which you want to change server properties.

   You must define at least one IWA realm (using the **IWA Realms** page) before attempting to set IWA server properties. If the message **Realms must be added in the IWA Realms tab before editing this tab** is displayed in red at the bottom of this page, you do not currently have any IWA realms defined
3. Specify the host and port for the primary IWA server. The default port is **16101**.

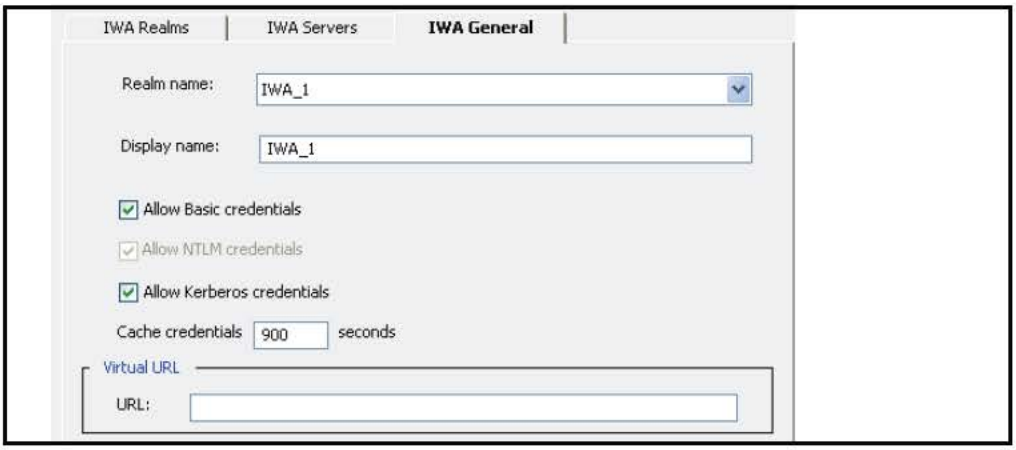
4. (Optional) Specify the host and port for the alternate IWA server. The default port is **16101**.
5. (Optional) Under **SSL Options**, click the **SSL enable** checkbox to enable SSL.
6. (Optional) By default, if SSL is enabled, the BCAAA certificate is verified. If you do not want to verify the BCAAA certificate, deselect this checkbox.
7. In the **Timeout Request** field, type the number of seconds the SG appliance allows for each request attempt before timing out. (The default request timeout is **60** seconds.)
8. Select **Apply** to commit the changes to the SG appliance.
9. Repeat the above steps for additional IWA realms, up to a total of 40.

Defining IWA Realm General Properties

The IWA General tab allows you to specify the display name, whether to support Basic and IWA credentials, the credential cache duration and a virtual URL.

To configure general settings:

1. Select **Configuration > Authentication > IWA > IWA General**.



2. From the **Realm Name** drop-down list, select the IWA realm for which you want to change properties.
3. If needed, change the IWA realm display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
4. You can enable or disable support for Basic credentials in the realm by selecting or deselecting the **Allow Basic credentials** checkbox.

   At least one Basic or NTLM/Kerberos credential must be enabled. Note that Basic credentials cannot be disabled in the IWA realm if the IWA realm is part of a sequence realm but is not the first realm in the sequence with **try IWA authentication only once** enabled.

   You can disable both NTLM and Kerberos credentials, leaving a realm that collects plaintext credentials but validates them against a Windows domain.

Important: The configuration of the realm can have significant security implications. If an IWA realm accepts Basic credentials, the client can automatically downgrade to sending the password in plaintext. Similarly, the client can use NTLM instead of Kerberos.

5. (Optional) You can enable or disable support for NTLM credentials in the realm by selecting or deselecting the **Allow NTLM credentials** checkbox. You can only enable support for Kerberos credentials in the realm if support for NTLM credentials has been enabled.
6. (Optional) You can enable or disable support for Kerberos credentials in the realm by selecting or deselecting the Allow Kerberos credentials.You can only enable support for Kerberos credentials in the realm if support for NTLM credentials has been enabled.
7. Specify the length of time, in seconds, that user and administrator credentials received from the IWA server are cached. Credentials can be cached for up to 3932100 seconds. The default cache duration is **900** seconds (15 minutes).

Note: If you specify **0**, traffic is increased to the IWA server because each authentication request generates an authentication and authorization request to the server.

8. In the Virtual URL field, enter the URL to redirect to when the user needs to be challenged for credentials if using a redirecting authenticate.mode.

Note: The virtual URL is not involved if the challenge does not redirect.

You can specify a virtual URL based on the individual realm. For more information on the virtual URL, see Chapter 3: "Controlling Access to the Internet and Intranet" on page 23.

When NTLM is in use, requests to the virtual URL must be sent to the proxy. This can be done either by transparent redirection or by making the virtual URL hostname resolve to an IP address of the proxy.

When Kerberos is in use:

- The virtual URL hostname must be part of the Kerberos realm (this is using the term *realm* in the Kerberos sense, not the SG appliance sense).
- For a forward proxy, this hostname should be added to the DNS server for the same domain as the Kerberos protected resources so that requests for this address go directly to the SG appliance.

In both NTLM and Kerberos, if single-sign on is desired, then the virtual URL hostname must have no dots and must not be proxied by the browser. The client must be able to resolve this hostname to an IP address of the proxy.

9. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure an IWA Realm

- To enter configuration mode:

```
SGOS#(config) security iwa create-realm realm_name
```

```
SGOS#(config) security iwa edit-realm realm_name
```

- The following subcommands are available:

```
#(config iwa realm_name) alternate-server host [port]
#(config iwa realm_name) cache-duration seconds
#(config iwa realm_name) credentials-basic {disable | enable}
#(config iwa realm_name) credentials-kerberos {disable | enable}
#(config iwa realm_name) credentials-ntlm {disable | enable}
#(config iwa realm_name) display-name display_name
#(config iwa realm_name) exit
#(config iwa realm_name) no alternate-server
#(config iwa realm_name) primary-server host [port]
#(config iwa realm_name) rename new_realm_name
#(config iwa realm_name) timeout seconds
#(config iwa realm_name) ssl {disable | enable}
#(config iwa realm_name) ssl-verify-server {disable | enable}
#(config iwa realm_name) view
#(config iwa realm_name) virtual-url url
```

Creating the CPL

You can create CPL policies now that you have completed IWA realm configuration. Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate for your purposes.

The examples below assume the default policy condition is *allow*. On new systems, the default policy condition is *deny*.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file layers.

- Every IWA-authenticated user is allowed access the SG appliance.

```
<Proxy>
  authenticate(IWARealm)
```

- Group membership is the determining factor in granting access to the SG appliance.

```
<Proxy>
  authenticate(IWARealm)
<Proxy>
  deny
```

Notes

- Forms authentication modes cannot be used with an IWA realm that allows only NTLM/Kerberos credentials. If a form mode is in use and the authentication realm is an IWA realm, you receive a configuration error.
- For Windows Internet Explorer IWA users who want true single-sign-on (allowing Internet Explorer to provide your credentials automatically when challenged), you must set the virtual URL to a hostname that is resolvable to the IP address of the SG appliance by the client machines. Dots (for example, `10.1.1.1`) are not allowed.

Note: Firefox (1.02 and higher) allows NTLM credentials for single sign-on but not Kerberos.

To define the information in Internet Explorer, navigate to **Internet Options > Security > Local intranet > Sites > Advanced... > Web sites**. (For XP, navigate to **Internet Options > Security > Internet > Custom Level**, then select **Automatic logon with current username and password**.)

For Windows Internet Explorer 6.x, add the virtual host address.

Chapter 9: LDAP Realm Authentication and Authorization

Many companies and organizations use the Lightweight Directory Access Protocol (LDAP) as the directory protocol of choice, enabling software to find an individual user without knowing where that user is located in the network topography.

This section discusses the following topics:

Overview

Blue Coat supports both LDAP v2 and LDAP v3, but recommends LDAP v3 because it uses Transport Layer Security (TLS) and SSL to provide a secure connection between the SG appliance and the LDAP server.

An LDAP directory, either version 2 or version 3, consists of a simple tree hierarchy. An LDAP directory might span multiple LDAP servers. In LDAP v3, servers can return referrals to others servers back to the client, allowing the client to follow those referrals if desired.

Directory services simplify administration; any additions or changes made once to the information in the directory are immediately available to all users and directory-enabled applications, devices, and SG appliances.

The SG appliance supports the use of external LDAP database servers to authenticate and authorize users on a per-group or per-attribute basis.

LDAP group-based authentication for the SG appliance can be configured to support any LDAP-compliant directory including:

- Microsoft Active Directory Server
- Novell NDS/eDirectory Server
- Netscape/Sun iPlanet Directory Server
- Other

The SG appliance also provides the ability to search for a single user in a single root of an LDAP directory information tree (DIT), and to search in multiple Base Distinguished Names (DNs).

You can configure a LDAP realm to use SSL when communicating to the LDAP server.

Configuring LDAP involves the following steps:

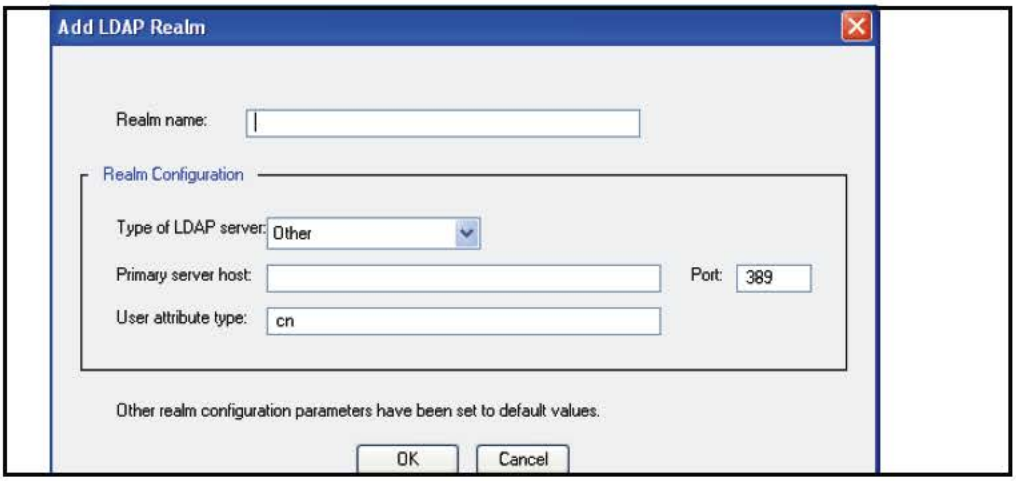
- Creating a realm (up to 40) and configuring basic settings.

- Configuring an LDAP server
- Defining LDAP Base Distinguished Names
- Defining Authorization and Group information
- Configuring general LDAP realm settings
- Creating policy

Creating an LDAP Realm

To create an LDAP realm:

1. Select **Configuration > Authentication > LDAP > LDAP Realms**.
2. Click **New**.



3. In the Real name field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter.
4. From the **Type of LDAP server** drop-down list, select the specific LDAP server.
5. Specify the host and port for the primary LDAP server. The host must be entered. The default port number is **389**.
6. In the **User attribute type** field, specify the default user attribute type for the type of LDAP server.

Microsoft Active Directory Server	`sAMAccountName=`
Novell NDS/eDirectory Server/Other	`cn=`
Netscape/iPlanet Directory Server	`uid=`

7. Click **OK**.
8. Select **Apply** to commit the changes to the SG appliance.

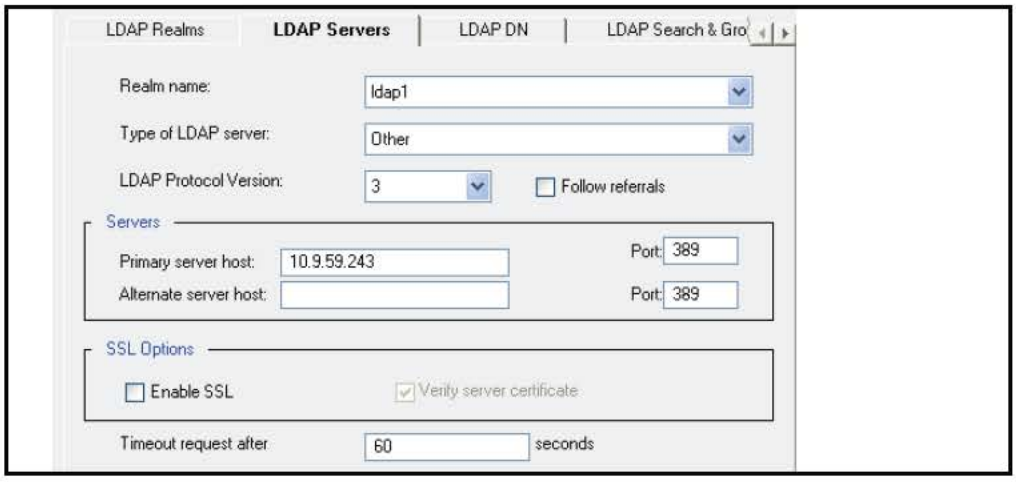
LDAP Servers

Once you have created an LDAP realm, you can use the LDAP Servers page to change the current default settings.

To edit LDAP server properties:

Note that the default values exist. You do not need to change these values if the default settings are acceptable.

1. Select **Configuration > Authentication > LDAP > LDAP Servers**.



2. From the **Realm Name** drop-down list, select the LDAP realm for which you want to change server properties.
3. From the **Type of LDAP server** drop-down list, select the specific LDAP server.
4. From the **LDAP Protocol Version** drop-down list, select **v2** for LDAP v2 support. LDAP v3 is the default.

   If you use LDAP v3, you can select **Follow referrals** to allow the client to follow referrals to other servers. (This feature is not available with LDAP v2.) The default is **Disabled**.
5. Specify the host and port for the primary LDAP server. The host must be entered. The default port number is **389**.
6. (Optional) Specify the host and port for the alternate LDAP server. The default port is **389**.
7. (Optional) Under **SSL Options**, select **Enable SSL** to enable SSL. You can only select this option if you are using LDAP v3.
8. (Optional) By default, if SSL is enabled, the LDAP server certificate is verified. If you do not want to verify the server certificate, disable this setting.
9. (Optional) Change the timeout request for the server from its default of **60** seconds.
10. Select **Apply** to commit the changes to the SG appliance.
11. Repeat the above steps for additional LDAP realms, up to a total of 40.

Defining LDAP Base Distinguished Names

The SG appliance allows you to specify multiple Base Distinguished Names (DNs) to search per realm, along with the ability to specify a specific branch of a Base DN.

A *Base DN* identifies the entry that is starting point of the search. You must specify at least one non-null base-DN for LDAP authentication to succeed.

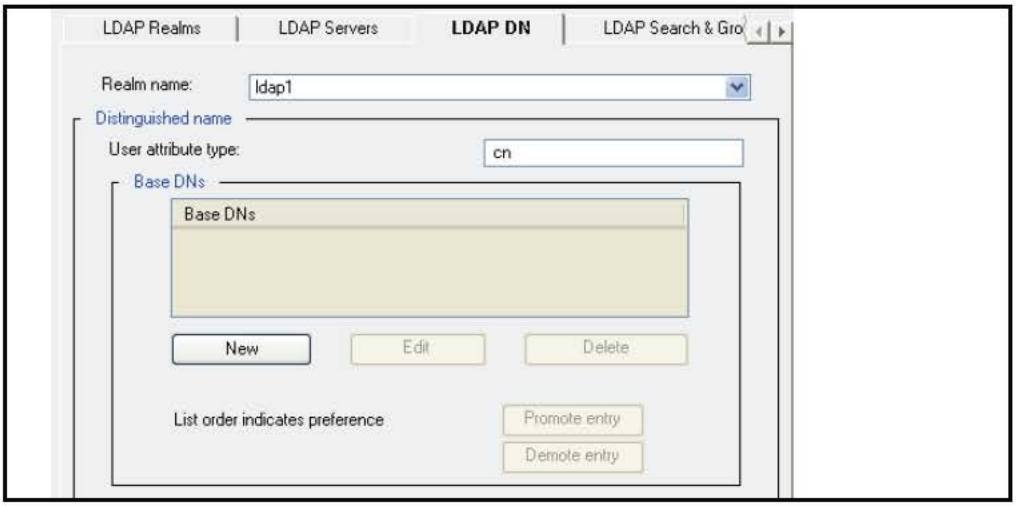
You must enter complete DNs. See the table below for some examples of distinguished name attributes.

Table 9-1. Distinguished Name Attributes

DN Attribute Syntax	Parameter Description
`c=`*`country`*	Country in which the user or group resides. Examples: `c=US`, `c=GB`.
`cn=`*`common name`*	Full name of person or object defined by the entry. Examples: `cn=David Smith`, `cn=Administrators`, `cn=4th floor printer`
`mail=`*`e-mail address`*	User or group e-mail address.
`givenName=`*`given name`*	User's first name.
`l=`*`locality`*	Locality in which the user or group resides. This can be the name of a city, country, township, or other geographic regions. Examples: `l=Seattle`, `l=Pacific Northwest`, `l=King County`.
`o=`*`organization`*	Organization to which the user or group is a member. Examples: `o=`Blue Coat `Inc`, `o=UW`.
`ou=`*`organizational unit`*	Unit within an organization. Examples: `ou=Sales`, `ou=IT`, `ou=Compliance`.
`st=`*`state or province`*	State or province in which the user or group resides. Examples: `st=Washington`, `st=Florida`.
`userPassword=`*`password`*	Password created by a user.
`streetAddress=`*`street address`*	Street number and address of user or group defined by the entry. Example: `streetAddress= 650 Almanor Avenue Sunnyvale, California 94085-3515`.
`sn=`*`surname`*	User's last name.
`telephoneNumber=`*`telephone`*	User or group telephone number.
`title=`*`title`*	User's job title.
`uid=`*`user ID`*	Name that uniquely identifies the person or object defined by the entry. Examples: `uid=ssmith`, `uid=kjones`.

To define searchable LDAP base DNs:

1. Select **Configuration > Authentication > LDAP > LDAP DN**.

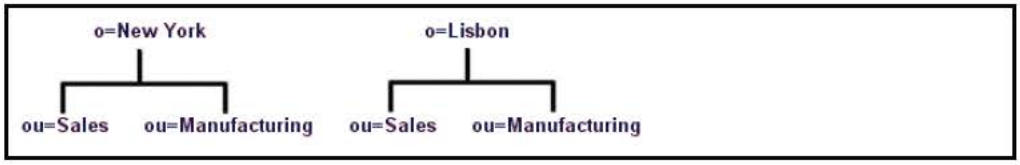


2. From the **Realm Name** drop-down list, select the LDAP realm for which you want to change DN properties.
3. In the **User attribute type** field, the SG appliance has entered the default user attribute type for the type of LDAP server you specified when creating the realm.

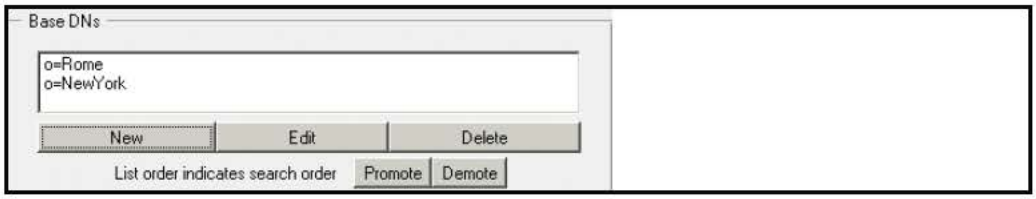
Microsoft Active Directory Server	`sAMAccountName=`
Novell NDS/eDirectory Server/Other	`cn=`
Netscape/iPlanet Directory Server	`uid=`

If you entered information correctly when creating the realm, you do not need to change the User attribute type in this step. If you do need to change or edit the entry, do so directly in the field.

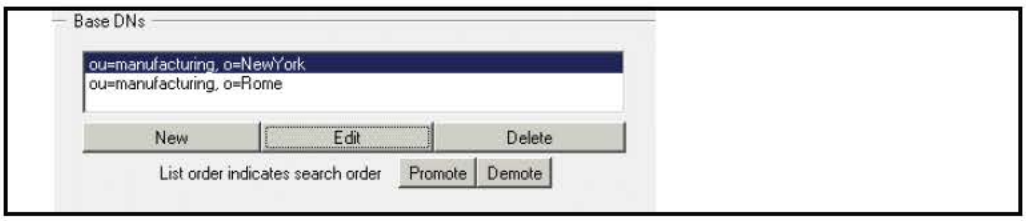
4. Enter as many Base DNs as you need for the realm. Assume, for example, that Sample_Company has offices in New York and Lisbon, each with its own Base DN. A simplified directory information tree is illustrated below.



To specify entries for the **Base DN**s field, click **New**, enter the Base DN, and click **OK**. Repeat for multiple Base DNs. To search all of Sample_Company, enter `o` values:



To search the manufacturing organizations, rather than starting at the top, enter *ou* and *o* values:



You can add, edit, and delete Base DNs for an SG appliance to search. You can also select an individual DN and move it up or down in the list with the **Promote** and **Demote** buttons. The appliance searches multiple DNs in the order listed, starting at the top and working down.

1. Select **Apply** to commit the changes to the SG appliance.

LDAP Search & Groups Tab (Authorization and Group Information)

After creating an LDAP realm, providing at least the required fields of the LDAP server for that realm, and defining base DNs for the realm, you must define authorization properties for each LDAP realm you created.

Note: Authorization decisions are completely handled by policy. The groups that the appliance looks up and queries are derived from the groups specified in policy in group= conditions, attribute= conditions, and has Attribute conditions. If you do not have any of those conditions, then Blue Coat does not look up any groups or attributes to make policy decisions based on authorization.

To define LDAP realm authorization properties:

1. Select **Configuration > Authentication > LDAP > LDAP Search & Groups**.

2. From the **Realm Name** drop-down list, select the LDAP realm for which you want to specify authorization information.
3. Specify whether to allow anonymous search or to enforce user authentication before allowing a search.

Some directories require a valid user to be able to perform an LDAP search; they do not allow *anonymous bind*. (Active Directory is one such example.) For these directories, you must specify a valid fully-qualified distinguished username and the password that permits directory access privileges. (For example, **cn=user1**,**cn=users**,**dc=bluecoat**,**dc=com** is a possible fully-qualified distinguished name.)

To permit users to anonymously bind to the LDAP service, select **Anonymous Search Allowed**. For example, with Netscape/iPlanet Directory Server, when anonymous access is allowed, no username or password is required by the LDAP client to retrieve information.

The LDAP directory attributes available for an anonymous client are typically a subset of those available when a valid user distinguished name and password have been used as search credentials.

To enforce user authentication before binding to the LDAP service, deselect **Anonymous Search Allowed**, and set the **Search User DN** and **Search User Password**. Enter a user distinguished name in the **Search User DN** field. This username can identify a single user or a user object that acts as a proxy for multiple users (a pool of administrators, for example). A search user distinguished name can be up to 512 characters long.

You can set or change the user password by clicking **Change Password**. This password can be up to 64 alphanumeric characters long.

You might want to create a separate user (such as Blue Coat, for example) instead of using an Administrator distinguished name and password.

The **Dereference level** field has four values—**always, finding, never, searching**—that allow you to specify when to search for a specific object rather than search for the object's alias. The default is **Always**.

4. Group Information

   Membership type and Membership attribute: The SG appliance enters the appropriate default:

   - Microsoft Active Directory:
     Membership type: `user`
     Membership attribute type: `memberOf`
   - Netscape/Sun iPlanet:
     Membership type:`group`
     Membership attribute type:`uniqueMember`
   - Novell NDS eDirectory
     Membership type:`group`
     Membership attribute type:`member`
   - Other
     Membership type:`user`
     Membership attribute type:`member`

   Username type to lookup: Select either **FQDN** or **Relative**. Only one can be selected at a time.

   - **Relative** can only be selected in the membership type is **Group**.
   - **FQDN** indicates that the lookup is done only on the user object. **FQDN** can be selected when the membership type is either **Group** or **User**.

5. Select **Apply** to commit the changes to the SG appliance.

Customizing LDAP Objectclass Attribute Values

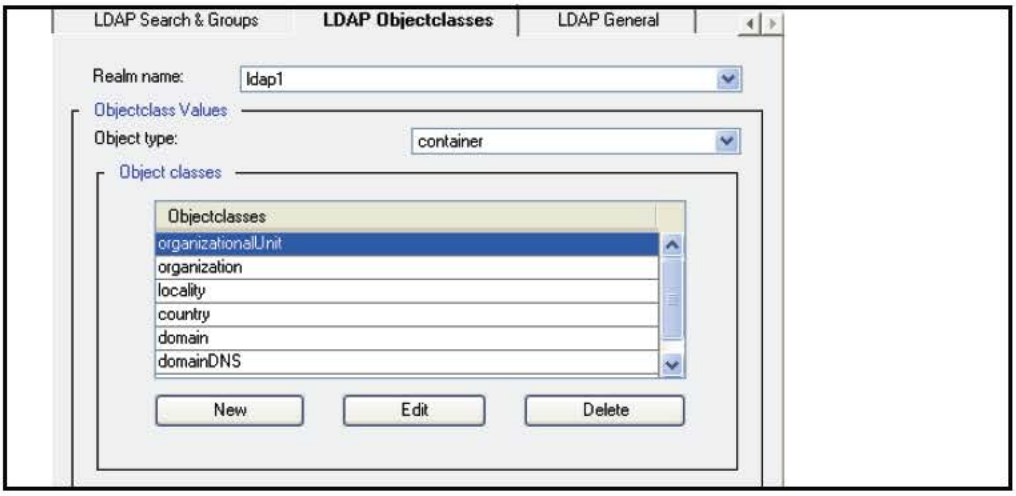
The *objectclass* attributes on an LDAP object define the type of object an entry is. For example, a user entry might have an `objectclass` attribute value of *person* while a group entry might have an `objectclass` attribute value of *group*.

The `objectclass` attribute values defined on a particular entry can differ among LDAP servers. The `objectclass` attribute values are attribute values only, they are not DNs of any kind.

Currently, the `objectclass` attribute values are used by Blue Coat during a VPM browse of an LDAP server. If an administrator wants to browse the groups in a particular realm, the SG appliance searches the LDAP server for objects that have `objectclass` attribute values matching those in the group list and in the container list. The list of `objectclass` attribute values in the container list is needed so that containers that contain groups can be fetched and expanded correctly.

To customize LDAP objectclass attribute values:

1. **Select Configuration > Authentication > LDAP > LDAP Objectclasses.**



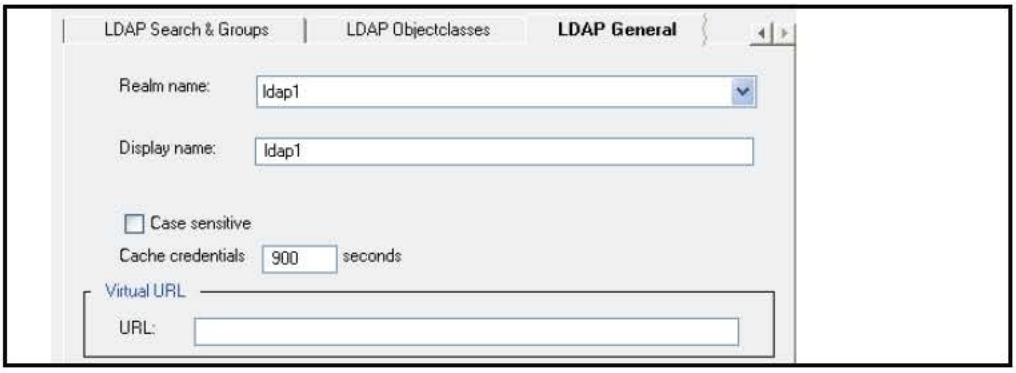
2. From the **Realm name** drop-down list, select the LDAP realm whose objectclasses you want to modify.
3. From the **Object type** drop-down list, select the type of object: **container**, **group**, or **user**.
4. To create or edit an object for the specified objectclass, click **New** or **Edit**. (The only difference is whether you are adding or editing an objectclass value.)
5. Enter or edit the objectclass, and click **OK.**
6. Select **Apply** to commit the changes to the SG appliance.

Defining LDAP General Realm Properties

The LDAP General page allows you to indicate whether an LDAP server is configured to expect case-sensitive usernames and passwords, the length of time that credentials are cached, the display name, and if you want to use a special virtual host for this realm.

To configure general LDAP settings:

1. Select **Configuration > Authentication > LDAP > LDAP General.**



2. From the **Realm Name** drop-down list, select the LDAP realm for which you want to change properties.
3. If needed, give the LDAP realm a display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
4. If the LDAP server is configured to expect case-sensitive usernames and passwords, select **Case sensitive**.
5. Specify the length of time in seconds that user and administrator credentials received from the LDAP server are cached. Credentials can be cached for up to 3932100 seconds. The default value is **900** seconds (15 minutes).

Note: If you specify **0**, this increases traffic to the LDAP server because each authentication request generates an authentication and authorization request to the server.

---

6. You can specify a virtual URL based on the individual realm. For information on the virtual URL, see Chapter 3: "Controlling Access to the Internet and Intranet" on page 23.
7. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Manage an LDAP Realm

- To enter configuration mode:

```
SGOS#(config) security ldap create-realm {ad | iplanet | nds | other}
realm_name [base_dn] primary_host [primary_port]
#(config) security ldap edit-realm realm_name
```

- The following subcommands are available:

```
#(config ldap realm_name) alternate-server host [port]
#(config ldap realm_name) cache-duration seconds
#(config ldap realm_name)  case-sensitive {disable | enable}
#(config ldap realm_name) default-group-name default_group_name
#(config ldap realm_name) display-name display_name
#(config ldap realm_name) distinguished-name user-attribute-type
user_attribute_type
```

```
#(config ldap realm_name) distinguished-name base-dn {add | demote |
promote | remove} {base_dn | clear}
#(config ldap realm_name) exit
#(config ldap realm_name) membership-attribute attribute_name
#(config ldap realm_name) membership-type {group | user}
#(config ldap realm_name) membership-username {full | relative}
#(config ldap realm_name) no alternate-server
#(config ldap realm_name) no default-group-name
#(config ldap realm_name) no membership-attribute
#(config ldap realm_name) objectclass container {add | remove}
{container_objectclass | clear}
#(config ldap realm_name) objectclass group {add | remove}
{group_objectclass | clear}
#(config ldap realm_name) objectclass user {add | remove}
{user_objectclass | clear}
#(config ldap realm_name) protocol-version {2 | 3}
#(config ldap realm_name) referrals-follow {disable | enable}
#(config ldap realm_name) rename new_realm_name
#(config ldap realm_name) search anonymous {disable | enable}
#(config ldap realm_name) search dereference {always | finding | never
| searching}
#(config ldap realm_name) search  encrypted-password
encrypted_password
#(config ldap realm_name) search password password
#(config ldap realm_name) search user-dn user_dn
#(config ldap realm_name) server-type {ad | iplanet | nds | other}
#(config ldap realm_name) spoof-authentication {none | origin | proxy}
#(config ldap realm_name) ssl {disable | enable}
#(config ldap realm_name) ssl-verify-server {disable | enable}
#(config ldap realm_name) timeout seconds
#(config ldap realm_name) validate-authorized-user {disable | enable}
#(config ldap realm_name) view
#(config ldap realm_name) virtual-url url
```

Creating the CPL

Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate for your purposes.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file layers.

Be aware that the default policy condition for these examples is *allow*. The default policy condition on new SGOS 5.x systems is *deny*.

- Every LDAP-authenticated user is allowed access the SG appliance.

```
<Proxy>
   authenticate(LDAPRealm)
```

- Group membership is the determining factor in granting access to the SG appliance.

```
<Proxy>
   authenticate(LDAPRealm)
<Proxy>
   group="cn=proxyusers, ou=groups, o=myco"
   deny
```

- A subnet definition determines the members of a group, in this case, members of the Human Resources department.

```
<Proxy>
   authenticate(LDAPRealm)
<Proxy>
   Define subnet HRSubnet
      192.168.0.0/16
      10.0.0.0/24
   End subnet HRSubnet
   [Rule] client_address=HRSubnet
       url.domain=monster.com
       url.domain=hotjobs.com
       deny
.
.
.
   [Rule]
     deny
```

Chapter 10: Local Realm Authentication and Authorization

Using a Local realm is appropriate when the network topography does not include external authentication or when you want to add users and administrators to be used by the SG appliance only.

The Local realm (you can create up to 40) uses a *Local User List*, a collection of users and groups stored locally on the SG appliance. You can create up to 50 different Local User Lists. Multiple Local realms can reference the same list at the same time, although each realm can only reference one list at a time. The default list used by the realm can be changed at any time.

This section discusses the following topics:

Creating a Local Realm

To create a local realm:

1. Select **Configuration > Authentication > Local > Local Realms**.
2. Click **New**.

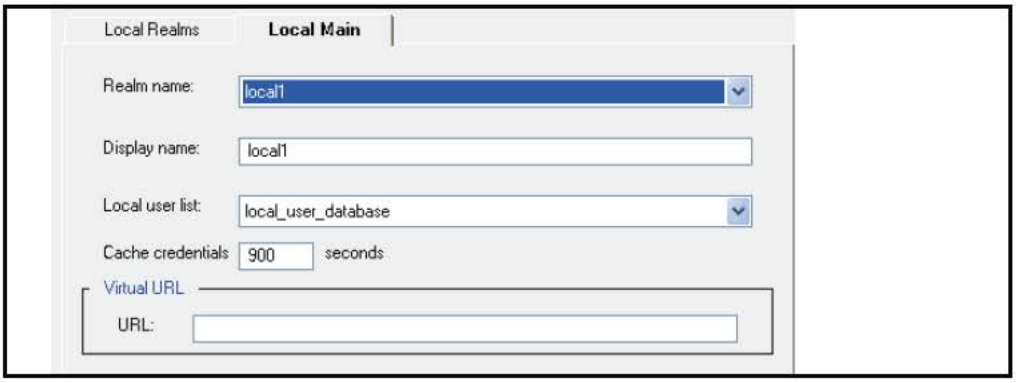
3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name must start with a letter.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

Changing Local Realm Properties

Once you have created a Local realm, you can modify the properties.

To define or change local realm properties:

1. Select **Configuration > Authentication > Local > Local Main**.



2. **Display name**: The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
3. **Local User List**: Specify the local user list from the drop-down list.
4. Specify the length of time, in seconds, that user and administrator credentials received from the Local password file are cached. Credentials can be cached for up to 3932100 seconds. The default is **900** seconds (15 minutes).
5. You can specify a virtual URL based on the individual realm. For information on using virtual URLs, see Chapter 3: "Controlling Access to the Internet and Intranet" on page 23.
6. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Define or Change Local Realm Properties

- To enter configuration mode:

```
SGOS#(config) security local create-realm realm_name
SGOS#(config) security local edit-realm realm_name
```

- The following subcommands are available:

```
#(config local realm_name) cache-duration seconds
#(config local realm_name) default-group-name default_group_name
#(config local realm_name) display-name display_name
#(config local realm_name) exit
#(config local realm_name) local-user-list local_user_list_name
#(config local realm_name) no default-group-name
#(config local realm_name) rename new_realm_name
#(config local realm_name) spoof-authentication {none | origin |
proxy}
#(config local realm_name) validate-authorized-user {disable |
enable}
#(config local realm_name) view
#(config local realm_name) virtual-url url
```

Defining the Local User List

Defining the local user list involves the following steps:

- Create a list or customize the default list for your needs.
- Upload a user list or add users and groups through the CLI.

- Associate the list with the realm.

Creating a Local User List

The user list *local_user_database* is created on a new system or after an upgrade. It is empty on a new system. If a password file existed on the SG appliance before an upgrade, then the list contains all users and groups from the password file; the initial default user list is *local_user_database*. If a new user list is created, the default can be changed to point to it instead by invoking the `security local-user-list default list` *list name* command. You can create up to 50 new lists with 10,000 users each.

Lists can be uploaded or you can directly edit lists through the CLI. If you want to upload a list, it must be created as a text file using the `.htpasswd` format of the SG appliance.

Each user entry in the list consists of:

- username
- List of groups
- Hashed password
- Enabled/disabled boolean searches

A list that has been populated looks like this:

```
SGOS#(config) security local-user-list edit list_name
SGOS#(config local-user-list list_name) view
list20
Lockout parameters:
  Max failed attempts: 60
  Lockout duration:    3600
  Reset interval:      7200
Users:
admin1
  Hashed Password: $1$TvEzpZE$Z2A/OuJU3w5LnEONDHkmg.
  Enabled: true
  Groups:
    group1
admin2
  Hashed Password: $1$sKJvNB3r$xsInBU./2hhBz6xDAHpND.
  Enabled: true
  Groups:
    group1
    group2
admin3
  Hashed Password: $1$duuCUt30$keSdIkZVS4RyFz47G78X20
  Enabled: true
  Groups:
    group2
Groups:
  group1
  group2
```

To create a new empty local user list:

```
SGOS#(config) security local-user-list create list_name
```

Username

The username must be case-sensitively unique, and can be no more than 64 characters long. All characters are valid, except for a colon (`:`).

A new local user is enabled by default and has an empty password.

List of Groups

You cannot add a user to a group unless the group has previously been created in the list. The group name must be case-sensitively unique, and can be no more than 64 characters long. All characters are valid, except for colon (`:`).

The groups can be created in the list; however, their user permissions are defined through policies only.

Hashed Password

The hashed password must be a valid UNIX DES or MD5 password whose plain-text equivalent cannot be more than 64 characters long.

To populate the local user list using an off-box `.htpasswd` file, continue with the next section. To populate the local user list using the SG appliance CLI, go to "Defining the Local User List" on page 104.

Populating a List using the .htpasswd File

To add users to a text file in `.htpasswd` format, enter the following UNIX `htpasswd` command:

```
prompt> htpasswd [-c] .htpasswd username
```

The `-c` option creates a new `.htpasswd` file and should only be used for the very first `.htpasswd` command. You can overwrite any existing `.htpasswd` file by using the `-c` option.

After entering this command, you are prompted to enter a password for the user identified by *username*. The entered password is hashed and added to the user entry in the text file. If the `-m` option is specified, the password is hashed using MD5; otherwise, UNIX DES is used.

Important: Because the -c option overwrites the existing file, do not use the option if you are adding users to an existing .htpasswd file.

Once you have added the users to the `.htpasswd` file, you can manually edit the file to add user groups. When the `.htpasswd` file is complete, it should have the following format:

```
user:encrypted_password:group1,group2,…
user:encrypted_password:group1,group2,…
```

Note: You can also modify the users and groups once they are loaded on the SG appliance. To modify the list once it is on the appliance, see "Populating a Local User List through the SG Appliance" on page 107.

Uploading the .htpasswd File

When the `.htpasswd` file is uploaded, the entries from it either replace all entries in the default local user list or append to the entries in the default local user list. One default local user list is specified on the SG appliance.

To set the default local user list use the command `security local-user-list default list list_name`. The list specified must exist.

To specify that the uploaded `.htpasswd` file replace all existing user entries in the default list, enter `security local-user-list default append-to-default disable` before uploading the .htpasswd file.

To specify that the `.htpasswd` file entries should be appended to the default list instead, enter `security local-user-list default append-to-default enable.`

To upload the .htpasswd file:

The `.htpasswd` file is loaded onto the SG appliance with a Perl script found at:

```
http://download.bluecoat.com/release/tools/set_auth.zip
```

Unzip the file, which contains the `set_auth.pl` script.

Note: To use the `set_auth.pl` script, you must have Perl binaries on the system where the script is running.

To load the .htpasswd file:

```
prompt> set_auth.pl username password
path_to_.htpasswd_file_on_local_machine ip_address_of_the_SG
```

where *username* and *password* are valid administrator credentials for the SG appliance.

Populating a Local User List through the SG Appliance

You can populate a local user list from scratch or modify a local user list that was populated by loading an `.htpasswd` file.

To create a new, empty local user list:

```
SGOS#(config) security local-user-list create list_name
```

To modify an existing local user list (can be empty or contain users):

- To enter configuration mode:

```
SGOS#(config) security local-user-list edit list_name
SGOS#(config local-user-list list_name)
```

- The following subcommands are available:

Note: To add users and groups to the list, enter the following commands, beginning with groups, since they must exist before you can add them to a user account.

```
SGOS#(config local-user-list list_name) group create group1
SGOS#(config local-user-list list_name) group create group2
SGOS#(config local-user-list list_name) group create group3
SGOS#(config local-user-list list_name) user create username
SGOS#(config local-user-list list_name) user edit username
SGOS#(config local-user-list list_name username) group add groupname1
SGOS#(config local-user-list list_name username) group add groupname2
SGOS#(config local-user-list list_name username) password password
-or-
SGOS#(config local-user-list list_name username) hashed-password
hashed-password
```

Note: If you enter a plain-text password, the SG appliance hashes the password. If you enter a hashed password, the appliance does not hash it again.

1. (Optional) The user account is enabled by default. To disable a user account:

```
SGOS#(config local-user-list list_name username) disable
ok
```

2. Repeat the above steps for each user you want added to the list.

To view the results of an individual user account:

Remain in the user account submode and enter the following command:

```
SGOS#(config local-user-list list_name username) view
admin1
  Hashed Password: $1$TvEzpZE$Z2A/OuJU3w5LnEONDHkmg.
  Enabled: true
  Failed Logins: 6
  Groups:
    group1
```

Note: If a user has no failed logins, the statistic does not display.

To view the users in the entire list:

Exit the user account submode and enter:

```
SGOS#(config local-user-list list_name username) exit
SGOS#(config local-user-list list_name) view
list20
Lockout parameters:
  Max failed attempts: 60
  Lockout duration:    3600
  Reset interval:      7200
Users:
admin1
  Hashed Password: $1$TvEzpZE$Z2A/OuJU3w5LnEONDHkmg.
  Enabled: true
  Groups:
    group1
admin2
  Hashed Password: $1$sKJvNB3r$xsInBU./2hhBz6xDAHpND.
  Enabled: true
  Groups:
    group1
    group2
admin3
  Hashed Password: $1$duuCUt30$keSdIkZVS4RyFz47G78X20
  Enabled: true
  Groups:
    group2
Groups:
  group1
  group2
```

To view all the lists on the SG appliance:

```
SGOS#(config) show security local-user-list
Default List: local_user_database
Append users loaded from file to default list: false
local_user_database
Lockout parameters:
  Max failed attempts: 60
  Lockout duration:    3600
  Reset interval:      7200
  Users:
    Groups:
test1
  Users:
    Groups:
```

To delete groups associated with a user:

```
SGOS#(config local-user-list list_name username) group remove
group_name
```

To delete users from a list:

```
SGOS#(config local-user-list list_name) user delete username
This will permanently delete the object. Proceed with deletion?
(y or n) y
ok
```

To delete all users from a list:

```
SGOS#(config local-user-list list_name) user clear
ok
```

The groups remain but have no users.

To delete all groups from a list:

```
SGOS#(config local-user-list list_name) group clear
ok
```

The users remain but do not belong to any groups.

Enhancing Security Settings for the Local User List

You can configure a local user database so that each user account is automatically disabled if too many failed login attempts occur for the account in too short a period, indicating a brute-force password attack on the SG appliance. The security settings are available through the CLI only.

Available security settings are:

- Maximum failed attempts: The maximum number of failed password attempts allowed for an account. When this threshold is reached, the account is disabled (locked). If this is zero, there is no limit. The default is 60 attempts.
- Lockout duration: The time after which a locked account is re-enabled. If this is zero, the account does not automatically re-enable, but instead remains locked until manually enabled. The default is 3600 seconds (one hour).
- Reset interval: The time after which a failed password count resets after the last failed password attempt. If this is zero, the failed password count resets only when the account is enabled or when its password is changed. The default is 7200 seconds (two hours).

These values are enabled by default on the system for all user account lists. You can change the defaults for each list that exists on the system.

To change the security settings for a specific user account list:

1. Enter the following commands from the `(config)` prompt:

```
SGOS#(config) security local-user-list edit list_name
SGOS#(config local-user-list list_name) lockout-duration seconds
SGOS#(config local-user-list list_name) max-failed-attempts attempts
SGOS#(config local-user-list list_name) reset-interval seconds
```

2. (Optional) View the settings:

```
SGOS#(config local-user-list list_name) view
listname
Lockout parameters:
  Max failed attempts: 45
  Lockout duration:    3600
  Reset interval:      0
```

3. (Optional) To disable any of these settings:

```
SGOS#(config local-user-list list_name) no [lockout-duration | max-failed-attempts | reset-interval]
```

Creating the CPL

Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate for your purposes. (The default policy in these examples is deny.)

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file layers.

- Every Local-authenticated user is allowed access the SG appliance.

```
<Proxy>
   authenticate(LocalRealm)
```

- Group membership is the determining factor in granting access to the SG appliance.

```
<Proxy>
   authenticate(LocalRealm)
<Proxy>
   group="group1" allow
```

- A subnet definition determines the members of a group, in this case, members of the Human Resources department.

```
<Proxy>
   authenticate(LocalRealm)
<Proxy>
   Define subnet HRSubnet
      192.168.0.0/16
      10.0.0.0/24
   End subnet HRSubnet
   [Rule] client_address=HRSubnet
     url.domain=monster.com
     url.domain=hotjobs.com
     deny
```

```
.
.
.
   [Rule]
          deny
```

Chapter 11: Netegrity SiteMinder Authentication

The SG appliance can be configured to consult a SiteMinder policy server for authentication and session management decisions. This requires that a SiteMinder realm be configured on the SG appliance and policy written to use that realm for authentication.

Access to the SiteMinder policy server is done through the Blue Coat Authentication and Authorization Agent (BCAAA), which must be installed on a Windows 2000 system or higher with access to the SiteMinder policy servers.

Understanding SiteMinder Interaction with Blue Coat

Within the SiteMinder system, BCAAA acts as a custom Web agent. It communicates with the SiteMinder policy server to authenticate the user and to obtain a SiteMinder session token, response attribute information, and group membership information.

Custom header and cookie response attributes associated with **OnAuthAccept** and **OnAccessAccept** attributes are obtained from the policy server and forwarded to the SG appliance. They can (as an option) be included in requests forwarded by the *appliance*.

Within the SG system, BCAAA acts as its agent to communicate with the SiteMinder server. The SG appliance provides the user information to be validated to BCAAA, and receives the session token and other information from BCAAA.

Each SG SiteMinder realm used causes the creation of a BCAAA process on the Windows host computer running BCAAA. A single host computer can support multiple SG realms (from the same or different SG appliances); the number depends on the capacity of the BCAAA host computer and the amount of activity in the realms.

Note: Each (active) SiteMinder realm on the SG appliance should reference a different agent on the Policy Server.

Configuration of the SG's realm must be coordinated with configuration of the SiteMinder policy server. Each must be configured to be aware of the other. In addition, certain SiteMinder responses must be configured so that BCAAA gets the information the SG appliance needs.

Configuring the SiteMinder Policy Server

Note: Blue Coat assumes you are familiar with configuration of SiteMinder policy servers and Web agents.

Since BCAAA is a Web agent in the SiteMinder system, it must be configured on the SiteMinder policy server. Configuration of BCAAA on the host computer is not required; the agent obtains its configuration information from the SG appliance.

A suitable Web agent must be created and configured on the SiteMinder server. This must be configured to support 5.x agents, and a shared secret must be chosen and entered on the server (it must also be entered in the SG SiteMinder realm configuration).

SiteMinder protects resources identified by URLs. An SG realm is associated with a single protected resource. This could be an already existing resource on a SiteMinder server, (typical for a reverse proxy arrangement) or it could be a resource created specifically to protect access to SG services (typical for a forward proxy).

Important: The request URL is not sent to the SiteMinder policy server as the requested resource; the requested resource is the entire SG realm. Access control of individual URLs is done on the SG appliance using CPL or VPM.

The SiteMinder realm that controls the protected resource must be configured with a compatible authentication scheme. The supported schemes are Basic (in plain text and over SSL), Forms (in plain text and over SSL), and X.509 certificates. Configure the SiteMinder realm with one of these authentication schemes.

Note: Only the following X.509 Certificates are supported: X.509 Client Cert Template, X.509 Client Cert and Basic Template, and X.509 Client Cert and Form Template.

The SG appliance requires information about the authenticated user to be returned as a SiteMinder response. The responses should be sent by an `OnAuthAccept` rule used in the policy that controls the protected resource.

The responses must include the following:

- A Web-Agent-HTTP-Header-variable named `BCSI_USERNAME`. It must be a user attribute; the value of the response must be the simple username of the authenticated user. For example, with an LDAP directory this might be the value of the cn attribute or the uid attribute.
- A Web-Agent-HTTP-Header-variable named `BCSI_GROUPS`. It must be a user attribute and the value of the response must be `SM_USERGROUPS`.

If the policy server returns an LDAP FQDN as part of the authentication response, the SG appliance uses that LDAP FQDN as the FQDN of the user.

Once the SiteMinder agent object, configuration, realm, rules, responses and policy have been defined, the SG appliance can be configured.

Additional SiteMinder Configuration Notes

Note: Additional configuration might be needed on the SiteMinder server depending on specific features being used.

- If using single-sign on (SSO) with off-box redirection (such as to a forms login page), the forms page must be processed by a 5.x or later Web Agent, and that agent must be configured with `fcccompatmode=no`. This precludes that agent from doing SSO with 5.x agents.
- For SSO to work with other Web agents, the other agents must have the `AcceptTPCookie=YES` as part of their configuration. This is described in the SiteMinder documentation.
- Blue Coat does not extract the issuerDN from X.509 certificates in the same way as the SiteMinder agent. Thus, a separate certificate mapping might be needed for the SGOS agent and the SiteMinder agents.

For example, the following was added to the SiteMinder policy server certificate mappings:

```
CN=Waterloo Authentication and Security Team,OU=Waterloo R&D, O=Blue Coat\, Inc.,L=Waterloo,ST=ON,C=CA
```

- In order to use off-box redirection (such as an SSO realm), all agents involved must have the setting `EncryptAgentName=no` in their configurations.
- The SG appliance's credential cache only caches the user's authentication information for the smaller of the time-to-live (TTL) configured on the SG appliance and the session TTL configured on the SiteMinder policy server.

Configuring the SG Realm

The SG realm must be configured so that it can:

- Find the Blue Coat agent(s) that acts on its behalf (hostname or IP address, port, SSL options, and the like).
- Provide BCAAA with the information necessary to allow it to identify itself as a Web agent (agent name, shared secret).
- Provide BCAAA with the information that allows it to find the SiteMinder policy server (IP address, ports, connection information.)
- Provide BCAAA with the information that it needs to do authentication and collect authorization information (protected resource name), and general options (server fail-over and off-box redirection)

For more information on configuring the SG SiteMinder realm, see "Creating a SiteMinder Realm" on page 116.

Note: All SG appliance and agent configuration is done on the appliance. The appliance sends the necessary information to BCAAA when it establishes communication.

Participating in a Single Sign-On (SSO) Scheme

The SG appliance can participate in SSO with other systems that use the same SiteMinder policy server. Users must supply their authentication credentials only once to any of the systems participating. Participating in SSO is not a requirement, the SG appliance can use the SiteMinder realm as an ordinary realm.

When using SSO with SiteMinder, the SSO token is carried in a cookie (`SMSESSION`). This cookie is set in the browser by the first system that authenticates the user; other systems obtain authentication information from the cookie and so do not have to challenge the user for credentials. The SG appliance sets the `SMSESSION` cookie if it is the first system to authenticate a user, and authenticates the user based on the cookie if the cookie is present.

Since the SSO information is carried in a cookie, all the servers participating must be in the same cookie domain, including the SG appliance. This imposes restrictions on the `authenticate.mode()` used on the SG appliance.

- A reverse proxy can use any `origin` mode.
- A forward proxy must use one of the `origin-redirect` modes (such as `origin-cookie-redirect`). When using `origin-*-redirect` modes, the virtual URL hostname must be in the same cookie domain as the other systems. It cannot be an IP address and the default `www.cfauth.com` does not work either.

When using `origin-*-redirect`, the SSO cookie is automatically set in an appropriate response after the SG appliance authenticates the user. When using `origin` mode (in a reverse proxy), setting this cookie must be explicitly specified by the administrator. The policy substitution variable `$(x-agent-sso-cookie)` expands to the appropriate value of the set-cookie: header.

Avoiding SG Appliance Challenges

In some SiteMinder deployments all credential challenges are issued by a central authentication service (typically a Web server that challenges through a form). Protected services do not challenge and process request credentials; instead, they work entirely with the SSO token. If the request does not include an SSO token, or the SSO token is not acceptable, the request is redirected to the central service, where authentication occurs. Once authentication is complete, the request is redirected to the original resource with a response that sets the SSO token.

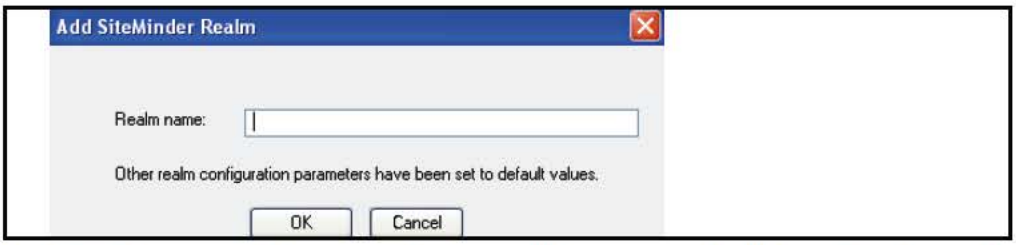
If the SiteMinder policy server is configured to use a forms-based authentication scheme, the above happens automatically. However, in this case, the SG realm can be configured to redirect to an off-box authentication service always. The URL of the service is configured in the scheme definition on the SiteMinder policy server. The SG realm is then configured with `always-redirect-offbox` enabled.

The SG appliance must not attempt to authenticate a request for the off-box authentication URL. If necessary, `authenticate(no)` can be used in policy to prevent this.

Creating a SiteMinder Realm

To create a SiteMinder realm:

1. Select **Configuration > Authentication > Netegrity SiteMinder > SiteMinder Realms**.
2. Click **New**.



3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter. The name should be meaningful to you, but it does not have to be the name of the SiteMinder policy server.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

Configuring Agents

You must configure the SiteMinder realm so that it can find the Blue Coat Authentication and Authorization Agent (BCAAA).

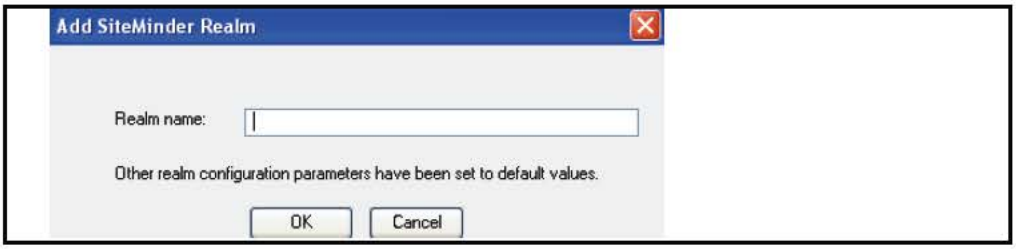
1. Select **Configuration > Authentication > Netegrity SiteMinder > Agents**.

2. Select the realm name to edit from the drop-down list.
3. In the Primary agent section, enter the hostname or IP address where the agent resides.
4. Change the port from the default of 16101 if necessary.
5. Enter the agent name in the **Agent name** field. The agent name is the name as configured on the SiteMinder policy server.
6. You must create a secret for the Agent that matches the secret created on the SiteMinder policy server. Click **Change Secret**. SiteMinder secrets can be up to 64 characters long and are always case sensitive.
7. (Optional) Enter an alternate agent host and agent name in the **Alternate agent** section.
8. (Optional) Click **Enable SSL** to enable SSL between the SG appliance and the BCAAA.
9. (Optional) By default, if SSL is enabled, the SiteMinder BCAAA certificate is verified. To not verify the agent certificate, disable this setting.

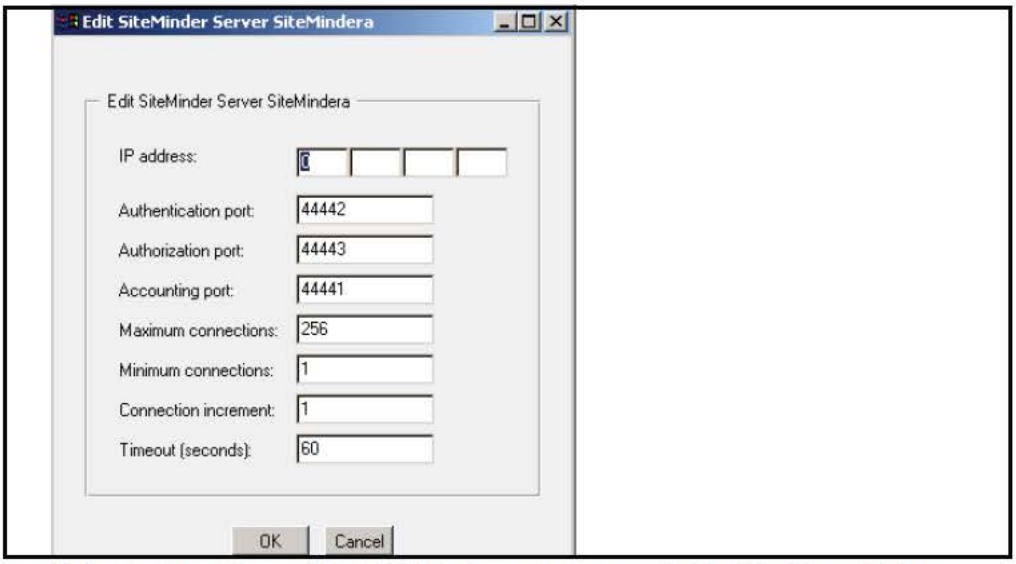
Configuring SiteMinder Servers

Once you create a SiteMinder realm, use the SiteMinder Servers page to create and edit the list of SiteMinder policy servers consulted by the realm.

1. Select **Configuration > Authentication > Netegrity SiteMinder > SiteMinder Servers**.
2. From the **Realm Name** drop-down list, select the SiteMinder realm for which you want to add servers or change server properties.
3. To create a new SiteMinder policy server, click **New**.



4. Enter the name of the server in the dialog. This name is used only to identify the server in the SG appliance's configuration; it usually is the real hostname of the SiteMinder policy server.
5. Click **OK**.
6. To edit an existing SiteMinder policy server, click **Edit**.



   a. Enter the IP address of the SiteMinder policy server in the **IP address** field.
   b. Enter the correct port number for the **Authentication**, **Authorization**, and **Accounting** ports. The ports should be the same as the ports configured on their SiteMinder policy server. The valid port range is 1-65535.
   c. The maximum number of connections is 32768; the default is **256**.
   d. The connection increment specifies how many connections to open at a time if more are needed and the maximum is not exceeded. The default is **1**.
   e. The timeout value has a default of **60** seconds, which can be changed.
7. Click **OK**.
8. Select **Apply** to commit the changes to the SG appliance.

Defining SiteMinder Server General Properties

The **SiteMinder Server General** tab allows you to specify the protected resource name, the server mode, and whether requests should always be redirected off box.

To configure general settings:

1. Select **Configuration > Authentication > Netegrity SiteMinder > SiteMinder Server General**.

2. From the **Realm Name** drop-down list, select the SiteMinder realm for which you want to change properties.
3. Enter the protected resource name. The protected resource name is the same as the resource name on the SiteMinder policy server that has rules and policy defined for it.
4. In the **Server mode** drop-down list, select either **failover** or **round-robin**. Failover mode falls back to one of the other servers if the primary one is down. Round-robin modes specifies that all of the servers should be used together in a round-robin approach. Failover is the default.

Note: The server mode describes the way the agent (BCAAA) interacts with the SiteMinder policy server, not the way that SG appliance interacts with BCAAA.

---

5. To force authentication challenges to always be redirected to an off-box URL, select **Always redirect off-box**.

Note: All SiteMinder Web agents involved must have the setting `EncryptAgentName=no` in their configurations to go off-box for any reason.

If using SiteMinder forms for authentication, the SG appliance always redirects the browser to the forms URL for authentication. You can force this behavior for other SiteMinder schemes by configuring the **always redirect off-box** property on the realm.

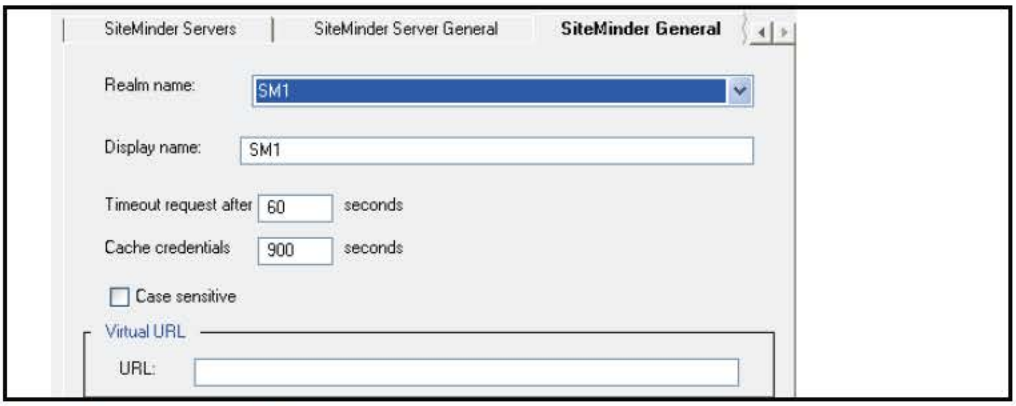
6. If your Web applications need information from the SiteMinder policy server responses, you can select **Add Header Responses**. Responses from the policy server obtained during authentication are added to each request forwarded by the SG appliance. Header responses replace any existing header of the same name; if no such header exists, the header is added. Cookie responses replace a cookie header with the same cookie name; if no such cookie header exists, one is added.
7. To enable validation of the client IP address, select **Validate client IP address**. If the client IP address in the SSO cookie can be valid yet different from the current request client IP address, due to downstream proxies or other devices, deselect **Validate client IP address** for the realm. SiteMinder agents participating in SSO with the SG appliance should also be modified; set the **TransientIPCheck** variable to **yes** to enable IP address validation and **no** to disable it.
8. Select **Apply** to commit the changes to the SG appliance.

Configuring General Settings for SiteMinder

The SiteMinder General tab allows you to set a display name, cache credentials, timeout value, and create a virtual URL.

To manage general settings for the SiteMinder realm:

1. Select **Authentication > Netegrity SiteMinder > SiteMinder General**.



2. From the **Realm Name** drop-down list, select the SiteMinder realm for which you want to change properties.
3. If needed, change the SiteMinder realm display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
4. Specify the length of time, in seconds, that user and administrator credentials received from the SiteMinder policy server are cached. Credentials can be cached for up to 3932100 seconds. The default cache-duration is **900** seconds (15 minutes).
5. If you want group comparisons for SiteMinder groups to be case sensitive, select **Case sensitive**.
6. The virtual hostname must be in the same cookie domain as the other servers participating in the SSO. It cannot be an IP address or the default, `www.cfauth.com`.
7. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure a SiteMinder Realm

- To enter configuration mode:

```
#(config) security siteminder create-realm realm_name
#(config) security siteminder edit-realm realm_name
```

- The following subcommands are available:

```
#(config siteminder realm_name) add-header-responses {enable | disable}
#(config siteminder realm_name) alternate-agent agent-name
#(config siteminder realm_name) alternate-agent encrypted-secret encrypted-shared-secret
#(config siteminder realm_name) alternate-agent host
#(config siteminder realm_name) alternate-agent port
#(config siteminder realm_name) alternate-agent shared-secret secret
#(config siteminder realm_name) alternate-agent always-redirect-offbox
```

```
#(config siteminder realm_name) always-redirect-offbox {enable |
disable}
#(config siteminder realm_name) cache-duration seconds
#(config siteminder realm_name) case-sensitive {enable | disable}
#(config siteminder realm_name) display-name display_name
#(config siteminder realm_name) exit
#(config siteminder realm_name) no alternate-agent
#(config siteminder realm_name) primary-agent agent-name
#(config siteminder realm_name) primary-agent encrypted-secret
encrypted-shared-secret
#(config siteminder realm_name) primary-agent host
#(config siteminder realm_name) primary-agent port
#(config siteminder realm_name) primary-agent shared-secret secret
#(config siteminder realm_name) primary-agent always-redirect-offbox
#(config siteminder realm_name) protected-resource-name resource-name
#(config siteminder realm_name) rename new_realm_name
#(config siteminder realm_name) server-mode {failover | round-robin}
#(config siteminder realm_name) validate-client-ip {enable | disable}
#(config siteminder realm_name) siteminder-server create server_name
#(config siteminder realm_name) siteminder-server delete server_name
#(config siteminder realm_name) siteminder-server edit server_name
#(config siteminder realm_name server_name)
   #(config siteminder realm_name server_name) accounting-port
   port_number
   #(config siteminder realm_name server_name) authentication-port
   port_number
   #(config siteminder realm_name server_name) authorization-port
   port_number
   #(config siteminder realm_name server_name) connection-increment
   number
   #(config siteminder realm_name server_name) exit
   #(config siteminder realm_name server_name) ip-address ip_address
   #(config siteminder realm_name server_name) max-connections number
   #(config siteminder realm_name server_name) min-connections number
   #(config siteminder realm_name server_name) timeout seconds
   #(config siteminder realm_name server_name) view
   #(config siteminder realm_name) ssl {enable | disable}
   #(config siteminder realm_name) ssl-verify-agent {enable | disable}
   #(config siteminder realm_name) timeout seconds
   #(config siteminder realm_name) view
   #(config siteminder realm_name) virtual-url url
```

Creating the CPL

You can create CPL policies now that you have completed SiteMinder realm configuration. Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate for your purposes.

The examples below assume the default policy condition is *allow*. On new SGOS 5.x systems, the default policy condition is *deny*.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file `<Proxy>` and other layers.

- Every SiteMinder-authenticated user is allowed access the SG appliance.

```
<Proxy>
  authenticate(SiteMinderRealm)
```

- Group membership is the determining factor in granting access to the SG appliance.

```
<Proxy>
   authenticate(LDAPRealm)
<Proxy>
   group="cn=proxyusers, ou=groups, o=myco"
   deny
```

Chapter 12: Policy Substitution Realm Authentication

A Policy Substitution realm provides a mechanism for identifying and authorizing users based on information in the request to the SG appliance. The realm uses information in the request and about the client to identify the user. The realm is configured to construct user identity information by using policy substitutions.

If authorization data (such as group membership) is needed, the realm can be configured with the name of an associated authorization realm (such as LDAP or local). If an authorization realm is configured, the fully-qualified username is sent to the authorization realm's authority to collect authorization data.

You can use policy substitutions realms in many situations. For example, a Policy Substitution realm can be configured to identify the user:

- based on the results of a NetBIOS over TCP/IP query to the client computer.
- based on the results of a reverse DNS lookup of the client computer's IP address.
- based on the contents of a header in the request. This might be used when a downstream device is authenticating the user.
- based on the results of an Ident query to the client computer.

The Policy Substitution realm is used typically for best-effort user discovery, mainly for logging and subsequent reporting purposes, without the need to authenticate the user. Be aware that if you use Policy Substitution realms to provide granular policy on a user, it might not be very secure because the information used to identify the user can be forged.

This section discusses the following topics:

How Policy Substitution Realms Work

The realm is configured the same way as other realms, except that the realm uses policy substitutions to construct the username and full username from information available in and about the request. Any policy substitution whose value is available at client logon can be used to provide information for the name.

The Policy Substitution realm, in addition to allowing you to create and manipulate realm properties, such as the name of the realm and the number of seconds that credential cache entries from this realm are valid, also contains two other attributes:

- A user field: A string containing policy substitutions that describes how to construct the simple username.
- A full username field: A string containing policy substitutions that describes how to construct the full username, which is used for authorization realm lookups. This can either be an LDAP FQDN when the authorization realm is an LDAP realm, or a simple name when local realms are being used for authorization.

Note: Policy Substitution realms never challenge for credentials. If the username and full username cannot be determined from the configured substitutions, authentication in the Policy Substitution realm fails.

Remember that Policy Substitution realms do not require an authorization realm. If no authorization realm is configured, the user is not a member of any group. The effect this has on the user depends on the authorization policy. If the policy does not make any decisions based on groups, you do not need to specify an authorization realm. Also, if your policy is such that it works as desired when all Policy Substitution realm users are not in any group, you do not have to specify an authorization realm.

Once the Policy Substitution realm is configured, you must create policy to authenticate the user.

Note: If all the policy substitutions fail, authentication fails. If any policy substitution works, authentication succeeds in the realm.

Example

The following is an example of how to use substitutions with Policy Substitution realms.

Assumptions:

- The user susie.smith is logged in to a Windows client computer at IP address 10.25.36.47.
- The Windows messenger service is enabled on the client computer.
- The client computer is in the domain AUTHTEAM.
- The customer has an LDAP directory in which group information is stored. The DN for a user's group information is

  `cn=`*`username`*`,cn=users,dc=`*`computer_domain`*`,dc=`*`company`*`,dc=`*`com`*

  where *`username`* is the name of the user, and *`computer_domain`* is the domain to which the user's computer belongs.
- A login script that runs on the client computer updates a DNS server so that a reverse DNS lookup for `10.25.36.47` results in `susie.smith.authteam.location.company.com`.

Results:

Under these circumstances, the following username and full username attributes might be used:

- **Username**: `$(netbios.messenger-username)@$(client.address)`.

  This results in `SUSIE.SMITH@10.25.36.47`.
- **Full username**: `cn=$(netbios.messenger-username),cn=users, dc=$(netbios.computer-domain),dc=company,dc=com`.

  This results in `cn=SUSIE.SMITH,cn=users, dc=AUTHTEAM,dc=company,dc=com`.
- **Username**: `$(netbios.computer-domain)\$(netbios.messenger-username)`.

  This results in `AUTHTEAM\SUSIE.SMITH`.

- **Username:** `$(client.host:label(6)).$(client.host:label(5))`.

  This results in `SUSIE.SMITH`.

Creating a Policy Substitution Realm

To create a policy substitution realm:

1. Select **Configuration > Authentication > Policy Substitution > Policy Substitution Realms**.
2. Click **New**.

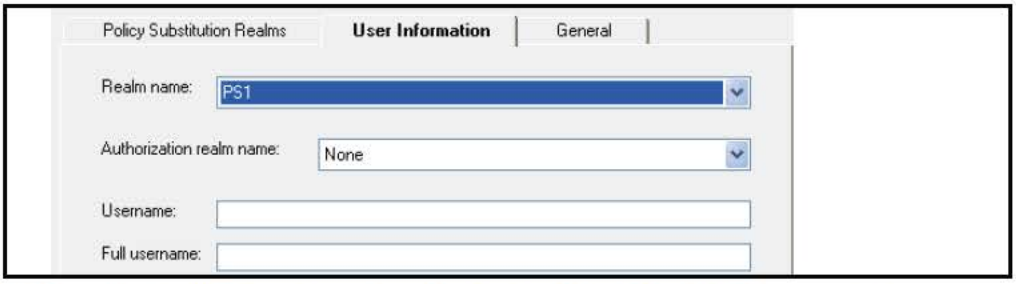
3. In the **Realm name** field, enter a realm name. The name can be up to 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

Defining a Policy Substitution Realm

You can define a Policy Substitution realm through either the Management Console or the CLI.

To define policy substitution user information:

1. Select **Configuration > Authentication > Policy Substitution > User Information**.



2. From the **Realm Name** drop-down list, select the Policy Substitution realm for which you want to change realm properties.
3. (Optional) From the **Authorization Realm Name** drop-down list, select the realm you want to use to authorize users.

Note: Remember that Policy Substitution realms do not require an authorization realm. If the policy does not make any decisions based on groups, you do not need to specify an authorization realm.

---

4. To construct usernames and full usernames, remember that the **Username** and **Full username** attributes are character strings that contain policy substitutions. When authentication is required for the transaction, these character strings are processed by the policy substitution mechanism, using the current transaction as input. The resulting string becomes the user's identity for the current transaction. For an overview of usernames and full usernames, see "How Policy Substitution Realms Work" on page 123.

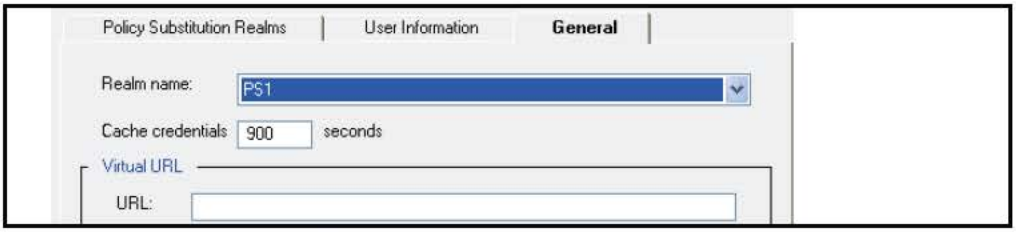
5. Select **Apply** to commit the changes to the SG appliance.

Defining Policy Substitution Realm General Properties

The Policy Substitution General tab allows you to specify the display name and a virtual URL.

To configure policy substitution realm general settings:

1. Select **Configuration > Authentication > Policy Substitution > General**.



2. From the **Realm name** drop-down list, select the Policy Substitution realm for which to change properties.
3. Specify the length of time, in seconds, that user and administrator credentials are cached. Credentials can be cached for up to 3932100 seconds. The default cache-duration is **900** seconds (15 minutes).
4. You can specify a virtual URL. For more information on the virtual URL, see Chapter 3: "Controlling Access to the Internet and Intranet" on page 23.
5. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure a Policy Substitution Realm

- To enter configuration mode:

```
SGOS#(config) security policy-substitution create-realm realm_name
#(config) security policy-substitution edit-realm realm_name
```

- The following subcommands are available:

```
#(config policy-substitution realm_name) authorization-realm-name realm_name
#(config policy-substitution realm_name) cache-duration seconds
#(config policy-substitution realm_name) exit
#(config policy-substitution realm_name) full-username construction_rule
#(config policy-substitution realm_name) no authorization-realm-name
#(config policy-substitution realm_name) rename new_realm_name
#(config policy-substitution realm_name) username construction_rule
```

```
#(config policy-substitution realm_name) view
#(config policy-substitution realm_name) virtual-url url
```

Tips

- Following is an example of how to configure three different types of Policy Substitution realms. For a list of available substitutions, refer to *Volume 9: Access Logging*.

  - Identity to be determined by sending a NetBIOS over TCP/IP query to the client computer, and using LDAP authorization

    ```
    SGOS#(config) security policy-substitution create-realm netbios
    SGOS#(config) security policy-substitution edit-realm netbios
    SGOS#(config policy-substitution netbios) username \
    $(netbios.messenger-username)
    SGOS#(config policy-substitution netbios) full-username \
    cn=$(netbios.messenger-username),cn=users,dc=company,dc=com
    SGOS#(config policy-substitution netbios) authorization-realm-
    name ldap
    ```

  - Identity to be determined by reverse DNS, using local authorization. Blue Coat assumes login scripts on the client computer update the DNS record for the client.

    ```
    SGOS#(config) security policy-substitution create-realm RDNS
    SGOS#(config) security policy-substitution edit-realm RDNS
    SGOS#(config policy-substitution RDNS) username \
    $(client.host:label(5)).$(client.host:label(6))
    #SGOS#(config policy-substitution RDNS) full-username \
    $(client.host:label(5)).$(client.host:label(6))
    SGOS#(config policy-substitution RDNS) authorization-realm-name
    local
    ```

  - Identity to be determined by a header in the request, using LDAP authorization.

    ```
    SGOS#(config) security policy-substitution create-realm header
    SGOS#(config) security policy-substitution edit-realm header
    SGOS#(config policy-substitution header) username \
    $(request.x_header.username)
    SGOS#(config policy-substitution header) full-username \
    cn=$(request.x_header.username),cn=users,dc=company,dc=com
    SGOS#(config policy-substitution header) username \
    authorization-realm-name ldap
    ```

- If you need to change the NetBIOS defaults of 5 seconds and 3 retries, use the `nbstat requester` option from the `netbios` command submode. (For more information on using the NetBIOS commands, refer to *Volume 12: Blue Coat SG Appliance Command Line Reference*.)

Creating the Policy Substitution Policy

When you complete Policy Substitution realm configuration, you must create CPL policies for the policy-substitution realm to be used. Be aware that the example below is just part of a comprehensive authentication policy. By themselves, they are not adequate.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file `<Proxy>` and other layers.

Be aware that the default policy condition for this example is *allow*. On new SGOS 5.x systems, the default policy condition is *deny*.

Note: The Policy Substitution realm cannot be used in an <Admin> layer.

- Every Policy Substitution realm authenticated user is allowed to access the SG appliance.

```
<Proxy>
  authenticate(PolicySubstitutionRealm)
```

Notes

The Policy Substitution realm can lose the logon if the NetBIOS computer name cannot by determined through a DNS query or a NetBIOS query. The DNS query can fail if the NetBIOS name is different than the DNS host name or if the computer is in a different DNS domain than the BCAAA computer and the BCAAA computer is not set up to impute different DNS domains.

The NetBIOS query can fail because the NetBIOS broadcast does not reach the target computer. This can happen if the computer is behind a firewall that is not forwarding NetBIOS requests or if the computer is on a subnet that is not considered to be local to the BCAAA server.

To prevent this issue, the BCAAA machine must be configured to be able to query the NetBIOS name of any computer of interest and get the correct IP address.

One workaround is to use a WINS server. This works like a DNS server but handles NetBIOS lookups.

Chapter 13: RADIUS Realm Authentication and Authorization

RADIUS is often the protocol of choice for ISPs or enterprises with very large numbers of users. RADIUS is designed to handle these large numbers through centralized user administration that eases the repetitive tasks of adding and deleting users and their authentication information. RADIUS also inherently provides some protection against sniffing.

Some RADIUS servers support one-time passwords. One-time passwords are passwords that become invalid as soon as they are used. The passwords are often generated by a token or program, although pre-printed lists are also used. Using one-time passwords ensures that the password cannot be used in a replay attack.

The SG appliance's one-time password support works with products such as Secure Computing SafeWord synchronous and asynchronous tokens and RSA SecurID tokens.

The SG appliance supports RADIUS servers that use challenge/response as part of the authentication process. SafeWord asynchronous tokens use challenge/response to provide authentication. SecurID tokens use challenge/response to initialize or change PINs.

Note: For this release, HTTP is the only supported protocol.

The challenge is displayed as the realm information in the authentication dialog; Blue Coat recommends that you use form authentication if you create a challenge/response realm, particularly if you use SecurID tokens.

If you set an authentication mode that uses forms, the system detects what type of question is being asked. If it is a yes/no question, it displays the query form with a *yes* and *no* button. If it is a new PIN question, the system displays a form with entry fields for the new PIN.

For information on using form authentication, see Chapter 7: "Forms-Based Authentication" on page 73.

Using policy, you can fine-tune RADIUS realms based on RADIUS attributes. If you use the Blue Coat attribute, groups are supported within a RADIUS realm.

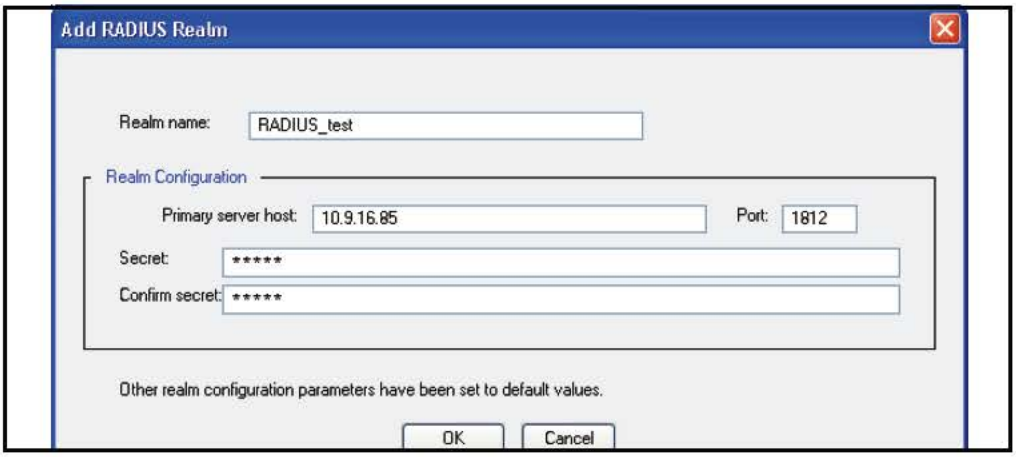
This section discusses the following topics:

Creating a RADIUS Realm

To create a RADIUS realm:

You can create up to 40 RADIUS realms.

1. Select **Configuration > Authentication > RADIUS > RADIUS Realms.**
2. Click **New.**



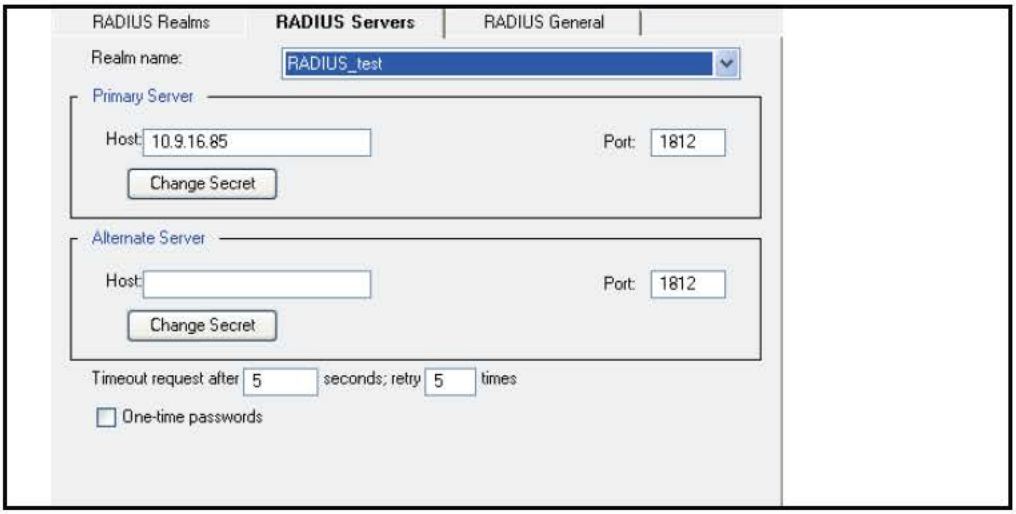
3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter.
4. Specify the host and port for the primary RADIUS server. The default port is **1812**.
5. Specify the RADIUS secret. RADIUS secrets can be up to 64 characters long and are always case sensitive.
6. Confirm the secret.
7. Click **OK.**
8. Select **Apply** to commit the changes to the SG appliance.

Defining RADIUS Realm Properties

Once you have created the RADIUS realm, you can change the primary host, port, and secret of the RADIUS server for that realm.

To re-define RADIUS server properties:

1. Select **Configuration > Authentication > RADIUS > RADIUS Servers.**



2. Specify the host and port for the primary RADIUS server. The default port is **1812**. (To create or change the RADIUS secret, click **Change Secret**. RADIUS secrets can be up to 64 characters long and are always case sensitive.)
3. (Optional) Specify the host and port for the alternate RADIUS server. The default port is **1812**. (To create or change the RADIUS secret, click **Change Secret**. RADIUS secrets can be up to 64 characters long and are always case sensitive.)
4. In the **Timeout Request** field, enter the number of seconds the SG appliance allows for each request attempt before giving up on a server and trying another server. Within a timeout multiple packets can be sent to the server, in case the network is busy and packets are lost. The default request timeout is 10 seconds.
5. In the **Retry** field, enter the number of attempts permitted before marking a server offline. The client maintains an average response time from the server; the retry interval is initially twice the average. If that retry packet fails, then the next packet waits twice as long again. This increases until it reaches the timeout value. The default number of retries is **10**.
6. If you are using one-time passwords, select the One-time passwords checkbox. You must enable one-time passwords if you created a challenge/response realm.
7. Select **Apply** to commit the changes to the SG appliance.

Defining RADIUS Realm General Properties

The RADIUS General tab allows you to specify the display name and a virtual URL.

To configure general settings:

1. Select **Configuration > Authentication > RADIUS > RADIUS General**.
2. If needed, change the RADIUS realm display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be empty.
3. If the RADIUS server is configured to expect case-sensitive usernames and passwords, make sure the **Case sensitive checkbox is selected**.

4. Specify the length of time, in seconds, that user credentials received from the RADIUS server are cached. Credentials can be cached for up to 3932100 seconds. The default is **900** seconds (15 minutes).

   Note: If you specify **0**, traffic is increased to the RADIUS server because each authentication request generates an authentication and authorization request. That is, if a Web page has 15 images and is loaded, you must authenticate 16 times—once for the Web page and once for each image.

5. (Optional) You can specify a virtual URL based on the individual realm. For more information on the virtual URL, see Chapter 3: "Controlling Access to the Internet and Intranet" on page 23.

6. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure a RADIUS Realm

- To enter configuration mode:

```
SGOS#(config) security radius create-realm realm_name secret primary-server_host [primary-server_port]
-or-
SGOS#(config) security radius create-realm-encrypted realm_name encrypted_secret primary_host [primary_port]
```

- The following subcommands are available:

```
#(config radius realm_name) alternate-server encrypted-secret encrypted_secret
#(config radius realm_name) alternate-server host [port]
#(config radius realm_name) alternate-server secret secret
#(config radius realm_name) cache-duration seconds
#(config radius realm_name) case-sensitive {disable | enable}
#(config radius realm_name) display-name display_name
#(config radius realm_name) exit
#(config radius realm_name) no alternate-server
#(config radius realm_name) one-time-passwords {disable | enable}
#(config radius realm_name) primary-server encrypted-secret encrypted_secret
#(config radius realm_name) primary-server host [port]
#(config radius realm_name) primary-server secret secret
#(config radius realm_name) rename new_realm_name
#(config radius realm_name) timeout seconds
#(config radius realm_name) server-retry count
#(config radius realm_name) spoof-authentication {none | origin | proxy}
#(config radius realm_name) view
#(config radius realm_name) virtual-url url
```

Creating the Policy

Fine-tune RADIUS realms through attributes configured by policy—CPL or VPM. You can also create RADIUS groups. To fine-tune RADIUS realms, continue with the next section. To create RADIUS groups, see "Creating RADIUS Groups" on page 134.

Note: RADIUS groups can only be configured through policy. This feature is not available through either the Management Console or the CLI.

Fine-Tuning RADIUS Realms

Fine-tune RADIUS Realms by using the following attributes in the `attribute.<name>` and `has_attribute.<name>` CPL conditions and source objects in VPM.

Table 13-1. RADIUS Attributes for the attribute.<name> and has_attribute.<name> Conditions

RADIUS Attribute Name	CPL Gesture Name	Type (Possible Value)
Callback-ID	attribute.Callback-ID	String
Callback-Number	attribute.Callback-Number	String
Filter-ID	attribute.Filter-ID	String
Framed-IP-Address	attribute.Framed-IP-Address	IP Address
Framed-IP-Netmask	attribute.Framed-IP-Netmask	IP Address
Framed-MTU	attribute.Framed-MTU	Integer
Framed-Pool	attribute.Framed-Pool	Strong
Framed-Protocol	attribute.Framed-Protocol	Integer (1-6)
Framed-Route	attribute.Framed-Route	String
Idle-Timeout	attribute.Idle-Timeout	Integer
Login-LAT-Group	attribute.Login-LAT-Group	String
Login-LAT-Node	attribute.Login-LAT-Node	String
Login-LAT-Port	attribute.Login-LAT-Port	Integer
Login-LAT-Service	attribute.Login-LAT-Service	String
Login-IP-Host	attribute.Login-IP-Host	IP Address
Login-Service	attribute.Login-Service	Integer (0-7)
Login-TCP-Port	attribute.Login-TCP-Port	Integer (0-65535)
Port-Limit	attribute.Port-Limit	Integer
Service-Type	attribute.Service-Type	Integer (1-11)
Session-Timeout	attribute.Session-Timeout	Integer
Tunnel-Assignment-ID	attribute.Tunnel-Assignment-ID	String
Tunnel-Medium-Type	attribute.Tunnel-Medium-Type	Integer (1-15)
Tunnel-Private-Group-ID	attribute.Tunnel-Private-Group-ID	String
Tunnel-Type	attribute.Tunnel-Type	Integer (1-12)
Blue-Coat-Group	attribute.Blue-Coat-Group	String

Creating RADIUS Groups

You can create a RADIUS realm group by using the custom Blue Coat attribute, which can appear multiple times within a RADIUS response. It can be used to assign a user to one or more groups. Values that are found in this attribute can be used for comparison with the group condition in CPL and the group object in VPM. The group name is a string with a length from 1-247 characters. The Blue Coat Vendor ID is 14501, and the Blue-Coat-Group attribute has a Vendor Type of 1.

If you are already using the Filter-ID attribute for classifying users, you can use that attribute instead of the custom Blue-Coat-Group attribute. While the Filter-ID attribute does not work with the CPL group condition or the group object in VPM, the `attribute.Filter-ID` condition can be used to manage users in a similar manner.

CPL Example

Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file layers.

- Every RADIUS-authenticated user is allowed access the SG appliance if the RADIUS attribute service-type is set.

```
<Proxy>
   authenticate(RADIUSRealm)
<Proxy>
   allow has_attribute.Service-Type=yes
   deny
```

- A group called RegisteredUsersGroup is allowed to access the SG appliance if the allow group gesture is defined.

```
<proxy>
  authenticate(RADIUSRealm)
<proxy>
  allow group=RegisteredUsersGroup
  deny
```

Troubleshooting

One of five conditions can cause the following error message:

Your request could not be processed because of a configuration error: "The request timed out while trying to authenticate. The authentication server may be busy or offline."

- The secret is wrong.
- The network is so busy that all packets were lost to the RADIUS server.
- The RADIUS server was slow enough that the SG appliance gave up before the server responded.
- The RADIUS servers are up, but the RADIUS server is not running. In this case, you might also receive ICMP messages that there is no listener.
- RADIUS servers machines are not running/unreachable. Depending on the network configuration, you might also receive ICMP messages.

Chapter 14: Sequence Realm Authentication

Once a realm is configured, you can associate it with other realms to allow Blue Coat to search for the proper authentication credentials for a specific user. That is, if the credentials are not acceptable to the first realm, they are sent to the second, and so on until a match is found or all the realms are exhausted. This is called *sequencing*.

For example, if a company has one set of end-users authenticating against an LDAP server and another using NTLM, a sequence realm can specify to attempt NTLM authentication first; if that fails due to a user-correctable error (such as credentials mismatch or a user not in database) then LDAP authentication can be specified to try next. You can also use sequences to fall through to a policy substitution realm if the user did not successfully authenticate against one of the earlier realms in the sequence.

Note: Errors such as *server down* do not fall through to the next realm in the sequence. Those errors result in an exception returned to the user. Only errors that are end-user correctable result in the next realm in the sequence being attempted.

This section discusses the following topics:

Adding Realms to a Sequence Realm

Keep in mind the following rules for using realm sequences:

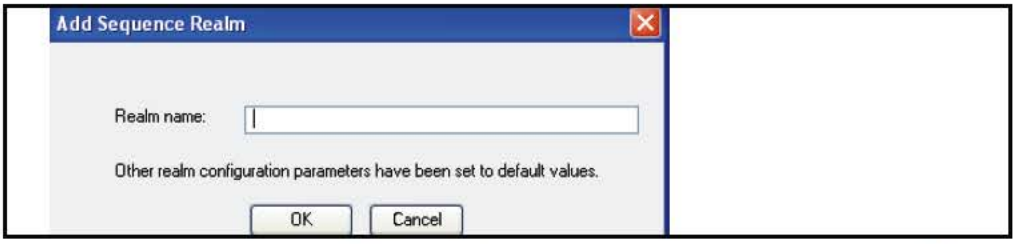
- Ensure the realms to be added to the sequence are customized to your needs. Check each realm to be sure that the current values are correct. For IWA, verify that the **Allow Basic Credentials** checkbox is set correctly.
- All realms in the realm sequence must exist and cannot be deleted or renamed while the realm sequence references them.
- Only one IWA realm is allowed in a realm sequence.
- If an IWA realm is in a realm sequence, it must be either the first or last realm in the list.
- If an IWA realm is in a realm sequence and the IWA realm does not support Basic credentials, the realm must be the first realm in the sequence and try IWA authentication once must be enabled.
- Multiple Basic realms are allowed.
- Multiple Windows SSO realms are allowed.
- Connection-based realms, such as Certificate, are not allowed in the realm sequence.
- A realm can only exist once in a particular realm sequence.
- A realm sequence cannot have another realm sequence as a member.

- If a realm is down, an exception page is returned. Authentication is not tried against the other later realms in the sequence.

Creating a Sequence Realm

To create a sequence realm:

1. Select **Configuration > Authentication > Sequences > Sequence Realms**.
2. Click **New**.

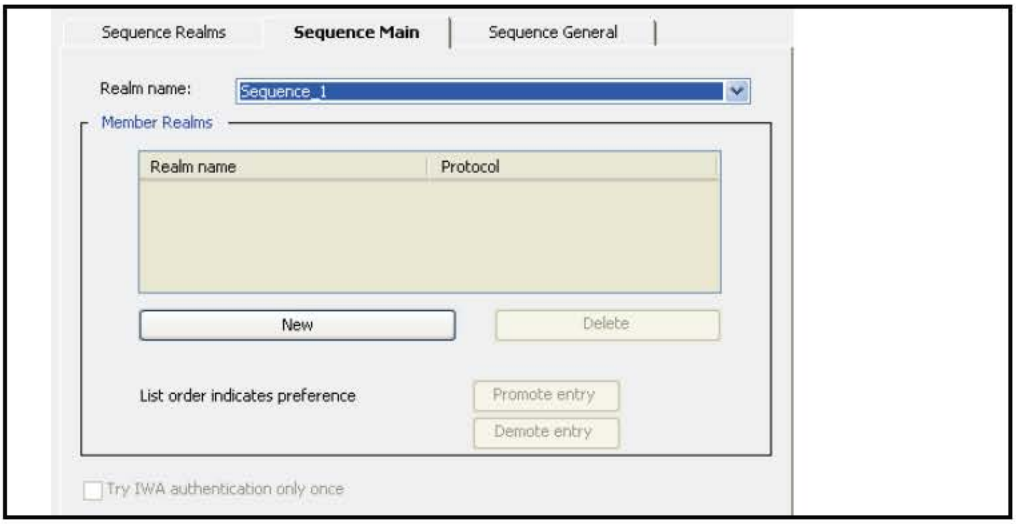


3. In the **Realm name**, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name must start with a letter.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

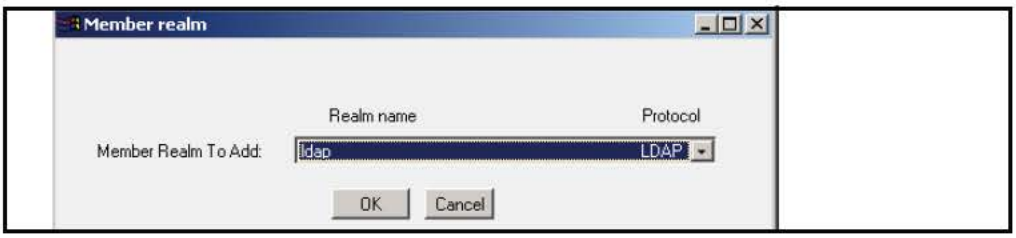
Adding Realms to a Sequence Realm

To add realms to a sequence realm:

1. Select **Configuration > Authentication > Sequences > Sequence Main**.



2. Click **New** to add an existing realm to the realm sequence from the drop-down list. Remember that each realm can be used only once in a realm sequence.



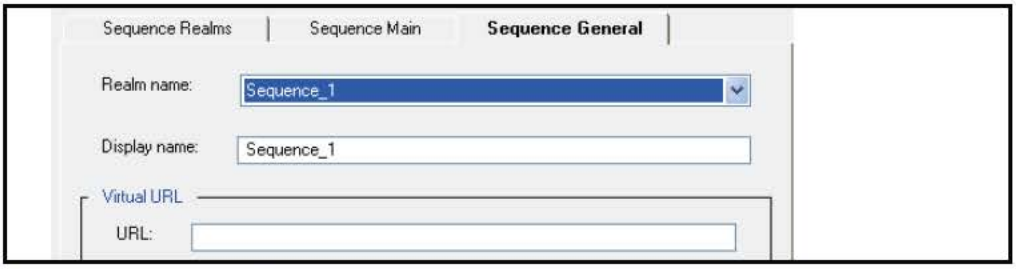
3. From the drop-down list, select the Sequence realm you wanted added to the realm sequence.
4. Click **OK.**

   You are returned to the main Sequences menu.
5. Select **Apply** to commit the changes to the SG appliance.
6. Repeat from 2 until you have added all necessary realms.
7. To change the order that the realms are checked, use the **promote/demote** buttons. When you add an IWA realm, it is placed first in the list and you can allow the realm sequence to **try IWA authentication only once.** If you demote the IWA entry, it becomes last in the sequence and the default of checking IWA multiple times is enabled.
8. Select **Apply** to commit the changes to the SG appliance.

Defining Sequence Realm General Properties

The Sequence General tab allows you to specify the display name and a virtual URL.

1. Select **Configuration > Authentication > Sequences > Sequence General**.



2. From the **Realm name** drop-down list, select the Sequence realm for which you want to change properties.
3. If needed, change the Sequence realm display name. The default value for the display name is the realm name. The display name cannot be longer than 128 characters and it cannot be null.
4. You can specify a virtual URL based on the individual realm sequence. For more information on the virtual URL, see Chapter 3: "Controlling Access to the Internet and Intranet" on page 23.
5. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Configure a Sequence Realm

- To enter configuration mode:

```
SGOS#(config) security sequence create-realm realm_sequence_name
(config) security sequence edit-realm realm_sequence_name
```

- The following subcommands are available:

```
#(config sequence realm_sequence_name)
#(config sequence realm_sequence_name) display-name display_name
#(config sequence realm_sequence_name) exit
#(config sequence realm_sequence_name) IWA-only-once {disable | enable}
#(config sequence realm_sequence_name) realm {add | demote | promote | remove} {realm_name | clear}
#(config sequence realm_sequence_name) rename new_realm_name
#(config sequence realm_sequence_name) view
#(config sequence realm_sequence_name) virtual-url url
```

Tips

- Explicit Proxy involving a sequence realm configured with an NTLM/IWA realm and a substitution realm.

  Internet Explorer (IE) automatically sends Windows credentials in the Proxy-Authorization: header when the SG appliance issues a challenge for NTLM/IWA. The prompt for username/password appears only if NTLM authentication fails. However, in the case of a sequence realm configured with an NTLM/IWA realm and a substitution realm, the client is authenticated as a guest in the policy substitution realm, and the prompt allowing the user to correct the NTLM credentials never appears.

- Transparent Proxy setup involving a sequence realm configured with an NTLM/IWA realm and a substitution realm.

  The only way the SG appliance can differentiate between a domain and non-domain user is though the NTLM/IWA credentials provided during the authentication challenge.

  IE does not offer Windows credentials in the Proxy-Authorization: header when the Proxy issues a challenge for NTLM/IWA unless the browser is configured to do so. In this case, the behavior is the same as for explicit proxy.

  If IE is not configured to offer Windows credentials, the browser issues a prompt for username/password, allowing non-domain users to be authenticated as guests in the policy substitution realm by entering worthless credentials.

Chapter 15: Windows Single Sign-on Authentication

The Windows Single Sign-on (SSO) realm is an authentication mechanism available on Windows networks.

This section discusses the following topics:

How Windows SSO Realms Work

In a Windows SSO realm, the client is never challenged for authentication. Instead, the BCAAA agent collects information about the current logged on user from the domain controller and/or by querying the client machine. Then the IP address of an incoming client request is mapped to a user identity in the domain. If authorization information is also needed, then another realm (LDAP or local) must be created. For more information, see "How Windows SSO Authorization Works" on page 140.

Note: The Windows SSO realm works reliably only in environments where one IP address maps to one user. If an IP address cannot be mapped to a single user, authentication fails. Those with NAT systems, which uses one set of IP addresses for intranet traffic and a different set for Internet traffic, should use a different realm for authentication.

To authenticate a user, the Windows SSO realm uses two methods, either separately or together:

- Domain Controller Querying: The domain controller is queried to identify which users are connecting to, or authenticating with, the domain controller. This can be used to infer the identity of the user at a particular workstation.
- Client Querying: The client workstation is queried to determine who the client workstation thinks is logged in.
- When Domain Controller Querying and Client Querying are both used, the Domain Controller Query result is used if it exists and is still within the valid time-to-live as configured in the `sso.ini` file. If the Domain Controller Query result is older than the configured time-to-live, the client workstation is queried.

Note: Before Domain Controller Querying or Client Querying can be used, the `sso.ini` file, located in the same directory as the BCAAA service, must be modified. For information on modifying this file, see "Modifying the Windows sso.ini File" on page 145.

For the most complete solution, an IWA realm could be configured at the same time as the Windows SSO realm and both realms added to a realm sequence. Then, if the Windows SSO realm failed to authenticate the user, the IWA realm could be used. For information on using a sequence realm, see Chapter 14: "Sequence Realm Authentication" on page 135.

How Windows SSO Works with BCAAA

The server side of the authentication exchange is handled by the Blue Coat Authentication and Authorization Agent (BCAAA). Windows SSO uses a single BCAAA process for all realms and proxies that use SSO.

BCAAA must be installed on a domain controller or member server. By default, the BCAAA service authenticates users in all domains trusted by the computer on which it is running. When using Domain Controller Querying, the BCAAA service can be configured to only query certain domain controllers in those trusted domains.

By default the BCAAA service is installed to run as LocalSystem. For a Windows SSO realm to have correct permissions to query domain controllers and clients, the user who BCAAA runs under must be an authenticated user of the domain.

When the Windows SSO realm is configured to do Client Querying, the user that BCAAA runs under must be an authenticated user of the domain. For failover purposes, a second BCAAA can be installed and configured to act as an alternate BCAAA in the Windows SSO realm. The alternate BCAAA service is used in the event of a failure with the primary BCAAA service configured in the realm.

Note: For information on configuring the BCAAA service as an authenticated user of the domain, see Appendix B: "Using the Authentication/Authorization Agent" on page 157.

How Windows SSO Authorization Works

The Windows SSO realm, in addition to allowing you to create and manipulate realm properties, such as the query type and the number of seconds that credential cache entries from this realm are valid, also contains the authorization username and the name of the realm that will do authorization for the Windows SSO realm. The authorization username is a string containing policy substitutions that describes how to construct the username for authorization lookups. This can either be an LDAP FQDN when the authorization realm is an LDAP realm, or a simple name when local realms are being used for authorization.

Note: Windows SSO realms never challenge for credentials. If the authorization username cannot be determined from the configured substitutions, authorization in the Windows SSO realm fails.

Keep in mind that Windows SSO realms do not require an authorization realm. If no authorization realm is configured, the user is not considered a member of any group. The effect this has on the user depends on the authorization policy. If the policy does not make any decisions based on groups, you do not need to specify an authorization realm. Also, if your policy is such that it works as desired when all Windows SSO realm users are not in any group, you do not have to specify an authorization realm.

Creating a Windows SSO Realm

The **Configuration > Authentication > Windows SSO > Windows SSO Realms** tab allows you to create a new Windows SSO realm.

To create a Windows SSO realm:

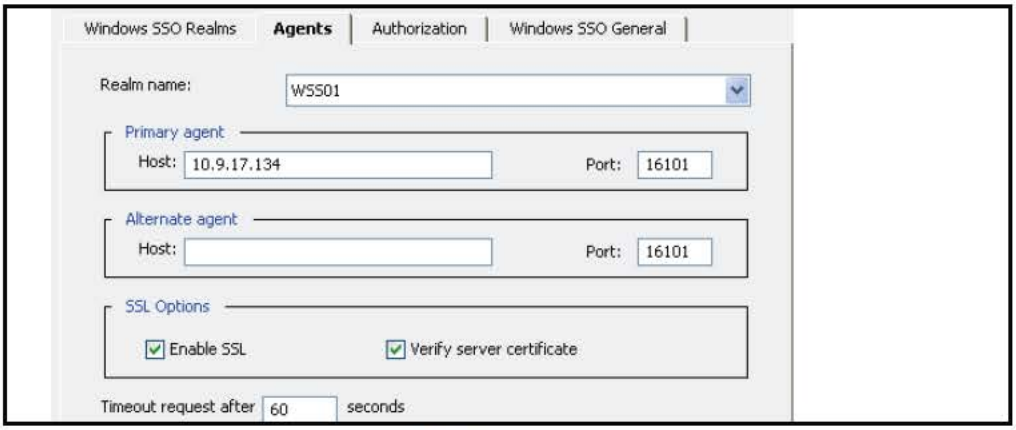
1. Select **Configuration > Authentication > Windows SSO > Windows SSO Realms**.
2. Click **New**.

3. In the **Realm name** field, enter a realm name. The name can be 32 characters long and composed of alphanumeric characters and underscores. The name *must* start with a letter.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

Windows SSO Agents

You must configure the Windows realm so that it can find the Blue Coat Authentication and Authorization Agent (BCAAA).

1. Select **Configuration > Authentication > Windows SSO > Agents**.



2. Select the realm name to edit from the drop-down list.

Note: You must have defined at least one Windows SSO realm (using the Windows SSO Realms tab) before attempting to configure the BCAAA agent. If the message **Realms must be added in the Windows SSO Realms tab before editing this tab** is displayed in red at the bottom of this page, you do not currently have any Windows SSO realms defined.

3. In the Primary agent section, enter the hostname or IP address where the BCAAA agent resides.
4. Change the port from the default of 16101 if necessary.
5. (Optional) Enter an alternate agent host and agent name in the **Alternate agent** section.
6. The primary and alternate BCAAA server must work together to support fail-over. If the primary BCAAA server fails, the alternate server should be able to provide the same mappings for the IP addresses.
7. (Optional) Click **Enable SSL** to enable SSL between the SG appliance and the BCAAA service.
8. (Optional) By default, if SSL is enabled, the Windows SSO BCAAA certificate is verified. To not verify the agent certificate, disable this setting.
9. In the **Timeout Request** field, type the number of seconds the SG appliance allows for each request attempt before timing out. (The default request timeout is **60** seconds.)
10. Select **Apply** to commit the changes to the SG appliance.

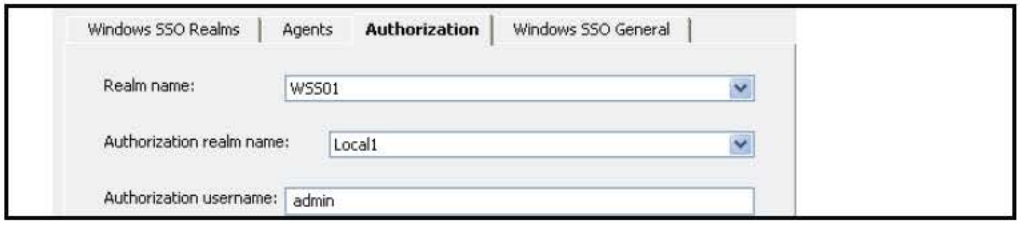
Configuring Authorization

After the Windows SSO realm is created, you can use the Windows SSO Authorization tab to configure authorization for the realm.

Note: Windows SSO realms do not require an authorization realm. If the policy does not make any decisions based on groups, you do not need to specify an authorization realm.

---

1. Select **Configuration > Authentication > Windows SSO > Authorization**.



2. From the **Realm name** drop-down list, select the Windows SSO realm for which you want to change realm properties.

Note: You must have defined at least one Windows SSO realm (using the Windows SSO Realms tab) before attempting to set Windows SSO realm properties. If the message **Realms must be added in the Windows SSO Realms tab before editing this tab** is displayed in red at the bottom of this page, you do not currently have any Policy Substitution realms defined.

---

3. (Optional) From the **Authorization realm name** drop-down list, select the realm you want to use to authorize users.

4. To construct usernames, keep in mind that the authorization username attributes is a string. that contains policy substitutions. When authorization is required for the transaction, the character string is processed by the policy substitution mechanism, using the current transaction as input. The resulting string becomes the user's authorization name for the current transaction.

Table 15-1. Common Substitutions Used in the Authorization username Field

ELFF Substitution	CPL Equivalent	Description
`x-cs-auth-domain`	`$(user.domain)`	The Windows domain of the authenticated user.
`cs-username`	`$(user.name)`	The relative username of the authenticated user.

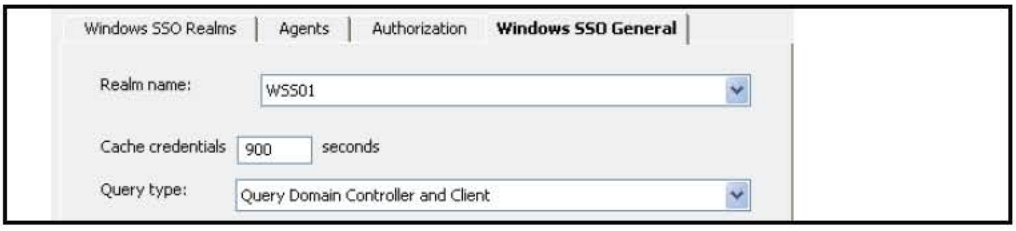
5. Select **Apply** to commit the changes to the SG appliance.

Defining Windows SSO Realm General Properties

The **Windows SSO General** tab allows you to specify the display name, the credential cache duration, and the type of authentication querying you want to do. After you select the type of authentication querying, you must modify the `sso.ini` file to enable the authentication querying. For information on modifying the `sso.ini` file, see "Modifying the Windows sso.ini File" on page 145.

To configure general settings:

1. Select **Configuration > Authentication > Windows SSO > Windows SSO General**.



2. From the **Realm Name** drop-down list, select the Windows SSO realm for which you want to change properties.

Note: You must have defined at least one Windows SSO realm (using the Windows SSO Realms tab) before attempting to set Windows SSO general properties. If the message **Realms must be added in the Windows SSO Realms tab before editing this tab** is displayed in red at the bottom of this page, you do not currently have any Windows SSO realms defined.

---

3. Specify the length of time, in seconds, that user and administrator credentials received from the Windows SSO BCAAA service are cached. Credentials can be cached for up to 3932100 seconds. The default cache duration is **900** seconds (15 minutes).

Note: If you specify **0**, traffic is increased to the Windows SSO BCAAA service and the authorization server (if configured) because each authentication request generates an authentication request to the BCAAA service and an authorization request to the authorization server.

4. In the **Query Type** field, select the method you want to use from the drop-down menu.

   By default the Windows SSO realm is configured for **Domain Controller Querying** only.

   If all of the client computers can be queried directly, then the most accurate results can be provided by the **Query Clients** option.

Note: Client Querying is blocked by the Windows XP SP2 firewall. This can be overridden through domain policy. If the firewall setting "Allow remote administration exception" or "Allow file and printer sharing exception" or "Define port exceptions" (with port 445) is enabled, the query will work.

   If an authentication mode without surrogates is being used (Proxy or Origin authenticate mode), then the **Query Domain Controller and Client** and **Query Client o**ptions can cause too much traffic when querying the clients, as each authentication request results in a request to the BCAAA service, which can result in a client workstation query depending on the client query time-to-live. If the client workstation querying traffic is a concern, the **Query Domain Controllers** option should be used instead.

5. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Manage a Windows SSO Realm

- To enter configuration mode:

```
SGOS#(config) security windows-sso create-realm realm_name
SGOS#(config) security windows-sso edit-realm realm_name
```

- The following subcommands are available:

```
SGOS#(config windows-sso realm_name) primary-agent {host hostname | port port_number}
SGOS#(config windows-sso realm_name) alternate-agent {host hostname | port port_number}
SGOS#(config windows-sso realm_name) ssl enable
SGOS#(config windows-sso realm_name) ssl-verify-agent enable
SGOS#(config windows-sso realm_name) authorization realm-name authorization-realm-name
SGOS#(config windows-sso realm_name) authorization username authorization-username
SGOS#(config windows-sso realm_name) cache-duration seconds
SGOS#(config windows-sso realm_name) sso-type {query-client | query-dc | query-dc-client}
```

Modifying the Windows sso.ini File

To enable the method of authentication querying you choose, you must modify the `sso.ini` file by adding domain controllers you want to query and user accounts you want to ignore.

The `sso.ini` file is located in **the** BCAAA installation directory.

If you are only using one method of querying, you only need configure the specific settings for that method. If you plan to use both methods to query, you must configure all the settings.

Note: The changes to the `sso.ini` file will have no effect until the BCAAA service is restarted.

To configure the sso.ini file for domain controller querying:

1. Open the file in a text editor.
2. In the section DCQSetup, uncomment the line: DCQEnabled=1.
3. In the section DCQDomainControllers, list the domain controllers you want to query or the IP address ranges of interest.
4. By default all domain controllers that are in the forest or are trusted are queried. In large organizations, domain controllers that are not of interest for the SG appliance installation might be queried. The `sso.ini` file can be used to list the domain controllers of interest or IP address ranges of interest.
5. In the section SSOServiceUsers, list the domain names of users who can access the domain controller on behalf of the service and mask the identity of the logged-on user.

   Listing these users here forces the BCAAA service to ignore them for authentication purposes.
6. Save the `sso.ini` file.

To configure the sso.ini file for client querying:

Note: Before you use the Windows SSO realm, you must change the BCAAA service to run as a domain user, and, if using XP clients, update the domain policy to allow the client query to pass through the firewall.

For information on installing and configuring the BCAAA service, see Appendix B: "Using the Authentication/Authorization Agent" on page 157.

---

1. Open the file in a text editor.
2. Review the TTL times in the section ClientQuerySetup to be sure they are appropriate for your network environment.
3. Update the section SSOServiceUsers to ignore domain users used for services.
4. Save the `sso.ini` file.

Creating the CPL

You can create CPL policies now that you have completed Windows SSO realm configuration. Be aware that the examples below are just part of a comprehensive authentication policy. By themselves, they are not adequate for your purposes.

The examples below assume the default policy condition is *allow*. On new systems, the default policy condition is *deny*.

Note: Refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide* for details about CPL and how transactions trigger the evaluation of policy file layers.

- Every Windows SSO-authenticated user is allowed access the SG appliance.

```
<Proxy>
   authenticate(WSSORealm)
```

- Group membership is the determining factor in granting access to the SG appliance.

```
<Proxy>
   authenticate(WSSORealm)
<Proxy>
  group=”cn=proxyusers, ou=groups, o=myco” ALLOW
deny
```

Notes

- The Windows SSO realm works reliably only in environments where one IP address maps to one user.
- This realm never uses a password.
- When doing domain controller querying, the Windows SSO realm can lose the logon if the NetBIOS computer name cannot by determined through a DNS query or a NetBIOS query. The DNS query can fail if the NetBIOS name is different than the DNS host name or if the computer is in a different DNS domain than the BCAAA computer and the BCAAA computer is not set up to impute different DNS domains.

  The NetBIOS query can fail because the NetBIOS broadcast does not reach the target computer. This can happen if the computer is behind a firewall that is not forwarding NetBIOS requests or if the computer is on a subnet that is not considered to be local to the BCAAA server.

  To prevent this issue, the BCAAA machine must be configured to be able to query the NetBIOS name of any computer of interest and get the correct IP address.

  One workaround is to use a WINS server. This works like a DNS server but handles NetBIOS lookups.

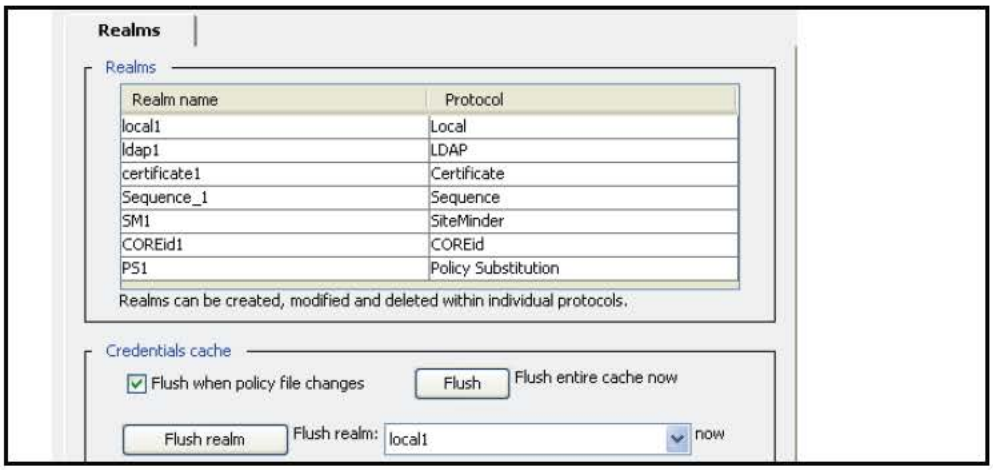
Chapter 16: Managing the Credential Cache

When you have configured all your realms, you can view your realms and manage the credentials cache for a specific realm.

Important: Credential caching is applicable only for authentication modes involving surrogates.

To manage the credential cache:

1. Select **Configuration > Authentication > Realms**.



2. To manage the credential cache:
   - To purge the credentials cache when you make policy changes, select **Flush When Policy File Changes** (this option is selected by default).
   - To flush the entire credentials cache immediately, click **Flush** and confirm.
   - To flush only the entries for a particular realm in the credentials cache, select the realm from the drop-down list, click **Flush Realm** confirm.

   All of these actions force users to be re-authenticated.
3. Select **Apply** to commit the changes to the SG appliance.

Related CLI Syntax to Manage the Credential Cache

```
SGOS#(config) security flush-credentials [on-policy-change {enable | disable} | realm realm]
```

Tips

- For all realms except IWA, SiteMinder, and COREid, the maximum number of entries stored in the credential cache is 80,000.

  For IWA, SiteMinder, and COREid authentication, the maximum number of entries stored in the credential cache is dependent on the system. You can have at least 2500 entries but potentially more depending on the system resources.

- XFTP users are not prompted for proxy authentication if the credentials are in the cache and the credentials have not expired.

Appendix A: Glossary

Term	Description
ADN Optimize Attribute	Controls whether to optimize bandwidth usage when connecting upstream using an ADN tunnel.
Asynchronous Adaptive Refresh (AAR)	This allows the Proxy*SG* to keep cached objects as fresh as possible, thus reducing response times. The AAR algorithm allows HTTP proxy to manage cached objects based on their rate of change and popularity: an object that changes frequently and/or is requested frequently is more eligible for asynchronous refresh compared to an object with a lower rate of change and/or popularity.
Asynchronous Refresh Activity	Refresh activity that does not wait for a request to occur, but that occurs *asynchronously* from the request.
Attributes (Service)	The service attributes define the parameters, such as explicit or transparent, cipher suite, and certificate verification, that the SG appliance uses for a particular service. .
Authenticate-401 Attribute	All transparent and explicit requests received on the port always use transparent authentication (cookie or IP, depending on the configuration). This is especially useful to force transparent proxy authentication in some proxy-chaining scenarios
authentication	The process of identifying a specific user.
authorization	The permissions given to a specific user.
Bandwidth Gain	A measure of the difference in client-side and server-side Internet traffic expressed in relation to server-side Internet traffic. It is managed in two ways: you can enable or disable bandwidth gain mode or you can select the Bandwidth Gain profile (this also enables bandwidth gain mode)..
Bandwidth Class	A defined unit of bandwidth allocation. An administrator uses bandwidth classes to allocate bandwidth to a particular type of traffic flowing through the SG appliance.
Bandwidth Class Hierarchy	Bandwidth classes can be grouped together in a class hierarchy, which is a tree structure that specifies the relationship among different classes. You create a hierarchy by creating at least one parent class and assigning other classes to be its children.
Bandwidth Policy	The set of rules that you define in the policy layer to identify and classify the traffic in the SG appliance, using the bandwidth classes that you create. You must use policy (through either VPM or CPL) in order to manage bandwidth.
Bypass Lists	The bypass list allows you to exempt IP addresses from being proxied by the SG appliance. The bypass list allows either <All> or a specific IP prefix entry for both the client and server columns. Both UDP and TCP traffic is automatically exempted.

Term	Description
Byte-Range Support	The ability of the Proxy*SG* to respond to byte-range requests (requests with a `Range:` HTTP header).
Cache-hit	An object that is in the Proxy*SG* and can be retrieved when an end user requests the information.
Cache-miss	An object that can be stored but has never been requested before; it was not in the Proxy*SG* to start, so it must be brought in and stored there as a side effect of processing the end-user's request. If the object is cacheable, it is stored and served the next time it is requested.
Child Class (Bandwidth Gain)	The child of a parent class is dependent upon that parent class for available bandwidth (they share the bandwidth in proportion to their minimum/maximum bandwidth values and priority levels). A child class with siblings (classes with the same parent class) shares bandwidth with those siblings in the same manner.
Client consent certificates	A certificate that indicates acceptance or denial of consent to decrypt an end user's HTTPS request.
Compression	An algorithm that reduces a file's size but does not lose any data. The ability to compress or decompress objects in the cache is based on policies you create. Compression can have a huge performance benefit, and it can be customized based on the needs of your environment: Whether CPU is more expensive (the default assumption), server-side bandwidth is more expensive, or whether client-side bandwidth is more expensive.
Default Proxy Listener	See " Proxy Service (Default)" .
Detect Protocol Attribute	Detects the protocol being used. Protocols that can be detected include: HTTP, P2P (eDonkey, BitTorrent, FastTrack, Gnutella), SSL, and Endpoint Mapper.
Directives	Directives are commands that can be used in installable lists to configure forwarding. See also *forwarding Configuration*.
Display Filter	The display filter is a drop-down list at the top of the Proxy Services pane that allows you to view the created proxy services by service name or action.
Early Intercept Attribute	Controls whether the proxy responds to client TCP connection requests before connecting to the upstream server. When early intercept is disabled, the proxy delays responding to the client until after it has attempted to contact the server.
Emulated Certificates	Certificates that are presented to the user by ProxySG when intercepting HTTPS requests. Blue Coat emulates the certificate from the server and signs it, copying the subjectName and expiration. The original certificate is used between the Proxy*SG* and the server.
ELFF-compatible format	A log type defined by the W3C that is general enough to be used with any protocol.
Encrypted Log	A log is encrypted using an external certificate associated with a private key. Encrypted logs can only be decrypted by someone with access to the private key. The private key is not accessible to the SG appliance.

Term	Description
explicit proxy	A configuration in which the browser is explicitly configured to communicate with the proxy server for access to content. This is the default for the SG appliance, and requires configuration for both browser and the interface card.
Fail Open/Closed	Failing open or closed applies to forwarding hosts and groups and SOCKS gateways. Fail Open/Closed applies when the health checks are showing sick for each forwarding or SOCKS gateway target in the applicable fail-over sequence. If no systems are healthy, the SG appliance fails open or closed, depending on the configuration. If closed, the connection attempt simply fails. If open, an attempt is made to connect without using any forwarding target (or SOCKS gateway). Fail open is usually a security risk; fail closed is the default if no setting is specified.
Forwarding Configuration	Forwarding can be configured through the CLI or through adding directives to a text file and installing it as an installable list. Each of these methods (the CLI or using directives) is equal. You cannot use the Management Console to configure forwarding.
Forwarding Host	Upstream Web servers or proxies.
forward proxy	A proxy server deployed close to the clients and used to access many servers. A forward proxy can be explicit or transparent.
Freshness	A percentage that reflects the objects in the Proxy*SG* cache that are expected to be fresh; that is, the content of those objects is expected to be identical to that on the OCS (origin content server).
Gateway	A device that serves as entrance and exit into a communications network.
Global Default Settings	You can configure settings for all forwarding hosts and groups. These are called the global defaults. You can also configure private settings for each individual forwarding host or group. Individual settings override the global defaults.
FTP	See Native FTP; Web FTP.
Host Affinity	Host affinity is the attempt to direct multiple connections by a single user to the same group member. Host affinity is closely tied to load balancing behavior; both should configured if load balancing is important.
Host Affinity Timeout	The host affinity timeout determines how long a user remains idle before the connection is closed. The timeout value checks the user's IP address, SSL ID, or cookie in the host affinity table.
Inbound Traffic (Bandwidth Gain)	Network packets flowing into the SG appliance. Inbound traffic mainly consists of the following: • Server inbound: Packets originating at the origin content server (OCS) and sent to the SG appliance to load a Web object. • Client inbound: Packets originating at the client and sent to the SG appliance for Web requests.

Term	Description
Installable Lists	Installable lists, comprised of directives, can be placed onto the SG appliance in one of several methods: through creating the list through the SG text editor, by placing the list at an accessible URL, or by downloading the directives file from the local system.
Integrated Host Timeout	An integrated host is an Origin Content Server (OCS) that has been added to the health check list. The host, added through the `integrate_new_hosts` property, ages out of the integrated host table after being idle for the specified time. The default is 60 minutes.
IP Reflection	Determines how the client IP address is presented to the origin server for explicitly proxied requests. All proxy services contain a reflect-ip attribute, which enables or disables sending of client's IP address instead of the SG's IP address.
Issuer keyring	The keyring that is used by the SG appliance to sign emulated certificates. The keyring is configured on the appliance and managed through policy.
Listener	The service that is listening on a specific port. A listener can be identified by any destination IP/subnet and port range. Multiple listeners can be added to each service.
Load Balancing	The ability to share traffic requests among multiple upstream targets. Two methods can be used to balance the load among systems: `least-connections` or `round-robin`.
Log Facility	A separate log that contains a single logical file and supports a single log format. It also contains the file's configuration and upload schedule information as well as other configurable information such as how often to rotate (switch to a new log) the logs at the destination, any passwords needed, and the point at which the facility can be uploaded.
Log Format	The type of log that is used: NCSA/Common, SQUID, ELFF, SurfControl, or Websense. The proprietary log types each have a corresponding pre-defined log format that has been set up to produce exactly that type of log (these logs cannot be edited). In addition, a number of other ELFF type log formats are also pre-defined (im, main, p2p, ssl, streaming). These can be edited, but they start out with a useful set of log fields for logging particular protocols understood by the SG appliance. It is also possible to create new log formats of type ELFF or Custom which can contain any desired combination of log fields.
Log Tail:	The access log tail shows the log entries as they get logged. With high traffic on the SG appliance, not all access log entries are necessarily displayed. However, you can view all access log information after uploading the log.
Maximum Object Size	The maximum object size stored in the Proxy*SG*. All objects retrieved that are greater than the maximum size are delivered to the client but are not stored in the Proxy*SG*.
NCSA common log format	A log type that contains only basic HTTP access information.

Term	Description
Negative Responses	An error response received from the OCS when a page or image is requested. If the Proxy*SG* is configured to cache such negative responses, it returns that response in subsequent requests for that page or image for the specified number of minutes. If it is not configured, which is the default, the Proxy*SG* attempts to retrieve the page or image every time it is requested.
Native FTP	Native FTP involves the client connecting (either explicitly or transparently) using the FTP protocol; the SG appliance then connects upstream through FTP (if necessary).
Outbound Traffic (Bandwidth Gain)	Network packets flowing out of the SG appliance. Outbound traffic mainly consists of the following: • Client outbound: Packets sent to the client in response to a Web request. • Server outbound: Packets sent to an OCS or upstream proxy to request a service.
Origin Content Server (OCS)	
Parent Class (Bandwidth Gain)	A class with at least one child. The parent class must share its bandwidth with its child classes in proportion to the minimum/maximum bandwidth values or priority levels.
PASV	Passive Mode Data Connections. Data connections initiated by an FTP client to an FTP server.
proxy	Caches content, filters traffic, monitors Internet and intranet resource usage, blocks specific Internet and intranet resources for individuals or groups, and enhances the quality of Internet or intranet user experiences. A proxy can also serve as an intermediary between a Web client and a Web server and can require authentication to allow identity based policy and logging for the client. The rules used to authenticate a client are based on the policies you create on the SG appliance, which can reference an existing security infrastructure—LDAP, RADIUS, IWA, and the like.
Proxy Service	The proxy service defines the ports, as well as other attributes. that are used by the proxies associated with the service.
Proxy Service (Default)	The default proxy service is a service that intercepts all traffic not otherwise intercepted by other listeners. It only has one listener whose action can be set to bypass or intercept. No new listeners can be added to the default proxy service, and the default listener and service cannot be deleted. Service attributes can be changed.
realms	A realm is a named collection of information about users and groups. The name is referenced in policy to control authentication and authorization of users for access to Blue Coat Systems SG services. Multiple authentication realms can be used on a single SG appliance. Realm services include IWA, LDAP, Local, and RADIUS.
Reflect Client IP Attribute	Enables the sending of the client's IP address instead of the SG's IP address to the upstream server. If you are using an Application Delivery Network (ADN), this setting is enforced on the concentrator proxy through the **Configuration>App. Delivery Network>Tunneling** tab.

Term	Description
Refresh Bandwidth	The amount of bandwidth used to keep stored objects fresh. By default, the Proxy*SG* is set to manage refresh bandwidth automatically. You can configure refresh bandwidth yourself, although Blue Coat does not recommend this.
reverse proxy	A proxy that acts as a front-end to a small number of pre-defined servers, typically to improve performance. Many clients can use it to access the small number of predefined servers.
rotate logs	When you rotate a log, the old log is no longer appended to the existing log, and a new log is created. All the facility information (headers for passwords, access log type, and so forth), is re-sent at the beginning of the new upload. If you're using Reporter (or anything that doesn't understand the concept of "file," such as streaming) the upload connection is broken and then re-started, and, again, the headers are re-sent.
serial console	A device that allows you to connect to the SG appliance when it is otherwise unreachable, without using the network. It can be used to administer the SG appliance through the CLI. You must use the CLI to use a serial console. Anyone with access to the serial console can change the administrative access controls, so physical security of the serial console is critical.
Server Certificate Categories	The hostname in a server certificate can be categorized by BCWF or another content filtering vendor to fit into categories such as banking, finance, sports.
Sibling Class (Bandwidth Gain)	A bandwidth class with the same parent class as another class.
SOCKS Proxy	A generic way to proxy TCP and UDP protocols. The SG appliance supports both SOCKSv4/4a and SOCKSv5; however, because of increased username and password authentication capabilities and compression support, Blue Coat recommends that you use SOCKS v5..
SmartReporter log type	A proprietary ELFF log type that is compatible with the SmartFilter SmartReporter tool.
Split proxy	Employs co-operative processing at the branch and the core to implement functionality that is not possible in a standalone proxy. Examples of split proxies include : Mapi Proxy SSL Proxy
SQUID-compatible format	A log type that was designed for cache statistics.
SSL	A standard protocol for secure communication over the network. Blue Coat recommends using this protocol to protect sensitive information.
SSL Interception	Decrypting SSL connections.
SSL Proxy	A proxy that can be used for any SSL traffic (HTTPS or not), in either forward or reverse proxy mode.

Term	Description
static routes	A manually-configured route that specifies the transmission path a packet must follow, based on the packet's destination address. A static route specifies a transmission path to another network.
SurfControl log type	A proprietary log type that is compatible with the SurfControl reporter tool. The SurfControl log format includes fully-qualified usernames when an NTLM realm provides authentication. The simple name is used for all other realm types.
Traffic Flow (Bandwidth Gain)	Also referred to as *flow*. A set of packets belonging to the same TCP/UDP connection that terminate at, originate at, or flow through the SG appliance. A single request from a client involves two separate connections. One of them is from the client to the SG appliance, and the other is from the SG appliance to the OCS. Within each of these connections, traffic flows in two directions—in one direction, packets flow out of the SG appliance (outbound traffic), and in the other direction, packets flow into the SG (inbound traffic). Connections can come from the client or the server. Thus, traffic can be classified into one of four types: • Server inbound • Server outbound • Client inbound • Client outbound These four traffic flows represent each of the four combinations described above. Each flow represents a single direction from a single connection.
transparent proxy	A configuration in which traffic is redirected to the SG appliance without the knowledge of the client browser. No configuration is required on the browser, but network configuration, such as an L4 switch or a WCCP-compliant router, is required.
Variants	Objects that are stored in the cache in various forms: the original form, fetched from the OCS; the transformed (compressed or uncompressed) form (if compression is used). If a required compression variant is not available, then one might be created upon a cache-hit. (Note: policy-based content transformations are not stored in the Proxy*SG*.)
Web FTP	Web FTP is used when a client connects in explicit mode using HTTP and accesses an ftp:// URL. The SG appliance translates the HTTP request into an FTP request for the OCS (if the content is not already cached), and then translates the FTP response with the file contents into an HTTP response for the client.
Websense log type	A proprietary log type that is compatible with the Websense reporter tool.

Term	Description
Wildcard Services	When multiple non-wildcard services are created on a port, all of them must be of the same service type (a wildcard service is one that is listening for that port on all IP addresses). If you have multiple IP addresses and you specify IP addresses for a port service, you cannot specify a different protocol if you define the same port on another IP address. For example, if you define HTTP port 80 on one IP address, you can only use the HTTP protocol on port 80 for other IP addresses. Also note that wildcard services and non-wildcard services cannot both exist at the same time on a given port. For all service types except HTTPS, a specific listener cannot be posted on a port if the same port has a wildcard listener of any service type already present.

Appendix B: Using the Authentication/Authorization Agent

The Blue Coat Systems Authentication and Authorization Agent (BCAAA) allows SGOS 4.x to manage authentication and authorization for IWA, Netegrity SiteMinder realms, and Oracle COREid realms. The agent is installed and configured separately from SGOS 5.x and is available at the Blue Coat Website.

The BCAAA service must be installed on a domain controller or member server, allowing the SG to access domain controllers. The BCAAA service authenticates users in all domains trusted by the computer on which it is running. A single installation of the BCAAA service can support multiple appliances.

Multiple versions of BCAAA can run on the same machine. This allows you to use the same machine to support versions of the SG that have different BCAAA version requirements.

The BCAAA install directory can include multiple executable programs.

- The program `bcaaa.exe` (`bcaaa` on Solaris) handles connections from SG appliances and hands them off to the correct version of the processor.
- The program `bcaaa-99.exe` (`bcaaa-99` on Solaris) handles communication with versions of the SG prior to SGOS 4.2.
- The program `bcaaa-100.exe` (`bcaaa-100` on Solaris) handles communication with SGOS 4.2 and higher.

When a new version of BCAAA is installed in the same installation directory as earlier versions, the earlier versions are not removed.

This allows SG appliances that were communicating with the old version to continue to operate.

Using the BCAAA Service

Several realms use the BCAAA service:

- IWA: The BCAAA service talks directly to an Integrated Windows Authentication (IWA) or NTLM server. When using IWA, the network typically chooses automatically whether to use NTLM or Kerberos (IWA).
  - NTLM: NTLM is a subset of IWA, meant to be used with Windows NT systems.
  - IWA: If using Kerberos, the BCAAA service must share a secret with a Kerberos server (called a KDC) and register an appropriate Service Principal Name (SPN). For information on sharing a secret and registering an SPN, see "Creating Service Principal Names for IWA Realms" on page 164.
  - SiteMinder and COREid: When a SiteMinder or COREid realm is referenced in policy, a BCAAA process is created. The SG then sends a configuration request that describes the servers to use. The BCAAA service logs in to the appropriate servers and determines configuration information to be passed back to the SG (such as the kind of credentials required). Responses from the SiteMinder and COREid policy servers are translated into appropriate BCAAA protocol responses and returned to the SG.

Before you can use the BCAAA service with SiteMinder or COREid, you must configure the appropriate SG realm to work with the SiteMinder or COREid servers. The realm can be configured from the SiteMinder or COREid configuration tabs in the Management Console or from the CLI.

Note: Each (active) SiteMinder realm on the SG should reference a different agent on the Policy Server.

For specific information about configuring the SiteMinder realm to work with the Netegrity policy servers, see Chapter 11: "Netegrity SiteMinder Authentication" on page 113. For specific information about configuring the COREid realm to work with Oracle COREid Access Servers, see Chapter 6: "Oracle COREid Authentication" on page 65.

- Windows Single Sign-on (SSO): The BCAAA service is used to supply mappings for IP addresses to logged on users. The Windows SSO realm can use domain controller querying, or client querying, or both domain controller and client querying to determine the logged-on user.

Note: To use domain controller querying, you must configure the `sso.ini` file to enable it and to add the domain controllers you want to query. For information on configuring the `sso.ini` file, see "Modifying the Windows sso.ini File" on page 145.

Performance Notes

Blue Coat recommends that the Windows BCAAA service be installed on a dedicated Windows machine. Installation of any other non-essential software might degrade the BCAAA service performance, which in turn degrades the user experience.

This is because the BCAAA server is in the client data path for accessing protected resources. Users make client requests to the SG appliance, which in turn proxies authentication requests to the BCAAA. service. The user must wait for the authentication request to complete before the SG appliance responds to the user with a protected resource.

Operating system requirements are:

- IWA and COREid: Windows® 2000 or later.
- SiteMinder: Windows 2000 or later or Solaris™ 5.8 or 5.9.

The appendix discusses:

- "Installing the BCAAA Service on a Windows System"
- "Installing the BCAAA Service on a Solaris System" on page 164
- "Creating Service Principal Names for IWA Realms" on page 164
- "Troubleshooting Authentication Agent Problems" on page 166
- "Common BCAAA Event Messages" on page 166

Installing the BCAAA Service on a Windows System

All images in this section are from a Windows 2000 system.

Note: If you have an existing CAASNT service on your system, it is stopped and deleted as part of the BCAAA installation procedure.

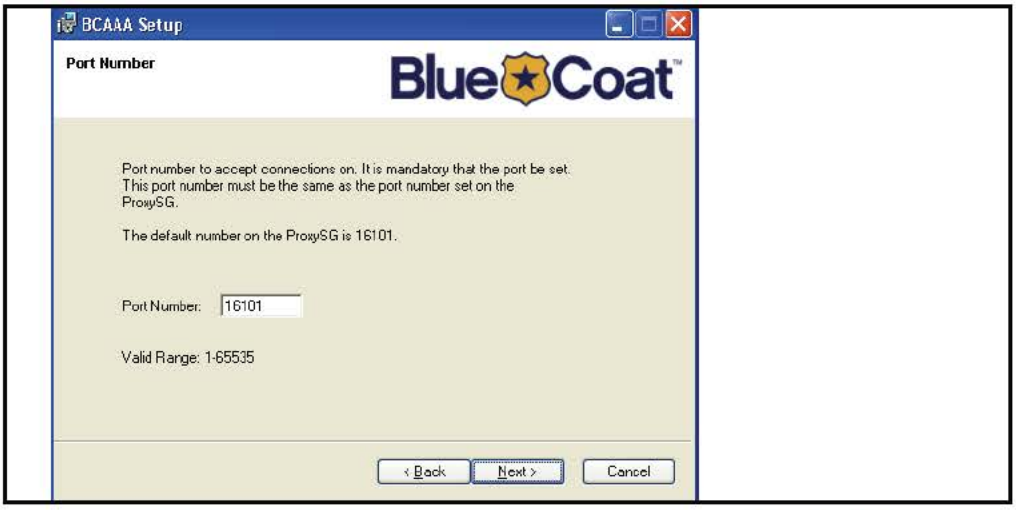
To install the authentication agent:

1. Download the file from the Blue Coat download site at https://download.bluecoat.com/
2. Launch the install wizard.
3. Click **Next** to select the destination folder.

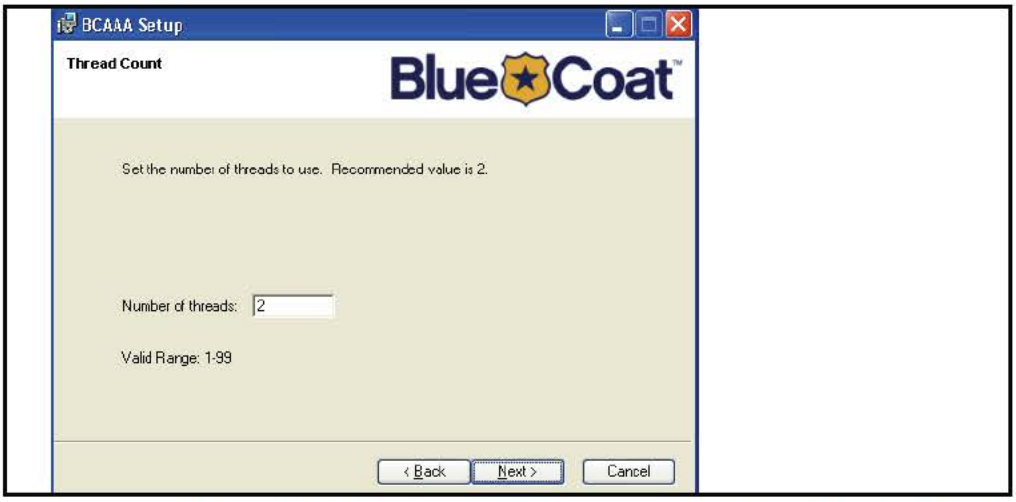
Note: When doing an upgrade from one version of BCAAA to another version of BCAAA, you must install into the previous BCAAA folder to retain your settings. If you install to a different folder, a new `.ini` file with default settings is created.

When upgrading from CAASNT to BCAAA, the settings from CAASNT are copied to the new installation directory.

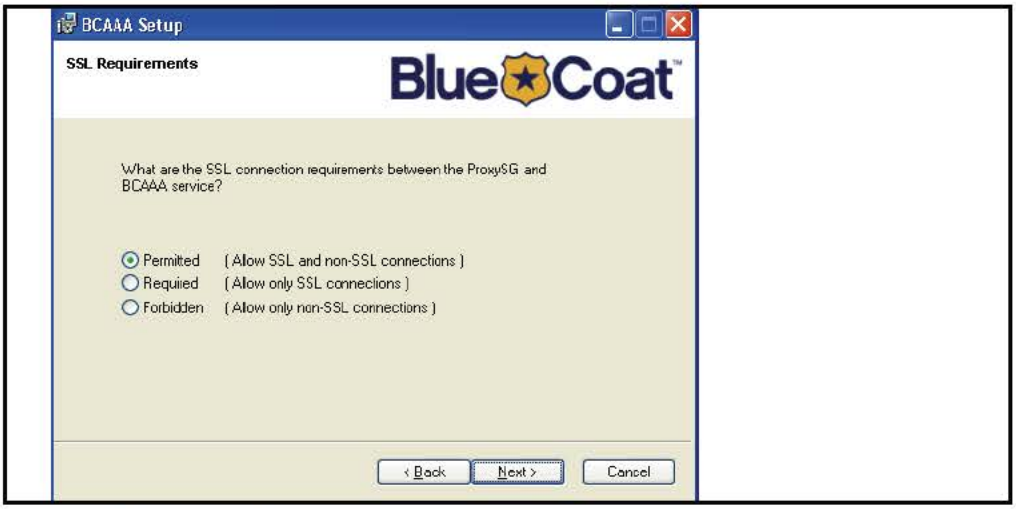
4. Click **Browse** to select a different destination folder for the BCAAA service.
5. Click **Next** to accept the default and select the port number.



6. The port number must match the port number you specify on the SG for the BCAAA service. The default is 16101.
7. Click **Next** to select the number of threads.



8. The recommended (and default) value is 2. The maximum number of threads allowed is 99 per SG. After selecting the number, click **Next** to specify the SSL requirements.



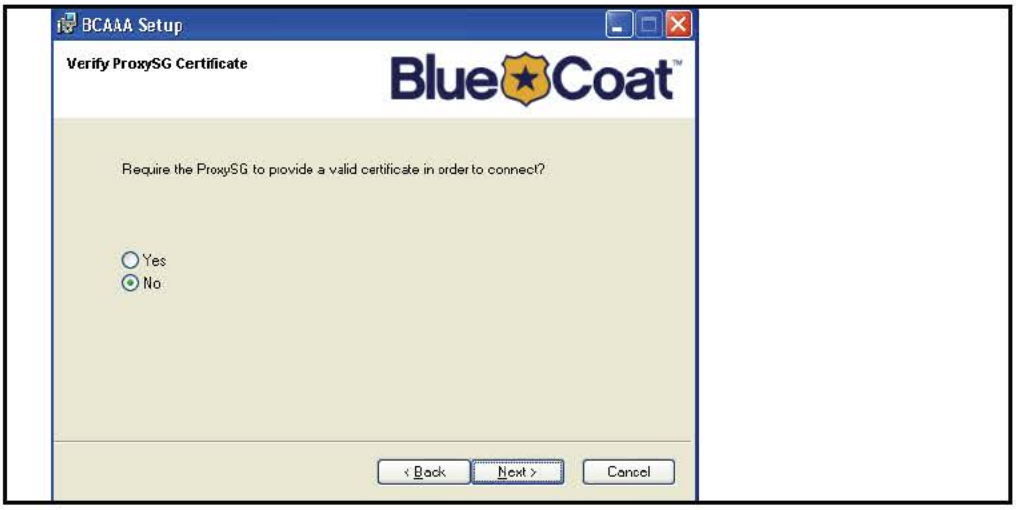
9. The default is that SSL is Permitted, allowing both SSL and non-SSL connections. This setting must be compatible with the setting on the SG appliance.
10. Click **Next** to specify the subject of the SSL certificate.

11. Specify the subject of the certificate.

    The BCAAA service looks up the specified subject in the service's certificate store. If it finds the subject, it uses it instead of generating a new certificate. If not, it generates a self-signed certificate with that subject. This generated certificate can be saved (as specified on the next screen).

12. Click **Next** to specify save options for the certificate.

13. Click **Next** to specify whether the SG appliance must provide a valid certificate when connecting to the BCAAA service.



14. To force the SG to provide a valid certificate to connect to the BCAAA service, select the **Yes** radio button. The default is **No**.
15. Click **Next** to view the summary of the changes you made.
16. Click **Install** to install the BCAAA service using the settings you configured.

    When installation completes, the final BCAAA screen displays.

To modify settings or uninstall the authentication agent:

1. Launch the install wizard.

2. Click **Modify** to re-enter the installation wizard; click **Remove** to uninstall the BCAAA service from the system.

Note: For instructions on using the installation wizard, see "Installing the BCAAA Service on a Windows System" on page 158.

3. Click **Next** to start the procedure.
4. Click **Finish** to exit the uninstall application.

To view the application event log

The BCAAA service logs all errors to the Windows 2000 Application Event Log under the name BCAAA.

1. Launch the Event Log.
2. Doubleclick the information message BCAAA service to see that the BCAAA service has been automatically started.

To view the BCAAA service

The BCAAA service logs all errors to the Windows 2000 Application Event Log under the name BCAAA.

1. Launch the Event Viewer.
2. Right-click on **BCAAA** and select **Properties** to manage the service. For example, to make the BCAAA service start only manually, set the **Startup Type** to **Manual**. (**Automatic** is the default setting.)

Completing Setup for the BCAAA Service

Once the BCAAA service is installed, you must complete BCAAA setup by configuring the service to work with Windows.

To configure the BCAAA service:

1. Open the properties panel for the BCAAA service
   a. Select the **Log-on** tab.
   b. Change the account to the one you created for the BCAAA service, and enter the password.
   c. Click **OK**. You might be warned that the account has been given **logon as service** privileges.
2. Verify in Local Security Policy's **User Rights Assignment** folder that the BCAAA Service user account has been added to the list of the **Log on as a service** policy.

Note: You must have modify/write privileges in the BCAAA folder.

---

3. (Optional) If group-based authorization is being done, then:
   - Ensure that the user impersonation privilege is set for the SERVICE group. For more information setting the user impersonation privilege, see: http://support.microsoft.com/default.aspx?scid=kb;en-us;831218.
   - Ensure that the Active Directory computer account running the BCAAA service has the **Trust computer for delegation** configuration property enabled.
4. (Optional) For all users authenticating to the SG using IWA realms, user accounts in the Active Directory must have permission to log onto the machine where the BCAAA server is running.
5. Go to the user's account properties user account tab.
6. Click the **Log On To...** button to specify the domain that computers can log onto. If the network environment restricts users to specific computers, then each user must have the name of the host running the BCAAA service added to their list.

Installing the BCAAA Service on a Solaris System

To install the BCAAA service on Solaris, complete the following instructions. You must be `root` to complete installation.

Note: For successful installation of the BCAAA service on a Solaris system, you will need `libstdc++.so.5`", usually installed with `package SFWgcc32` gcc-3.2 - GNU Compiler Collection Version 3.2

1. Download the shell script to your system.
2. Execute the shell script:

   ```
   # sh bcaaa-version_number-SOLARIS-install.sh
   ```

3. Answer the questions to install the service on your Solaris system. A sample session is shown below:

   ```
   Enter a path to a scratch directory [/tmp]:
   Install Blue Coat Systems Authentication and Authorization Agent
   (BCAAA)? (y/n)y
   Enter user that should own the installed files [root]
   Enter group for the installed files [root]
   /usr/local/bin/bcaaa installed
   /usr/local/bin/bcaaa-100 installed
   Libraries installed in /usr/local/lib/BlueCoatSystems/
   /usr/local/etc/bcaaa.ini installed
   If you use inetd, append the following line to /etc/services
   bcaaa           16101/tcp                #Blue Coat Systems
   Authentication Agent
   If you use inetd, append the following line to /etc/inetd.conf, then
   signal inetd to re-read the configuration file
   If you use something else, make the equivalent changes
   bcaaa  stream  tcp  nowait  root /usr/local/bin/bcaaa bcaaa -c  /usr/
   local/etc/bcaaa.ini
   Installation complete
   ```

Creating Service Principal Names for IWA Realms

For the BCAAA service to participate in an IWA Kerberos authentication exchange, it must share a secret with the Kerberos server (called a KDC) and have registered an appropriate Service Principal Name (SPN).

You can share the secret two ways:

- LocalSystem

  In this approach the SPN is registered with the NetBIOS name of the machine on which BCAAA is running. BCAAA runs under LocalSystem (the default for services), and uses the machine's shared secret.

  The primary advantage of this approach is convenience: it works with the default settings for service installation. The disadvantage is that only one BCAAA server is allowed for the realm, so you cannot have a backup server.

  Note: Handling of the shared secret is done by Windows when the machine joins the domain; there is no explicit knowledge of the shared secret by SGOS or by BCAAA.

- Service Account

  You can also create a service account for the BCAAA service and register the SPN on the service account. This allows multiple servers to run BCAAA all using the same account.

  The advantage is the ability to have a backup BCAAA server. The disadvantage is that it requires additional configuration on the Active Directory server, the domain controller, and on each BCAAA machine. It is also less secure, since the BCAAA account password is shared among multiple machines.

To share a secret by creating a service account:

Note: All steps require administrator privileges.

1. Go to the Active Directory server.
2. Create an account for use by the BCAAA service.
3. Create a password.
4. On the domain controller, open the domain policy console and modify the Local Policy's user rights assignment and allow the account you created in on the Active Directory to have the right to "act as the operating system."
5. Run the following command:

   `setspn -A HTTP/FQDN-of-host` *`name`*

   where *name* is the name of the account created in step 1 and the FQDN is the virtual URL that was set in the authentication realm. For example:

   ```
   setspn -A HTTP/krbproxy.authteam.waterloo.bluecoat.com authteam\krb-bcaaa
   ```

   Note: The `setspn` application might have to be downloaded from Microsoft. It is installed by default in `program files\resource kit`.

(Optional) To create a group account (a BCAAA user account capable of doing group-based authorization):

If group-based authorization is being done, then:

1. Ensure that the user impersonation privilege is set for the SERVICE group.

   Note: For information on setting the user impersonation privilege, see

   http://support.microsoft.com/default.aspx?scid=kb;en-us;831218

2. Ensure that the Active Directory computer account running the BCAAA service has the "Trust computer for delegation" configuration property enabled.

On each machine where you want to run the BCAAA service:

1. Install the BCAAA service as normal.
2. Open the Properties panel for the BCAAA service and select the Logon tab. Change the account to the one you created on the Active Directory server, and enter the password. When you click OK, it might warn you that the account has been granted "Log On as A Service right".

3. Change the security on the BCAAA install directory to give the account created on the Active Directory server full control.

All these machines now share the same secret with the KDC and can decrypt service tickets intended for the service described by the SPN.

Troubleshooting Authentication Agent Problems

This section describes some common problems you might encounter when setting up or using the BCAAA service on a Windows platform.

To troubleshoot the BCAAA service, launch the event viewer.

The Properties pane displays, providing information about the status of the BCAAA service at that time. Note the Type and the Event ID. The description below the Type/Event ID lists the problem. You can often find more information about the problem and suggestions for its solution in "Common BCAAA Event Messages" on page 166.

Common problems:

- If an attempt to start the BCAAA service is issued when BCAAA is already started, the following error message displays:

  The requested service has already been started.

- If another application is using the same port number as the BCAAA service, the following messages are displayed:

  The BCAAA service could not be started.

  A system error has occurred.

  System error 10048 has occurred.

  Only one usage of each socket address (protocol/network address/port) is normally permitted.

Common BCAAA Event Messages

Following are the most common event messages that can be logged to the Windows 2000 Application Event Log. Most of the event messages not listed here are error status messages returned by Win32 function calls. When a Win32 call fails, the error code and error text containing the reason for the error displays in the event log under the name BCAAA.

To view the BCAAA event log:

1. Right click on **My Computer** and select **Manage**.
2. Select **System Tools > Event Viewer > Application**.

For each BCAAA event message, the event message is displayed along with the event number.

Table B-1. BCAA Event Messages

Message ID	Message	Description
200	Various messages	The associated message provides information about a condition that is not an error.
300	Various messages	The associated message warns about an unexpected condition that does not prevent operation.

Table B-1. BCAA Event Messages (Continued)

Message ID	Message	Description
400	Various messages	The associated message describes an error condition that prevents normal operation.
1001	Authentication Agent service started: port=# threads=# socket=0x# process id=# agent version=# Proxy*SG* Appliance version=#	This indicates successful startup and provides information about the agent.
1002	Authentication Agent stopped	This indicates normal shutdown of the service.
1003	Proxy*SG* Appliance (a.b.c.d) connected; Process # spawned as #	This indicates a Proxy*SG* has connected to the agent (Windows only).
1004	Proxy*SG* Appliance agent process exited (normal logout)	This indicates normal logout by a Proxy*SG*.
1005	Process %d has terminated, ExitCode=0x#, link=0x#	This indicates an unexpected termination of an agent process (Windows only).
1006	Service dispatcher exited.	This indicates an unexpected termination of the service dispatcher.
1007	CreateNamedPipe failed, pipe='%s'	The agent dispatcher could not create the named pipe for the reason given.
1008	ConnectNamedPipe failed, pipe='%s'	The agent process could not obtain the information from the dispatcher on the named pipe for the reason given.
1009	WriteFile failed, pipe='%s'	The dispatcher could not write information to the named pipe for the reason given.
1011	CreateThread (ProcessTimerThread) failed	The dispatcher could not create its timer thread.
1012	Failed to create Proxy*SG* Appliance process '%s'	The BCAAA server does not have the same version of BCAAA available as the SG is expecting.
1019	Various	The dispatcher was unable to determine the exit status of an agent process.
1020	Terminating Proxy*SG* Appliance process #, ProcNum=# Handle=0x#	An agent process was active when the Windows service was shut down.
1022	Various	The associated message reports the status of a Proxy*SG* login attempt.
1101	BasicAuth: CloseHandle failed; user 'xx\\xx'	The agent was unable to close the login handle for the specified user.
1102	Username: '%s\\%s' too long	The Proxy*SG* offered the specified username, which is too long.
1106	Various	An attempted authentication using BASIC credentials failed for the reason given.
1107	User Right 'Act as part of the operating system' required for Basic Authentication	The agent does not have the necessary privileges to do BASIC authentication

Table B-1. BCAA Event Messages (Continued)

Message ID	Message	Description
1108	Various	The agent was unable to determine information about the user for the reason given.
1202	Unable to create GroupsOfInterest mutex 'xx' - already exists	The agent could not create the Windows mutex needed for group authorization checks because it already exists.
1203	Unable to create GroupsOfInterest mutex 'xx	The agent could not create the Windows mutex needed for group authorization checks.
1204	OpenMutex failed for AuthGroups mutex '%s', group='%s'	The agent was unable to open the Windows mutex needed for group authorization checks.
1205	Various	The agent was unable to close the Windows mutex named for the reason given.
1207	GetAclInformation failed	The agent was unable to obtain ACL information needed to do group authorization checks.
1209	GetKernelObjectSecurity failed for AuthGroup='%s'	The agent was unable to obtain security information about the specified group.
1210	SetKernelObjectSecurity failed	The agent was unable to set up security information for the reason specified.
1211	InitializeSecurityDescriptor failed	The agent was unable to initialize the security descriptor for the reason specified.
1212	GetSecurityDescriptorDacl failed	The agent was unable to get the discretionary access control list (DACL) for the reason specified.
1213	SetSecurityDescriptorDacl failed	The agent was unable to set the discretionary access control list (DACL) for the reason specified.
1214	InitializeAcl failed	The agent was unable to initialize the access control list (ACL) for the reason specified.
1215	GetUserName failed for AuthGroup='%s'	The agent was unable to determine the username while processing the specified group.
1217	GetAce failed for AuthGroup='%s'	The agent was unable to get the access control entry (ACE) for the specified group.
1218	AddAce failed	The agent was unable to add the necessary access control entry (ACE) for the reason specified.
1219	AddAccessAllowedAce failed	The agent was unable to add the necessary "access allowed" access control entry (ACE).
1220	Could not establish groups-of-interest: result=0x##	The agent was unable to initialize groups-of-interest checking.
1221	AuthGroup '%s' does not exist	The specified group does not exist.
1222	IWA RevertSecurityContext failed, user='%s'	The agent could not revert the security context for the specified user.
1223	BASIC: RevertToSelf failed, user='%s'	The agent could not revert the security context for the specified user.

Table B-1. BCAA Event Messages (Continued)

Message ID	Message	Description
1224	Error calling OpenProcessToken	The agent's call to OpenProcessToken failed for the specified reason.
1225	Error calling LookupPrivilegeValue	The agent could not get information about a needed privilege.
1226	Error calling AdjustTokenPrivileges	The agent could not adjust its privileges as required.
1227	ImpersonateLoggedOnUser failed; Group access denied for user '%s'	The agent could not impersonate the specified user.
1228	IWA: ImpersonateSecurityContext failed; Group access denied for user '%s'	The agent could not impersonate the specified user.
1301	NOTE: Pending ContextLink=### timed out; deleting SecurityContext h=## TS=## now=##	The Proxy*SG* did not provide a response to a challenge quickly enough.
1302	Various	An authentication request from a Proxy*SG* referenced an in-progress request that has timed out or does not exist.
1304	Various	The agent was unable to delete a security context for the reason given.
1305	AcceptSecurityContext failure, SEC_E_INVALID_HANDLE, ContextLink=### count=#	The agent was provided with an invalid context handle.
1306	Various	The client provided an invalid token to the authentication system.
1308	AcceptSecurityContext failure, ContextLink=# count=#, detail=#(xxx)	Windows rejected the authentication attempt for the reason given.
1310	Various	This records the failure of NTLM authentication or group authorization.
1311	3:Failed NTLM Authentication for user: '%s'	This records the failure of NTLM authentication; the user name was supplied by the client.
1312	Various	The agent could not determine the username from the NTLM type 3 message supplied by the client.
1313	Invalid Type3 message	The client provided an NTLM type 3 message that was invalid.
1314	BASE64_Decode: Length of token exceeds max (%d)	The client provided an NTLM token that was too long.
1316	Unsupported version in request: %d(0x%x)	The Proxy*SG* sent a request with an unsupported version number.
1401	Various	The agent lost communication with the Proxy*SG*.
1403	Various	The agent is aborting for the reason given.
1402	Unexpected thread 0 exit	The agent exited unexpectedly.
1404	Unable to get ProcessInfo from parent process.	The agent could not obtain its information from the dispatcher.

Table B-1. BCAA Event Messages (Continued)

Message ID	Message	Description
1405	CreateFile failed, pipe='xx'	The agent could not create a handle for the dispatcher's named pipe.
1406	WaitNamedPipe failed, pipe='%s'	The agent could not wait for the dispatcher's named pipe.
1407	ReadFile failed, pipe='%s'	The agent could not read information from the dispatcher's named pipe.
1409	Various	The agent could not create the specified thread for the reason given.
1412	Various	The agent could not create a required Windows event object.
1413	AuthMethod 'xxs' not supported: returning _AuthResult=0x##	The Proxy*SG* requested an unsupported authentication mechanism.
1414	Various	The specified request is unsupported.
1500	Various	The agent has a problem with memory allocation; typically this means there is not enough memory.
1501	Unable to allocate memory for ProcLink buffer.	The agent could not allocate some needed memory.
1502	Unable to allocate memory for ContextLink buffer.	The agent could not allocate some needed memory.
1503	Various	The agent was unable to allocate needed memory.
1604	Service dispatch failed	The Windows service dispatcher failed to start.
1605	RegisterServiceCtrlHandler failed	The agent dispatcher was unable to register the service control handler.
1608	SetServiceStatus failed, g_StatusHandle=%d	The agent was unable to set the service's status.
1610	Unsupported service control code: #	Windows sent a service control code that the agent does not support.
1701	WSASocket failed	The agent could not create a Windows socket for the reason given.
1702	WSAStartup failed.	The agent could not start the Windows socket for the reason given.
1703	Various	The agent could not send data to the Proxy*SG* for the reason given.
1704	Various	The agent could not receive data from the Proxy*SG* for the reason given.
1705	accept failed	The agent dispatcher could not initialize to accept new connections.
1706	bind failed, PortNumber=#	The agent dispatcher could not bind to the specified port.
1707	listen failed.	The agent dispatcher could not listen for new connections.

Table B-1. BCAA Event Messages (Continued)

Message ID	Message	Description
1708	Various	Windows reported an event wait failure to the agent while doing I/O on the socket.
1709	The agent is already running or the agent's port # is in use by another process	Some other process is already using the port needed by the agent.
1710	WSARecv failed reading bytes from socket	Windows reported an error when the agent tried to receive bytes from the Proxy*SG*.
1711	WSASend failed sending bytes to socket.	Windows reported an error when the agent tried to send bytes to the Proxy*SG*.
1712	Various	A socket I/O operation did not complete successfully.
1801	Error calling AcquireCredentialsHandle	The agent could not acquire its credentials from Windows.
1803	Various	The agent could not load a needed library (DLL).
1804	Various	The agent could not locate the needed services in a library (DLL).
1805	Unsupported SSPI Windows platform; PlatformId=#	The reported Windows platform is not supported for NTLM authentication.
1806	Error calling QueryContextAttributes	The agent could not determine the authenticated user's security attributes.
1807	QuerySecurityPackageInfo failed	The agent could not get needed security information from Windows.
1808	Max Token size too long (#); max size is #	The client supplied an NTLM token that is too long.
1809	FreeContextBuffer failed	An attempt to free the NTLM context buffer failed.
1811	Username 'x\\y' too long	The reported user name is too long.
1901	Admin Services Error: Access denied to domain/user/group information	The agent was unable to access necessary information.
1902	Admin Services Error: Invalid computer from which to fetch information	The computer to be used to get security information is invalid.
1903	Admin Services Error: Group not found	The requested group could not be found.
1904	Various	The reported error was encountered while browsing.
1905	Admin services error: could not translate context to Unicode	The requested object for browsing could not be translated to Unicode
1906	Admin service out of memory	The browsing service ran out of memory.
1907	Search request object too long: # > #	The requested object for browsing is too long.
2000	AcquireCredentialsHandle failed: 0x#	The agent could not acquire the credentials needed for an SSL session.

Table B-1. BCAA Event Messages (Continued)

Message ID	Message	Description
2001	Various	The agent was unable to negotiate an SSL session for the reason given.
2002	Various	An I/O error occurred during an SSL session .
2003	Various	The specified cryptographic error occurred during an SSL session.
2004	Various	The specified problem occurred with a certificate during SSL negotiation.

Appendix C: Managing the SSL Client

Understanding the SSL Client

The SSL client is used to determine various SSL parameters for outgoing HTTPS connections. Specifically, its role is to:

- Identify the SSL protocol version the SG uses in negotiations with origin servers.
- Identify the cipher suites used.
- Determine which certificate can be presented to origin servers by associating a keyring with the SSL client.

Creating an SSL Client

The SG is configured with a default SSL client.

Creation of the SSL client means that for every HTTPS connection to the destination server, the SG picks the parameters needed for negotiating the SSL connection from the SSL-client configuration. Thus, multiple SSL connections to different HTTPS destination servers can be supported with a single SSL-client configuration. This is similar to a browser where one configuration is used to negotiate multiple connections with different hosts.

When the SG is acting as an SSL client (SSL origination), SSL sessions are re-used until the server forces a fresh handshake or until the same session ID has been used 255 times.

If you just need to change the protocol, the cipher suites, or the keyring associated with the SSL client, you do not need to recreate the client. Continue with "Associating a Keyring and Protocol with the SSL Client" on page 173 or "Changing the Cipher Suites of the SSL Client" on page 174.

To create the SSL client:

```
SGOS#(config ssl) create ssl-client default
defaulting protocol to SSLv2v3TLSv1
defaulting associated keyring-id to default
  ok
```

To delete the SSL client:

```
SGOS#(config ssl) delete ssl-client default
  ok
```

Associating a Keyring and Protocol with the SSL Client

The SSL client, called default, already exists on the SG. Keyrings that are not used to authenticate encrypted connections do not need to be associated with the SSL client.

Important: Only one keyring can be associated with the SSL client at a time.

To associate a keyring with the SSL client and change the protocol version:

1. Select **Configuration>SSL>SSL Client**.
2. Verify **Use SSL Client** is selected.

3. Only keyrings with certificates can be associated with the SSL client, displayed in the **Keyring** drop-down list. Select the keyring used to negotiate with origin content servers through an encrypted connection.
4. You can change the SSL Versions default from **SSLv2v3TLSv1** to any other protocol listed in the drop-down list.
5. Click OK; click **Apply**

Related CLI Syntax to Associate a Keyring and Protocol with the SSL Client

```
SGOS#(config) ssl
SGOS#(config ssl) edit ssl-client default
SGOS#(config ssl ssl-client default) keyring-id keyring_id
SGOS#(config ssl ssl-client default) protocol {sslv2 | sslv3 | tlsv1 |
sslv2v3 | sslv2tlsv1 | sslv3tlsv1 | sslv2v3tlsv1}
```

Changing the Cipher Suites of the SSL Client

The cipher suite sets the encryption method used by the SG. As the encryption key strength is determined by the signed certificate, configuring a higher cipher suite than defined by the certificate has no affect. Conversely, the cipher suite configuration must be high enough to accommodate certification encryption values.

This can only be done through the CLI.

To change the cipher suite of the SSL client:

The default is to use all ciphers.

You have a choice of using the interactive or non-interactive `create` command.

Note: Director uses non-interactive commands in profiles and overlays to create cipher suites. For more information on Director, refer to the *Blue Coat Director Configuration and Management Guide*.)

To change the cipher suites used through the:

- interactive command: continue with the next procedure.
- non-interactive command: skip to "To change the cipher suites non-interactively:" on page 175.

To change the cipher suites using the interactive cipher-suites command:

Note that the `Use` column in the `set cipher-suite` output below indicates that the default is to use all ciphers.

1. Choose the cipher suites you want to use at the prompt.

```
SGOS#(config) ssl
SGOS#(config ssl) edit ssl-client default
SGOS#(config ssl ssl-client default) cipher-suite
SSL-Client Name      Keyring Name    Protocol
--------------       ------------    ------------
default             default          SSLv2v3TLSv1
```

```
Cipher#   Use   Description            Strength
-------   ---   --------------------   --------
1         yes     RC4-MD5                Medium
2         no      RC4-SHA                Medium
3         no      DES-CBC3-SHA           High
4         no      DES-CBC3-MD5           High
5         no      RC2-CBC-MD5            Medium
6         no      RC4-64-MD5             Low
7         no      DES-CBC-SHA            Low
8         no      DES-CBC-MD5            Low
9         no      EXP1024-RC4-MD5        Export
10        no      EXP1024-RC4-SHA        Export
11        no      EXP1024-RC2-CBC-MD5    Export
12        no      EXP1024-DES-CBC-SHA    Export
13        no      EXP-RC4-MD5            Export
14        no      EXP-RC2-CBC-MD5        Export
15        no      EXP-DES-CBC-SHA        Export
16        no      AES128-SHA             Medium
17        no      AES256-SHA             High
Select cipher numbers to use, separated by commas: 1,3,4
ok
```

2. (Optional) View the results. Notice the change in the `Use` column.

```
SGOS#(config ssl ssl-client default) view
SSL-Client Name      Keyring Name     Protocol
---------------       ------------     ------------
default               default          SSLv2v3TLSv1

Cipher#   Use   Description            Strength
-------   ---   --------------------   --------
1         yes     RC4-MD5                Medium
2         no      RC4-SHA                Medium
3         yes     DES-CBC3-SHA           High
4         yes     DES-CBC3-MD5           High
5         no      RC2-CBC-MD5            Medium
6         no      RC4-64-MD5             Low
7         no      DES-CBC-SHA            Low
8         no      DES-CBC-MD5            Low
9         no      EXP1024-RC4-MD5        Export
10        no      EXP1024-RC4-SHA        Export
11        no      EXP1024-RC2-CBC-MD5    Export
12        no      EXP1024-DES-CBC-SHA    Export
13        no      EXP-RC4-MD5            Export
14        no      EXP-RC2-CBC-MD5        Export
15        no      EXP-DES-CBC-SHA        Export
16        no      AES128-SHA             Medium
17        no      AES256-SHA             High
```

To change the cipher suites non-interactively:

Enter the following commands:

```
SGOS#(config) ssl
SGOS#(config ssl) edit ssl-client default
SGOS#(config ssl ssl-client default) cipher-suite cipher-suite cipher-suite
```

where [*cipher-suite*] can be any combination of the following:

1. rc4-md5
2. rc4-sha
3. des-cbc3-sha
4. des-cbc3-md5
5. rc2-cbc-md5
6. rc4-64-md5
7. des-cbc-sha
8. des-cbc-md5
9. exp1024-rc4-md5
10. exp1024-rc4-sha
11. exp1024-rc2-cbc-md5
12. exp1024-des-cbc-sha
13. exp-rc4-md5
14. exp-rc2-cbc-md5
15. exp-des-cbc-sha
16. aes128-sha
17. aes256-sha

Notes:

- If you do not specify any attributes, the interactive mode is assumed and the cipher suites cannot be used by Director in profiles or overlays.
- Multiple cipher suites can be specified on the command line.

Example

```
SGOS#(config ssl ssl-client default) cipher-suite rc4-md5 des-cbc3-md5
exp1024-rc4-md5 exp-des-cbc-sha
  ok
SGOS#(config ssl ssl-client default) view
SSL-Client Name      Keyring Name     Protocol
--------------       ------------     ------------
default               default          SSLv2v3TLSv1

Cipher#   Use   Description            Strength
-------   ---   --------------------   --------
1         no      RC4-MD5                Medium
2         no      RC4-SHA                Medium
3         no      DES-CBC3-SHA           High
4         no      DES-CBC3-MD5           High
5         no      RC2-CBC-MD5            Medium
6         no      RC4-64-MD5             Low
7         no      DES-CBC-SHA            Low
8         no      DES-CBC-MD5            Low
9         no      EXP1024-RC4-MD5        Export
10        no      EXP1024-RC4-SHA        Export
11        no      EXP1024-RC2-CBC-MD5    Export
12        no      EXP1024-DES-CBC-SHA    Export
13        no      EXP-RC4-MD5            Export
14        no      EXP-RC2-CBC-MD5        Export
15        yes     EXP-DES-CBC-SHA        Export
16        no      AES128-SHA             Medium
17        no      AES256-SHA              High
```

Troubleshooting Server Certificate Verification

Server certificate verification can be disabled for all upstream hosts or specific upstream hosts. The SG, by default, verifies the SSL certificate presented by the upstream HTTPS server. However, it fails to negotiate the SSL connection if SSL certificate verification fails.

The two most common causes of server certificate verification failure are:

- The absence of a suitable CA certificate on the SG. Be sure that the SG is configured with the relevant CA certificates to avoid unwanted verification failures.
- If a forwarding host of type HTTPS server is being used, you can override the default behavior by changing the `ssl-verify-server` option on a per-host basis .
- The server is using a self-signed certificate. In this case, you need to change the keyring to one that has a CA certificate.

Setting the SSL Negotiation Timeout

The SSL negotiation timeout value dictates the time a SG waits for a new SSL handshake to complete. This value applies to both the HTTPS Reverse Proxy and SSL origination.

You can change the default SSL negotiation timeout value if the default, 300 seconds, is not sufficient for your environment. This value can only be changed through the CLI; it cannot be set from the Management Console.

To change the HTTPS Reverse Proxy timeout period, enter the follow commands from the command prompt:

```
SGOS#(config) ssl
SGOS#(config ssl) view ssl-nego-timeout
300
SGOS#(config ssl) ssl-nego-timeout seconds
```

Index

D

E

F

H

I

R

S

T