STEEL-BELTED

RADIUS®

Administration Guide

Enterprise Edition

Release 5.4

March 2006

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA
(408) 745-2000
www.juniper.net

M06310

Contents

Chapter 8 Administering RADIUS Tunnels

Chapter 9 Administering Address Pools

Chapter 10 Setting Up Administrator Accounts

Chapter 11 Setting Up Authentication Policies

Chapter 13 LDAP Configuration Interface

Chapter 14 Configuring SQL Authentication

Appendix F Configuring IPv6 Support

Appendix G Stopping and Starting Steel-Belted Radius

Index

About This Guide

The *Steel-Belted Radius Administration Guide*/Enterprise Edition describes how to configure and administer the Steel-Belted Radius software on a server running the Solaris operating system, the Linux operating system, or the Windows 2000/Windows XP/Windows Server 2003 operating system.

Before You Begin

This manual assumes that you have installed the Steel-Belted Radius software and the SBR Administrator. For information on how to install the Steel-Belted Radius software, refer to the Steel-Belted Radius *Getting Started* manual.

Audience

This manual is intended for network administrators responsible for implementing and maintaining authentication, authorization, and accounting services. This manual assumes that you are familiar with general RADIUS (Remote Authentication Dial In User Service) and networking concepts and the specific environment in which you are installing Steel-Belted Radius.

If you use Steel-Belted Radius with third-party products such as Oracle or RSA SecurID, you should be familiar with their installation, configuration, and use.

What's In This Manual

This manual contains the following chapters and appendixes:

- Chapter 1, "About Steel-Belted Radius," presents an overview of Steel-Belted Radius and describes licensing requirements for Steel-Belted Radius.
- Chapter 2, "RADIUS Basics," summarizes important concepts relating to the operation of Steel-Belted Radius.

- Chapter 3, "Using SBR Administrator," describes how to use the SBR Administrator to configure Steel-Belted Radius.
- Chapter 4, "Administering RADIUS Clients," describes how to set up remote access server (RAS) devices as Steel-Belted Radius clients.
- Chapter 5, "Administering Users," describes how to set up users in the Steel-Belted Radius database.
- Chapter 6, "Administering Profiles," describes how to set up user profiles to simplify user administration.
- Chapter 7, "Administering Proxy Targets," describes how to identify proxy RADIUS targets.
- Chapter 8, "Administering RADIUS Tunnels," describes how to set up secure RADIUS tunnels.
- Chapter 9, "Administering Address Pools," describes how to set up IPv4 and IPX address pools.
- Chapter 10, "Setting Up Administrator Accounts," describes how to identify who can administer Steel-Belted Radius.
- Chapter 11, "Setting Up Authentication Policies," presents an overview of Extensible Authentication Protocol (EAP) types and describes how to configure and sequence RADIUS authentication methods.
- Chapter 12, "Configuring Replication," describes how to configure and use the centralized configuration management (CCM) feature to coordinate Steel-Belted Radius server settings in a replication environment
- Chapter 13, "LDAP Configuration Interface," describes how to use public domain LDAP utilities to populate a Steel-Belted Radius server database.
- Chapter 14, "Configuring SQL Authentication," describes how to configure authentication against records stored in an external SQL database.
- Chapter 15, "Configuring SQL Accounting," describes how to configure Steel-Belted Radius to write accounting information to an external SQL database.
- Chapter 16, "Configuring LDAP Authentication," describes how to configure authentication against records stored in an external LDAP database.
- Chapter 17, "Displaying Statistics," describes how to use the monitoring facilities in Steel-Belted Radius.
- Chapter 18, "Logging and Reporting," describes how to use the logging and reporting facilities in Steel-Belted Radius.
- Appendix A, "Glossary," provides brief explanations for RADIUS terminology used in this and other Steel-Belted Radius manuals.
- Appendix B, "When to Restart Steel-Belted Radius," provides a summary of critical operational information.
- Appendix C, "Technical Notes," presents tips for configuring Steel-Belted Radius to interoperate with equipment and facilities from other vendors.

- Appendix D, "Authentication Protocols," provides a matrix of authentication methods and their supported authentication protocols.
- Appendix E, "Importing and Exporting Data," describes how to import and export information in a Steel-Belted Radius database to and from an XML file.
- Appendix F, "Configuring IPv6 Support," presents an overview of IPv6 features and concepts and describes how to enable experimental IPv6 functions in Steel-Belted Radius.
- Appendix G, "Stopping and Starting Steel-Belted Radius," describes how to stop and restart the Steel-Belted Radius service (Windows) or RADIUS daemon (Solaris/Linux).

Typographical Conventions

This manual uses the following conventions to present special types of text.

Computer Text

Filenames, directory names, IP addresses, URLs, commands, and file listings appear in a plain fixed-width font:

For more information, go to `http://www.funk.com`...

```
[EventDilutions]
SQLConnectFailure=8
```

In examples, text that you type literally is shown in a bold font.

```
C:\>cd \Radius\Service
```

Screen Interaction

Text related to the SBR Administrator's user interface appears in **bold sans serif type**.

Click the **OK** button.

Enter your username in the **Login** field.

Menu commands are presented as the name of the menu, followed by the **>** sign and the name of the command. If a menu item opens a submenu, the complete menu path is given.

Choose **Edit > Cut**.
Choose **Edit > Paste As... > Text**.

Variable Text

Variable text that you must replace with your own information appears in *italics*. For example, you would enter your name and password in place of ***YourName*** and ***YourPassword*** in the following interaction.

```
Enter your name: YourName
```

```
Password: YourPassword
```

File names and computer text can also be displayed in italics to indicate that the you should replace the values shown with values appropriate for your enterprise. For example, you would enter your own information in place of the italicized text in the following example:

```
[EventDilutions]
EventName=DilutionCount
```

Key Names

Names of keyboard keys appear in SMALL CAPS. When you need to press two or more keys simultaneously, the key names are joined by a + sign:

Press RETURN.

Press CTRL+ALT+DEL.

Syntax

- *radiusdir* represents the directory into which Steel-Belted Radius has been installed. By default, this is `C:\Radius` for Windows systems and `/opt/funk/radius` on Linux and Solaris systems.
- Brackets [ ] enclose optional items in format and syntax descriptions. In the following example, the first *Attribute* argument is required; you can include an optional second *Attribute* argument by entering a comma and the second argument (but not the square brackets) on the same line.

  ```
  <add | replace> = Attribute [,Attribute]
  ```

  In configuration files, brackets identify section headers:

  ...the [Configuration] section of `radius.ini`...

  In screen prompts, brackets indicate the default value. For example, if you press ENTER without entering anything at the following prompt, the system uses the indicated default value (`/opt`).

  ```
  Enter install path [/opt]:
  ```
- Angle brackets < > enclose a list from which you must choose an item in format and syntax descriptions.
- A vertical bar (|) separates items in a list of choices. In the following example, you must specify `add` or `replace` (but not both):

  ```
  [AttributeName]
  <add | replace> = Attribute [,Attribute]
  ```

Related Documentation

The following documents supplement the information in this manual.

Steel-Belted Radius Documentation

Please review the `ReleaseNotes.txt` file that accompanies your Steel-Belted Radius software for late-breaking information not available in this manual.

In addition to this manual, the Steel-Belted Radius documentation includes the following manuals:

- The *Steel-Belted Radius Getting Started* manual describes how to install, configure, and administer the Steel-Belted Radius software on a server running the Solaris operating system, the Linux operating system, or the Windows 2000/Windows XP/Windows Server 2003 operating system.
- The *Steel-Belted Radius Reference Guide* describes the configuration files and settings used by Steel-Belted Radius.

Requests for Comments (RFCs)

The Internet Engineering Task Force (IETF) maintains an online repository of Request for Comments (RFC)s online at `http://www.ietf.org/rfc.html`.

- RFC 2548, *Microsoft Vendor-specific RADIUS Attributes*. G. Zorn. March 1999.
- RFC 2618, *RADIUS Authentication Client MIB*. B. Aboba, G. Zorn. June 1999.
- RFC 2619, *RADIUS Authentication Server MIB*. G. Zorn, B. Aboba. June 1999.
- RFC 2620, *RADIUS Accounting Client MIB*. B. Aboba, G. Zorn. June 1999.
- RFC 2621, *RADIUS Accounting Server MIB*. G. Zorn, B. Aboba. June 1999.
- RFC 2809, *Implementation of L2TP Compulsory Tunneling via RADIUS*. B. Aboba, G. Zorn. April 2000.
- RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*. C. Rigney, S. Willens, A. Rubens, W. Simpson. June 2000.
- RFC 2866, *RADIUS Accounting*. C. Rigney. June 2000.
- RFC 2867, *RADIUS Accounting Modifications for Tunnel Protocol Support*. G. Zorn, B. Aboba, D. Mitton. June 2000.
- RFC 2868, *RADIUS Attributes for Tunnel Protocol Support*. G. Zorn, D. Leifer, A. Rubens, J. Shriver, M. Holdrege, I. Goyret. June 2000.
- RFC 2869, *RADIUS Extensions*. C. Rigney, W. Willats, P. Calhoun. June 2000.
- RFC 2882, *Network Access Servers Requirements: Extended RADIUS Practices*. D. Mitton. July 2000.
- RFC 3162, *RADIUS and IPv6*. B. Aboba, G. Zorn, D. Mitton. August 2001.

Third-Party Products

For more information about configuring your access servers and firewalls, consult the manufacturer's documentation provided with each device.

Contacting Technical Support

If you are located in the U.S. and Canada, you can contact Funk Software Technical Support in the following ways:

Telephone:	(617) 491-6503
Email:	`support@funk.com`
Web:	Go to `http://www.funk.com`, click **Tech Support**, and then click **Steel-Belted Radius.**
From within SBR Administrator:	Choose **Web > Steel-Belted Radius User Page**.

If you are located outside the U.S. and Canada, please contact the authorized Funk Software partner in your area to obtain support.

- Our Technical Support department is open weekdays between 9:00 AM and 5:30 PM (Eastern) to customers who are on warranty support, who are evaluating the product, or who wish to purchase on-demand support.
- Our Technical Support department is open weekdays between 9:00 AM and 8:00 PM (Eastern) to customers who hold a current annual maintenance and support contract.

When you call, please have the following information available:

- The Steel-Belted Radius product edition and release number.
- Information about the server configuration and operating system, including any OS patches that have been applied.
- For licensed products under a current maintenance agreement, your license or support contract number.
- Question or description of the problem, with as much detail as possible.
- Any documentation that may help in resolving the problem, which could include error messages, memory dumps, compiler listings, error logs, etc.

You can use the Funk Software website (`http://www.funk.com`) to register your software, display answers to frequently asked questions, search the Steel-Belted Radius technical support database, and download product documentation in Adobe Reader (.pdf) format.

Chapter 1
About Steel-Belted Radius

Thank you for selecting Steel-Belted Radius®/Enterprise Edition.

Steel-Belted Radius is a complete implementation of the RADIUS (Remote Authentication Dial In User Service) protocol. Steel-Belted Radius interfaces with a wide variety of network access equipment, and authenticates remote and wireless LAN (WLAN) users against numerous back-end databases, allowing you to consolidate the administration of all your remote and WLAN users.

Steel-Belted Radius/Enterprise Edition delivers a total RADIUS solution, designed to meet the access control and policy management requirements of enterprises. It interfaces with a wide variety of remote access servers, including VPN and dial-in servers and wireless LAN access points, and authenticates remote and WLAN users against your existing security infrastructure. This lets you control who can access your network and what resources are available to them. Steel-Belted Radius logs all access usage, so you can track and document usage statistics.

Steel-Belted Radius Features

- Centralized management of user access control and security simplifies access administration.
- Flexible, powerful proxy RADIUS features let you easily distribute authentication and accounting requests to the appropriate RADIUS server for processing.
- Authentication against a local database permits network access by employees.
- Flexible authentication options let you use your existing OS-based authentication database, token systems from RSA Security and other vendors, and external SQL/LDAP databases for remote and WLAN user authentication.
- Support for a wide variety of 802.1X-compliant access points and other network access servers ensures compatibility in your network environment.
- You can configure Steel-Belted Radius by means of a graphical SBR Administrator .
- Administrative access levels can be defined and applied to user or group accounts on the server machine. Read, write, and read/write access can be applied selectively

to various categories of configuration data, including users, RADIUS clients, proxy targets, and statistics.

How Steel-Belted Radius Works

RADIUS is an industry-standard method for providing authentication, authorization, and accounting services:

- **Authentication** is the process of verifying a user is who he/she claims to be.
- **Authorization** is the process of determining whether the user is allowed on the network and controlling the network resources that the user can access on the protected network.
- **Accounting** is the process of recording accounting records generated by a RAS that can be used for billing, system diagnosis, and usage planning.

A RADIUS-based remote access environment typically involves three types of components:

- An access client is a user who initiates a network connection. An access client might be a user dialing into a service provider network, a router at a small office/home office connecting to an enterprise network to provide network access, or a wireless client connecting to an 802.1X access point.
- A remote access server (RAS) is a device that recognizes and processes connection requests from outside the network edge. A RAS can be a wireless access point, a modem pool, a network firewall, or any other device that needs to authenticate users. When the RAS receives a user's connection request, it may perform an initial access negotiation with the user to obtain identity/password information. The RAS then passes this information to the RADIUS server as part of an authentication/authorization request.
- The RADIUS server matches data from the authentication/authorization request with information in a trusted database, such as the database on the Steel-Belted Radius server; on some other type of authentication server, such as SecurID or TACACS+; in a SQL or LDAP database; or on some other RADIUS server for which this server is a proxy. If a match is found and the user's credentials are correct, the RADIUS server sends an Access-Accept message to the RAS; if a match is not found or a problem is found with the user's credentials, the server returns an Access-Reject message. The RAS then establishes or terminates the user's connection. The RAS may then forward accounting information to the RADIUS server to document the transaction; the RADIUS server may store or forward this information as needed to support billing for the services provided.

Figure 1 illustrates a simple RADIUS environment.

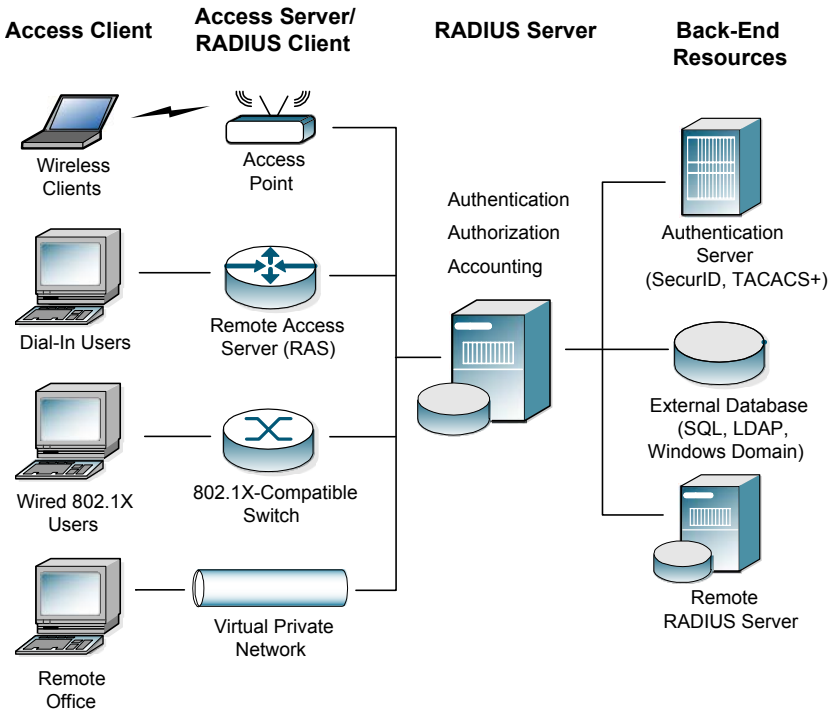


***Figure 1** RADIUS Environment*

Licensing

If you want to install the Steel-Belted Radius server software for a 30-day evaluation, you do not need a license key.

If you want to install a permanent (non-evaluation) copy of Steel-Belted Radius, you must have a single-seat software license key.

If you have more than one copy of the Steel-Belted Radius software installed, you must have a site license key or you must have a separate license key for each installation.

The SBR Administrator can be installed on as many workstations as you require. The SBR Administrator does not require a license key.

For details about licensing, please refer to the Steel-Belted Radius license agreement or contact Funk Software.

Chapter 2
RADIUS Basics

This chapter presents a conceptual overview of RADIUS (Remote Authentication Dial In User Service) authentication, authorization, and accounting services.

RADIUS Overview

RADIUS is an industry-standard protocol for providing authentication, authorization, and accounting services.

- **Authentication** is the process of verifying a user's identity and associating additional information (attributes) to the user's login session.
- **Authorization** is the process of determining whether the user is allowed on the network and controlling network access values based on a defined security policy.
- **Accounting** is the process of generating log files that record session statistics used for billing, system diagnosis, and usage planning.

A RADIUS-based remote access environment typically involves three types of components:

- An access client is a user who initiates a network connection. An access client might be a user dialing into a service provider network, a router at a small office/home office connecting to an enterprise network to provide network access, or a wireless client connecting to an 802.1X access point.
- A remote access server (RAS) is a device that recognizes and processes connection requests from outside the network edge. A RAS can be a wireless access point, a modem pool, a network firewall, or any other device that needs to authenticate users. When the RAS receives a user's connection request, it may perform an initial access negotiation with the user to obtain identity/password information. The RAS then passes this information to the RADIUS server as part of an authentication/authorization request.
- The RADIUS server matches data from the authentication/authorization request with information in a trusted database, such as the database on the Steel-Belted Radius server; on some other type of authentication server, such as SecurID or TACACS+; in a SQL or LDAP database; or on some other RADIUS server for which this server is a proxy. If a match is found and the user's credentials

are correct, the RADIUS server sends an Access-Accept message to the RAS; if a match is not found or a problem is found with the user's credentials, the server returns an Access-Reject message. The RAS then establishes or terminates the user's connection. The RAS may then forward accounting information to the RADIUS server to document the transaction; the RADIUS server may store or forward this information as needed to support billing for the services provided.Proxy RADIUS support let you distribute authentication and accounting requests to the appropriate RADIUS server for processing.

Figure 2 illustrates a simple RADIUS authentication/authorization sequence.

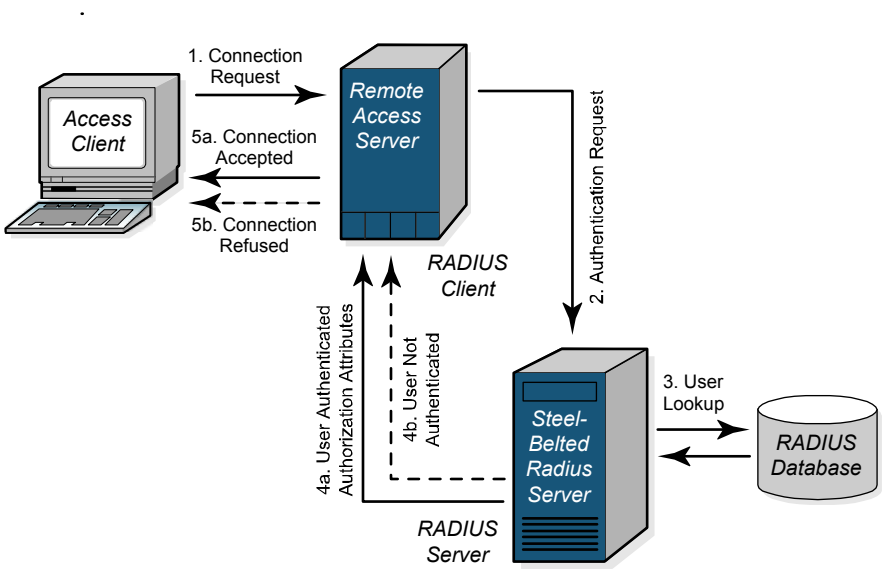


***Figure 2** RADIUS Authentication*

1 A RADIUS *access client*, which could be a dial-up device, a mobile device with wireless network access, or a computer at a remote office, sends an authentication request containing identification and connection information to a *remote access server* (RAS), which might be a wireless Access Point, an ISDN bridge, or a modem pool.

***NOTE:** The terms "remote access server" (or RAS) and "network access server" (or NAS) are interchangeable. This manual uses "RAS," though some attribute names and parameters retain the older "NAS" in their names.*

2 When the RAS receives a user's connection request, it typically performs an initial access negotiation with the user (EAP, PPP or SLIP) to establish connection information (username, password, RAS device identifier, RAS port number, and so on). The RAS then forwards the user information in an authentication request to the RADIUS server.

3 The RADIUS server looks up the user information in a local or remote RADIUS authentication database. The RADIUS server verifies that the user's name and password are valid. It can also enforce fine-grained security rules by using an *access checklist* to verify specific attributes in the authentication request.

4 If a match is found, the RADIUS server returns an Access-Accept message (4a). The RADIUS server might also send *return list* information stored in the database, such as the user's authorization or connection parameters, back to the RAS.

If a match is not found, the RADIUS server returns an Access-Reject message (4b).

If third-party software such as RSA SecurID is used, the RADIUS server may prompt the user for more information before accepting or rejecting the authentication request.

5 Based on the information it receives from the RADIUS server, the RAS accepts or refuses the connection request.

After the user is authenticated and the connection established, the RAS may forward accounting data to the RADIUS server to document the transaction; the RADIUS server can store or forward this data to support billing for the services provided.

RADIUS Packets

A RADIUS client and RADIUS server communicate by means of RADIUS packets. RADIUS packets carry messages between the RADIUS client and RADIUS server in a series of request/response transactions: the client sends a request and expects a response from the server. If the response does not arrive, the client can retry the request periodically.

Each RADIUS packet supports a specific purpose: authentication or accounting. A packet can contain values called *attributes*. The specific attributes to be found in each packet depend upon the type of packet (authentication or accounting) and the device that sent it (for example, the specific make and model of the RAS device acting as a RADIUS client).

For information on RADIUS authentication packet structures and attributes, see RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*. For information on RADIUS accounting packet structures and attributes, see RFC 2866, *RADIUS Accounting*.

RADIUS Configuration

You must configure a RADIUS client and RADIUS server before they can communicate. If the client and server are on the same network, one administrator may be able to configure both sides of the RADIUS communication. If the client and server are on different networks, you may have to coordinate RADIUS configuration details with the administrators of other networks.

RADIUS Server Configuration

To configure Steel-Belted Radius to respond to RADIUS clients, run the SBR Administrator, open the RADIUS Clients panel, and enter the following information for each RADIUS client:

- The IP address of the client device.
- The RADIUS shared secret used by Steel-Belted Radius and the client device. For information on RADIUS shared secrets, see "RADIUS Shared Secret" on page 8.
- The make and model of the client device, selected from a list of devices that Steel-Belted Radius supports. If a specific make and model is not listed, select **- Standard Radius -**.

Additionally, you must configure the UDP ports the server will use when sending and receiving RADIUS authentication and accounting packets. The UDP ports you configure on the RADIUS server must match the UDP ports that the RADIUS client is using for the same purposes. For more information, see "RADIUS Ports" on page 10.

RADIUS Client Configuration

You must tell each RADIUS client how to contact its RADIUS server. To configure a client to work with a Steel-Belted Radius server, log in to the client device, run its administration program, bring up its RADIUS configuration interface, and enter the following information:

- The IP address of the Steel-Belted Radius server.
- The RADIUS shared secret to be used by Steel-Belted Radius and the client device. For information on RADIUS shared secrets, see "RADIUS Shared Secret" on page 8.
- The UDP ports on which to send and receive RADIUS authentication and accounting packets. These must match the UDP ports that Steel-Belted Radius is using for the same purposes. For more information, see "RADIUS Ports" on page 10.

Multiple RADIUS Servers

You can distribute the RADIUS workload among several servers, as follows:

- You can set up separate servers for RADIUS authentication and accounting services. When RADIUS authentication and accounting services are performed by separate servers, each client device must be configured to send its authentication packets to one RADIUS server and its accounting packets to another.
- You can provide redundancy by pairing RADIUS servers to work in tandem. Most RAS configuration interfaces let you designate primary and secondary servers for authentication and accounting.

If both measures for distributing the RADIUS workload are implemented, client configuration involves identifying four servers for each client device: a primary RADIUS accounting server, a secondary RADIUS accounting server, a primary RADIUS authentication server, and a secondary RADIUS authentication server.

RADIUS Shared Secret

A RADIUS shared secret is a case-sensitive test string used to validate communications between two RADIUS devices. You should configure shared secrets that are long enough and random enough to resist attack, and you should avoid using the same shared secret throughout your network. To maximize the security of your server's shared secret, consider using Funk's free Password Amplifier utility, which takes an ordinary shared secret or password (`swordfish`) and hashes it repeatedly to produce a 16-character amplified secret (`g8QvQuRgRsl1AQ1E`). You can paste this amplified secret into your server configuration to maximize security.

NOTE: *For more information on Funk's free Password Amplifier utility, see* `http://www.funk.com/radius/News/pass_amp_pn.asp`.

Steel-Belted Radius uses three types of shared secrets:

- RADIUS secret – Used to authenticate communication between a RADIUS server and a RADIUS client
- Replication secret – Used to authenticate communication between a primary server and a replica server
- Node secret – If you use RSA SecurID, Steel-Belted Radius uses a node secret to authenticate communication between a RADIUS server and an RSA Authentication Manager server.

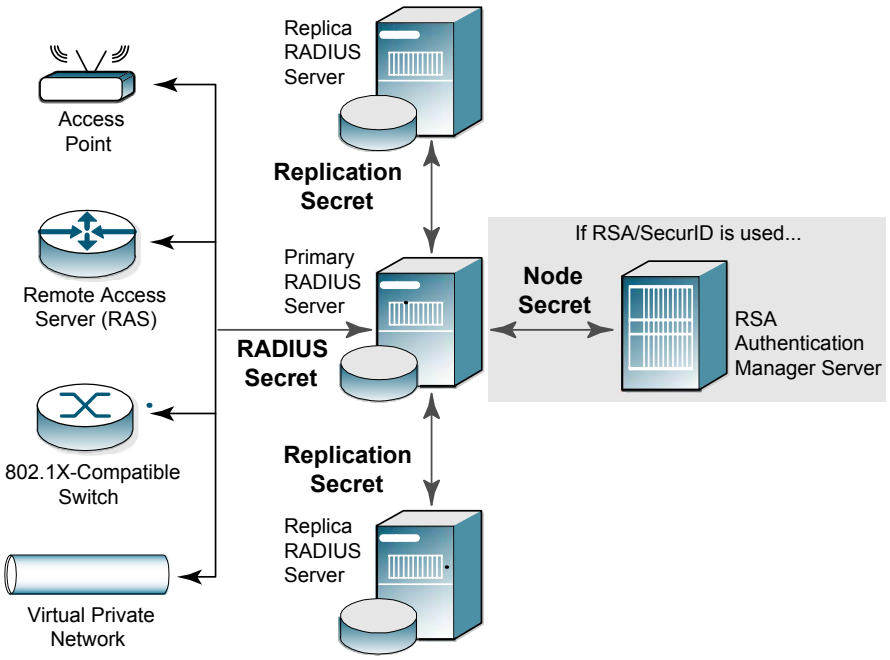


Figure 3 *Shared Secrets*

RADIUS Secret

A RADIUS shared secret is a case-sensitive password used to validate communications between a RADIUS server, such as Steel-Belted Radius, and a RADIUS client, such as an Access Point (AP) or Remote Access Server (RAS). Steel-Belted Radius supports shared secrets of up to 127 alphanumeric characters, including spaces and the following special characters:

```
~!@#$%^&*()_+|\=-`{}[]:"';<>?/.,
```

Identical shared secrets must be configured on both sides of the RADIUS communication link.

NOTE: *Not all RAS devices support shared secrets of up to 127 alphanumeric/special characters. You should select shared secrets that are fully supported by RADIUS devices in your network.*

Most RADIUS clients allow you to configure different secrets for authentication and accounting. If the interface also permits you to identify primary and secondary RADIUS servers, you can set up as many as four secrets (primary accounting, secondary accounting, primary authentication, and secondary authentication).

On the server side, the configuration interface allows you to create a list of known RADIUS clients (RAS devices). You will need to configure the RADIUS server with the authentication shared secret and the accounting shared secret for each client on this list.

During an authentication transaction, password information must be transmitted securely between the RADIUS client (RAS or AP) and Steel-Belted Radius.Password security can be addressed using a variety of protocols, such as PAP, CHAP, or MS-CHAP (described in "Password Protocols" on page 16). When PAP is used, the password is encrypted and decrypted using the authentication shared secret.

No encryption is involved in transmitting accounting data between a RADIUS client and RADIUS server. However, the accounting shared secret is used by each device to verify that it can "trust" any RADIUS communications it receives from the other device.

During an authentication transaction, password information must be transmitted securely between the RADIUS client and the RADIUS server.

Replication Secret

A replication secret is a text string used to authenticate communications between a primary server and a replica server. You do not need to configure the replication secret for a realm: the primary server generates it automatically, and each replica server in a realm receives the replication secret as part of its configuration package.

See "Centralized Configuration Management" on page 19 for information on primary and replica servers.

Node Secret

If you use Steel-Belted Radius with RSA SecurID, the RSA Authentication Manager software views the Steel-Belted Radius service as a *host agent*. You must configure a node secret to authenticate communication between Steel-Belted Radius and the RSA Authentication Manager. A node secret is a pseudorandom string known only to the Steel-Belted Radius and RSA Authentication Manager. Before the Steel-Belted Radius sends an authentication request to the RSA Authentication Manager, it encrypts the data using a symmetric node secret key.

RADIUS Ports

The RADIUS standard initially used UDP ports 1645 and 1646 for RADIUS authentication and accounting packets. The RADIUS standards group later changed the port assignments to 1812 and 1813, but many organizations still use the old 1645/1646 port numbers for RADIUS.

Any two devices that exchange RADIUS packets must use compatible UDP port numbers. That is, if you are configuring a RAS to exchange authentication packets with a RADIUS server, you must find out which port the server uses to receive authentication

packets from its clients (1812, for example). You must then configure the RAS to send authentication packets on the same port (1812). The same is true for RADIUS accounting.

Steel-Belted Radius can listen on multiple ports. For compatibility, the server listens to the old and new default RADIUS ports: ports 1645 and 1812 for authentication, and ports 1646 and 1813 for accounting. To add, change, or disable the ports on which Steel-Belted Radius listens, modify the `radius.ini` file or edit the `services` (Windows) or `/etc/services` (Solaris/Linux) file. The `radius.ini` file and the `services` file are described in the *Steel-Belted Radius Reference Guide*.

Authentication

RADIUS uses different types of messages during user authentication. Table 1 summarizes the conditions under which each type of RADIUS authentication message is issued, and the purpose of any RADIUS attributes the message contains.

***Table 1.** RADIUS Authentication Messages and Attributes*

Message Conditions	Purpose of Message Attributes
When a RAS receives a connection request from a user, the RAS requests authentication by sending an Access-Request to its RADIUS server.	Identify the user. Describe the type of connection the user is trying to establish.
When a RADIUS server is able to authenticate a user, it returns a RADIUS Access-Accept to the RAS.	Allow the RAS to complete access negotiations. Configure connection details such as providing the RAS with an IP address it can assign to the user. Enforce time limits and other “class of service” restrictions on the connection.
When a RADIUS server is unable to authenticate a connection request, it returns an Access-Reject to the RAS.	Terminate access negotiations. Identify the reason for the authorization failure.
If initial authentication conditions are met but additional input is needed from the user, the RADIUS server returns an Access-Challenge to the RAS.	Enable the RAS to prompt the user for more authentication data. Complete the current Access-Request, so the RAS can issue a new one.

Authentication Methods

Each time an Access-Request message reaches the server, an authentication transaction begins. During this transaction, the server attempts to authenticate the request by sequentially trying its configured and enabled authentication methods. The server consults its list of authentication methods to determine which methods to try and in which order to try them. You can view and edit the authentication methods list by running the SBR Administrator and opening the Authentication Policies panel.

Native User Authentication

Native user authentication references user accounts stored on the Steel-Belted Radius server. When trying the native user method, Steel-Belted Radius searches its database for an entry whose User-Type is `Native User`, and whose username matches the User-Name in the Access-Request.

- If the entry cannot be found, or if it is found and the password is invalid, Steel-Belted Radius tries the next enabled method in the authentication methods list.
- If an entry for the user is found but the entry's checklist does not match attributes found in the Access-Request, Steel-Belted Radius returns an Access-Reject message to the RAS.
- If the entry is found and its password and checklist match perfectly, Steel-Belted Radius formats an Access-Accept message using the entry's return list, and returns it to the RAS.

Pass-Through Authentication

Pass-through authentication methods permit Steel-Belted Radius to begin the authentication by asking another entity to validate the username and password found in the Access-Request.

Steel-Belted Radius can pass authentication requests through to a Windows security database, RSA Authentication Manager (RSA SecurID), or TACACS+ server.

Proxy RADIUS Authentication

Steel-Belted Radius can convey an Access-Request to some other RADIUS server, which then (1) attempts to authenticate the connection request according to its own conventions and (2) returns a response to Steel-Belted Radius. Steel-Belted Radius then relays this response to the RAS. The set of conventions for relaying packets between cooperating RADIUS servers is known as *proxy RADIUS*.

External Authentication

External authentication methods enable Steel-Belted Radius to authenticate users by referring to external SQL or LDAP databases. During external authentication, Steel-Belted Radius queries the database for authentication data, and uses the results to format a response packet. Steel-Belted Radius then relays this response to the RAS.

For information on using Steel-Belted Radius with SQL databases, see Chapter 14, "Configuring SQL Authentication" on page 181. For information on using Steel-Belted Radius with LDAP databases, see Chapter 16, "Configuring LDAP Authentication" on page 205.

HTTP Digest Access Authentication

HTTP Digest Access authentication provides a simple challenge-response authentication mechanism that an HTTP server can use to challenge an HTTP client request.

Steel-Belted Radius supports two forms of HTTP Digest Access authentication:

- HTTP Digest Access authentication, which is described in `draft-ietf-radext-digest-auth-05.txt`, uses 13 new RADIUS attributes to authenticate access requests from an HTTP server. When HTTP Digest Access authentication is used:
  a. An HTTP client sends a request without an authorization header to an HTTP server.
  b. The HTTP server sends a challenge containing a random value (nonce) to the HTTP client.
  c. The HTTP client creates an MD5 hash containing the username, password, nonce value, and other information, and returns this MD5 hash to the HTTP server in a request with an authentication header.
  d. The HTTP server sends an Access-Request message containing special RADIUS attributes to Steel-Belted Radius.
  e. Steel-Belted Radius verifies the HTTP client's credentials and returns a RADIUS Access-Accept or Access-Reject message to the HTTP server.
- The Ericsson ViG version of HTTP Digest Access authentication, which is described in `draft-sterman-aaa-sip-05.txt`, uses two Ericsson vendor-specific attributes (a Digest-Response attribute and one or more Digest-Attributes attributes) to authenticate access requests. When Ericsson ViG HTTP Digest Access authentication is enabled, Steel-Belted Radius looks for the ViG VSAs when it parses incoming packets, and, if it finds them, converts them to AVPs compatible with HTTP Digest Access authentication.

You must edit settings in the `radius.ini` file to enable HTTP Digest Access authentication.

Authenticate-Only Requests

Steel-Belted Radius supports requests to authenticate a user where the server performs no other processing. The RAS specifies this type of request by setting the `Service-Type` field to a value of `Authenticate-Only` (numeric value 8). The server responds with either an Access-Reject or an Access-Accept (without any attributes).

You can disable this feature (so that attributes are always returned in the response packet) by setting the `AuthenticateOnly` field in the [Configuration] section of the `radius.ini` file to `0`. For more information on `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.

Configuring the Authentication Sequence

After you configure authentication methods for Steel-Belted Radius, the Authentication Policies panel in the SBR Administrator displays them in the order in which the server tries them. Enabled methods are displayed in black text; disabled methods are displayed in gray text. During an authentication transaction, the server works down the list, skipping disabled methods.

You can enable or disable methods or re-order methods in the list by using the controls in the Authentication Methods tab of the Authentication Policies panel. For information on setting up authentication sequences, see Chapter 11, "Setting Up Authentication Policies" on page 111.

Configuring Authentication Methods

Each authentication method in Steel-Belted Radius performs a different type of processing on information in an incoming Access-Request packet. Table 2 summarizes what you need to do to configure each authentication method.

Table 2. *Authentication Method Configuration*

Method	How to Configure	See
Native User	Create native user entries in the Steel-Belted Radius database.	"Setting Up Native Users" on page 61
OS Pass-Through Security	This method assumes that you already have users, groups, and passwords defined in your local security database. Create user entries in the Steel-Belted Radius database. Choose User-types as appropriate.	Chapter 5, "Administering Users" on page 59
RSA SecurID	This method assumes that you already have PIN/token code pairs defined on an RSA SecurID server. First, configure Steel-Belted Radius to communicate with the RSA SecurID server. Then create user entries in the Steel-Belted Radius database. Choose SecurID User, <ANY>, SecurID Prefix, and SecurID Suffix User-types.	"Setting Up SecurID Users" on page 71
TACACS+	This method assumes that you have username/password pairs defined on a TACACS+ server. First, configure Steel-Belted Radius to communicate with the TACACS+ server. Then, create user entries in the Steel-Belted Radius database. Assign TACACS+ User, <ANY>, TACACS+ Prefix, and TACACS+ Suffix user-types.	"Setting Up TACACS+ Users" on page 73
Proxy RADIUS	Add a single target. You can set up single targets that are not associated with any realm.	Chapter 7, "Administering Proxy Targets" on page 85

Table 2. *Authentication Method Configuration (Continued)*

Method	How to Configure	See
EAP-TTLS	This method provides a means for an authentication request to be sent directly from the client to the server through a TLS connection. The act of establishing the TLS connection authenticates the server to the client and the authentication request sent through the tunnel authenticates the client to the server. Create a Steel-Belted Radius `ttlsauth.aut` file that specifies options for the TLS connection and the manner in which Steel-Belted Radius routes the inner authentication request. Stop and restart Steel-Belted Radius. Subsequently, the EAP-TTLS authentication method appears in the Authentication Methods tab in the Authentication Policies panel. You can use the Authentication Policies panel to enable, disable, and re-order EAP-TTLS methods.	"Configuring EAP-TTLS and EAP-PEAP" on page 123
External SQL Database	This method assumes that you have user records stored in a SQL database. Create a Steel-Belted Radius `.aut` file that connects to a SQL database and issues a SELECT query based upon the username and password. Give the `.aut` file a unique InitializationString value. Stop and restart Steel-Belted Radius. Subsequently, the SQL authentication method appears in the Authentication Methods tab of the Authentication Policies panel, using the InitializationString value as its name. You can use the Authentication Policies panel to enable, disable, and re-order the SQL authentication method.	"Configuring SQL Authentication" on page 183
External LDAP Database	This method assumes that you have user records stored in an LDAP database. Create a Steel-Belted Radius `.aut` file that validates the username and password based upon `Bind` and `Search` requests to an LDAP database. Give the `.aut` file a unique InitializationString value. Stop and restart Steel-Belted Radius. Subsequently, the LDAP authentication method appears in the Authentication Methods tab of the Authentication Policies panel, using the InitializationString value as its name. You can use the Authentication Policies panel to enable, disable, and re-order the LDAP authentication method as desired	"Configuring LDAP Authentication" on page 208

Two-Factor Authentication

Extensible Authentication Protocol - Tunneled Transport Layer Security (EAP-TTLS) provides for certificate-based mutual authentication between a client and a network through an encrypted tunnel. A typical implementation of EAP-TTLS uses certificates

on authentication servers to create a network-to-user encryption tunnel, and then uses EAP inside the TLS tunnel for user-to-network authentication.

An enhanced version of EAP-TTLS uses certificates on the client side to provide *two-factor authentication*: the end user must have both a private key for a valid certificate and the password to an active account to obtain network access.

When client certificate support in EAP-TTLS is enabled on the server, you must provide a list of trusted root certificate from which offered client certificates must derive. These certificates must be provided in DER-encoded form and must be placed in the root subdirectory of the server directory.

Optionally, you can enable certificate revocation list (CRL) checking as part of the EAP-TTLS authentication process. CRL checking verifies that an unexpired certificate has not been revoked by its issuing Certificate Authority (CA) for any reason, such as a suspected security breach. Enabling CRL checking means that, every time the client requests a connection, Steel-Belted Radius checks the CRL to confirm that the client certificate has not been revoked. This improves security but increases processing overhead.

Note that, if client certificate support is not enabled in EAP-TTLS, any trusted root certificates and CRL checking options are ignored.

Password Protocols

During an authentication transaction, password information is transmitted between the RAS and the RADIUS server. This password information originally comes from the user, for example during PPP negotiations between a user and a RAS. Steel-Belted Radius supports four protocols (PAP, CHAP, MS-CHAP, and MS-CHAP-V2) for receiving the password from the RAS. Steel-Belted Radius also supports the *Extensible Authentication Protocol*.

Table 3 lists supported protocols according to the authentication methods with which each protocol can be used.

***Table 3.** Authentication Methods and Password Protocols*

Method	PAP	CHAP	MS-CHAP	MS-CHAP-V2
LDAP	Yes	**Yes**, if BindName is used and the password is in clear text form or is encrypted with enc-md5	**Yes**, if BindName is used and the password is in clear text form or is encrypted with enc-md5	**Yes**, can return clear-text password or MD4 hash of Unicode form of password.
		No, if Bind is used	No, if Bind is used	No, if Bind is used
Native	Yes	Yes	Yes	Yes
Windows Domain Group	Yes	No	**Yes**, if the user is in local or trusted domain	**Yes**, if the user is in local or trusted domain

Table 3. *Authentication Methods and Password Protocols (Continued)*

Method	PAP	CHAP	MS-CHAP	MS-CHAP-V2
Windows Domain User	Yes	No	**Yes**, if the user is in local or trusted domain	**Yes**, if the user is in local or trusted domain
Proxy RADIUS	Yes	Yes	Yes	Yes
SecurID	Yes	No	No	No
SQL	Yes	**Yes**, if the password is available in clear text form in the database or is encrypted with enc-md5	**Yes**, if the password is available in clear text form in the database or is encrypted with enc-md5	**Yes**, can return clear-text password or MD4 hash of Unicode form of password.
TACACS+	Yes	Yes	No	No
UNIX User	**Yes**	No	No	No
UNIX Group	Yes	No	No	No

Password Authentication Protocol

When the Password Authentication Protocol (PAP) is used, a remote user negotiates with the RAS "in the clear," and no encryption is used to send the password to the RAS. After the RAS has enough information from the user to create an Access-Request, the RAS encrypts the password (using its RADIUS authentication shared secret) before sending an Access-Request packet to Steel-Belted Radius.

Upon receiving the Access-Request, Steel-Belted Radius looks for attributes within the packet that identify the RAS that sent it. Steel-Belted Radius decrypts the password by using the shared secret configured for the RADIUS client entry associated with the sending RAS.

Ultimately, Steel-Belted Radius has the password in clear text form for authentication.

All Steel-Belted Radius authentication methods support PAP.

Challenge Handshake Authentication Protocol

The Challenge Handshake Authentication Protocol (CHAP) avoids sending passwords in clear text over any communication link. Under CHAP, during password negotiations the RAS generates a *challenge* (a random string) and sends it to the user. The user's PPP client creates a *digest* (the password concatenated with the challenge), encrypts the digest using one-way encryption, and sends the digest to the RAS.

The RAS sends this digest as the password in the Access-Request.

Because the encryption is one-way, Steel-Belted Radius cannot recover the password from the digest. Instead, it performs an identical operation, using the RAS's challenge value (provided in the Access-Request packet) and its own copy of the user's password to generate its own digest. If the two digests match, the password is the same.

Steel-Belted Radius must be able to perform the digest operation to support CHAP. Therefore, it must have access to its own copy of the user's password. Native User

passwords are stored in the Steel-Belted Radius database. SQL or LDAP BindName authentication retrieves the password by means of a query to the database; the retrieved password can be used to create a digest if it is in clear text form. A TACACS+ server provides CHAP support and handles the digest operation itself after Steel-Belted Radius sends the username and password through. No other authentication methods support CHAP at this time.

MS-CHAP and MS-CHAP-V2

The two varieties of MS-CHAP (Microsoft Challenge Handshake Authentication Protocol) are Microsoft authentication protocols that, like CHAP, avoid sending passwords in clear text. Steel-Belted Radius supports both 40-bit and 128-bit MS-CHAP methods. Steel-Belted Radius must be able to perform a digest operation similar to CHAP to support MS-CHAP. Therefore, it must have access to its own copy of the user's password. Native User passwords are stored in the Steel-Belted Radius database. SQL or LDAP BindName authentication retrieves the password by means of a query to the database; the retrieved password can be used to create a digest if it is in clear text form.

MS-CHAP and MS-CHAP-V2 communicate users' requests to change their passwords to a RADIUS server. Steel-Belted Radius supports this feature, although it must also be supported by whatever application the user is using to log in.

MS-CHAP and MS-CHAP-V2 operate in the same way, but they use different attributes. An MS-CHAP client won't accept MS-CHAP-V2 attributes, and vice-versa; be careful to use the appropriate set of attributes.

For more information about MS-CHAP and MS-CHAP-V2, see RFC 2433, *Microsoft PPP CHAP Extensions*; RFC 2548, *Microsoft Vendor-specific RADIUS Attributes*; and RFC 2759, *Microsoft PPP CHAP Extensions, Version 2*.

Tunnels

A *tunnel* is a uniquely secure type of remote connection. A tunnel passes data between a remote site and a home network, providing an additional layer of encrypted protocol "wrapper" around the data. A tunnel offers authentication and encryption features that help secure the connection against network vandals and eavesdroppers. In addition, a tunnel can provide quality of service features such as guaranteed bandwidth.

***NOTE:** Steel-Belted Radius does not add tunnel functionality to your network. Steel-Belted Radius is able to support the authentication and accounting needs of any tunnels that you've already set up.*

Administration and configuration of the tunnel happens at the remote site, since this is the side of the connection that requests remote access and opens the tunnel. An administrator at the remote site must configure the tunnel with various attributes: its destination IP address, what security protocols it supports, its password, and so on. These attributes are stored in a database and retrieved when needed to set up a connection.

Storing tunnel attributes on a RADIUS server simplifies tunnel connections. At connection time, the tunnel is established by a RAS device at the remote site. The RAS retrieves the tunnel configuration attributes from the RADIUS server and uses them to open the tunnel into the enterprise. After the tunnel is open, the user can be authenticated at the enterprise.

A RADIUS server is said to support tunnels if it has the ability to store and retrieve the configuration data that a RAS needs to open a tunnel. Steel-Belted Radius fully supports tunnels:

- Steel-Belted Radius can determine from the attributes in the incoming Access-Request whether the connection request involves a tunnel, and if so, which tunnel.
- Steel-Belted Radius can store and retrieve tunnel configuration data.

Steel-Belted Radius can track the number of tunnels currently in use, compare to a maximum number, and refuse the connection if the number is exceeded.

Centralized Configuration Management

Steel-Belted Radius supports the replication of RADIUS configuration data from a *primary server* to a maximum of 10 *replica servers* within a *replication realm*. Replica servers help balance the load of authentication requests coming in from RADIUS clients, and ensure that authentication services are not interrupted if the primary or other replica servers stops working.

For example, Figure 4 illustrates an environment where RADIUS traffic is load-balanced by configuring each RAS device to authenticate users through a different RADIUS server (solid line). If a RADIUS server becomes unavailable, the RAS can fail over to its backup RADIUS server (dotted line).

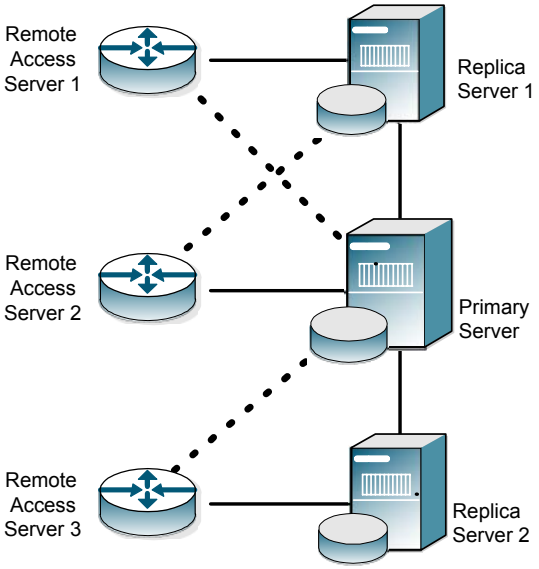


Figure 4 *Using Replication for Redundancy and Load Balancing*

All the servers within a realm reflect the current configuration specified by the network administrator: the network administrator modifies the configuration on the primary server, and the primary server propagates the new configuration to its replica servers. For example, after a network administrator configures a new RADIUS client or profile on the primary server, the network administrator tells the primary server to publish a configuration package file (`replica.ccmpkg`) that contains the updated configuration information. After publication, the primary server notifies each replica server that a new configuration package is ready. Each replica then downloads and installs the configuration package to update its settings.

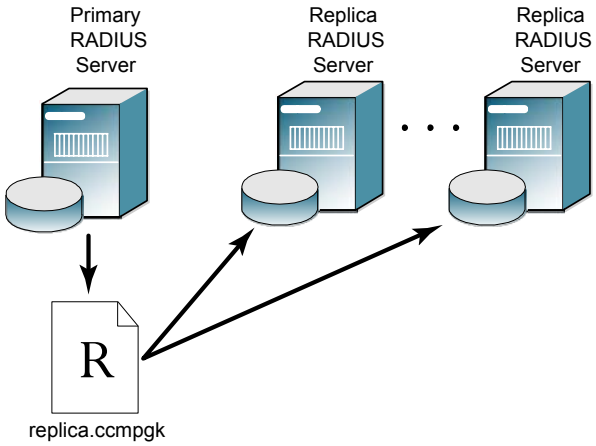


Figure 5 *Replication Package*

The primary server maintains a list of the replica servers that have registered with it. The primary server uses this list to track which servers to notify after it publishes an updated configuration package to resynchronize the configuration of replica servers.

If the primary server needs to be taken out of service, the network administrator promotes one of the replica servers to be the new primary server. Thereafter, the other replica servers copy the configuration package from the promoted primary server.

Accounting

To understand the Steel-Belted Radius accounting sequence, you will need an overview of RADIUS accounting messages. Table 4 explains the conditions under which each type of message is issued, and the purpose of any RADIUS attributes that a message contains.

Table 4. *Message Conditions and Attributes*

Message Conditions	Purpose of Message Attributes
The RADIUS client sends accounting data to Steel-Belted Radius using an Accounting-Request message. The RADIUS client is responsible for verifying that the server receives accounting requests. Most clients retry periodically until the server responds.	Depending on the value of the Acct-Status-Type attribute, the message type is considered to be Start, Stop, Interim-Acct, Accounting-On, or Accounting-Off.
Upon receipt of an Accounting-Request message, the server sends an Accounting-Response.	Complete the request/response cycle.
After receiving an Access-Accept from the server, the RAS completes its access negotiation with the user. The RAS then sends a Start message to the server.	Record connection data, such as username, RAS identifier, RAS port identifier, port type, and connection start time.
At intervals of approximately every six minutes, the RAS sends an Interim-Acct message to the server.	Record a “snapshot” of statistics regarding the connection. One message contains the current value of every statistic that this RAS is capable of recording about this type of connection.
After a connection is terminated, the RAS sends a Stop message to the server.	Record statistics regarding the connection. One message contains the final value of every statistic that this RAS is capable of recording about this type of connection.
Every time a client device comes online, whether after a crash or after an orderly shutdown, it sends an Accounting-On message to the server.	Identify the device that is going online and clear all session information.
Every time a client device experiences an orderly shutdown, before completing its shutdown sequence it sends an Accounting-Off message to the server.	Identify the device that is going offline and clear all session information.

Accounting Sequence

A RAS can issue an Accounting-Request whenever it chooses, for example upon establishing a successful connection. Each time an Accounting-Request message reaches

Steel-Belted Radius, an accounting transaction begins. During this transaction, the server handles the message by examining the Acct-Status-Type and other attributes within the message, and taking the appropriate action.

Comma-Delimited Log Files

When the Steel-Belted Radius accounting log is enabled, all of the RADIUS accounting attributes that the server receives are reformatted and logged to a comma-separated value (CSV) text file, which is easily imported into spreadsheets and database programs for report generation and billing.

External Accounting

See "About SQL Accounting" on page 193.

External accounting methods permit Steel-Belted Radius to record accounting data to external databases. Configuration files specify how Steel-Belted Radius communicates with an external database and how to insert accounting data into that database.

SQL is the only external accounting method currently supported by Steel-Belted Radius.

Tunneled Accounting

During authentication, a user is typically identified by attributes such as User-Name (in the authentication request) and Class (in the authentication accept response). Standard RADIUS accounting requests typically include these attributes in messages flagging Start, Interim, and Stop events so that the user's identity can be recorded for accounting and auditing purposes.

When an organization uses a tunneled authentication protocol such as EAP/TTLS or EAP/PEAP, the identity of a user requesting authentication may be concealed from the RAS; the User-Name attribute carried by the outer authentication protocol is typically a non-unique value such as "anonymous." As a result, the outer User-Name value included in accounting requests may not be sufficient to determine a user's identity. Class attributes provided by an authentication server cannot be included in cleartext in an outer Access-Accept message because they might contain clues about the user's identity, thereby defeating the identity-hiding feature of the tunneled protocol.

Tunneled accounting allows Steel-Belted Radius to pass user identity information to accounting processes without exposing user identities to a RAS or AP that should not see them. When tunneled accounting is enabled, RADIUS attributes are encrypted and encapsulated in a Class attribute. If the information for a Class attribute exceeds the attribute payload size (253 octets), Steel-Belted Radius returns more than one Class attribute for a user.

The tunneled accounting transaction sequence is:

1 The Steel-Belted Radius server acting as the tunnel endpoint for EAP/TTLS or EAP/PEAP encrypts a user's inner User-Name and Class attributes when it authenticates the user.

2 The server returns the encrypted information to the RAS or AP encapsulated in a Class attribute in the outer Access-Accept message. The RAS or AP associates this encapsulated identity attribute with the user, and echoes the encapsulated identity attribute whenever it generates an accounting request for the user.

3 When Steel-Belted Radius receives an accounting request from a RAS or Access Point, it scans the request for an encapsulated identity attribute.

4 If Steel-Belted Radius finds an encapsulated identity attribute, it de-encapsulates and decrypts the attributes to reconstitute the original inner User-Name and Class attributes.

5 Steel-Belted Radius substitutes the decrypted attributes for the ones returned from the RAS or AP.

6 Steel-Belted Radius processes the accounting request locally or forwards the accounting request through the proxy to its intended target.

For an overview of how EAP/TTLS and EAP/PEAP work, refer to “EAP Types” on page 118.

To implement tunneled accounting, you must configure the `classmap.ini` file to specify how attributes should be presented, and you must configure the `spi.ini` file to specify the keys that are used to encrypt and decrypt users’ identity information. The `classmap.ini` file and the `spi.ini` file are described in the *Steel-Belted Radius Reference Guide*.

Attributes

You work with RADIUS attributes while setting up users, profiles, and RADIUS clients in Steel-Belted Radius. The SBR Administrator program lets you select RADIUS attributes by name from a predefined list. For each attribute, the SBR Administrator prompts you to enter values using familiar data types such as string, integer, telephone number, or network address.

This section provides background information you need to work with attributes in Steel-Belted Radius.

Dictionaries

Steel-Belted Radius uses *dictionary files* to store lists of RADIUS attributes. Steel-Belted Radius uses these dictionaries to parse authentication/accounting requests and generate responses.

The main Steel-Belted Radius dictionary file (`radius.dct`) lists attributes defined by the RADIUS standard. The `radius.dct` file resides in the same directory as the Steel-Belted Radius service (usually `C:\RADIUS\Service` on Windows computers or *`radiusdir`* on Solaris/Linux computers).

Vendor-Specific Attributes

In addition to the standard attributes, many RASs use vendor-specific attributes (VSAs) to complete a connection. Steel-Belted Radius supports a large number of specific RAS devices by providing vendor-specific, proprietary dictionary files. These files also reside in the server directory and use the filename extension `.dct`.

Make/Model Field

During Steel-Belted Radius configuration, when you make a selection in the RADIUS client **Make/model** field, you are telling the server which dictionary file contains the VSAs for this client device. Thereafter, whenever the server receives a RADIUS packet from this client device, it can consult this dictionary file for any non-standard attributes that it encounters in the packet. Standard RADIUS attributes are always defined by the `radius.dct` file. If you do not know the make and model of a RADIUS client, choose **- Standard RADIUS -**.

For the most part, the selections currently available in the **Make/model** field are devices whose vendors have provided attribute dictionaries for use with Steel-Belted Radius.

Updating Attribute Information

If your RAS vendor announces a new product, a new attribute, or a new value for an attribute, you can add this information to your Steel-Belted Radius configuration. You can edit the dictionary file for that vendor to add new attributes or attribute values, or you can create a new vendor-specific dictionary file that contains new attributes and values. For more information on dictionary files, refer to the *Steel-Belted Radius Reference Guide*.

User Attribute Lists

Each User entry in the Steel-Belted Radius database provides the information necessary for the server to try to authenticate a connection request using a specific authentication method. When you view a User entry using the Administrator program, this method is identified in the **User type** field.

You can control authentication and authorization at finer levels of detail than simple username/password checking allow. The checklist, return list, or profile fields in user entries provide powerful tools for the authentication and authorization of users. These fields tell the server how to handle RADIUS attributes while authenticating a connection request, and can be used to configure the authorization of the session.

Checklist Attributes

A checklist is a set of attributes that must accompany a connection request before the request can be authenticated. The RAS must send attributes that match the checklist associated with a user entry; otherwise, Steel-Belted Radius rejects the user even if the user's name and password are valid.

By including appropriate attributes in the checklist, a variety of rules can be enforced. For example, only specific users might be permitted to use ISDN or dial-in connections to a particular RAS, or Caller ID might be used to validate a user against a list of acceptable originating telephone numbers.

A checklist is created by selecting attributes from a list of all RADIUS attributes known to Steel-Belted Radius. This list can include a variety of vendor-specific attributes.

During authentication, Steel-Belted Radius filters the checklist based on the dictionary for the RADIUS client that sent the authentication request. The server ignores any checklist attribute that is not valid for this device.

Return List Attributes

A *return list* is a set of attributes that Steel-Belted Radius must return to the RAS after authentication succeeds. The return list usually provides additional parameters that the RAS needs to complete the connection, typically as part of PPP negotiations. Return list attributes can thus be considered to be "authorization configuration parameters."

By including appropriate attributes in the return list, you can create a variety of connection policies. Specific users can be assigned particular IP addresses or IPX network numbers; IP header compression can be turned on or off; or a time limit can be assigned to the connection.

You create a return list by selecting attributes from a list of all RADIUS attributes known to Steel-Belted Radius. This list can include a variety of vendor-specific attributes.

During authentication, Steel-Belted Radius filters the return list based on the dictionary for the RADIUS client that sent the authentication request. The server omits any return list attribute that is not valid for this device.

Attribute Values

The value of each RADIUS attribute has a well-defined data type: numeric, string, list, IP or IPX address, time, or hexadecimal. For example, `Callback-Number` is of type `string` and contains a telephone number. `RAS-Port-Type` is an item from a list, and can be `Sync`, `Async`, and so forth.

NOTE: *Steel-Belted Radius supports signed integers (negative numbers) for attributes received in packets and processing relating to those attributes. However, SBR Administrator does not support signed integers, and treats signed and unsigned integers as unsigned integers.*

Multi-Valued Attributes

Attributes can be single- or multi-valued. Single-valued attributes appear at most once in the checklist or return list; multi-valued attributes may appear several times.

If an attribute appears more than once in the checklist, this means that any one of the values is valid. For example, you can set up a checklist to include both `Sync` and `Async` values for attribute `RAS-Port-Type`. This means that the user can dial into a Sync port or an Async port, but not one of the ISDN ports.

If an attribute appears more than once in the return list, each value of the attribute is sent as part of the response packet. For example, to enable both IP and IPX header compression for a user, the `Framed-Compression` attribute appears twice in the return list: once with the value `VJ-TCP-IP-header-compression` and once with the value `IPX-header-compression`.

Orderable Attributes

Certain multi-valued return list attributes are also orderable; that is, the attribute can appear more than once in a RADIUS response, and the order in which the attributes appear is significant.

For example, the `Reply-Message` attribute allows text messages to be sent back to the user for display. A multi-line message is sent by including this attribute multiple times in the return list, with each line of the message in its proper sequence.

System Assigned Values

Some attributes do not allow the administrator to set a value. Steel-Belted Radius retrieves the appropriate value for this attribute when it is needed.

Echo Property

Using the echo property, you can force an attribute from the RADIUS request to be echoed in the RADIUS response. For example, you might add `Callback-Number` to the return list and select the **echo** checkbox. Steel-Belted Radius takes the value of the Callback-Number it receives in the RADIUS request and echoes it back to the client in the RADIUS response; if it receives no Callback-Number, it echoes nothing.

You enter `Callback-Number` one or more times into the checklist. This indicates that one of the callback numbers you supplied must be present in the RADIUS request, and that number should be echoed in the RADIUS response.

Default Values

Selecting **default** for a checklist attribute specifies that, if the RADIUS request does not include this attribute, the request should not be rejected. Instead, the value supplied as the default should be used as if it were received as part of the request. One use for default values is to require that an attribute in a RADIUS request must have one of several values, or must not be present at all. Another use is to provide a default value for an attribute in conjunction with the echo property in the return list.

Steel-Belted Radius can provide alternate values when an attribute appears in the checklist marked as **default**, and the same attribute appears in the return list marked as **echo**. The server echoes the actual value of the attribute in the RADIUS response if the attribute appears in the RADIUS request and echoes the default value (from the checklist) in the response if the attribute does not appear in the RADIUS request.

If you add multiple values of the same attribute to the checklist, only one of them can be marked as default.

For example, an administrator adds several Callback-Number values to the checklist and marks one of them as default. The administrator adds `Callback-Number` to the return list and specifies it as echo.

- If a Callback-Number value is present in the RADIUS request, it must match one of the checklist values or the user is rejected.

- If it does match, the user is accepted and the value supplied is echoed in the RADIUS response.
- If no `Callback-Number` is supplied in the request, the user is accepted and the default value is echoed in the response.

Other checklist attributes are used to provide configuration for the user, such as time-of-day and concurrent-login-limit information.

Proxy RADIUS

Steel-Belted Radius can forward a RADIUS request to another server for processing and relay the other server's result back to its client. Steel-Belted Radius is acting as a *proxy* for the *target server*, and that Steel-Belted Radius is *proxy-forwarding* the request to the target server.

Any Steel-Belted Radius server can act as proxy or target for authentication or accounting messages (or both).

Proxy RADIUS Authentication

Figure 6 illustrates how RADIUS authentication messages are proxy-forwarded:

1 A RADIUS client sends an authentication request to a RADIUS proxy server.

2 The proxy RADIUS server forwards the message to a RADIUS target server.

3 The target RADIUS server performs the authentication services indicated by the message, then returns a response message to the proxy RADIUS server.

4 The proxy RADIUS server relays the response message to the RADIUS client.

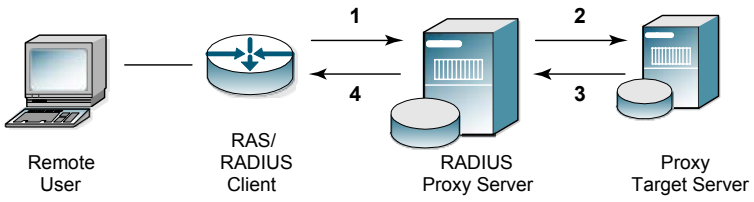


***Figure 6** RADIUS Proxy Forwarding*

Proxy RADIUS Accounting

RADIUS accounting messages are proxy-forwarded as follows:

1 A RADIUS server receives an accounting request.

2 Depending on its configuration, the RADIUS server forwards the accounting message to a target server, records accounting attributes locally on the proxy server, or records the information in both places.

3 If the proxy server does not receive an acknowledgement of the forwarded packet, it periodically re-sends the packet according to its retry policy.

RADIUS Client Groups

To simplify large-scale deployment of RADIUS clients, you can set up a RADIUS *client group* instead of multiple individual RADIUS clients. A RADIUS client group uses a contiguous range of IP addresses instead of a specific IP address. When the Steel-Belted Radius server receives a RADIUS request from an IP address that belongs to the range assigned to a client group, it

If your RADIUS clients use the same RADIUS attributes and have contiguous IP addresses, you can configure one or more RADIUS client groups and specify an address range consisting of as many as 500 IP addresses for each client group. When Steel-Belted Radius receives a RADIUS request that includes a source IP address in this range, it uses the RADIUS client group to determine the appropriate shared secret, make/model, and IP address pool.

Please note the following when you set up address ranges for RADIUS client groups:

- Address ranges are for IPv4 networks only. Steel-Belted Radius does not support address ranges for IPv6 or IPX.
- The address range assigned to one RADIUS client group cannot overlap the address ranges assigned to other RADIUS client groups.
- The starting address of the address range assigned to a RADIUS client group cannot match the IP address of an individual RADIUS client.
- If an individual RADIUS client entry has an IP address that falls within an address range assigned to a RADIUS client group, Steel-Belted RADIUS uses the make/model for the individual RADIUS client. For example, if RADIUS client RAS1 is configured with IP address 192.168.21.55 and if RADIUS client group BLDG1RAS is configured with an IP address range 192.168.21.50–192.168.21.60, Steel-Belted Radius uses the client information for RAS1 if it receives a RADIUS request from 192.168.21.55, and it uses the client information for BLDG1RAS if it receives a RADIUS request from 192.168.21.56.
- A RADIUS client group cannot use a Class D, E, or F IP address (that is, an address greater than 223.255.255.0).

See "Adding a RADIUS Client or Client Group" on page 54 for information on how to configure IPv4 address ranges for RADIUS clients.

IP Address Assignment

Steel-Belted Radius can assign IP addresses to users in several ways:

- Static assignment – The same IP address is assigned to a user each time the user connects. For example, if the user `Kevin` has a `Framed-IP-Address` attribute set to `123.11.245.123`, then the IP address 123.11.245.123 is assigned each time Kevin connects to the network.
- Assignment from a specific address pool – An address is assigned from a specific pool when the user connects. For example, if user `Kevin` has a `Framed-IP-Address` attribute set to the `Sales` IP address pool, the next available IP address from `Sales` is assigned when Kevin connects to the network.
- Assignment from the RADIUS client's IP address pool (or set of IP address pools) – An address is assigned from one of the pools associated with the RADIUS client that makes the connection when a user connect. For example, assume that a RADIUS client called `RAS1` uses IP address pool A and a RADIUS client called `RAS2` uses IP address pool B. A User entry called `Kevin` has a `Framed-IP-Address` attribute value of `pool associated with RADIUS Client`. When user Kevin gets a port on `RAS1`, an IP address from pool A is assigned. On the next call, Kevin might connect to `RAS2`; in this case an address from pool B is assigned.

  Alternatively, if a user has been associated with a particular RAS-specific IP address pool (and suffix), an IP address from that pool is assigned.

Address Pools and Replication

Address pool information is not distributed with other configuration information in a replicated environment. If you are using IPv4 or IPX address pools in a replication environment, you must configure address pools separately on each replica server, and then use the same names to configure a master list of address pools on your primary server.

The master list of address pools configured on the primary server must include the names of all the pools on all of the replica servers. For example, Figure 7 illustrates a simple environment that uses four address pools. POOL1 and POOL2 are configured on one replica server and POOL4 is configured on a different replica server. As a consequence, the IP address pool list on the primary server must include POOL1, POOL2, POOL3 (the pool used by the primary server), and POOL4.

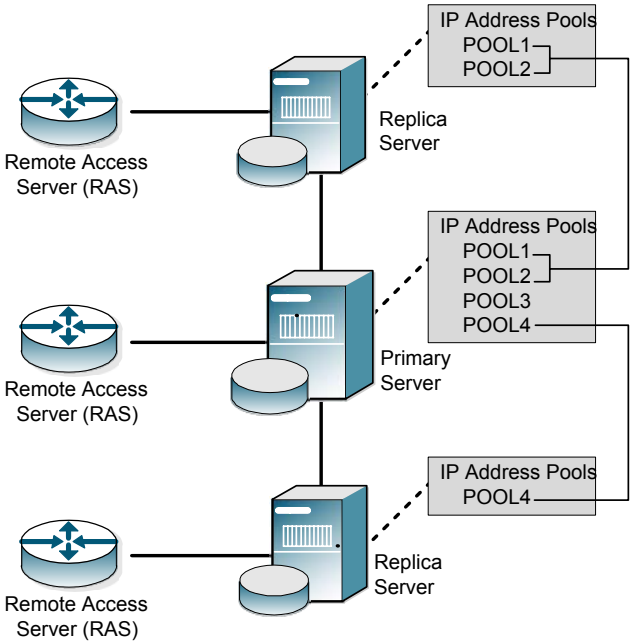


***Figure 7** IP Address Pools in a Replication Environment*

The network administrator must configure RADIUS clients (including the address pool associated with a RADIUS client) on the primary server. This RADIUS client/address pool association (but not the address pool information itself) is stored as part of the replication package passed from the primary server to the replica servers.

Hints

Steel-Belted Radius can treat the attribute `Framed-IP-Address` as a *hint*. This means that if this attribute appears in the Access-Request and the user return list is configured to allocate `Framed-IP-Address` from a pool, the IP address in the Access-Request is returned instead of the newly-allocated IP address.

This functionality is defined in the [Configuration] section of `radius.ini`:

```
[Configuration]
FramedIPAddressHint = <yes/no>
```

When hints are enabled, Steel-Belted Radius uses a hint to determine the value of the Framed-IP-Address attribute in the access response. This means that `Framed-IP-Address` in the Access-Request is returned in the Access-Accept, regardless of the `Framed-IP-Address` value stored in the user's account.

The default value is `no`.

Table 5 details the effect of hints:

***Table 5.** Effect of Hints*

Account Configuration	Framed-IP-Address returned without hints	Framed-IP-Address returned with hints
No `Framed-IP-Address`	No value	`Framed-IP-Address` from Access-Request
Static Address	Static address	Static address
Address from Pool	Next address from pool	`Framed-IP-Address` from Access-Request

***NOTE:** By using hints, you can assign the same IP address to multiple active accounts.*

Resource Management

This section explains how Steel-Belted Radius manages limited resources, such as network addresses, user or tunnel connections, and UDP ports.

Network Address Assignment

The Steel-Belted Radius address pooling feature allows you to set up one or more pools out of which unique network addresses are assigned dynamically as users require them. Each pool consists of a list of one or more ranges of IP addresses (an IP pool) or IPX network numbers (an IPX pool).

By using this feature, you can avoid allocating specific fixed addresses to individual users. You can make fewer addresses go farther, and you can consolidate address assignment across all your RAS devices.

How Address Assignment Works

Proper operation of address assignment from a pool depends crucially on both RADIUS authentication and RADIUS accounting transactions, as follows:

1 During the RADIUS authentication transaction, if the user's attribute settings specify address assignment from a pool, an address is allocated for that user from that pool.

2 The address is reserved for that user until a RADIUS accounting transaction indicates that the user has terminated the connection.

For this reason, the RAS device must be configured for RADIUS accounting, and the same Steel-Belted Radius server must be specified for both authentication and accounting. If your RAS is not configured for accounting (or does not support accounting), you cannot use the address pooling feature because addresses would be assigned but never released.

Setting Return List Attributes

The `Framed-IP-Address` (or `Framed-IPX-Address`) return list attribute controls how the user's IP (or IPX) address is assigned. The `Framed-IP-Address` or `Framed-IPX-Address` attribute can be set for each user in the Steel-Belted Radius database.

Handling Address Leaks

Under optimal conditions, Steel-Belted Radius assigns and releases addresses automatically. In some circumstances, you can get *address leakage*, where an address remains reserved for a user after the user has terminated the connection.

Address leakage occurs when the address has been assigned during the authentication transaction, but the accounting transaction that would have released the address is never received by Steel-Belted Radius. This can occur for several reasons:

- The Steel-Belted Radius server might have been taken down for a period of time during which accounting transactions occurred.
- The RAS device might have failed or been taken down before the user terminated. (In many cases, however, Steel-Belted Radius might be able to prevent address leakage by recovering the addresses when the RAS starts up again.)
- The RAS might have sent the authentication and accounting transactions to different RADIUS servers.
- Despite a successful authentication, the user's PPP negotiation with the RAS might have terminated unsuccessfully for a variety of reasons. In such a case, some RAS devices might not initiate a subsequent accounting transaction.
- Routing problems might have prevented the accounting transaction from reaching Steel-Belted Radius.

An address that has "leaked" remains out of circulation until you manually release it by displaying the Sessions list and deleting the corresponding session. See "Deleting Entries from the Sessions List" on page 228.

Address Leakage Upon Stopping and Starting the Server

Steel-Belted Radius maintains all current address assignments in a persistent database on disk. If you shut down the server and then restart it, all the information about which address is assigned to which user is retained.

Note that if you leave Steel-Belted Radius turned off for a substantial period of time after addresses have been assigned, address leakage may occur. After you restart the server, review the Current Sessions list (described in "Displaying the Current Sessions List" on page 226) and delete entries you know are obsolete.

Overlapping Address Ranges

If you maintain multiple IP or IPX address pools, you can duplicate some of the addresses among the pools. The address tracking mechanism of Steel-Belted Radius, when it is enabled, ensures that, if an IP address appears in more than one pool, after it

is assigned out of any pool, it remains unavailable through any of the pools until it is released.

You must disable this type of address tracking if the server is assigning IP addresses from disjoint networks. In that configuration, two numerically identical IP addresses would signal a conflict, even though they actually belong to two different networks.

Order of Address Assignment

IP or IPX addresses are assigned on a first-in-first-out basis; that is, the address that was first released is the first to be reassigned. This ensures that addresses are out of use for as long as possible prior to reuse.

Concurrent Network Connections

The SBR Administrator program allows you to limit the number of active connections, on a per-user, or per-tunnel basis.

Concurrent User Connections

You can set a maximum limit on the number of concurrent connections that a user can have. Subsequently, when the user requests a new connection, Steel-Belted Radius compares the current number of connections to the maximum limit.

If a new connection would exceed the limit, Steel-Belted Radius can reject the additional connection or allow the connection, but log the event in the server log (described in "Using the Server Log File" on page 233).

***NOTE:** When counting connections, Steel-Belted Radius does not distinguish between multi-link connections and new user authentication attempts.*

For concurrent connection limits to work, each RAS must be configured for RADIUS accounting and the same Steel-Belted Radius server must be responsible for both authentication and accounting. These conventions give the server full access to the data it needs to track connections accurately.

The maximum number of concurrent connections can be set individually for any type of user by selecting the **Maximum Concurrent Connections** checkbox and entering a number in the accompanying field in the appropriate Add User/Edit User window. See Chapter 5, "Administering Users" on page 59, especially "Concurrent Connection Limits" on page 78.

When a concurrent connection limit is set up for a user, it affects only that user. When a concurrent connection limit is set up for a group, every member of the group receives the same connection limit. For example, if GroupA has a connection limit of two, then each member of the group can have two concurrent connections.

Authentication methods that do not require user entries must provide alternate mechanisms for supporting concurrent connection limits. For example, if you are using external database authentication there is an alias mechanism you can use in the SQL or LDAP configuration file. Concurrent connection limits can be supported under proxy authentication only if the target server supports them.

NOTE: *Concurrent user connections can be tracked across multiple Steel-Belted Radius servers by adding the Concurrency Server package.*

Concurrent Tunnel Connections

Steel-Belted Radius uses its Current Sessions list to determine the number of active connections for each tunnel. The Sessions list summarizes all of the RADIUS accounting data currently available to the server. Tunnel connections appear in the Sessions list using a special display convention that distinguishes them from user connections.

You can set a maximum limit on the number of concurrent connections that can be open using a specific tunnel. Subsequently, when a user requests a new connection through that tunnel, Steel-Belted Radius compares the current number of connections to the maximum limit. If a new connection would exceed the limit, Steel-Belted Radius rejects the additional connection.

For concurrent connection limits to work, it is essential that each RAS that can open a tunnel be configured for RADIUS accounting and that the same Steel-Belted Radius server be specified for both authentication and accounting. This permits the server's Sessions list to be kept up to date and available to every RAS that needs to authenticate tunnel connections.

NOTE: *Concurrent tunnel connections cannot be tracked across multiple Steel-Belted Radius servers without additional software extensions. Contact Funk Software for more information.*

Phantom Records

When Steel-Belted Radius allocates resources such as IP addresses, IPX addresses, user connections, and tunnel connections, to its clients, it generates a *phantom* accounting record for its internal use. Phantom records are not written to the RADIUS accounting database, but they are displayed in the Current Sessions list (described on page 226). Phantom records resemble accounting start records, except that the session ID for phantom records is displayed as `N/A`.

After Steel-Belted Radius receives the corresponding accounting start request packet from the client, it discards the phantom record and replaces `N/A` with the actual `Session-ID` number returned by the client device in the Current Sessions list.

In some cases, Steel-Belted Radius can allocate a resource and create a phantom record, but then never receive a corresponding start packet from the client. To avoid committing the resource indefinitely, Steel-Belted Radius waits for a limited period for the start packet to confirm the transaction. The default time that Steel-Belted Radius waits is 180 seconds. You can configure this time by editing the `radius.ini` file, which is described in detail in the *Steel-Belted Radius Reference Guide*.

File Permissions for Log Files (Solaris/Linux)

When you run Steel-Belted Radius on a Solaris or Linux server, you can specify the users who are authorized to read or edit important files, such as authentication and accounting log files. For example, you can specify that system administrators who install and configure Steel-Belted Radius have read/write access for system log files and that network operators who monitor Steel-Belted Radius have read-only (or no) access for system log files.

Security Groups and Permissions

Each file and directory on a Solaris or Linux server has three security groups associated with it:

- The Owner identifies the person who created or owns the file.
- The Group security group identifies the set of users who are members of the group or groups to which the file Owner belongs. Group members can exercise special privileges with respect to that file. A user can belong to more than one group.
- The Other security group consists of the set of all users who do not belong to Owner or Group.

Each security group has three flags that control what privileges that group can exercise with respect to the file or directory:

- The Read flag (r) determines whether the file can be read. The Read flag has an octal value of 4.
- The Write flag (w) determines whether the security group can create, modify, or delete the file. The Write flag has an octal value of 2.
- The Execute flag (x) determines whether the security group can run a script or executable file. The Execute flag has an octal value of 1.

For example, a file owner might have `rwx` permission for a file, which indicates the file owner has read/write/execute access to the file. Similarly, Other might have `r--` permission (where - indicates no permission), which means that the user can read but not edit or execute the file.

You can add the octal values for permission flags to generate a numeric representation of the file permissions for Owner, Group, and Other:

- 1 = execute only
- 2 = write only
- 3 = write and execute (1+2)
- 4 = read only
- 5 = read and execute (4+1)
- 6 = read and write (4+2)
- 7 = read and write and execute (4+2+1)

The security permissions exercised by Owner/Group/Other are typically expressed as string or a three-digit number. Table 6 provides examples of different file permissions.

Table 6. *File Permissions*

Permission	Octal value	What It Means
`-rwxrwxrwx`	777	Read, write and executable for Owner/Group/Other
`-rw-rw-r--`	664	Read and write for Owner/Group; read access for Other
`-rw-rw----`	660	Read and write for Owner/Group; no access for Other
`-rwx------`	700	Read, write and executable for owner only
`-rw-rw-rw`	666	Read and write for owner, group and all others

The UNIX `chown` command lets you change the owner or group (or both) associated with a file or directory. The UNIX `chmod` command lets you change the permissions of files and directories.

Using the User File Creation Mode Mask

The user file mode creation mode mask (often abbreviated as `umask`) determines the default file system mode for newly created files of the current process. Solaris/Linux hosts typically have a hierarchy of `umask` values: a server-level `umask` value, which can be overridden by a user-, shell-, or application-level `umask` value. The result is an *ambient umask value*, which determines what file permissions are used when files are created by any given process.

The `umask` value is a three-digit octal number. The first digit sets the mask for Owner, the second for Group, and the third for Other. The `umask` value identifies the permissions that are *withheld* when a file is created: the `umask` value is subtracted from the full access mode value (777) to determine the access permissions for a new file. For example, if the `umask` value for a process is set to 022, the Write permission for Group and Other are withheld from the full access mode value (777), resulting in a file permission of 755 (`rwxr-xr-x`). Similarly, if the `umask` value of 177 is configured for a process (explicitly or by virtue of the ambient umask), files created by the process have a file permission of 600 (`rw-------`). Table 7 summarizes the result of using different octal numbers in a `umask` value.

Table 7. *Summary of umask Permissions*

Octal Number	Access	Permission Resulting From umask Value
0	rwx	Read, Write, Execute
1	rw-	Read, Write
2	r-x	Read, Execute only
3	r--	Read only
4	-wx	Write, Execute only
5	-w-	Write only
6	--x	Execute only
7	---	No permissions

The umask value affects a file's access permissions only when the file is created. If you change the umask value, access permissions for existing files are not affected. Similarly, you can use the chown and chmod commands to change a file's access permissions after the file has been created.

Implementing Default File Permissions in Steel-Belted Radius

The RADIUSMASK parameter in the sbrd.conf file specifies the application-level umask value used to establish access permissions for all files created by Steel-Belted Radius. Refer to the *Steel-Belted Radius Reference Guide* for information on configuring the sbrd.conf file.

If you do not specify a value for the RADIUSMASK parameter, Steel-Belted Radius uses the ambient umask value established by the server-, user- or shell-level umask value to determine the access permissions for files it creates.

Some log files have explicit controls that let you override the umask value established by the RADIUSMASK parameter or the ambient umask value. See "Implementing Override File Permissions in Steel-Belted Radius" for more information on overriding the application-level default umask value.

As previously noted, the umask value affects a file's access permissions only when the file is created. If you change the RADIUSMASK setting, new files created by Steel-Belted Radius are assigned the access permission specified by the new setting. This includes files that roll over periodically; the existing file would retain the access file permission it received when it was created, and the new file would be assigned the access permission specified by the new RADIUSMASK value.

***NOTE:** The Execute file permission value for files created by Steel-Belted Radius is always set to None for Owner, Group, and Other. Thus, a* umask *value of 0 (no restrictions) is equivalent to a umask value of 1 (read/write permission) for files created by Steel-Belted Radius.*

Implementing Override File Permissions in Steel-Belted Radius

To override file permissions established by the Steel-Belted Radius RADIUSMASK or the ambient umask for specific log files, you must modify the LogFilePermissions parameter in the applicable initialization (.ini) file.

Table 8 identifies the configuration file you must modify to configure non-default file permissions for Steel-Belted Radius log files.

***Table 8.** Configuration Files for Setting Log File Permissions*

Controlled Files	Configuration File
Server Diagnostics log (RADIUS log)	radius.ini
Authentication Reporting Library accepts log	authReportAccept.ini
Authentication Reporting Library bad shared secret log	authReportBadSharedSecret.ini
Authentication Reporting Library rejects log	authReportReject.ini

Table 8. Configuration Files for Setting Log File Permissions (Continued)

Controlled Files	Configuration File
Authentication Reporting Library unknown client log	`authReportUnknownClient.ini`
Authentication Logging Library logs and header check-point logs	`authlog.ini`
Accounting Library logs and header check-point logs	`account.ini`
Server Statistics logs and header check-point logs	`statlog.ini`

The syntax for the LogFilePermissions parameter is:

```
LogfilePermissions = owner:group mode
```

- Specify the *owner* and *group* settings by entering character strings or decimal integers, as used for arguments to the UNIX `chown(1)` command. For example, `ralphw:proj`, `ralphw:120`, or `1007:120`.
- Specify the *mode* setting as a character string or an octal integer. When permissions are specified as a character string, they follow the format that is used by the UNIX `ls(1)` command; for example, `rw-rw-rw-`. When permissions are specified as an octal integer, they follow the format used for arguments to the UNIX `chmod(1)` command; for example, `666`.

***NOTE:** You can specify only read/write permissions for a Steel-Belted Radius file. You cannot specify execute permissions for Steel-Belted Radius files.*

The value of each `LogFilePermissions` parameter is read when the Steel-Belted Radius server is started or restarted. The value of the LogFilePermissions parameter in the `radius.ini` file is also read when you issue a HUP command to the Steel-Belted Radius server.

- If you enter a valid value for a `LogfilePermissions` parameter, the ownership and permissions of the controlled log file are set as specified whenever the file is opened or created.
- If you do not enter a value for a `LogfilePermissions` parameter, the ownership and permissions of the controlled file are not changed. The controlled file is created using the ownership of the account that is executing the server and the permissions that are derived the default RADIUSMASK value or from the ambient `umask` setting. If the file already exists, new information is appended without changing the existing ownership and permissions of the controlled file.
- If you enter an invalid value for a `LogfilePermissions` setting, then the ownership of the controlled log file defaults to the effective user/group ID of the server process (normally `root:other` on Solaris and `root:root` on Linux), and the permissions for the controlled file default to `0600` (`-rw-------`). This ensures that the affected log file can always be opened without any escalation of file access privileges. Messages similar to the following are logged whenever an explicit file access control is misconfigured:

```
Invalid LogfilePermissions specified in radius.ini
[Configuration]: -rwx------
```

```
Server log file permissions defaulted to 0:0 0600
```

Chapter 3
Using SBR Administrator

The SBR Administrator is a Java-based application that lets you configure settings for Steel-Belted Radius. In minutes, you can set up new users, alter standard profiles, or configure new RAS devices from any computer on the network.

This chapter presents an overview of how to use the SBR Administrator.

Running the SBR Administrator

To run the SBR Administrator:

- **Windows**: Double-click the **SBR Administrator** icon or choose **Start > All Programs > Funk > SBR Administrator > SBR Administrator**.
- **Solaris/Linux**: Issue the following command:

  ```
  > radiusdir/radadmin/sbradmin/sbradmin&
  ```

NOTE: *On some Solaris hosts, you may need to execute* `xhost` *to grant permission for the SBR Administrator to be displayed. If you need to set the DISPLAY variable to specify where the SBR Administrator will be displayed, execute the following commands:*

```
> /bin/sh
> DISPLAY=ipaddress:0.0
> export DISPLAY
```

where `ipaddress` *is the IP address of your workstation.*

Logging Into a Server

When you run the SBR Administrator, you start with the Servers panel (Figure 8). You must use the Servers panel to connect to a Steel-Belted Radius server before you can display or use other SBR Administrator panels.

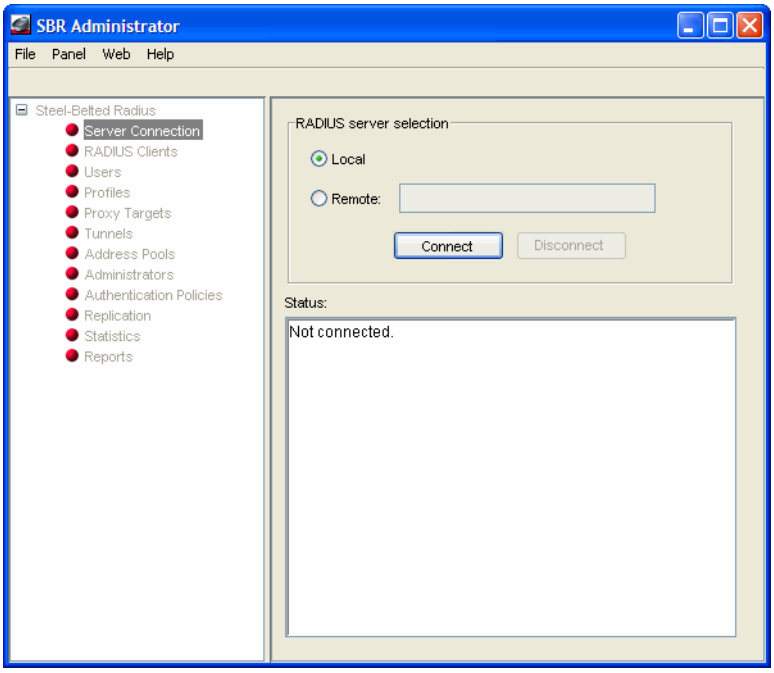


***Figure 8** Servers Panel*

To log into a Steel-Belted Radius server:

1 Identify the server you want to administer.

- ▷ To administer a Steel-Belted Radius server running on your local host, click **Local** and then click **Connect**.
- ▷ To administer a Steel-Belted Radius server running on a remote host, click **Remote**, enter the name or IPv4/IPv6 address of the remote host, and click **Connect**.

When you click **Connect**, SBR Administrator attempts to establish an HTTPS connection with the local or remote server. If it cannot establish a connection in 10 seconds, SBR Administrator times out and displays an error message.

***NOTE:** If a timeout occurs, verify that the Steel-Belted Radius service/daemon is running on the target server and that it is listening on the administration port SBR Administrator is using.*

2 When the Log Into Server window (Figure 9) opens, enter your administrator username and password and click **OK**.

SBR Administrator verifies that the username you entered is present in the `access.ini` file. If the username is found, SBR Administrator validates the password you entered against a local or remote password database.

Figure 9 Log Into Server Window

After you connect to a server, the **Status** pane lists various features of the running server, such as version, platform on which it is running, IP address, available authentication methods, license information, and any initialization errors that might have occurred.

***NOTE:** You do not have to disconnect from one Steel-Belted Radius server before you can open a connection to a different server. When you request a connection to a new server, SBR Administrator automatically terminates your current connection.*

Navigating in SBR Administrator

This section describes how to use the SBR Administrator panels, menus, windows, and toolbar.

SBR Administrator Panels

SBR Administrator uses a series of framed windows (panels) to configure server settings and display statistics and logs in SBR Administrator. Figure 10 illustrates the components of an SBR Administrator panel. To display a panel, you click an entry in the Navigation frame. SBR Administrator displays the specified panel in the Content frame.

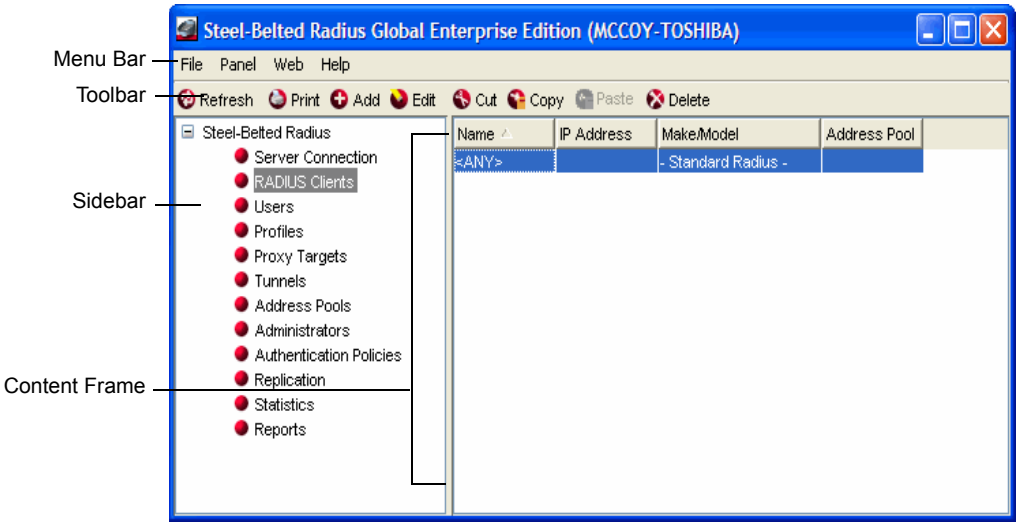
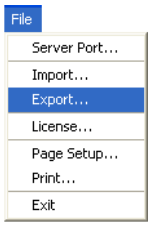


Figure 10 SBR Administrator Panel Layout

SBR Administrator Menus

The main SBR Administrator window has four menus: File, Panel, Web, and Help.

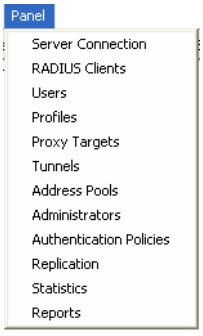


File Menu

Table 9 describes the functions of each entry in the File menu in the SBR Administrator.

Table 9. *File Menu Options*

Menu Entry	Function
Server Port	Specifies the TCP port the SBR Administrator uses to communicate with the Steel-Belted Radius server.
Import	Opens the Import window, which lets you import information from an XML file into the Steel-Belted Radius database. The Import window is described in Appendix E, "Importing and Exporting Data."
Export	Opens the Export window, which lets you export selected information from the Steel-Belted Radius to an XML file. The Export window is described in Appendix E, "Importing and Exporting Data."
License	Opens the Add a License for Server window, which lets you add a license string for your Steel-Belted Server software. For more information, see "Adding License Keys" on page 50.
Page Setup	Opens the Page Setup window, which lets you configure your printer settings.
Print	Prints the information in the active window. When you print the information in a panel, SBR Administrator preserves the column spacing used on screen. If a table is wider than the printed page, pages are printed in a matrix, with pages numbered 1-1, 1-2, 2-1, 2-2, etc.
Exit	Exits the Steel-Belted Radius application.



Panel Menu

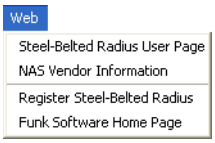
Table 10 describes the functions of each entry in the Panel menu in the SBR Administrator.

Table 10. *Panel Menu Options*

Menu Entry	Function
Servers	Displays the Servers panel in the SBR Administrator window. For more information, see "Logging Into a Server" on page 41.
RADIUS Clients	Displays the RADIUS Clients panel in the SBR Administrator window. For more information, see Chapter 4, "Administering RADIUS Clients" on page 53.
Users	Displays the Users panel in the SBR Administrator window. For more information, see Chapter 5, "Administering Users" on page 59.

Table 10. Panel Menu Options (Continued)

Menu Entry	Function
Profiles	Displays the Profiles panel in the SBR Administrator window. For more information, see Chapter 6, “Administering Profiles” on page 81.
Proxy Targets	Displays the Proxy Targets panel in the SBR Administrator window. For more information, see Chapter 7, “Administering Proxy Targets” on page 85.
Tunnels	Displays the Tunnels panel in the SBR Administrator window. For more information, see Chapter 8, “Administering RADIUS Tunnels” on page 91.
Address Pools	Displays the Address Pools panel in the SBR Administrator window. For more information, see Chapter 9, “Administering Address Pools” on page 99.
Administrators	Displays the Administrators panel in the SBR Administrator window. For more information, see Chapter 10, “Setting Up Administrator Accounts” on page 107.
Authentication Policies	Displays the Authentication Policies panel in the SBR Administrator window. For more information, see Chapter 11, “Setting Up Authentication Policies” on page 111.
Replication	Displays the Replication panel in the SBR Administrator window. For more information, see Chapter 12, “Configuring Replication” on page 141.
Statistics	Displays the Statistics panel in the SBR Administrator window. For more information, see Chapter 17, “Displaying Statistics” on page 217.
Reports	Displays the Reports panel in the SBR Administrator window. For more information, see Chapter 18, “Logging and Reporting” on page 225.

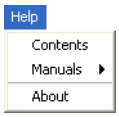


Web Menu

Table 11 describes the functions of each entry in the Web menu in the SBR Administrator.

Table 11. Web Menu Options

Menu Entry	Function
Steel-Belted Radius User Page	Opens the Steel-Belted Radius User web page, which lets you review product information, download documentation and technical notes, and access other resources.
NAS Vendor Information	Opens the RADIUS/AAA Compatibility Guide web page, which lets you review information about remote access devices and wireless LAN devices made by third-party vendors.
Register Steel-Belted Radius	Opens the Register Product web page, which lets you register your copy of Steel-Belted Radius online.
Funk Software Home Page	Opens the Home web page for Funk Software.



Help Menu

Table 12 describes the functions of each entry in the Help menu in the SBR Administrator.

Table 12. *Help Menu Options*

Menu Entry	Function
Contents	Opens the online help for the SBR Administrator.
Manuals	Displays the available Steel-Belted Radius manuals (in PDF format) and release notes (in text format).
About	Displays the About SBR Administrator window, which lists version information for the SBR Administrator. For more information, see "Displaying Version Information" on page 50.

SBR Administrator Toolbar

After you log into Steel-Belted Radius, you can use the toolbar to manipulate SBR Administrator objects. The buttons on the SBR Administrator toolbar change when you change panels to provide buttons appropriate for the current context.

Figure 11 *SBR Administrator Toolbar*

Table 13. *SBR Administrator Toolbar*

Toolbar Button	Function
Refresh	Refreshes the displayed list of items in the SBR Administrator window.
Print	Prints the contents of the active panel.
Add	Adds an object to the Steel-Belted Radius database.
Edit	Edits an existing object in the Steel-Belted Radius database. Active only when an object is selected in the active panel.
Cut	Deletes an existing object from the Steel-Belted Radius database and copies its information to the Clipboard. Active only when an object is selected in the active panel.
Copy	Copies settings for the selected object from the Steel-Belted Radius database to the Clipboard. Active only when an object is selected in the active panel.
Paste	Pastes an object from the Clipboard to the Steel-Belted Radius database. Active only after a Cut or Copy command has been used.
Delete	Deletes an existing object from the Steel-Belted Radius database.
Apply	In the Authentication Policies window, applies any changes you have made to the authentication policy settings.

Table 13. SBR Administrator Toolbar (Continued)

Toolbar Button	Function
Reset	In the Authentication Policies window, discards any changes you have made to the authentication policy settings. In the System Statistics window, resets statistics other than Server Up Time to zero.
EAP Setup (Authentication Policies panel only)	Opens the EAP Setup window, which lets you specify the active EAP methods that will be used for an authentication method. For more information, see "Configuring EAP Settings" on page 138.

SBR Administrator Windows

This section summarizes how to use SBR Administrator windows and controls.

Adding an Entry

To add an entry to the Steel-Belted Radius database, open the appropriate panel and click the **Add** button on the SBR Administrator toolbar. The SBR Administrator displays an Add window (Figure 12).

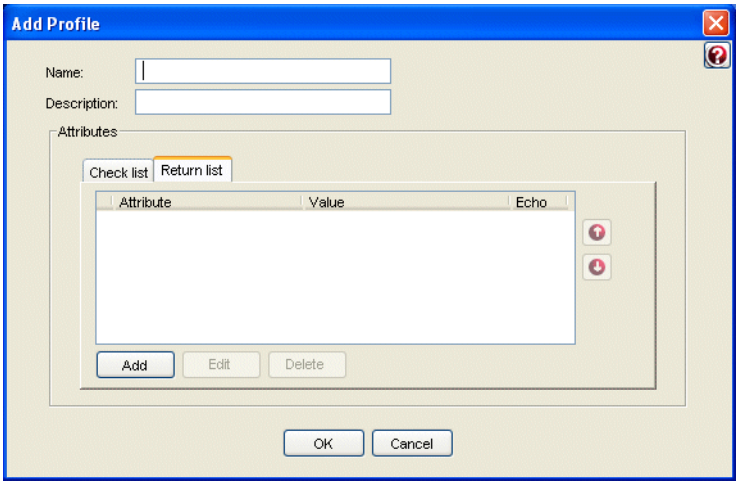


Figure 12 Sample Add Window

Every object of the same type must have a unique name. If the name you assign to an item is already being used by another item of the same type, the SBR Administrator displays a warning.

Editing an Entry

To edit an existing entry to the Steel-Belted Radius database, open the appropriate panel and double-click the item you want to change (or select the item and click the **Edit** button on the SBR Administrator toolbar). The SBR Administrator displays the settings for the item you selected in an Edit window (Figure 13). The **Save** button remains disabled until the contents of a field in the Edit window changes.

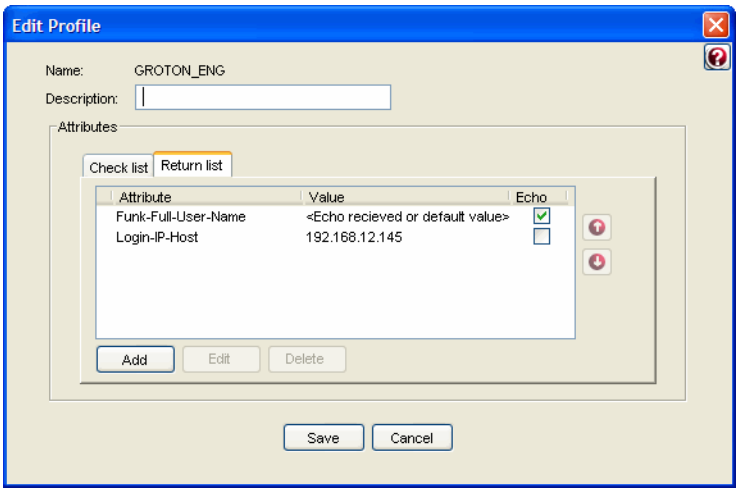


Figure 13 *Sample Edit Window*

NOTE: *You cannot change the name associated with an item in the Edit window. To change an item's name, you must cut/paste the item and assign it the name you want it to have.*

Cutting/Copying/Pasting Records

Panels displaying tables of items have **Cut**, **Copy**, and **Paste** buttons in the toolbar. You can select an item from the display and cut or copy it to the Clipboard, and then add a new record to the display by pasting it from the Clipboard.

The Clipboard can contain one item of each type (RADIUS client, user, etc.) If you copy an item to the Clipboard and then copy another item of the same type, the information for the second item overwrites the information for the first item. Clipboard contents are preserved until you exit the SBR Administrator.

When you paste an item, the SBR Administrator displays a window similar to the Add window with the pasted record's contents. The **Name** field is cleared; you must enter a unique name to save the pasted information as a new record. Cancelling from a Paste operation does not change the contents of the Clipboard.

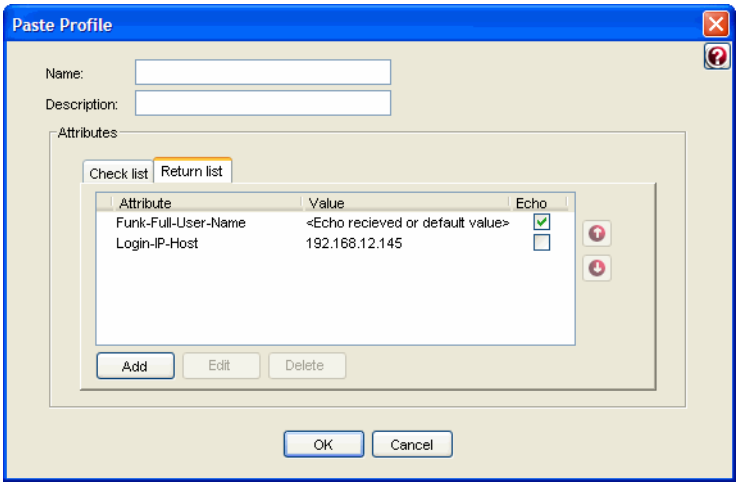


Figure 14 Sample Paste Window

Resizing Columns

You can resize columns in an SBR Administrator table by dragging the column header boundary to the left or right.

Changing Column Sequence

You can change the sequence of columns in an SBR Administrator table by dragging the column headers left and right.

Sorting Information

By default, items in SBR Administrator tables are sorted by Name. You can sort items in any order by clicking a column header.

Previously sorted tables retain their order when the table is sorted on another column. If you want to sort a table by more than one column (for example, sort by address pool and subsort by IP address), click the least-significant column (here, IP Address), and then click the more significant columns (here, Address Pool).

Using Context Menus

You can right-click an object in SBR Administrator windows to display a context menu for that object. The contents of the context menu depends on the type of item; for example, if you right-click a RADIUS client entry, the context menu provides options for copying, cutting, pasting, and deleting.

If you right-click a blank area in an SBR Administrator window, the context menu displays a different set of options. For example, if you right-click a blank space in the RADIUS Client panel, the context menu provides options for refreshing the display, adding or pasting and entry, and printing.

Adding License Keys

Depending upon your purchasing arrangements, your Steel-Belted Radius software may require a new license key at some point after its initial installation.

If you are provided with a new license key by your reseller or by Steel-Belted Radius, you can add the key to an existing Steel-Belted Radius installation as follows:

1. Start the SBR Administrator program and connect to your Steel-Belted Radius server.
2. Select **File > License**.
3. When the Add a License for Server window (Figure 15) appears, enter the license key and click **OK**.

Figure 15 *Add a License for Server Window*

   If the license key you have entered is invalid, the server displays an error message. If this occurs, click **OK** in the message window and enter the correct license key.

4. After you have entered a valid license key, the server displays a confirmation message and reminds you that you must restart the server. Click **OK**.

The server does not restart itself automatically after a new license key is added. You must restart Steel-Belted Radius manually to activate the new license key.

Accessing Online Help

To get help with the SBR Administrator, click the **?** (help) button, press F1, or select **Help > Contents**.

To view the PDF version of the Steel-Belted Radius manuals, select **Help > Manuals >** and choose the manual you want to open. Note that you must have the free Adobe Reader software installed on your computer to display the Steel-Belted Radius manuals.

Displaying Version Information

To identify the current version of the SBR Administrator, select **Help > About** to open the SBR Administrator window (Figure 16).

Figure 16 *About SBR Administrator Window*

Exiting the SBR Administrator

To exit the SBR Administrator, choose **File > Exit**.

Closing the SBR Administrator has no impact on the Steel-Belted Radius service or daemon.

Chapter 4
Administering RADIUS Clients

A RADIUS client is a network device or software application that contacts Steel-Belted Radius when it needs to authenticate a user or to record accounting information about a network connection.

A RADIUS client group is a collection of network devices or software applications that contacts Steel-Belted Radius to authenticate a user or to record accounting information about a network connection. Members of a RADIUS client group use a contiguous range of IP addresses and use identical settings, such as a shared secret or an IP address pool.

This chapter describes how to set up RADIUS clients and client groups.

RADIUS Clients Panel

The RADIUS Clients panel lets you identify the devices that you want to define as clients of Steel-Belted Radius.

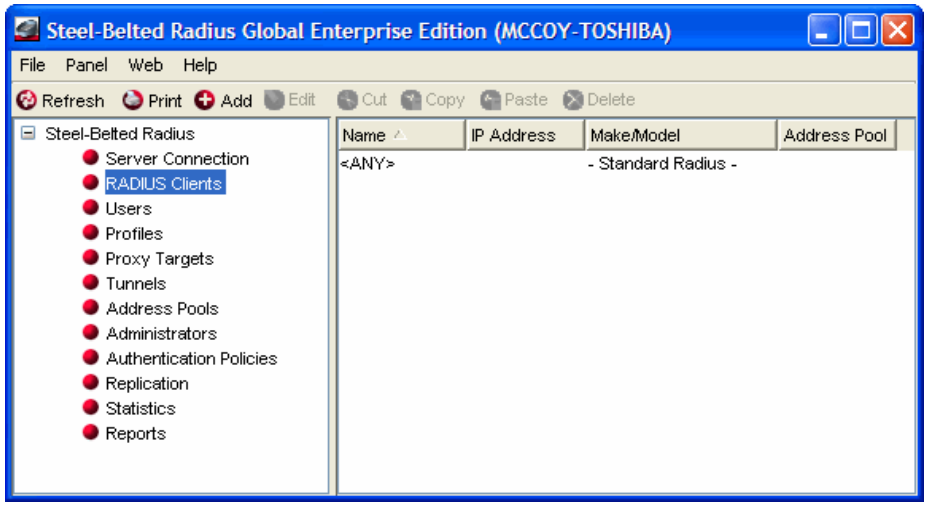


Figure 17 *RADIUS Clients Panel*

Adding a RADIUS Client or Client Group

To add a RADIUS client or client group:

1 Click the **Add** button.

The Add RADIUS Client window (Figure 18) appears.

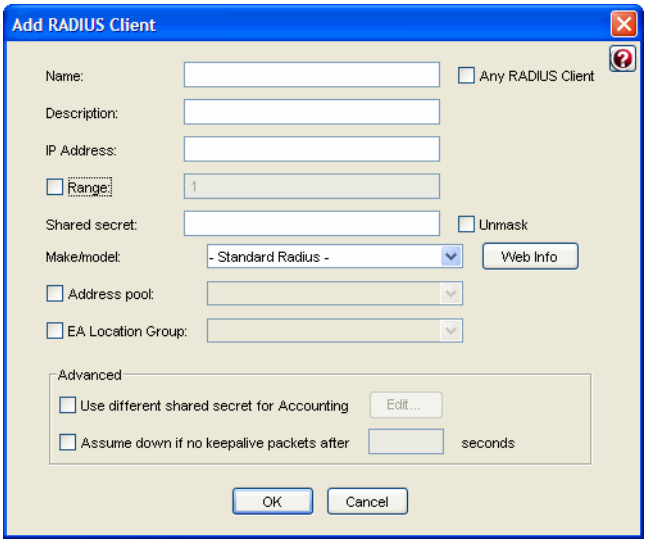


***Figure 18** Add RADIUS Client Window*

2 Enter the name of the RADIUS client or client group in the **Name** field.

Although you can assign any name to a RADIUS client entry, you should use the device's IP host name or hostname to avoid confusion.

You can create a special RADIUS client entry called <ANY> by clicking the **Any RADIUS Client** checkbox (Figure 19). The <ANY> RADIUS client allows Steel-Belted Radius to accept requests from any RAS or proxy RADIUS server, as long as the shared secret is correct.

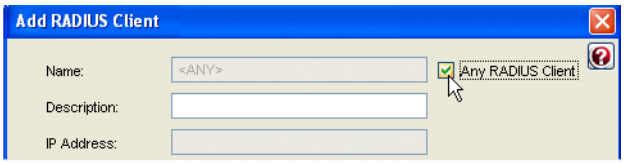


***Figure 19** Creating an <ANY> RADIUS Client*

Note that the **IP Address** field for an <ANY> RADIUS client cannot be edited. <ANY> implies that the server accepts requests from any IP address, provided that the shared secret is correct.

3 Optionally, enter a description of the RADIUS client in the **Description** field.

The description you associate with a RADIUS client is not used during processing.

4 Enter the IPv4 or IPv6 address of the RADIUS client in the **IP Address** field.

You can enter the IPv4 or IPv6 address of the RADIUS client. Alternatively, you can enter the DNS name of the device; the SBR Administrator resolves the name

you enter to its corresponding IP address and displays the result in the **IP Address** field.

If you want the RADIUS client to use an IPv4 address range, enter the starting address for the range in the **IP Address** field, click the **Range** checkbox, and enter the number of addresses in the range in the **Range** field (Figure 20). You can create an address range of as many as 500 addresses in an address range.

For more information on IPv4 address ranges for RADIUS clients, see "RADIUS Client Groups" on page 28.

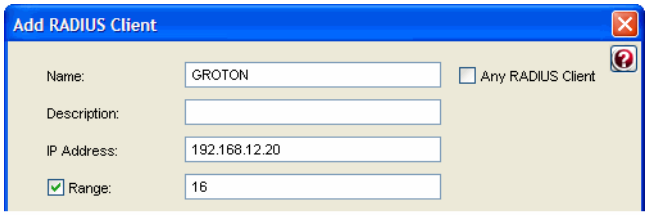


***Figure 20** Entering an IPv4 Address Range for a RADIUS Client*

Tip: See "RADIUS Shared Secret" on page 8.

5 Enter the authentication shared secret for the RADIUS client in the **Shared secret** field.

For privacy, asterisks are echoed as you type. You can check **Unmask shared secret** to display the characters in the shared secret.

After you complete configuration of the authentication shared secret on the server side, you must enter the same authentication shared secret when you configure the RAS device.

6 Use the **Make/model** list to select the make and model of your RADIUS client device.

The **Make/model** selection tells Steel-Belted Radius which dictionary of RADIUS attributes to use when communicating with this client. If you are not sure which make and model you are using or if your device is not in the list, select **- Standard RADIUS -**.

7 If you want the RADIUS client to obtain its IPv4 address from an address pool, click the **Address pool** checkbox and use the **Address pool** list to specify which address pool to use when authenticating an access request from this RADIUS client.

***NOTE:** You must configure IP address pools before you set up RADIUS clients if you want the clients to use address pools.*

8 If you are using Steel-Belted Radius in conjunction with Endpoint Assurance (EA) and you want to associate the RADIUS client with an EA location group, check the **EA Location Group** checkbox and use the **EA Location Group** list to specify the location group to which the RADIUS client belongs.

***NOTE:** The EA Location Group controls are disabled until you configure your Endpoint Assurance location groups in the* `ea.ini` *file. Refer to the Steel-Belted Radius Reference Guide for information on configuring the* `ea.ini` *file.*

9 Optionally, specify an accounting secret for the RADIUS client.

By default, Steel-Belted Radius uses the same shared secret for authentication and accounting. If you want the RADIUS client to use different shared secrets for authentication and accounting:

a Click the **Use different shared secret for accounting** checkbox.

b When the Accounting Shared Secret window (Figure 21) opens, enter the shared secret you want the RADIUS client to use for accounting.

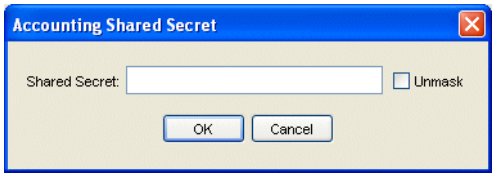


***Figure 21** Accounting Shared Secret Window*

For privacy, asterisks are echoed as you type. You can click the **Unmask** checkbox to display the characters in the shared secret.

c Click **OK**.

You must enter the same accounting shared secret when you configure the RADIUS client.

10 Optionally, indicate whether you want to enable keepalive processing and specify how long the server should wait for RADIUS packets from the client before assuming connectivity has been lost.

If you check the **Assume down if no keepalive packets after** checkbox, you can enter a value in the **(seconds)** field. If the server does not receive any RADIUS packets from this client after the specified number of seconds, the server assumes that the connection to the client is lost or that the client device has failed. When this happens, Steel-Belted Radius gracefully closes any user or tunnel connections it has authenticated for the client. Steel-Belted Radius releases any pooled IP or IPX addresses and adjusts the counts of concurrent user or tunnel connections appropriately.

***NOTE:** If the value you enter in the **(seconds)** field is too low, valid user or tunnel connections can be lost. For example, during low usage periods, a RAS device might send no RADIUS packets to the Steel-Belted Radius server, even though the device is still "up."*

Verifying a Shared Secret

To verify a shared secret on Steel-Belted Radius:

1 Open the RADIUS Clients panel.

2 Select the RADIUS client whose shared secret you want to verify and click the **Edit** button (or double-click the RADIUS client entry).

The Edit RADIUS Client window opens.

3 Enter the shared secret you think is assigned to the RADIUS client in the **Shared secret** field.

4 Click the **Validate** button.

The SBR Administrator displays a message confirming that the shared secret is what you think it is.

Deleting a RADIUS Client

To delete a RADIUS client:

1 Open the RADIUS Clients panel.

2 Select the RADIUS client entry you want to delete.

3 Click the **Delete** button on the SBR Administrator toolbar.

4 When you are prompted to confirm the deletion request, click **Yes**.

Chapter 5
Administering Users

This chapter describes how to add users to the Steel-Belted Radius database.

User Files

The following files establish settings for setting up users. For more information about these files, refer to the *Steel-Belted Radius Reference Guide*.

Table 14. *User Account Files*

File Name	Function
`redirect.ini`	Configures settings for when Steel-Belted Radius should redirect users after repeated failed login attempts.
`radius.ini`	Specifies (among other things) the settings relating to RSA SecurID support in Steel-Belted Radius.
`securid.ini`	Specifies the prompt strings returned to SecurID users during login and authentication.
`tacplus.ini`	Specifies the name of the TACACS+ server and the shared secret used to validate communication between the Steel-Belted Radius server and the TACACS+ server.

Users Panel

The Users panel (Figure 22) lets you administer user accounts. Each user entry in the Steel-Belted Radius database identifies one method by which the server can authenticate a specific user.

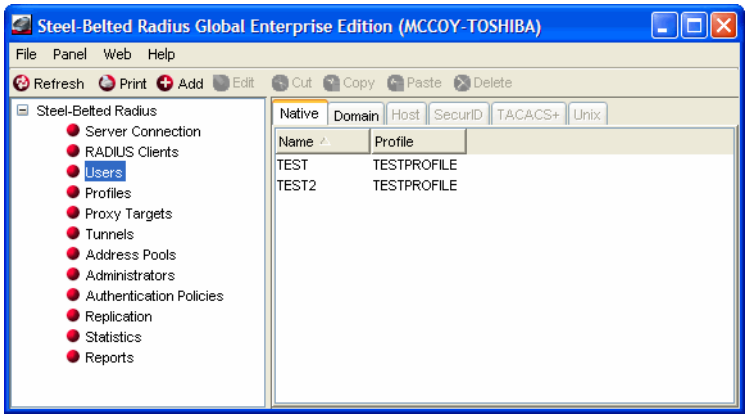


***Figure 22** Users Panel*

The Users panel has as many as six tabs, as described in Table 15.

***Table 15.** Users Panel Tabs*

Tab	Function	Available On Server Running OS
Native	Lists the native users in the local Steel-Belted Radius database. You must display the **Native** tab to add, edit, and delete native users.	Windows Linux Solaris
Domain	Lists the users authenticated using Windows Domain authentication. You must display the **Domain** tab to add, edit, and delete Domain users.	Windows
Host	Lists the users authenticated using host authentication. You must display the **Host** tab to add, edit, and delete host users.	Windows
SecurID	Lists the users authenticated using RSA SecurID authentication. You must display the **SecurID** tab to add, edit, and delete SecurID users.	Windows Linux Solaris
TACACS+	Lists the users authenticated using TACACS+. You must display the **TACACS+** tab to add, edit, and delete TACACS+ users.	Windows Linux Solaris
UNIX	Lists the users and groups authenticated using UNIX authentication. You must display the **UNIX** tab to add, edit, and delete UNIX users.	Linux Solaris

***NOTE:** You can populate the user database for Steel-Belted Radius by entering information in the Users panel or by importing data from other servers. For more information on importing user information, see Appendix E, "Importing and Exporting Data."*

Setting Up Native Users

Native user entries require you to enter the user's name and password into the Steel-Belted Radius database. For all other types of user entry, the server relies on another database to confirm the user's password.

***NOTE:** You must define a native user entry for every user who requires remote access to a Windows network. For example, you can accommodate Solaris, Linux, or Macintosh users by adding them as native users.*

Adding a Native User

To add a native user to the Steel-Belted Radius database:

1. Click the **Users** entry to display the Users panel (Figure 22).
2. Click the **Native** tab.
3. Click the **Add** button to display the Add Native User window (Figure 23).

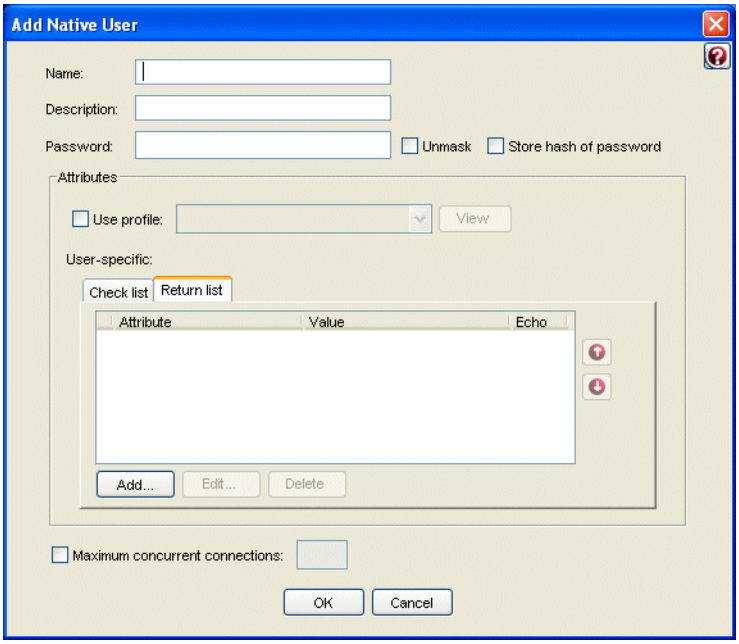


***Figure 23** Add Native User Window*

4. Enter the user's login name in the **Name** field.

   Native user entries in the Steel-Belted Radius database have all-uppercase names; names are converted to all-uppercase letters when the native user entry is created, and they remain all-uppercase for the life of the entry. For example, a native username entered as **realLife1** is stored as `REALLIFE1` in the Steel-Belted Radius database.

5. Optionally, enter a description of the user in the **Description** field.
6. Enter the user's login password in the **Password** field.

If you want the characters in the password (rather than asterisks) to appear as you type, click the **Unmask** checkbox. Note that passwords are case-sensitive: *swordfish*, *SwordFish*, and *SWORDFISH* are three different passwords.

7 Specify whether you want the user's password to be encrypted before it is stored.

- ▷ If this user requires only PAP authentication and you want to store the hash of the password in the Steel-Belted Radius database, click the **Store hash of password** checkbox. This option allows the user to authenticate using only PAP.
- ▷ If this user requires CHAP authentication, do not click the **Store hash of password** checkbox.

8 If you want to use a profile to assign checklist and return list attributes to the user, click the **Use profile** checkbox and use the **Profile** list to select the profile you want.

After you select a profile, you can click the **View** button to display the checklist and return list attributes in that profile.

For more information on profiles, refer to Chapter 6, "Administering Profiles."

9 If you want to specify checklist attributes or return list attributes for the user, click the **Checklist** tab or the **Return list** tab, and then click the **Add** button.

Refer to "Adding a Checklist or Return List Attribute for a User" on page 63 for information on how to add checklist and return list attributes.

10 After you have added the appropriate checklist and return list attributes for a user, select an attribute and use the Up and Down buttons to the right of the attribute list to put the attributes in the correct sequence.

11 If you want to specify the maximum number of concurrent connections this user can maintain, click the **Maximum concurrent connections** checkbox and enter a number in the accompanying field.

12 Click **OK**.

Editing a Native User

After you have added a user, you can modify any setting for that user except the username. To edit a native user who already exists in the Steel-Belted Radius database:

1 Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).

2 Click the **Native** tab.

3 Select the user entry you want to edit and click the **Edit** button (or right-click an entry and choose **Edit** from the context menu).

The Edit Native User window (Figure 24) opens.

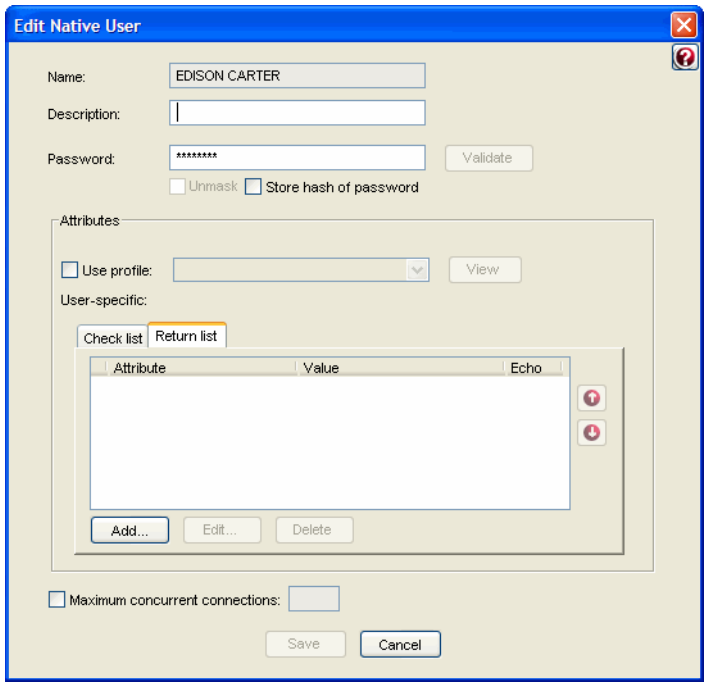


Figure 24 Edit Native User Window

4 Edit the settings for the user as appropriate.

Refer to "Adding a Native User" on page 61 for information on the fields in the native user windows.

You can modify any setting except the user's name. To edit a user's name, you must copy the user record to a new user entry.

5 Click **Save**.

Deleting a Native User

To delete a native user:

1 Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).

2 Click the **Native** tab.

3 Select the user entry you want to delete and click the **Delete** button (or right-click an entry and choose **Delete** from the context menu).

4 When you are asked to confirm the deletion, click **Yes**.

Adding a Checklist or Return List Attribute for a User

A checklist attribute is an item of information that must accompany a request for connection before the connection can be authenticated. A return list attribute is an item of information that Steel-Belted Radius includes in the Access-Accept message when a connection request is approved.

To add a checklist or return list attribute to a user's entry:

1. Open the appropriate user entry.
2. Click the **Checklist** tab or the **Return list** tab.
3. Click **Add**. The Add Checklist Attribute window or the Add Return List Attribute window (Figure 25) opens.

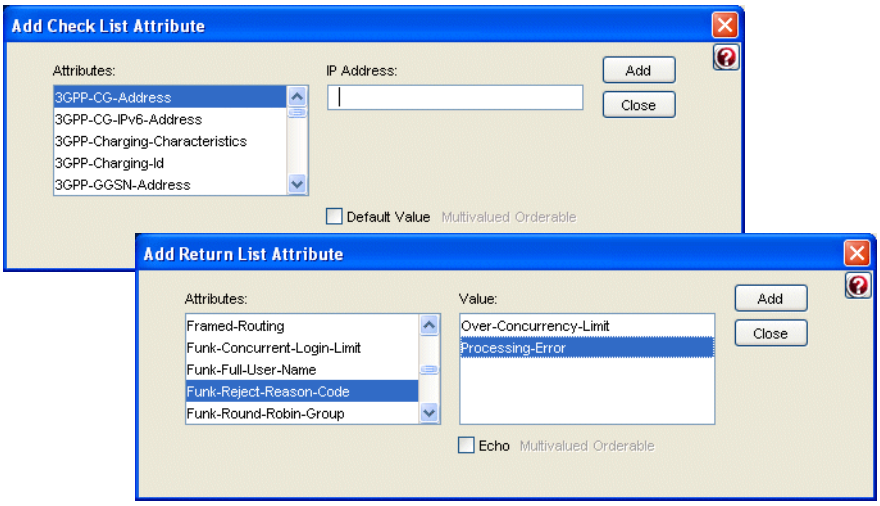


***Figure 25** Add Checklist Attribute and Add Return List Attribute Windows*

4. Select the attribute you want to add from the **Attributes** list.
5. Select or enter a value for the attribute.

   The window changes according to the attribute you choose. Some attributes require that you enter a value, string, or IP address. Other attributes require that you choose from a fixed list of values.

   If the **Multivalued** indicator is dimmed, an attribute can have only one value. If the **Multivalued** attribute is undimmed, you can add multiple values for the attribute.

   (Checklist attributes only) To set this value to the default value for the attribute (which is useful in situations where the attribute is not included in the RADIUS request), check the **Default value** checkbox.

   (Return list single-valued attributes only) If you do not want to specify a particular value, but want to make sure that whatever value of the attribute appears in the RADIUS request is echoed to the client in the RADIUS response, click the **Echo** checkbox.
6. Click **Add** to add this attribute/value pair to the list.
7. When you are finished adding attribute/value pairs, click **Close** to return to the Add User window.

Setting Up Windows Domain Users

Windows domain authentication is automatically enabled when you install the server. To use the Windows Domain Authentication plug-in, the RADIUS service must be run under the `LocalSystem` account on a Windows 2000 or Windows XP computer that is part of a domain. (The `LocalSystem` account is a standard authenticated domain user account in Windows.) Groups and users to be authenticated can reside in any domain within the forest, as well as in those domains outside the forest for which a trust relationship exists.

When you use Windows domain authentication, the domain name can be present in the User-Name attribute of the Access Request, which can be of the form `\\domain\user`, `domain\user`, or simply `user`. Additionally, the form `user@domain` can be used.

Prequalification Checklists

By default, when Steel-Belted Radius uses Windows domain membership to authenticate a user, it processes attributes for the first group that the user matches. The attributes consist of checklist and reply-list attributes, and checklist processing is performed to determine the user's authorization rights after authentication succeeds.

If an enterprise sets up separate Windows domain groups for different access methods (for example, one domain group for users accessing the network through a VPN and another domain group for users accessing the network through a WLAN Access Point) and then assigns users to more than one domain group (so that the users get different permissions based on what access method they use), Steel-Belted Radius can authenticate the user against the first group the user matches and process the wrong attributes for that user, causing checklist processing to fail and the user's access to be rejected.

Prequalification checklists allow a site to perform checklist processing before it authenticates a user, so that the attributes returned by every group a user belongs to can be evaluated (and the appropriate membership chosen) before authentication proceeds.

Example: CandyCorp sets up two groups (WLAN_USERS and VPN_USERS) in the CORP domain and creates access policies for each. Mary is a member of both groups; when she accesses the corporate network through a WLAN Access Point, her traffic should be tagged for a specific VLAN, and when she accesses the corporate network through a VPN, an Ascend-Data-Filter should be sent to the VPN gateway to restrict the internal hosts she can reach.

- Without prequalification checklist processing, Steel-Belted Radius responds to Mary's connection through an Access Point by using the first domain group membership it finds (which might be VPN_USERS), authenticating Mary and returning the attributes associated with that group, and then rejecting Mary because post-authentication checklist processing fails when the group used for authentication (VPN_USERS) didn't provide the appropriate access attributes.
- With prequalification checklist processing enabled, Steel-Belted Radius responds to Mary's connection through the Access Point by running checklist processing *before* it authenticates Mary: Steel-Belted Radius tests each group to which Mary belongs to

see if authentication and authorization will ultimately be successful. If checklist processing for a domain group fails, that group is skipped and the next group is tried; if checklist processing for all groups fails, Mary's access request is denied. If checklist processing successfully matches Mary to a domain group, authentication proceeds, and Mary's traffic is processed according to corporate policies (that is, it is tagged with the VLAN identifier appropriate for her WLAN access).

The application of prequalification checklist processing is not limited to domain groups. Prequalification checklists can be used to direct a user request to an appropriate domain user entry based on the presence of attributes in the user's request. For example, if a user's name ("ADMIN") is specified in an Access-Request and both \\CORP\ADMIN and \\LAB\ADMIN are listed in the Steel-Belted Radius database with the same password, prequalification checklist processing could be used to select the appropriate domain user object for authentication and authorization.

NOTE: *Prequalification checklist processing can be relatively expensive in terms of processing time. Each access request might entail multiple database operations, since Steel-Belted Radius must potentially review every domain group to find one with attributes that match the user's checklist requirements.*

Prequalification processing is enabled through the PrequalifyChecklist argument in the in the [Windows Domain] section of the `winauth.aut` file.

MS-CHAP Considerations

If the user is successfully authenticated, any appropriate encryption keys (obtained through either MS-CHAP or MS-CHAP-V2) are returned to Steel-Belted Radius and the user's profile is retrieved from the Steel-Belted Radius database. To enable encryption, the appropriate attributes, such as `Mppe-Send-Key` and `Mppe-Recv-Key`, must be included in the user's profile You do not need to prepend the username with the domain name to avoid timeout problems when dealing with a large number of domains.

The Windows Domain Authentication plug-in does not support EAP pre-fetch.

Configuration

As with other authentication plug-ins, `winauth.dll` is configured through a single `.aut` file (`winauth.aut`). The `winauth.aut` file must contain [Bootstrap] and [Windows Domain] sections:

```
[Bootstrap]
LibraryName=winauth.dll
Enable=1
InitializationString=Windows domain authentication.
```

Handling of users with expired passwords is configured in the [Windows Domain] section:

```
[Windows Domain]
AllowExpiredPasswordsForUsers = no
AllowExpiredPasswordsForGroups = no
RetryFailedAuthentications = no
AllowMachineLogin = yes
;ProfileForExpiredUsers = Profile
```

```
;ProfileForExpiredUsersInGroups = Profile
PrequalifyChecklist = no
```

NOTE: *MS-CHAP and MS-CHAP-V2 users with expired passwords are not accepted. They may be prompted to change password if their login application supports password changing.*

Expired Domain Passwords

The Windows domain authentication method allows users to be authenticated against domain security using an expired domain password. This lets Steel-Belted Radius handle security policies that force domain passwords to be changed automatically after a certain number of days. Typically, after the password expires, at the next attempt to log in, the domain recognizes the password supplied by the user as expired. The domain then returns a special status code to its client application indicating these conditions. Typically, the user is then prompted to change his or her domain password, but the client application (for example, Microsoft Remote Access Client) must support the ability to change passwords.

When Steel-Belted Radius passes a username/password pair through to a domain for authentication, the domain can indicate to Steel-Belted Radius that the password is expired. If so, Steel-Belted Radius's default response is to issue an Access-Reject. You can configure it to respond instead with an Access-Accept.

Adding a Domain User or Domain Group

Most medium to large Microsoft networking installations are organized into *domains* for security purposes. If your network is so organized, domain authentication should be preferred over host authentication for RADIUS purposes.

To use domain authentication, the Steel-Belted Radius service must run on a Windows workstation or server that belongs to a domain. The Windows host running Steel-Belted Radius does not need to be a domain controller.

It is possible to authenticate against domains other than the one in which the Steel-Belted Radius service is running, provided that the other domain is trusted by the domain of the RADIUS service. The trust relationship may not be mutual; the other domain does not have to trust the RADIUS domain.

Example: An enterprise has three domains: A, B, and C, and Steel-Belted Radius is running in A. A trusts B and C trusts A. You can use Domains A and B for authentication, but not C, because A does not trust C.

You can add a Domain User entry to provide for the authentication of a specific user defined within a specific domain under Microsoft networking. For more flexibility, you can add a Domain Group, to provide for the authentication of all users that belong to a specific group defined within a specific domain.

To add a domain user or domain group:

1. Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).
2. Click the **Domain** tab.

3 Click the **Add** button on the SBR Administrator toolbar to display the Add Domain User window (Figure 26).

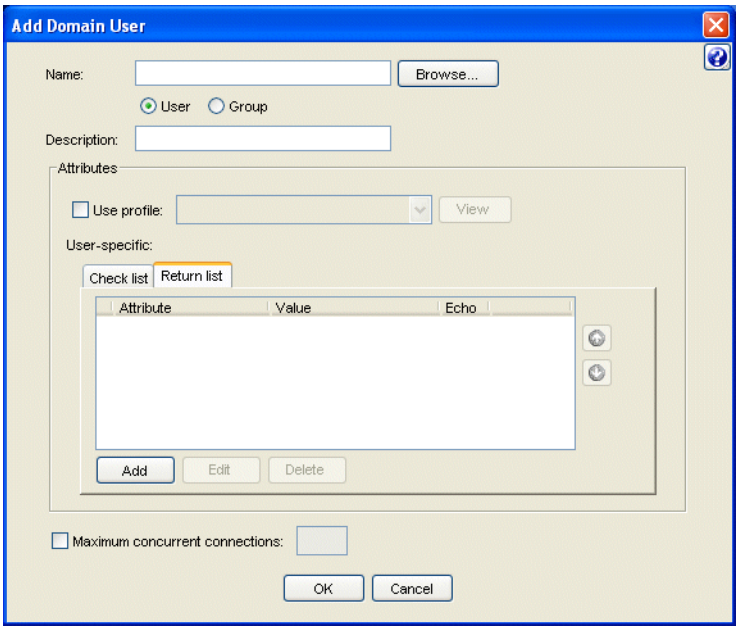


Figure 26 *Add Domain User Window*

4 Enter the user's login name in the **Name** field.

Domain usernames must be in the format ***DOMAIN******USER***. Domain usernames cannot contain the following characters:

\ / " [ ] : | < > + = ; , ? * @

If you want to browse for an existing user or group, click the **Browse** button. When the Browse for Domain User window (Figure 27) opens, click the name of the appropriate domain, and then click the name of the user or group in that domain you want to use. Click **OK** to finish.

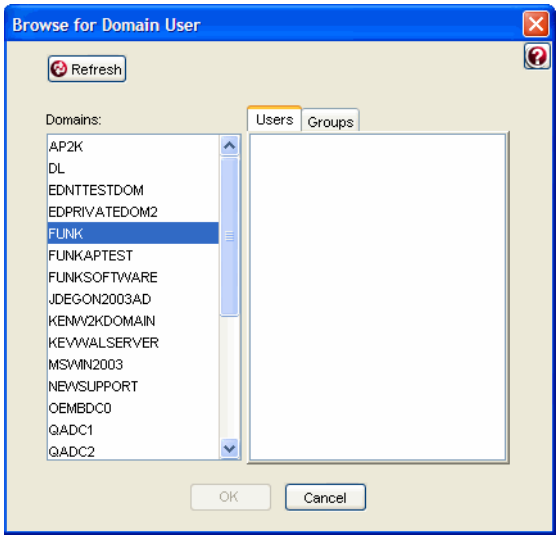


Figure 27 *Browse for Domain User Window*

5 Optionally, enter a description of the domain user or group in the **Description** field.

6 If you want to use a profile to assign checklist and return list attributes to the user, click the **Use profile** checkbox and use the **Profile** list to select the profile you want.

After you select a profile, you can click the **View** button to display the checklist and return list attributes in that profile.

7 If you want to specify checklist attributes or return list attributes for the domain user, click the **Checklist** tab or the **Return list** tab, and then click the **Add** button.

Refer to "Adding a Checklist or Return List Attribute for a User" on page 63 for information on how to add checklist and return list attributes.

8 After you have added the appropriate checklist and return list attributes for a user, select an attribute and use the Up and Down buttons to the right of the attribute list to put the attributes in the correct sequence.

9 If you want to specify the maximum number of concurrent connections this user can maintain, click the **Maximum concurrent connections** checkbox and enter a number in the accompanying field.

10 Click **OK**.

Setting Up Host Users

Host authentication is provided to accommodate networks that aren't organized into domains, or networks that mix domain and workgroup security. Be aware of the following restrictions and limitations to the use of Host authentication:

> Microsoft networking permits only one peer-to-peer connection between two machines at a time. Steel-Belted Radius performs Host authentication by attempting to log the user into the specified Host; if the server is already logged into that Host (for any reason), the new login attempt fails due to the one-connection restriction.

You can add a Host User entry to provide for the authentication of a specific user defined on a specific Windows 2000 machine. For more flexibility, you can add a Host Group, to provide for the authentication of all users that belong to a specific group defined on a specific Windows 2000 machine.

Adding a Host User or Group

To add a Host user or group:

1 Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).

2 Click the **Host** tab.

3 Click the **Add** button. The Add Host User or Group window (Figure 28) appears.

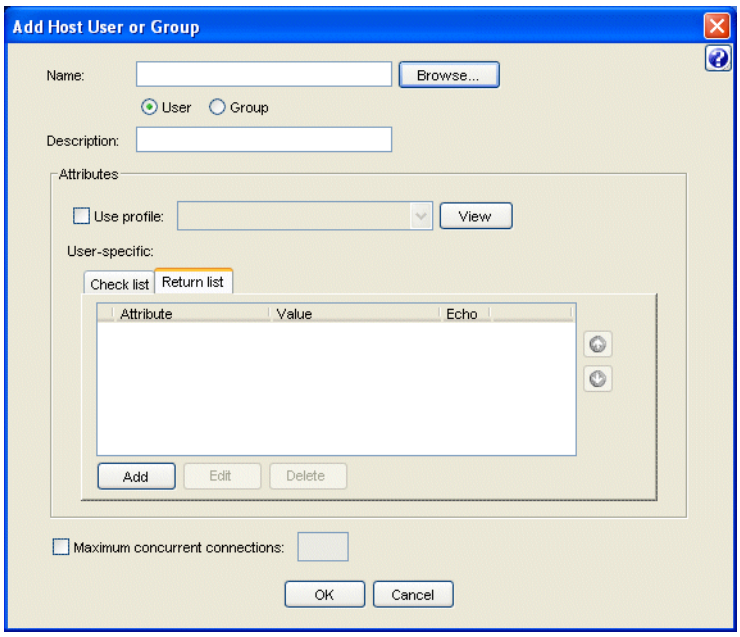


Figure 28 *Add Host User Window*

4 Enter the user's login name in the **Name** field.

If you want to browse for an existing user or group, click the **Browse** button. When the Browse for Host User window (Figure 29) opens, select the name of a Windows host from the list on the left and then select a Host user or group from the list on the right. Click **OK**. SBR Administrator returns you to the Add Host User window.

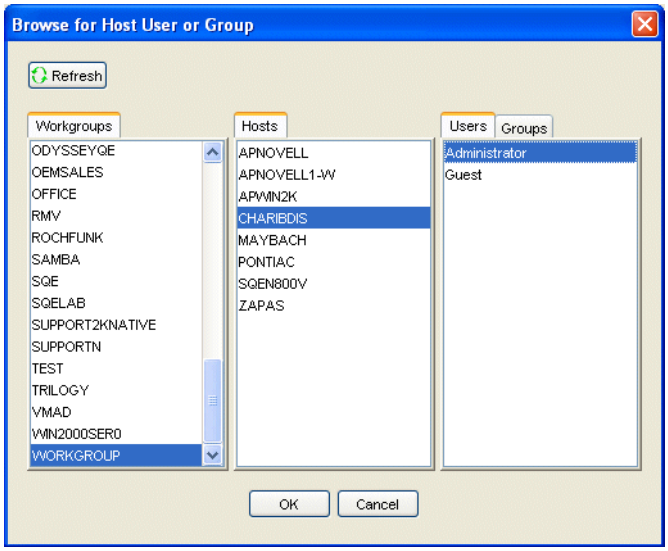


Figure 29 *Browse for Host User or Group Window*

5 Optionally, enter a description of the Host user or group in the **Description** field.

6 If you want to use a profile to assign checklist and return list attributes to the user, click the **Use profile** checkbox and use the **Profile** list to select the profile you want.

After you select a profile, you can click the **View** button to display the checklist and return list attributes in that profile.

7 If you want to specify checklist attributes or return list attributes for the user, click the **Checklist** tab or the **Return list** tab, and then click the **Add** button.

Refer to "Adding a Checklist or Return List Attribute for a User" on page 63 for information on how to add checklist and return list attributes.

8 After you have added the appropriate checklist and return list attributes for a user, use the Up and Down buttons to the right of the attribute list to put the attributes in the correct sequence.

9 If you want to specify the maximum number of concurrent connections this user can maintain, click the **Maximum concurrent connections** checkbox and enter a number in the accompanying field.

10 Click **OK**.

Setting Up SecurID Users

Usernames are case-sensitive. The case in which names are recorded depends on whether usernames are being stored in a local or external database. Usernames stored in an external database (UNIX, RSA SecurID, TACACS+) retain their case as stored in that database.

Adding a SecurID User

You can configure Steel-Belted Radius to use RSA SecurID authentication for your users by setting up communication between the RSA server and the RADIUS server (described in "Configuring SecurID Authentication" on page 136), and then adding SecurID users to the Steel-Belted Radius database using the instructions that follow.

Steel-Belted Radius attempts SecurID authentication only on usernames that match a SecurID entry in its User database. Steel-Belted Radius offers four types of SecurID entry, each providing a different matching rule:

- You can enter the name of a specific user.

  For example, you might create a SecurID user entry for the specific user `George`. This tells Steel-Belted Radius that SecurID can be used as an authentication method when an authentication request is received for username `George`. If username `George` is authenticated, the attributes of the user entry apply.

- You can enter a prefix.

  For example, you might create a SecurID user entry for the prefix `sales$`. This tells Steel-Belted Radius that SecurID can be used as an authentication method when an authentication request is received for a username such as `sales$Harry` or `sales$Cynthia`. Using a prefix lets you group multiple SecurID users into a single user entry instead of creating a separate entry for each SecurID user. If the `sales$` user is authenticated, the attributes of the user entry apply.

***NOTE:** Only the part of the username after the prefix (Harry or Cynthia in the example above) is sent to the RSA SecurID server.*

You can use different settings for different groups.

***NOTE:** The user must type in the prefix as part of the username when dialing in and requesting a connection.*

- You can enter a suffix.

  A suffix works like a prefix, but appears at the end of the username; for example, if the suffix were `!sales`, you might have usernames such as `Harry!sales` or `Cynthia!sales`.

- You can create an entry for `Any` user.

  This creates a single user entry named `<ANY>` that matches any username to be authenticated. SecurID can be used as an authentication method for any username, and, if successful, the attributes of the `<ANY>` entry apply.

  The `<ANY>` entry makes sense if a single set of attributes apply to all your SecurID users and if you want to make SecurID either the only authentication method used or the authentication method of last resort if other authentication methods fail.

To add a SecurID user entry:

1. Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).
2. Click the **SecurID** tab.
3. Click the **Add** button on the SBR Administrator toolbar to display the Add SecurID User window (Figure 30).

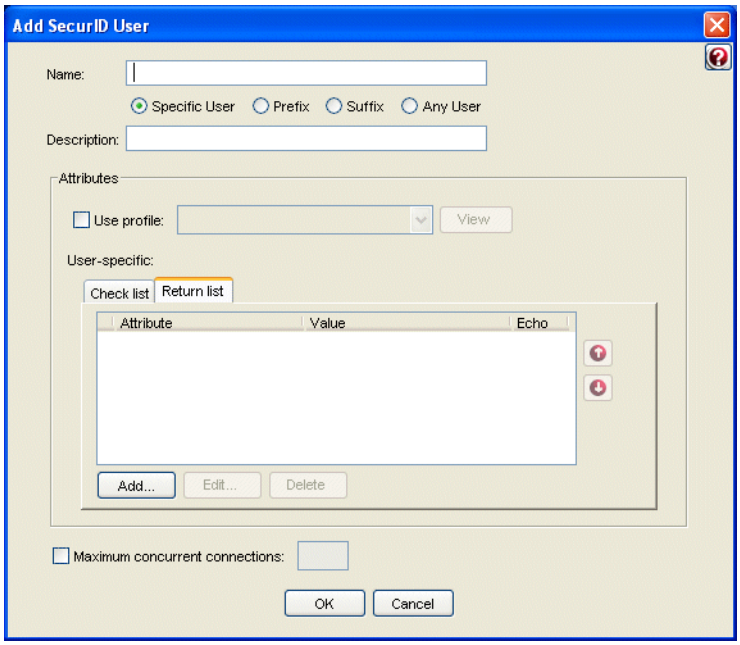


***Figure 30** Add SecurID User Window*

4. Enter the specific username, a prefix, or a suffix in the **Name** field.

5 Select the user type: **Specific user**, **Prefix**, **Suffix**, or **Any user**.

6 Optionally, enter a description of the user in the **Description** field.

7 If you want to use a profile to assign checklist and return list attributes to the user, click the **Use profile** checkbox and use the **Profile** list to select the profile you want.

After you select a profile, you can click the **View** button to display the checklist and return list attributes in that profile.

8 If you want to specify checklist attributes or return list attributes for the user, click the **Checklist** tab or the **Return list** tab, and then click the **Add** button.

Refer to "Adding a Checklist or Return List Attribute for a User" on page 63 for information on how to add checklist and return list attributes.

9 After you have added the appropriate checklist and return list attributes for a user, use the Up and Down buttons to the right of the attribute list to put the attributes in the correct sequence.

10 If you want to specify the maximum number of concurrent connections this user can maintain, click the **Maximum concurrent connections** checkbox and enter a number in the accompanying field.

11 Click **OK**.

Each new suffix or prefix entry that you add appears in the Users window with the username represented by the string `USERNAME`; for example, `!USERNAME` or `SALES<USERNAME>`.

Setting Up TACACS+ Users

You can configure Steel-Belted Radius to authenticate your users by querying a TACACS+ server.

***NOTE:** Before you add TACACS+ users, you must configure communication between the Steel-Belted Radius server and the TACACS+ server by editing the tacplus.ini file. For more information on the* `tacplus.ini` *file, refer to the Steel-Belted Radius Reference Guide.*

Steel-Belted Radius attempts TACACS+ authentication only on usernames that match a TACACS+ entry in its user database. Each type of TACACS+ entry specifies a different matching rule:

- You can enter the name of a specific user.

  For example, you might create a TACACS+ user entry for the specific user `George`. This tells Steel-Belted Radius that when an authentication request is received for username `George`, TACACS+ can be used as an authentication method and, if successful, the attributes of this user entry apply.

- You can enter a prefix.

  For example, you might create a TACACS+ user entry for the prefix `sales$`. This tells Steel-Belted Radius that when an authentication request is received for a

username such as `sales$Harry` or `sales$Cynthia`, TACACS+ can be used as an authentication method and, if successful, the attributes of this user entry apply.

NOTE: *Only the part of the username after the prefix (Harry or Cynthia in the example above) is sent to the TACACS+ server.*

Using a prefix lets you group multiple TACACS+ into a single user entry instead of creating a separate entry for each TACACS+ user. You can use different settings for different groups.

NOTE: *The user must type in the prefix as part of the username he or she is using to dial in and request a connection.*

- You can enter a suffix.

  A suffix works like a prefix, but appears at the end of the username; for example, if the suffix were `!sales`, you might have usernames such as `Harry!sales` or `Cynthia!sales`.

- You can create an entry for `Any` user.

  This creates a single user entry named `<ANY>` that matches any username to be authenticated. TACACS+ can be used as an authentication method for any username, and, if successful, the attributes of the `<ANY>` entry apply.

  The `<ANY>` entry makes sense if a single set of attributes apply to all your TACACS+ users and if you want to make TACACS+ either the only authentication method used or the authentication method of last resort if other authentication methods fail.

To add a TACACS+ user:

1. Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).
2. Click the **TACACS+** tab.
3. Click the **Add** button on the SBR Administrator toolbar to display the Add TACACS+ User window (Figure 31).

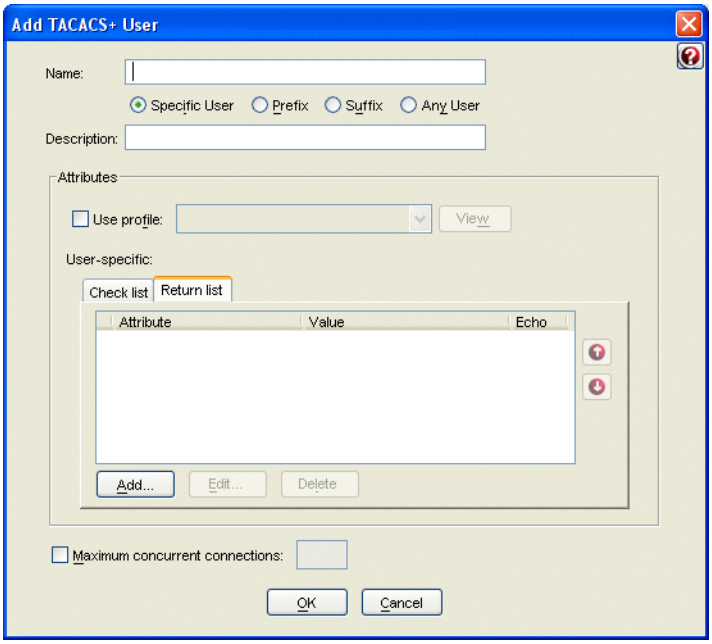


Figure 31 *Add TACACS+ User Window*

4. Enter the specific username, a prefix, or a suffix in the **Name** field.
5. Select the user type: **Specific user**, **Prefix**, **Suffix**, or **Any user**.
6. Optionally, enter a description of the user in the **Description** field.
7. If you want to use a profile to assign checklist and return list attributes to the user, click the **Use profile** checkbox and use the **Profile** list to select the profile you want.

   After you select a profile, you can click the **View** button to display the checklist and return list attributes in that profile.
8. If you want to specify checklist attributes or return list attributes for the user, click the **Checklist** tab or the **Return list** tab, and then click the **Add** button.

   Refer to "Adding a Checklist or Return List Attribute for a User" on page 63 for information on how to add checklist and return list attributes.
9. After you have added the appropriate checklist and return list attributes for a user, use the Up and Down buttons to the right of the attribute list to put the attributes in the correct sequence.
10. If you want to specify the maximum number of concurrent connections this user can maintain, click the **Maximum concurrent connections** checkbox and enter a number in the accompanying field.
11. Click **OK**.

Each new suffix or prefix entry that you add appears in the Users window with the username represented by the string `USERNAME`; for example, `!USERNAME` or `SALES<USERNAME>`.

Setting Up UNIX Users

You can add a UNIX user entry to provide for the authentication of a specific user defined on a Linux or Solaris server. For more flexibility, you can add a UNIX group to provide for the authentication of all users that belong to a specific group defined on the server.

To add a UNIX user or group:

1. Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).
2. Click the **UNIX** tab.
3. Click the **Add** button on the SBR Administrator toolbar to display the Add UNIX User window (Figure 32).

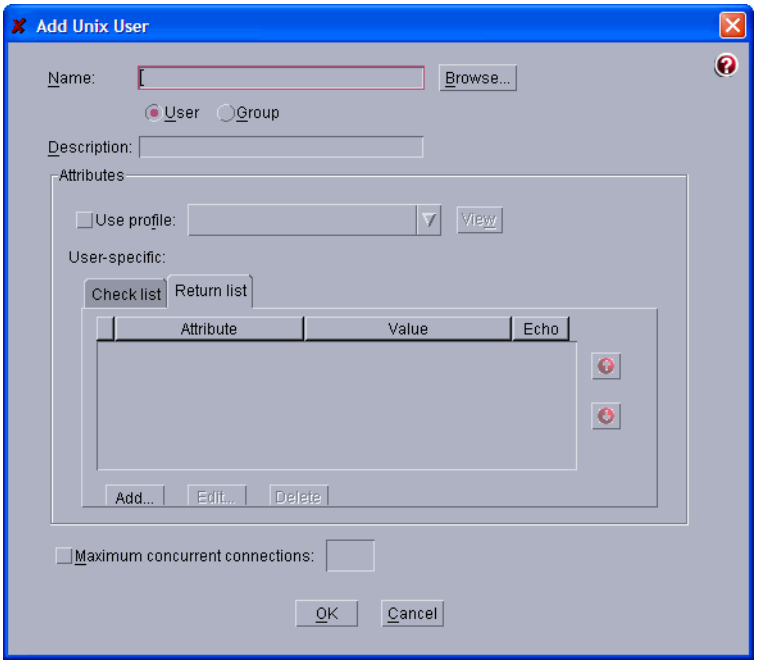


***Figure 32** Add UNIX User Window*

4. Click the **Browse** button and select a user or group from the list. Click **OK**.
5. Optionally, enter a description of the user in the **Description** field.
6. If you want to use a profile to assign checklist and return list attributes to the user, click the **Use profile** checkbox and use the **Profile** list to select the profile you want.

   After you select a profile, you can click the **View** button to display the checklist and return list attributes in that profile.
7. If you want to specify checklist attributes or return list attributes for the user, click the **Checklist** tab or the **Return list** tab, and then click the **Add** button.

   Refer to "Adding a Checklist or Return List Attribute for a User" on page 63 for information on how to add checklist and return list attributes.

8 After you have added the appropriate checklist and return list attributes for a user, use the Up and Down buttons to the right of the attribute list to put the attributes in the correct sequence.

9 If you want to specify the maximum number of concurrent connections this user can maintain, click the **Maximum concurrent connections** checkbox and enter a number in the accompanying field.

10 Click **OK**.

Editing User Settings

This section describes fields that you can set for any user entry, regardless of user type. For more information, see "User Attribute Lists" on page 24 and "About Profiles" on page 81.

Selecting a Profile

To select a profile for a user:

1 Open the Users panel.

2 Select the user whose entry you want to modify.

3 Click **Edit** (or double-click the user entry).

4 Click the **Use Profile** checkbox.

5 Select the list to select the profile you want to use.

To display the settings associated with the selected profile, click the **View** button.

6 When you are finished, click **Save**.

Setting Attribute Values

To change the value of an attribute already in the checklist or return list for a user entry:

1 Click the **Checklist** tab or **Return List** tab.

2 Select the attribute whose value you want to change.

3 Click **Edit** or double-click the attribute.

4 When the Change window opens, enter or select the new value.

Depending on the attribute, you can enter a new value or select a value from a list. For some attributes, Steel-Belted Radius retrieves the value from the server and you cannot enter a value in this window.

5 Click **OK**.

Removing Attribute/Value Pairs

To remove an attribute/value pair already in the checklist or return list for a User entry:

1. Click the **Checklist** tab or **Return List** tab.
2. Select the attribute/value pair you'd like to remove.
3. Click **Delete**.

Reordering Attributes

Certain attributes are multi-valued and orderable; that is, the attribute/value pair can appear more than once in a RADIUS response, and the order in which the attribute/value pairs appear is significant.

To reorder attributes in a User entry:

1. Click the **Checklist** tab or **Return List** tab.
2. Highlight an attribute/value pair in the list.
3. Click the Up or Down arrow to move the selected attribute within the list.
   - ▷ The Up arrow moves the selected attribute/value up in the list. If the attribute is not orderable, or if the selected item already the first value for this attribute, the button is disabled.
   - ▷ The Down arrow moves the selected attribute/value down in the list. If the attribute is not orderable, or if the selected item is already the last value for this attribute, the button is disabled.

Changing Attributes Inherited from a Profile

Checklist and return list attributes can be specified for a user, or they can be inherited from a profile associated with a user. Attributes inherited from a profile are overridden by attributes assigned to a specific user.

Concurrent Connection Limits

A maximum number of open connections can be set for each user entry by checking the **Maximum concurrent connections** checkbox and entering a number in the accompanying field. When the user requests access, the user can be authenticated using the given authentication method only if fewer than this number of connections are currently open for this user.

Deleting a User

To delete a user:

1. Click the **Users** entry to display the Users panel (Figure 22, "Users Panel," on page 60).
2. Click the tab that is appropriate for the type of user you want to delete.

   For example, if you want to delete a SecurID user, click the **SecurID** tab.

3 Click the **Delete** button from the SBR Administrator toolbar (or right-click the user entry and choose **Delete** from the context menu).

4 When you are prompted to confirm the deletion, click **Yes**.

Chapter 6
Administering Profiles

This chapter describes how to set up and administer user profiles.

About Profiles

Steel-Belted Radius lets you define default templates of checklist and return list pairs called *profiles*. A profile provides specific attributes for one or both lists. You can define as many profiles as you require. Profiles provide a powerful means of managing and configuring accounts.

When you edit a User entry, you can assign a profile to the User; the checklist and return list attributes of that profile then become the default settings for the User entry. After you assign a profile to a User entry, you can modify the new entries on the user's checklist and return list. Changes you make apply only to the specific user entry; they do not affect the profile itself. Assigning a profile and then overriding individual attributes is a convenient way to leverage Steel-Belted Radius's features to your advantage.

To change attributes settings across many users immediately, edit the profile that you have assigned to these users. The changes you make to a profile are automatically reflected in each user's checklist and return list.

Adding a Checklist or Return List Attribute for a Profile

A checklist attribute is an item of information that must accompany a request for connection before the connection can be authenticated. A return list attribute is an item of information that Steel-Belted Radius includes in the Access-Accept message when a connection request is approved.

Resolving Profile and User Attributes

If user-specific attributes are stored in an external database, Steel-Belted Radius determines the final set of attributes for a user by merging the attributes stored in the native database with those retrieved from the external database. This calculation is performed as follows:

1 The attributes from the profile (or Alias user) assigned to the user are first retrieved.

2 These attributes are then merged with the user-specific modifications to the attributes in the following manner:

- If the attribute is multi-valued, then the attribute(s) retrieved from the external database is added to the overall list of attributes.
- If the attribute is single-valued, then the attribute(s) retrieved from the external database replaces any attribute of the same name in the profile or associated with the alias.
- If the attribute is orderable, then the attribute(s) retrieved from the external database replaces any orderable attribute of the same name in the profile or associated with the alias.

Setting Up Profiles

Tip: See "User Attribute Lists" on page 24.

The Profiles window (Figure 33) lets you define sets of checklist and return list attributes. You can then assign these profiles to users to simplify user administration.

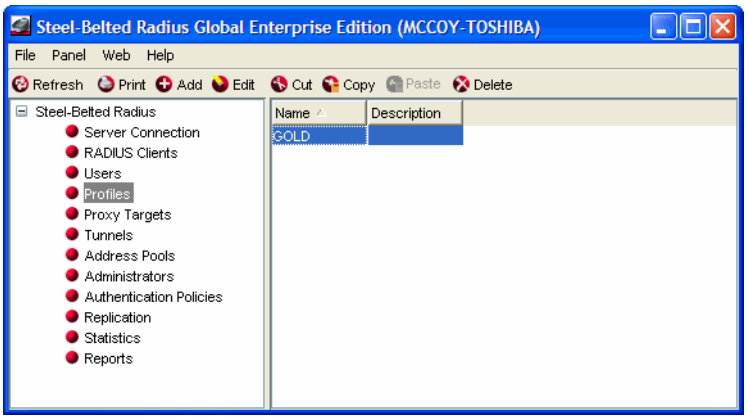


***Figure 33** Profiles Panel*

Adding a Profile

To add a profile:

1 Click **Profiles** to open the Profiles panel.

2 Click the **Add** button on the SBR Administrator toolbar.

The Add Profile window (Figure 34) appears.

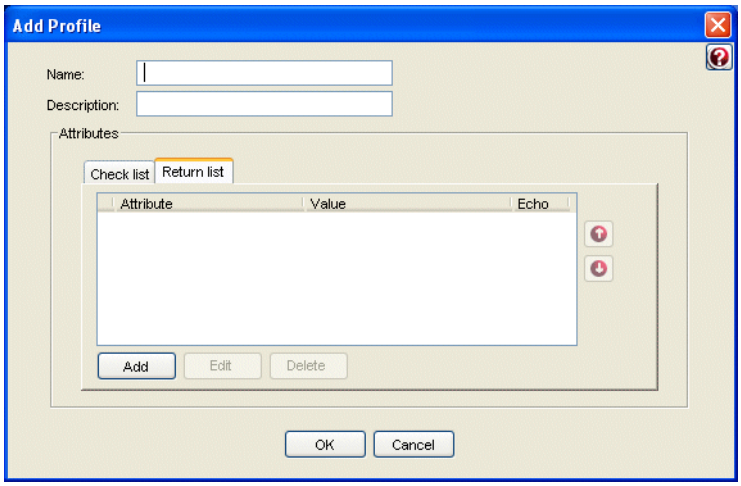


***Figure 34** Add Profile Window*

3 Enter a name for the new profile in the **Name** field.

4 Optionally, enter a description for the profile in the **Description** field.

5 Add checklist and return list attributes to the profile.

a Click the **Checklist** tab or the **Return list** tab.

b Click **Add**. The Add Checklist Attribute window or the Add Return List Attribute window (Figure 35) opens.

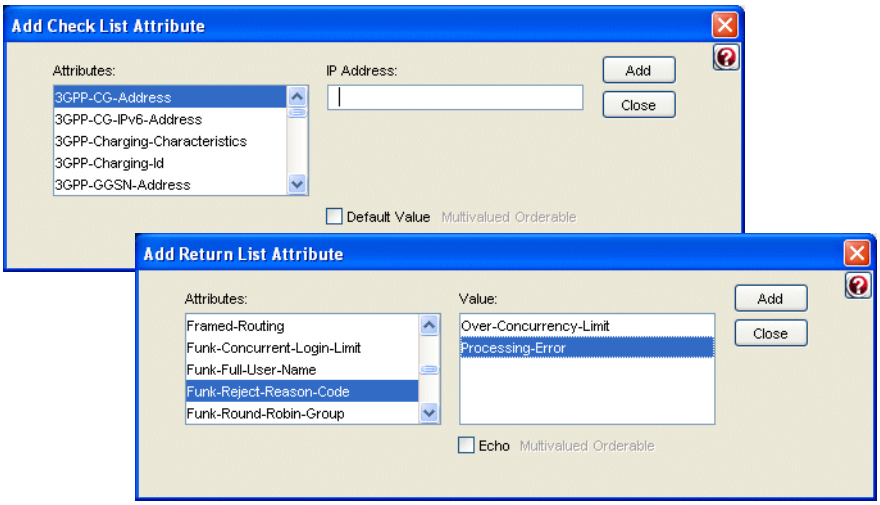


***Figure 35** Add Checklist Attribute and Add Return List Attribute Windows*

c Select the attribute you want to add from the **Attributes** list.

d Select or enter a value for the attribute.

The window changes according to the attribute you choose. Some attributes require that you enter a value, string, or IP address. Other attributes require that you choose from a fixed list of values.

If the **Multivalued** indicator is dimmed, an attribute can have only one value. If the **Multivalued** attribute is undimmed, you can add multiple values for the attribute.

(Checklist attributes only) To set this value to the default value for the attribute (which is useful in situations where the attribute is not included in the RADIUS request), check the **Default value** checkbox.

(Return list single-valued attributes only) If you do not want to specify a particular value, but want to make sure that whatever value of the attribute appears in the RADIUS request is echoed to the client in the RADIUS response, click the **Echo** checkbox.

e Click **Add** to add this attribute/value pair to the list.

f When you are finished adding attribute/value pairs, click **Close** to return to the Add Profile window.

6 Click **OK** to save the profile.

Removing a Profile

To remove a profile:

1 Open the Profiles panel.

2 Select the entry for the profile you want to remove.

3 Click the ***Delete*** button on the SBR Administrator toolbar (or right-click the profile entry and choose ***Delete*** from the context menu).

4 When you are prompted to confirm the deletion, click **Yes**.

Warning: *Do not delete a profile that is assigned to a user. If you delete an active profile, the attributes defined in the profile are removed from users settings, possibly resulting in authentication failures. Steel-Belted Radius warns you if you are deleting an active profile. If you delete the profile anyway, the attributes defined in the profile are removed from that user's settings; when you next display the user's settings, an error message asks you to edit and resave those settings.*

Chapter 7
Administering Proxy Targets

This chapter describes how to set up proxy targets.

Tip: See “Proxy RADIUS” on page 27.

The Proxy Target window (Figure 36) lets you configure Steel-Belted Radius to forward RADIUS packets to another RADIUS server.

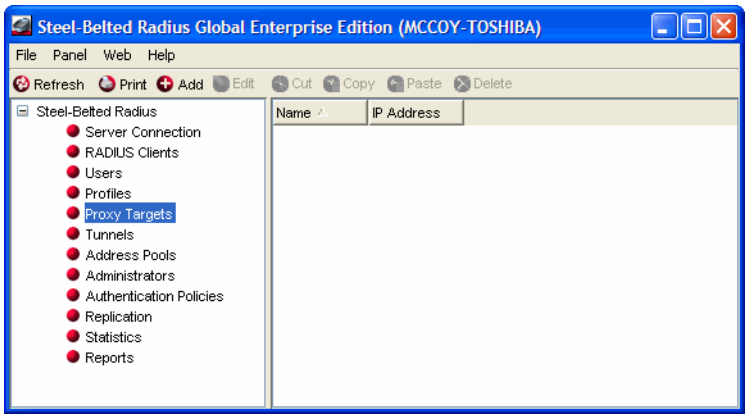


***Figure 36** Proxy Target Window*

Adding a Proxy Target

This section explains how to set up proxy forwarding from the Steel-Belted Radius server (the proxy) to another RADIUS server (the target).

To add a proxy target:

1. Open the Proxy Target panel.
2. Click the **Add** button on the SBR Administrator toolbar.

   The Add Proxy Target window (Figure 37) appears.

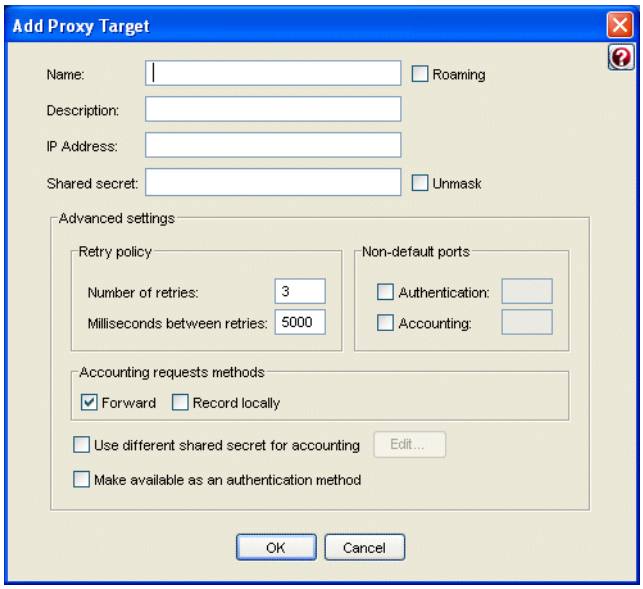


Figure 37 *Add Proxy Target Window*

3 Enter the name of the proxy target in the **Name** field.

The target name must not duplicate any other target name, realm name, or tunnel name in your Steel-Belted Radius configuration. The name you record for a proxy target is not used in processing; Steel-Belted Radius uses the proxy target's IP address to route RADIUS packets.

4 Enter a description for the proxy target in the **Description** field.

5 Enter the IP address or DNS name of the proxy target in the **IP Address** field.

If you enter the DNS name of the proxy target, the SBR Administrator resolves the name you enter to an IP address automatically.

Tip: See "RADIUS Shared Secret" on page 8.

6 Enter the shared secret for the proxy target in the **Shared Secret** field.

Note that shared secrets are case-sensitive.

If you want the characters in the shared secret (rather than asterisks) to appear as you type, click the **Unmask** checkbox.

The shared secret configured for the proxy target in Steel-Belted Radius must match the shared secret configured on the proxy target.

7 Specify how many times Steel-Belted Radius should try to reach the proxy target and how long to wait between attempts in the **Number of retries** and **Milliseconds between retries** fields.

When Steel-Belted Radius acts as a proxy, it emulates the characteristics of a RAS device. This includes the ability to retransmit a request if the first attempt does not get a timely response from the proxy target.

- The **Number of retries** field specifies the number of times a request is retransmitted if an acknowledgment from the target is not received; if the number of retries is exhausted, then the original request is rejected. By default, Steel-Belted Radius retries three times before giving up.

- The **Milliseconds between retries** field specifies the time interval between each retry in milliseconds (thousandths of a second). By default, Steel-Belted Radius waits 5000 milliseconds (5 seconds) between retries.

Tip: See "RADIUS Ports" on page 10.

8 If the proxy target uses non-default ports for authentication or accounting, click the **Authentication** or **Accounting** checkbox and enter the port number you want Steel-Belted Radius to use when exchanging RADIUS authentication or accounting information with the proxy target.

The port numbers configured for the proxy target in Steel-Belted Radius must match the port numbers configured on the proxy target. By default, Steel-Belted Radius uses port 1645 for authentication and port 1646 for accounting.

9 Specify whether you want accounting requests to be forwarded or recorded locally.

- If you click the **Forward** checkbox, Steel-Belted Radius forwards the accounting transaction to the same proxy target that received the authentication transaction.
- If you click the **Record locally** checkbox, Steel-Belted Radius logs the accounting transaction locally (regardless of whether an authentication request was forwarded to the proxy target.

You can click both checkboxes if you want accounting requests to be forwarded and logged locally.

10 If you want Steel-Belted Radius to use a different shared secret for accounting when communicating with the proxy target, click the **Use different shared secret for accounting** checkbox and click the **Edit** button to specify an accounting shared secret.

Refer to "Maintaining an Accounting Shared Secret" on page 88 for information on using the Edit Accounting Shared Secret window.

11 If you want to use a proxy target as an authentication method, click the **Make available as an authentication method** checkbox.

If you enable this option, the name of the proxy target appears in the Authentication Methods tab of the Authentication Policies panel as `proxy:name`. This is useful if you have user records defined on an older RADIUS server and you want to provide a seamless migration to Steel-Belted Radius. Using the older server as a proxy RADIUS target means that RADIUS requests that arrive addressed to this target are handled by Steel-Belted Radius automatically, without requiring end users to change their addressing conventions.

NOTE: *If the proxy target that you are configuring is a member of a proxy RADIUS realm, you should ensure that the* **Make available as an authentication method** *checkbox is unchecked.*

12 Click **OK**.

Ask the administrator at the target site to log into the target server's RADIUS configuration program and add Steel-Belted Radius as a RADIUS client of the target server. You will need to provide this administrator with the IP address of the Steel-Belted Radius server.

Maintaining an Accounting Shared Secret

To specify a shared secret for accounting:

1. Click the **Use different shared secret for accounting** checkbox.
2. Click the **Edit** button.
3. When the Accounting Shared Secret window (Figure 38) opens, enter the shared secret you want Steel-Belted Radius to use.

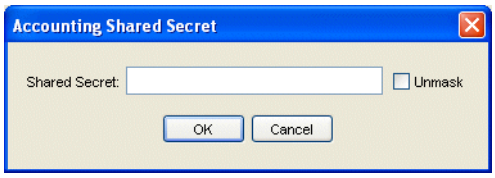


Figure 38 Accounting Shared Secret Window

If you want the characters in the shared secret (rather than asterisks) to appear as you type, click the **Unmask** checkbox. Note that shared secrets are case-sensitive.

4. Click **OK**.

Deleting a Proxy Target

To remove a target server from the proxy target list:

1. In the Proxy Target panel, select the target server you'd like to remove.
2. Click **Delete**.

Steel-Belted Radius as a Target

This section describes how to set up proxy forwarding from some other RADIUS server (the proxy) to the Steel-Belted Radius server (the target):

1. Set up the proxy as a RADIUS client of Steel-Belted Radius.

   Add the entry using the RADIUS Clients panel. Specify the proxy's name, its IP address, and the shared secret that you want to use for encryption between the proxy and Steel-Belted Radius.

2. Ask the administrator at the target site to log into the proxy's RADIUS configuration program and set up Steel-Belted Radius as a proxy RADIUS target. You will need to provide this administrator with the IP address of the Steel-Belted Radius server.

   ***NOTE:** Make sure that the same UDP port and shared secret are entered on both proxy and target sides.*

Dictionaries when Steel-Belted Radius is the Target

When Steel-Belted Radius receives a proxy-forwarded packet, it consults its RADIUS client entry for that proxy server. The **Make/model** field of this entry determines which attribute dictionary Steel-Belted Radius uses.

At various different times, Steel-Belted Radius can receive requests from the same proxy server that have originated from different RAS devices, possibly of different types. The single **Make/model** field that was entered for the proxy might not be adequate to handle the variety of RASs on the "other side" of the transaction.

One way to handle this problem is to add the originating RAS devices to Steel-Belted Radius's list of RADIUS clients. Steel-Belted Radius can be configured to examine each proxy-forwarded packet for clues as to the make and model of the originating device. If clues are found, Steel-Belted Radius does everything it can to map this information to a vendor-specific dictionary, and uses this dictionary in preference to the one for the proxy.

Accepting Packets from Any Proxy

If you'd like Steel-Belted Radius to be able to accept proxy requests from any IP address, you can use the RADIUS Clients panel to add a special entry called `<ANY>`, and specify a shared secret. The `<ANY>` entry permits forwarded requests from any proxy to be accepted, provided the shared secret is correct.

***NOTE:** This feature requires that proxies are configured to use the shared secret you provide in the <ANY> entry.*

Proxy RADIUS as an Authentication Method

Any target proxy RADIUS server can be configured as a Steel-Belted Radius authentication method by enabling the **Make available as an authentication method** checkbox in the Add Proxy Target/Edit Proxy Target window.

A target server can be set up as an authentication method even if the end users do not know anything about the target. That is, a user does not need to log in using a decorated username such as `User@TargetName` to be authenticated by the target server.

If you prioritize the `proxy: TargetName` authentication method above the Native User authentication method in the authentication methods list, the user can log in as `User` and Steel-Belted Radius automatically sends the request to the target for authentication. The authentication succeeds if the UserName and password are stored on the target, but if not, Steel-Belted Radius reaches the Native User method eventually, and the user can then be authenticated.

This technique is useful as a migration path to Steel-Belted Radius from other RADIUS servers. You can set up Steel-Belted Radius as the proxy and the old RADIUS server as the target. After proxy authentication is enabled (in the Proxy Targets panel) and prioritized (in the Authentication Policies panel), Steel-Belted Radius can authenticate users against the old RADIUS server, either as an automatic "first choice" or as an alternative when authentication against the new server's "native" database fails.

Chapter 8
Administering RADIUS Tunnels

This chapter describes how to set up and administer RADIUS tunnels.

About RADIUS Tunnels

A *tunnel* is a uniquely secure type of remote connection. A tunnel passes data between a remote site and an enterprise site, providing an additional layer of encrypted protocol "wrapper" around the data. A tunnel offers authentication and encryption features that help secure the connection against network vandals and eavesdroppers. In addition, a tunnel can provide quality of service features such as guaranteed bandwidth.

***NOTE:** Steel-Belted Radius does not add tunnel functionality to your network. Steel-Belted Radius is able to support the authentication and accounting needs of any tunnels that you've already set up.*

Administration and configuration of the tunnel happens at the remote site, since this is the side of the connection that requests remote access and opens the tunnel. An administrator at the remote site must configure the tunnel with various attributes: its destination IP address, what security protocols it supports, its password, and so on. These attributes are stored in a database to be retrieved when needed to set up a connection.

Storing tunnel attributes on a RADIUS server simplifies tunnel connections. At connection time, the tunnel is established by a RAS device at the remote site. The RAS retrieves the tunnel configuration attributes from the RADIUS server and uses them to open the tunnel into the enterprise. After the tunnel is open, the user can be authenticated at the enterprise.

A RADIUS server is said to support tunnels if it has the ability to store and retrieve the configuration data that a RAS needs to open a tunnel. Steel-Belted Radius fully supports tunnels:

- Steel-Belted Radius can determine from the attributes in the incoming Access-Request whether the connection request involves a tunnel, and if so, which tunnel.
- Steel-Belted Radius can store and retrieve tunnel configuration data.

- Steel-Belted Radius can track the number of tunnels currently in use, compare to a maximum number, and refuse the connection if the number is exceeded.

Tunnel Authentication Sequence

1 Steel-Belted Radius receives an Access-Request message:

2 Steel-Belted Radius checks if the Access-Request contains a Called-Station-Id attribute. If it does, Steel-Belted Radius searches its database for a tunnel entry that contains the indicated telephone number in its Called-Station-Id list.

If a match between the `Called-Station-Id` and a tunnel entry can be found, Steel-Belted Radius constructs an Access-Accept message using the Attributes list in the matching tunnel entry. It then returns the Access-Accept to the client RAS.

NOTE: *If realms are in use, Steel-Belted Radius also searches for this number in its realm configuration files. If a match is found, the Access-Request is routed to the realm, and the quest for a tunnel is abandoned. For this reason, it is important to ensure that DNIS numbers are unique across all tunnel entries and across all realm configuration files.*

3 Steel-Belted Radius checks if the Access-Request contains a username in the form *`User<Delimiter>TunnelName`* or *`TunnelName<Delimiter>User`*. *`<Delimiter>`* is a single character that must match the server's tunnel delimiter character. The order of the realm name relative to the username must match the server's tunnel naming convention (`prefix` or `suffix`). Both of these values are determined per server (that is, all tunnels that use this server must follow the same conventions) by entering them in the Name Parsing tab of the Tunnels panel.

4 Steel-Belted Radius searches its database for a tunnel entry whose name matches the incoming *`TunnelName`*. If a match can be found, Steel-Belted Radius constructs an Access-Accept message using the Attributes list in the matching tunnel entry. It then returns the Access-Accept to the client RAS.

5 If Steel-Belted Radius was able to match the Access-Request with a tunnel entry, the RAS uses the attributes returned in the Access-Accept message to open a tunnel into the enterprise site. Authentication of the User-Name is attempted, usually at the enterprise site. If user authentication succeeds, the connection is complete. Otherwise, the user's connection request is denied.

6 If no matching tunnel entry was found in steps 1 or 2, Steel-Belted Radius concludes that a tunnel is not involved in making this connection. It then continues with its User-Name parsing sequence determine a destination for the authentication request.

Configuring Tunnel Support

To configure Steel-Belted Radius to support a tunnel, you must open the Tunnels panel (describe on page 94) in the SBR Administrator and add a tunnel entry.

A tunnel entry allows you to specify a list of connection Attributes such as the tunnel password, the IP address of the RAS at the enterprise site, encryption conventions to use, and so on. You can also enter the maximum number of tunnels that can be open at one time. You will need to coordinate with the administrator at the enterprise site to get some of this information.

Called Station Id

DNIS (Dialed Number Information Services) refers to a capability that many RAS devices have to determine and use the telephone number that was dialed to make a connection request. The RADIUS standard supports DNIS by specifying the following attributes:

- `Calling-Station-Id` is the number from which the user originated the request.
- `Called-Station-Id` is the telephone number that was dialed to make the network connection.

When setting up a tunnel entry for the Steel-Belted Radius database, you can enter a telephone number or list of numbers in the **Called Station Id** list in the Tunnels panel. This list identifies Called-Station-Id attribute values that the server should expect to find in tunnel connection requests.

Dictionaries for Tunnel Support

The Tunnels panel allows you to create the Attributes list by selecting attributes from a drop-down list. The available selections include attributes from all standard and vendor-specific RADIUS dictionaries installed on the Steel-Belted Radius server.

When the server can accept a tunnel connection request, it consults the corresponding tunnel entry for the list of Attributes to return in the Access-Accept packet. Steel-Belted Radius always returns any standard RADIUS attributes that appear in the Attributes list. It also returns any vendor-specific attributes that are appropriate for the RAS that requested the tunnel connection. Vendor-specific attributes in the Attributes list that do not apply to the requesting RAS are ignored.

Concurrent Tunnel Connections

Steel-Belted Radius tracks the number of active connections for each tunnel. You can limit the number of concurrent connections that can be open through a specific tunnel. When a user requests a new connection through a tunnel, Steel-Belted Radius compares the number of active connections in a tunnel to the maximum number of connections: if a new connection would exceed the limit, Steel-Belted Radius rejects the additional connection.

For concurrent connection limits to work, each RAS that can open a tunnel must be configured for RADIUS accounting and the same Steel-Belted Radius server must be specified for both authentication and accounting.

***NOTE:** Concurrent tunnel connections cannot be tracked across multiple Steel-Belted Radius servers without additional software extensions. Contact Funk Software for more information.*

Configuring RADIUS Tunnels

The Tunnels panel (Figure 39) lets you configure Steel-Belted Radius to support tunnels. When you add a tunnel entry, you are not creating a tunnel; you are enabling

Steel-Belted Radius to support an existing tunnel's authentication and accounting needs and specifying how the server should parse tunnel names.

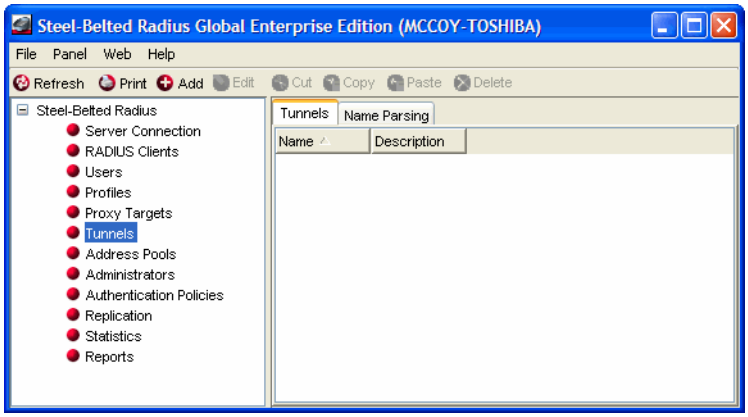


Figure 39 *Tunnels Panel: Tunnels Tab*

Adding a Tunnel

To add a Tunnel entry:

1. Open the Tunnels panel.
2. Click the **Add** button in the Steel-Belted Radius toolbar.

   The Add Tunnel window (Figure 40) opens.

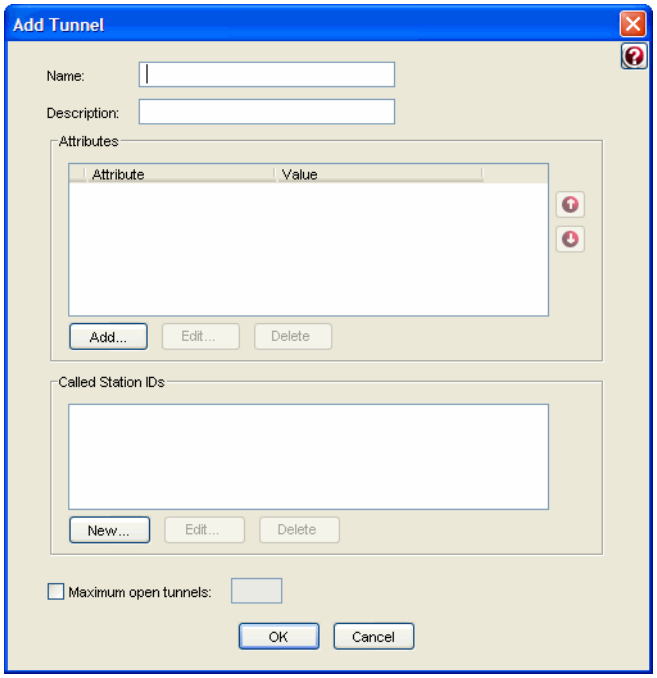


Figure 40 *Add Tunnel Window*

3. Enter the name of the tunnel name in the **Name** field.

Tunnel names do not need to match the actual node name of a client tunnel server. The name you assign to a tunnel must not match the name assigned to a proxy target, realm, or tunnel in your Steel-Belted Radius configuration.

4 Enter a description of the tunnel in the **Description** field.

Tunnel descriptions are used only for administrative purposes and do not affect tunnel connections. This field is typically used to identify the user or organization that uses the tunnel.

5 Associate attributes and values with the tunnel you are setting up.

When a tunnel is used to make a connection, the attributes associated with the tunnel are filtered according to the make/model of the RADIUS client used to establish the connection.

To associate attributes and values with a tunnel:

a Click the **Add** button below the **Attributes** list.

The Add Tunnel Attribute window (Figure 41) opens.

***Figure 41** Add Tunnel Attribute Window*

b Select the attribute you want to add from the **Attributes** list.

c Specify the string or IP address you want to use for the attribute value.

d Click **Add**.

e When you finish adding attributes for the tunnel, click **Close**.

6 Optionally, specify one or more Called Station IDs for the tunnel.

A Called Station ID is a telephone number that was dialed to make a network connection. The Called station ID list identifies the Called-Station-Id attribute values that the server expects to find in tunnel connection requests.

To add one or more Called Station ID numbers for a tunnel:

a Click the **New** button to the right of the Called Station ID list.

The Add Called Station ID window (Figure 42) opens.

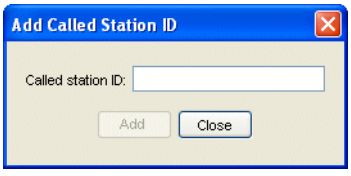


***Figure 42** Add Called Station ID Window*

b Enter the number you want to use in the **Called station ID** field.

c Click **Add**.

Repeat Steps a–c until you have added all called station IDs for the tunnel.

d When you are finished adding called station IDs, click **Close**.

7 If you want to limit the number of connections that can use the tunnel simultaneously, click the **Maximum open tunnels** checkbox and enter the maximum number of tunnels in the **Maximum open tunnels** field.

8 Click **OK**.

Editing a Tunnel

To edit a tunnel entry:

1 Open the Tunnels panel and select the tunnel you want to edit. Click **Edit**.

The Edit Tunnel window (Figure 43) appears.

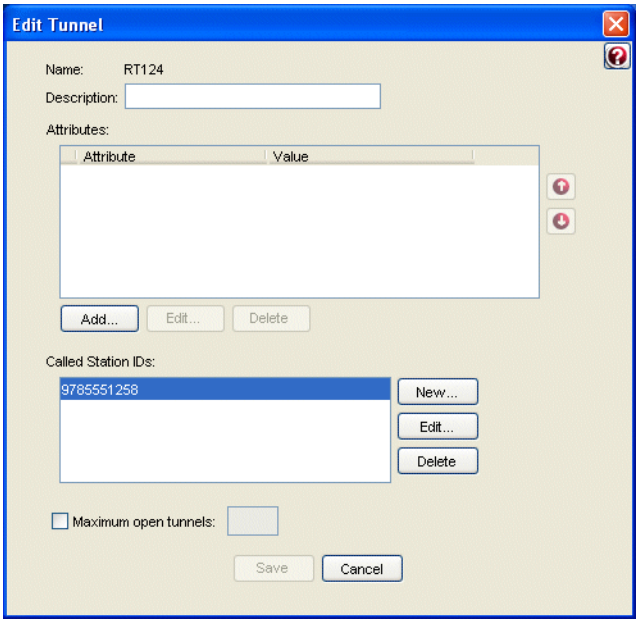


***Figure 43** Edit Tunnel Window*

2 Modify the settings for the tunnel as appropriate.

3 Refer to "Adding a Tunnel" on page 94 for information on how to use the fields and controls on the Edit Tunnel window.

4 When you are finished, click **Save**.

Deleting a Tunnel

To delete a tunnel entry from the Steel-Belted Radius database:

1 Open the Tunnels panel (Figure 39 on page 94).

2 Select the tunnel you want to delete and click the **Delete** button on the Steel-Belted Radius toolbar (or right-click the entry and choose **Delete** from the context menu that appears).

3 When the Confirm Delete window opens, click **Yes**.

Configuring Tunnel Name Parsing

Tunnel name parsing lets Steel-Belted Radius determine whether the name string provided by a user includes a tunnel name by looking for the character configured as the delimiter for tunnel information. Tunnel name parsing options apply to all tunnels maintained by Steel-Belted Radius. You cannot set name parsing options for individual tunnels.

To configure tunnel name parsing:

1 Open the Tunnels panel (Figure 39 on page 94).

2 Click the **Name Parsing** tab to display the Name Parsing window (Figure 44).

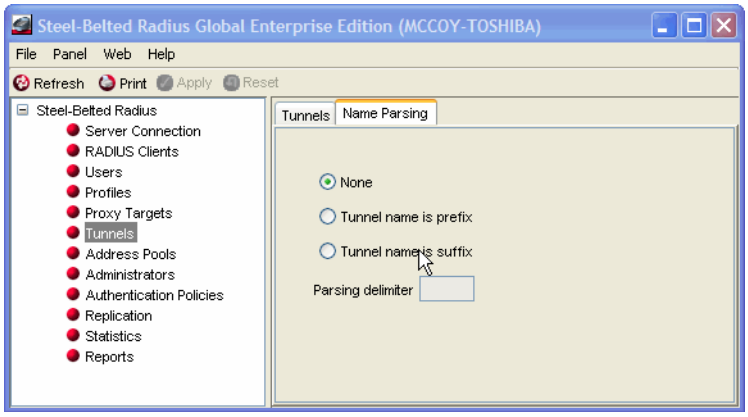


***Figure 44** Tunnel Panel: Name Parsing Tab*

3 Click one of the following radio buttons:

- **None** – Tunnel name parsing is not supported. If you choose this option, the tunnel authentication sequence is bypassed for each Access-Request; the server uses the standard username/password authentication sequence only.
- **Tunnel name is prefix** – If the tunnel delimiter character is detected, the User-Name is assumed to be *TunnelName<PrefixDelimiter>User*.
- **Tunnel name is suffix** – If the tunnel delimiter character is detected, the User-Name is assumed to be *User<SuffixDelimiter>TunnelName*.

The option you choose applies to all tunnels defined on the server.

4 If you clicked **Tunnel name is prefix** or **Tunnel name is suffix**, use the Parsing delimiter field to specify the character used to separate the tunnel name and the username.

The default delimiter character for tunnel name parsing is `@`.

***NOTE:** Choose different delimiter characters and different prefix/suffix name parsing conventions for tunnels and for proxies or realms.*

Chapter 9
Administering Address Pools

This chapter describes how to set up IPv4 and IPX address pools. Steel-Belted Radius does not support IPv6 address pools.

***NOTE:** Please contact Funk Technical Support If you need address pools larger than 65,535 (2^{16}) addresses.*

Address Pool Files

The following files establish settings for IP and IPX address pools. For more information about these files, refer to the *Steel-Belted Radius Reference Guide*.

***Table 16.** Address Pool Files*

File Name	Function
`radius.ini`	Specifies (among other things) the suffixes used to set up RAS-specific IP pools.

Setting Up IP Address Pools

The IP Pools panel allows you to set up one or more pools out of which unique IPv4 or IPX addresses are assigned as users require them. Each pool consists of a list of one or more ranges of addresses.

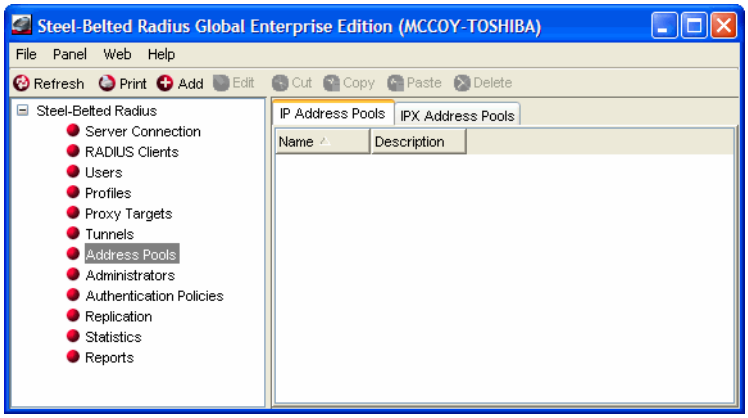


Figure 45 *Address Pools Panel: IP Address Pools Tab*

Adding an IPv4 Address Pool

An IP address pool consists of one or more ranges of IPv4 addresses. You can add or delete ranges and set an optional description for each address pool.

To add an IP address pool:

1. Click the **Address Pools** button to display the Address Pools panel.
2. If necessary, click the **IP Address Pools** tab to display the list of IP address pools that have been configured.
3. Click the **Add** button.

   The Add IP Address Pool window (Figure 46) appears.

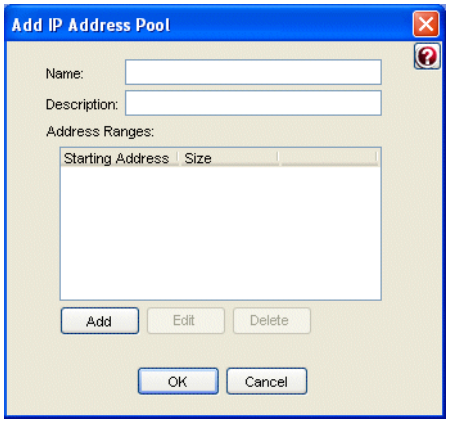


Figure 46 *Add IP Address Pool Window*

4. Enter the name of the IP address pool in the **Name** field.
5. Optionally, enter a description of the address pool in the **Description** field.
6. Identify the address ranges in the IP address pool.
   a. Click the **Add** button below the Address Ranges list.

      The Add IP Address Range window (Figure 47) opens.

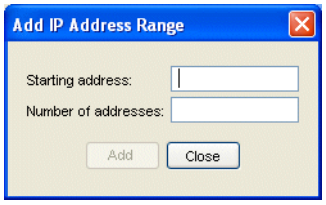


Figure 47 Add IP Address Range Window

b Enter the first address in the in the **Starting address** field.

c Enter the number of addresses in the address range in the **Number of addresses** field.

d Click **Add**.

e Repeat steps a–d for each address range in the IP address pool. When you are finished, click **Close**.

7 Click **OK**.

Editing an IP Address Pool

To edit an IP address pool:

1 Click the **Address Pools** button to display the Address Pools panel.

2 If necessary, click the **IP Address Pools** tab to display the list of IP address pools that have been configured.

3 Select the entry you want to modify and click the **Edit** button (or right-click the entry and choose **Edit**).

The Edit IP Address Pool window (Figure 48) appears.

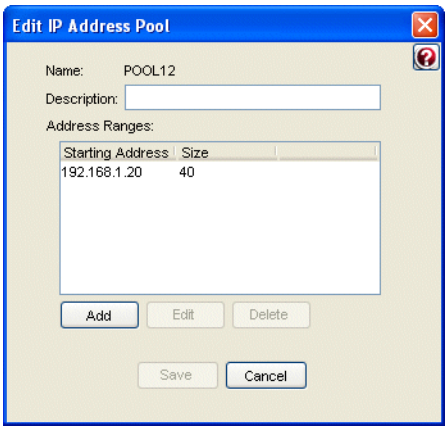


Figure 48 Edit IP Address Pool Window

4 Modify the settings for the address pool as needed.

- To add an address range to the address pool, click the **Add** button and specify the starting address and number of addresses in the range.
- To modify an address range, select it and click the **Edit** button.

- To delete an address range from the address pool, select it and click the **Delete** button.

5 When you are finished, click **Save**.

Removing an IP Address Pool

To delete an IP address pool:

1 Click the **Address Pools** button to display the Address Pools panel.

2 If necessary, click the **IP Address Pools** tab to display the list of IP address pools that have been configured.

3 Select the entry you want to remove and click the **Delete** button (or right-click the entry and choose **Delete**).

4 When you are prompted to confirm the deletion, click **Yes**.

Specifying an IP Address Pool for User/Profile Records

The Framed-IP-Address return list attribute controls how the server assigns an IP address to a user making a connection. When you add or edit the Framed-IP-Address attribute in the Users or Profiles window, the Add Attribute window (Figure 49) allows you to choose an IP address pool instead of specifying an IP address.

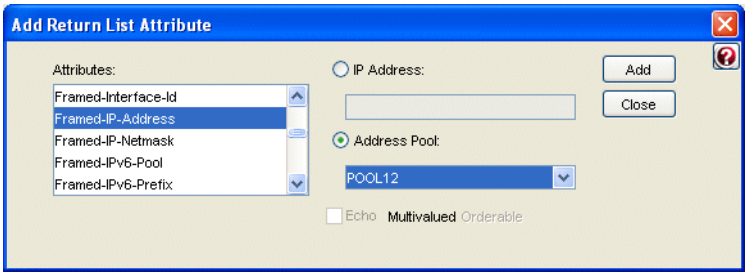


Figure 49 Editing the Framed-IP-Address

RAS-Specific IP Address Pools

Steel-Belted Radius allows you to define IP address pools that are specific to the RAS (RADIUS client) from which the user request was received. You can also define a set of suffixes that define categories of pools. For example, a pool category might correspond to the kinds of services available to users in that category. You might decide to define categories called `Bronze`, `Silver`, and `Gold`, indicating increasing packet routing priorities.

To create a RAS-specific address pool:

1 Define the IP address pool with the IP Pools window (Figure 50).

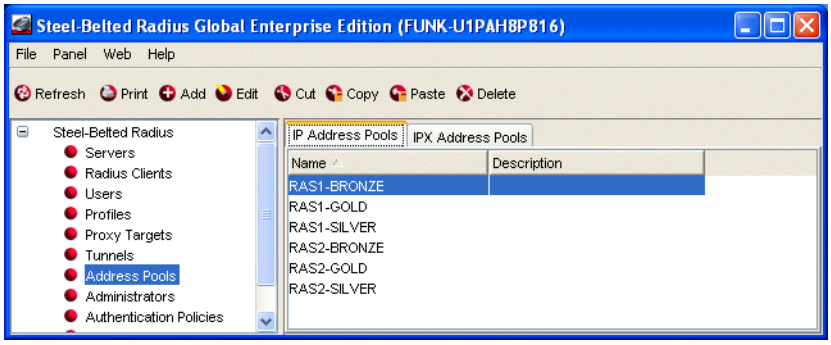


Figure 50 *Address Pools Panel: IP Address Pools Tab*

2 Associate the new IP address pool with the appropriate RAS by use of **IP Address Pool** field on the RADIUS Clients panel.

3 Assign the user to a RAS-specific IP address pool and suffix.

Create this association in the Users panel or the Profiles panel by adding a Framed-IP-Address return list attribute with a value of **pool associated with RAS Client** (Figure 51).

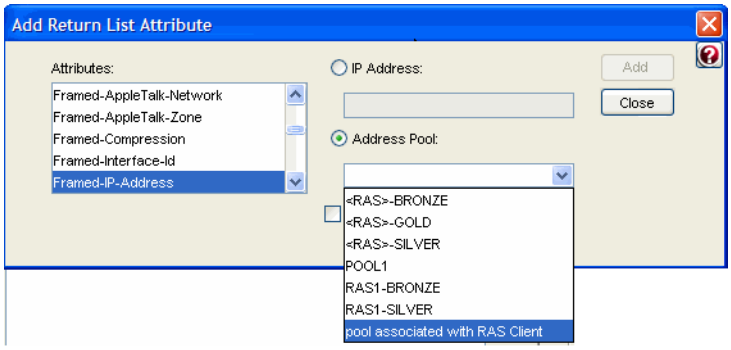


Figure 51 *Assigning the User to a RAS-Specific IP Address Pool*

Setting Up IPX Address Pools

The IPX Pools window (Figure 52) allows you to set up one or more pools out of which unique IPX network numbers are assigned as users require them. Each pool consists of a list of one or more ranges of IPX network numbers.

Warning: *Depending on your overall configuration, certain limitations might apply to this feature. See "How Address Assignment Works" on page 31.*

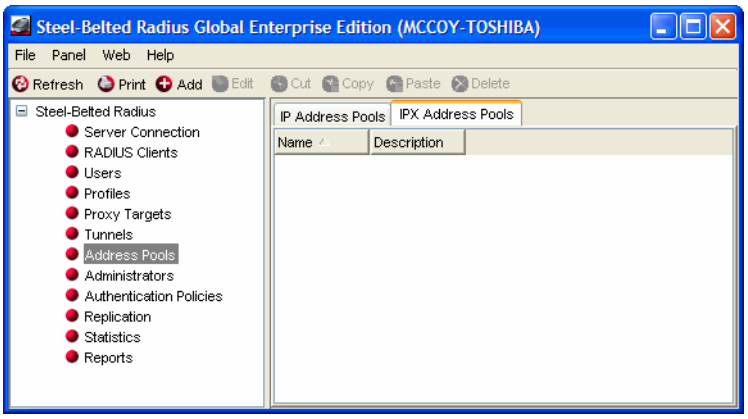


Figure 52 Address Pools Panel: IPX Pools Tab

Adding an IPX Pool

An IPX pool consists of one or more ranges of IPX network numbers. You can add or delete ranges and set an optional description for each address pool.

To add an IPX address pool:

1. Click the **Address Pools** button to display the Address Pools panel.
2. Click the **IPX Address Pools** tab to display the list of IPX address pools that have been configured.
3. Click the **Add** button.

   The Add IPX Address Pool window (Figure 53) appears.

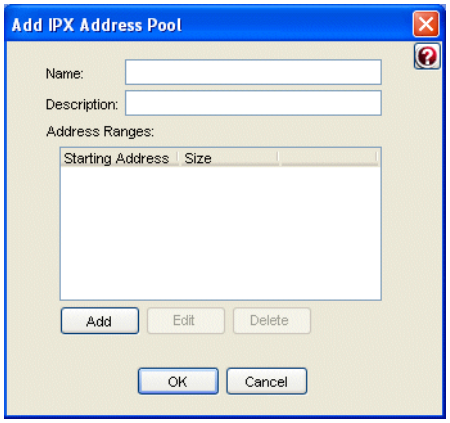


Figure 53 Add IPX Address Pool Window

4. Enter the name of the IPX address pool in the **Name** field.
5. Optionally, enter a description of the address pool in the **Description** field.
6. Identify the address ranges in the IP address pool.
   a. Click the **Add** button below the Address Ranges list.

      The Add IP Address Range window (Figure 54) opens.

Figure 54 Add IP Address Range Window

b Enter the first address in the in the **Starting address** field.

c Enter the number of IPX addresses in the address range in the **Number of addresses** field.

d Click **Add**.

e Repeat steps a–d for each address range in the IPX address pool. When you are finished, click **Close**.

7 Click **OK**.

Editing an IPX Address Pool

To edit an IPX address pool:

1 Click the **Address Pools** button to display the Address Pools panel.

2 Click the **IPX Address Pools** tab to display the list of IP address pools that have been configured.

3 Select the entry you want to modify and click the **Edit** button (or right-click the entry and choose **Edit**).

The Edit IPX Address Pool window (Figure 55) appears.

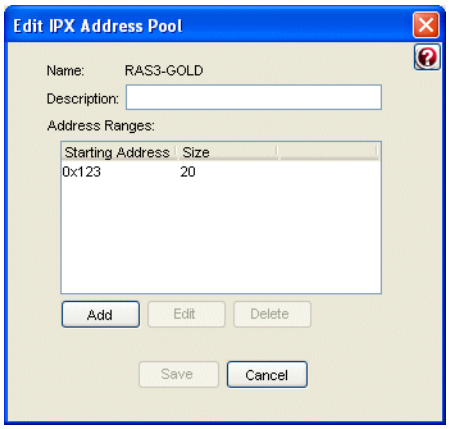


Figure 55 Edit IPX Address Pool Window

4 Modify the settings for the address pool as needed.

- To add an address range to the address pool, click the **Add** button and specify the starting address and number of addresses in the range.
- To modify an address range, select it and click the **Edit** button.

- To delete an address range from the address pool, select it and click the **Delete** button.

5 When you are finished, click **Save**.

Removing an IPX Address Pool

To delete an IPX address pool:

1 Click the **Address Pools** button to display the Address Pools panel.

2 Click the **IPX Address Pools** tab to display the list of IPX address pools that have been configured.

3 Select the entry you want to remove and click the **Delete** button (or right-click the entry and choose **Delete**).

4 When you are prompted to confirm the deletion, click **Yes**.

Specifying Pooled IPX Network Numbers in User/Profile Records

The Framed-IPX-Address return list attribute controls how Steel-Belted Radius assigns an IPX address to a user making a connection.

When you add or edit the Framed-IPX-Address attribute for a user or profile, the Framed-IPX-Address window appears. To select an IPX address assignment option, type an IPX address in the **IPX address** field or click the **IPX Address Pool** checkbox box and select the name of the IPX address pool you want to use from the list.

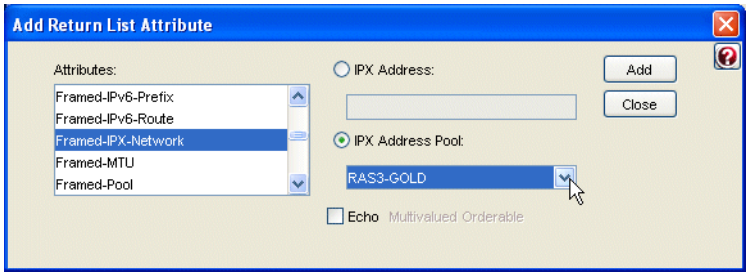


Figure 56 Specifying an IPX Pool for the Framed-IPX-Address Attribute

Chapter 10
Setting Up Administrator Accounts

This chapter describes how to set up Steel-Belted Radius administrators and specify what permissions an administrator holds.

Administrators Panel

The Administrators panel lets you grant and revoke the right to use the SBR Administrator to configure a Steel-Belted Radius server. Each time you request a connection from the Servers panel, SBR Administrator prompts you to authenticate yourself by entering an account name and password.

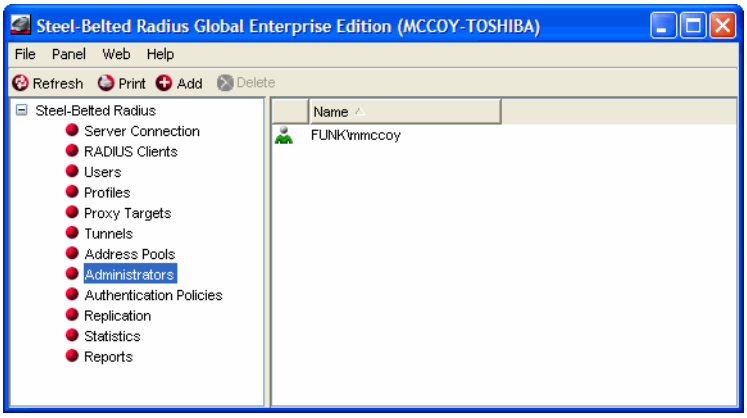


***Figure 57** Administrators Panel*

When a Steel-Belted Radius server is installed, any user who is a member of the group `Administrators` on a Steel-Belted Radius server implicitly has the right to use the SBR Administrator at its default (full) level of access. The Administrators panel lets you modify these default permissions.

The Administrators panel lists the users and groups who have been explicitly granted the right to run the SBR Administrator. Local users or groups are shown with their normal name. Remote users or groups are shown with the name of the domain, followed by a backslash and then the name of the domain user or group.

Adding a Local Administrator

To add a local administrator to the Steel-Belted Radius database:

1. Click the **Administrators** button display the Administrators panel.
2. Click the **Add** button to open the Browse for Administrator window (Figure 58).

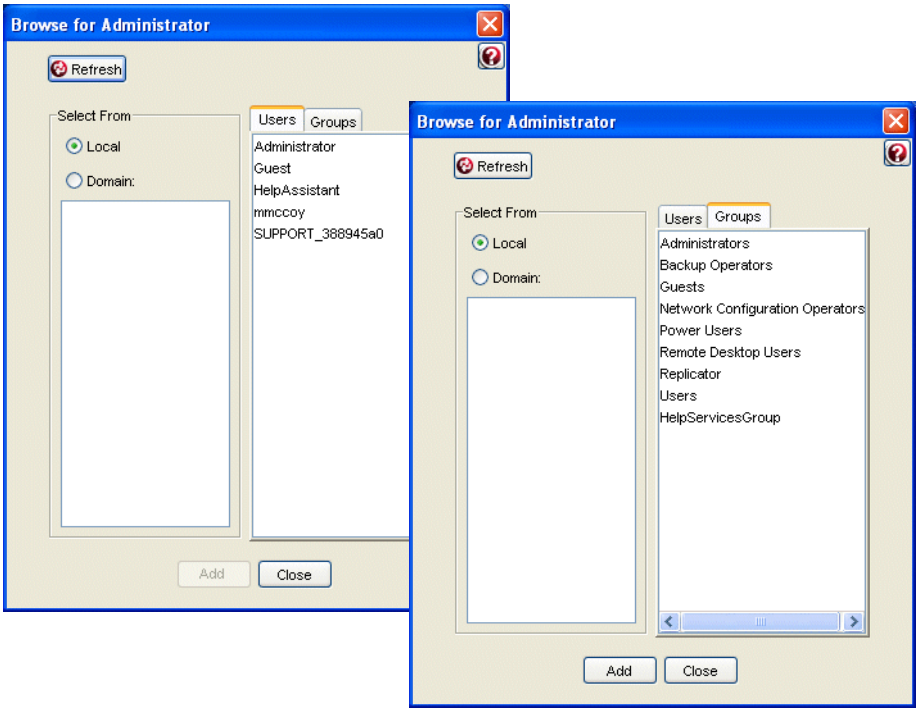


Figure 58 Browse for Administrator Windows

3. Click the **Local** radio button to specify you are adding a local user.
4. Identify the local users or groups you want to add.
   - ▷ If you want to add a local user, click the **Users** tab and select the name of a user.
   - ▷ If you want to add a local group, click the **Groups** tab and select the name of a user group.
5. Click **Add**.
6. Continue adding local users and groups until you are done, then click **Close**.

Adding a Remote Administrator

To grant access to a remote administrator within a domain:

1. Click the **Administrators** button display the Administrators panel.
2. Click the **Add** button to open the Browse for Administrator window (Figure 58).
3. Click the **Domain** radio button to specify you are adding a remote (domain) user.

4 When the list of domains appears, select the domain within which you would like to grant access.

5 Identify the remote users or groups within that domain who you want to add.

- If you want to add a remote user, click the **Users** tab and select the name of a user.
- If you want to add a remote group, click the **Groups** tab and select the name of a user group.

6 Click **Add**.

7 Continue adding domain users and groups until you are done, then click **Close**.

Deleting an Administrator

To revoke rights for a Steel-Belted Radius administrator:

1 Open the Administrators panel.

2 Select the user or group whose administration rights you want to revoke.

3 Click the **Delete** button on the SBR Administrator toolbar (or right-click the entry and choose **Delete**).

4 When you are prompted to confirm the deletion, click **Yes**.

Warning: *Be careful not to revoke your own rights. If you do so, you will no longer have access to Steel-Belted Radius administrative functions.*

Chapter 11
Setting Up Authentication Policies

This chapter presents an overview of concepts relating to the Extensible Authentication Protocol (EAP) and describes how to configure Steel-Belted Radius to use EAP authentication methods and plug-ins.

About the Extensible Authentication Protocol

Steel-Belted Radius supports the Extensible Authentication Protocol (EAP), a standard for communication between clients, access points (APs), remote access server (RAS) devices, and servers that provides for the future extensibility of authentication protocols.

EAP allows specialized knowledge about authentication protocols to be taken out of a RAS or Access Point so that it acts solely as a conduit between authentication server and client. This means that new types of authentication can be supported by adding the appropriate functionality to server and client, without any changes to PPP or RAS devices. When the authentication process is complete, the RADIUS server simply informs the RAS or Access Point of the result.

Steel-Belted Radius supports several EAP authentication mechanisms, such as TTLS, TLS, FAST, PEAP, LEAP, MD5-Challenge, and Generic Token. Support for EAP has been designed to anticipate other authentication types as they become available.

For technical details about EAP, see RFC 2284, "PPP Extensible Authentication Protocol (EAP)," and RFC 2869, "RADIUS Extensions."

Handling EAP Requests

The flow of RADIUS packets in an EAP scenario is quite different from the transactions using standard user credentials (for example, PAP or CHAP). Standard user credentials involve the transmission of a RADIUS request from the RAS (or Access Point) to Steel-Belted Radius and a response (either an Accept or Reject) from the server back to the RAS (or Access Point).

With EAP, the first packet sent from the RAS (or Access Point) to Steel-Belted Radius contains an `EAP-Message` attribute containing an EAP Identity Response. This is a signal sent by the system being authenticated that it wants to be authenticated by means

of EAP. It is now up to Steel-Belted Radius to select the EAP protocol with which it is to authenticate the end-user.

The contents of the `User-Name` attribute is the only guideline available to Steel-Belted Radius in selecting the appropriate EAP protocol. Should Steel-Belted Radius select an EAP protocol that is not supported by the client, the client has the opportunity to send an EAP-NAK and to request a specific alternate protocol.

NOTE: *Given this general flow, a RADIUS request with EAP credentials must incur a minimum of two network round-trips between the RAS (or Access Point) and the Steel-Belted Radius before reaching a successful conclusion.*

Automatic EAP Helpers

Automatic EAP helpers serve as intermediaries between EAP and traditional authentication methods. These helper modules may be configured (using an associated `.eap` file) to work with existing authentication methods to shield the authentication methods from the particulars of the selected EAP protocol.

Table 17 indicates whether each EAP type is implemented as an EAP helper or stand-alone module in Steel-Belted Radius.

Table 17. *EAP Implementations*

EAP-Type	Implemented As
EAP-FAST	Standalone Authentication Method Module
EAP-TTLS	Standalone Authentication Method Module
EAP-TLS	Standalone Authentication Method Module
EAP-TLS	Automatic EAP helper
LEAP	Automatic EAP helper for MS-CHAP-v1
EAP Generic-Token	Standalone Authentication Method Module (SecurID)
EAP MD5-Challenge	Automatic EAP helper for CHAP
EAP MS-CHAP-v2	Automatic EAP helper for MS-CHAP-v2 (needed for PEAP)

Whether an automatic EAP helper can be used in conjunction with a specific authentication method depends on what types of credentials the authentication method supports.

The automatic EAP helper that implements EAP MD5-Challenge generates CHAP credentials, while the helper that implements LEAP generates MS-CHAP-v1 credentials. As such, EAP MD5-Challenge can be used only with authentication methods that support CHAP, and LEAP can be used only with authentication methods that support MS-CHAP-v1.

Table 18 summarizes the support for MS-CHAP-v1 and CHAP in the Steel-Belted Radius authentication methods.

Table 18. MS-CHAP-v1 and CHAP Support

Authentication Method	MS-CHAP-V1	CHAP
LDAP	Yes for BindName (password must be stored in the clear or encrypted using enc-md5 in LDAP server), No for Bind	Yes for BindName (password must be stored in the clear or encrypted using enc-md5 in LDAP server), No for Bind
Native	Yes	Yes
Proxy RADIUS	Yes	Yes
SecurID	No	No
SQL	Yes if password is in clear or encrypted using enc-md5 in SQL database	Yes if password is in clear or encrypted using enc-md5 in SQL database
TACACS+	No	Yes
UNIX User	No	No
UNIX Group	No	No
Windows Domain User	Yes (server must be running under SYSTEM account)	No
Windows Domain Group	Yes (server must be running under SYSTEM account)	No

Authentication Request Routing

The order in which authentication methods and automatic EAP helpers are called to handle an authentication request depends on two factors:

- The ordered list of enabled authentication methods (viewable in the Authentication Policies panel in SBR Administrator). Refer to "Activating Authentication Methods" on page 138 for information on using the Authentication Policies panel.
- The EAP-related configuration for each of the enabled authentication methods in the `eap.ini` file, which you can configure from the Authentication Policies panel.

When Steel-Belted Radius receives an authentication request that does not contain EAP credentials, it passes the request to each enabled authentication method until one of the methods claims the request.

The EAP settings in the `eap.ini` file come into play only when a request with EAP credentials is received. An authentication request contains EAP credentials if it includes one or more `EAP-Message` attributes and contains no other form of user credentials (for example, `User-Password`).

EAP-Only Setting

When an authentication method's `EAP-Only` setting is 1, Steel-Belted Radius prevents the authentication method from being called for any request that does not contain EAP credentials. Under this setting, the authentication method is also bypassed if an authentication request specifically requests an EAP protocol that is not listed in the authentication method's `EAP-Type` list in the `eap.ini` file.

NOTE: *The PEAP authentication method plug-in converts the inner EAP/Generic Token credentials to PAP for security reasons. If you are using SecurID with PEAP, you should set the EAP-Only setting to 0.*

First-Handle-Via-Auto-EAP Setting

If your configuration involves clients using more than one EAP protocol, Steel-Belted Radius must select an initial EAP protocol with which to proceed when receiving an authentication request with EAP credentials.

Selecting the incorrect EAP protocol is not fatal; the client simply sends an EAP NAK in response to the server's selected protocol and suggests an alternate one. After one additional network round-trip, the correct EAP protocol becomes active.

Depending on the capabilities of the authentication methods being used, you may be able to cut out this additional network round-trip that affects a portion of your EAP-based authentication requests.

If an authentication method can check for the existence of a user and can retrieve the user's password information with only the information available in the authentication request (for example, the username), it is said to be *prefetch-capable*. A prefetch-capable authentication method could be consulted first to see if a user exists in its database before committing to a specific EAP protocol.

If your authentication method is prefetch-capable, you would set `First-Handle-Via-Auto-EAP` to 0, indicating that the authentication method should have the first chance to handle the request. You would also set `First-Handle-Via-Auto-EAP` to 0 if the authentication method is capable of handling EAP credentials all on its own (clearly, it would not expect an automatic helper EAP method to do work on its behalf in this case).

By configuring the authentication method to be called first, Steel-Belted Radius can delay selection of an EAP protocol until it has ascertained whether the user exists in a particular authentication method's database. This is a useful technique when you plan to use more than one EAP protocol, but you do not know which one the client will want. Even in this scenario, automatic EAP helpers may still end up performing the EAP protocol processing; they will take over after the authentication method has retrieved a user's password information, rather than before.

The goal of an automatic EAP helper is to generate credentials against which traditional authentication methods (ones that do not understand EAP) can operate. Once an automatic EAP helper has generated these credentials, the authentication method that triggered the use of the helper is checked first for a password/credential match. Should this match not be present, the same traditional credentials are passed to all remaining enabled authentication method in the master list (in the order in which they appear in the list).

Table 19. *Authentication Method Prefetch Capability*

Authentication Method	Prefetch Capable?
LDAP	Yes, if using BindName (rather than the Bind option)

Table 19. Authentication Method Prefetch Capability (Continued)

Authentication Method	Prefetch Capable?
Native User	Yes
SQL	Yes, if password does not need to be used as an input parameter in the SQL statement
UNIX User	No
Windows Domain	No

EAP-NAK Notifications

If you are supporting only one type of client or only one EAP protocol, Steel-Belted Radius selects that EAP protocol for all EAP-based authentication requests it receives. If you are planning to support multiple EAP protocols and do not intend to maintain databases that track the appropriate EAP protocol on a user-by-user basis, Steel-Belted Radius automatically selects the appropriate EAP protocol for you.

When multiple EAP protocols are in play, you should configure each authentication method you plan to use with all the EAP protocols that may be used with it. In this configuration, when Steel-Belted Radius receives an authentication request containing EAP information, it chooses the first EAP protocol listed for the first authentication method that claims the request. Should the client require a different EAP protocol, it sends back an EAP-NAK that specifies the EAP protocol it would prefer to use.

After receiving an EAP-NAK, Steel-Belted Radius performs a scan of the authentication methods, in search of the first authentication method that has the requested EAP protocol listed (the authentication method may support this EAP protocol directly or with the help of an automatic EAP helper).

If the requested EAP protocol does not appear in any of the authentication methods' lists of supported EAP protocols, Steel-Belted Radius rejects the authentication request.

Reauthenticating Connections

Most Access Points understand only a limited number of attributes that may be included in a RADIUS response to signal that the user has been accepted. The `Session-Timeout` attribute is of particular significance in a WLAN realm as it instructs the Access Point how long to allow the user to remain connected to a WLAN before having to re-authenticate to Steel-Belted Radius.

You can configure your choice of `Session-Timeout` settings using standard Steel-Belted Radius reply-list items on a user-by-user basis. If you are using EAP-TLS or EAP-TTLS to authenticate users, you can also have these modules automatically generate `Session-Timeout` attributes based on policies set in their configuration files. This level of control is necessary for EAP-TLS and EAP-TTLS as these modules also support *session resumption*, a quicker method of re-authenticating users. The value in the `Session-Timeout` attribute may need to be dynamically calculated in these cases.

NOTE: *Not all Access Points support the Session-Timeout attribute. You should check your Access Points' specifications to determine whether this configuration must be performed in a fixed manner on the Access Point or if the Access Point should defer to the server.*

Certificates

A *certificate* is an electronic data structure used to identify an individual, a server, a company, or some other entity, and to associate that identity with a public key and an associated private key. Like a passport, a certificate provides generally recognized proof of an entity's identity. Certificates bind public key values to entities, so that remote users of an entity's public key can be certain the associated private key is owned by the correct person or system. Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate will work with the corresponding private key possessed by the entity identified by the certificate. The most widely accepted format for certificates is defined by the ITU-T X.509 international standard, which is described in RFC 3280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile."

Certificate authorities (CAs) are entities that validate identities and issue certificates. An organization that wants to serve as its own CA can issue its own certificates, or an organization can purchase certificates from a trusted third-party CA. The methods used to validate an identity vary depending on the policies of a given CA. In general, before issuing a certificate, a CA must verify the identity of the entity and must digitally sign the certificate to ensure it cannot be modified. This ensures that a certificate issued by a CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee).

In addition to a public key, a certificate includes the name of the entity it identifies, an expiration date, the name of the CA that issued the certificate, a serial number, and the digital signature of the issuing CA, which creates a mathematical relationship between the signing CA certificate's public key and the public key of the certificate it signs. The CA's digital signature allows the certificate to function as a "letter of introduction" for users who know and trust the CA but don't know the entity identified by the certificate.

Because a certificate's expiration date is part of its signed contents, remote entities can verify that a certificate is valid and current.

Common types of certificates include the following:

- Certificate Authority certificates can sign other certificates.
- Server certificates are used on a server to enable a software client to verify the validity of the connection to a machine ("Am I really connecting to www.funk.com?") and to create an encrypted channel between a client and a server.
- Client certificates are used to allow a server to verify a client's identity (certificate based authentication) and to allow a user to digitally sign or encrypt data.

Certificate Chains

A certificate chain is a sequence of certificates, where each certificate in the chain is signed by the certificate above it in the chain. At the top of the chain is a self-signed

certificate. Each CA in the chain vouches for the identity in the entity to which it issues a signed digital certificate. Certificate chains establish a chain of trust; if you trust the CA at the top of the chain, this implies you can trust the signed certificates below it in the chain.

Certificate Revocation Lists

Under normal circumstances, a certificate remains valid until it reaches its expiration date. However, a certificate may become invalid before it expires. For example, if an employee whose identity is bound to a certificate terminates employment or if an enterprise suspects the confidentiality of the private key associated with a certificate's public key has been compromised, the certificate may be declared invalid and revoked.

When a CA revokes a certificate, it must let other entities know the certificate is no longer valid and should not be accepted. A *Certificate Revocation List* is a signed data structure that identifies the serial numbers certificates that have been issued and subsequently revoked by the CA. When a remote entity is asked to use a certificate to verify a remote user's identity, it can download a current copy of the applicable CRL and confirm that the certificate's serial number is not present.

CRLs can be issued by a CA on periodically (hourly, daily, or weekly) or as needed. When a certificate is revoked, its serial number is listed in the CRL, and that serial number remains in the CRL at least one period after the certificate's expiration date. CRLs, like certificates, can be distributed by untrusted servers and untrusted communications.

Under some circumstances, *latency* (the time between when a certificate is revoked and when the certificate's serial number appears on the CRL of the issuing CA) may be a concern. For example, if a revocation is reported today, that revocation will not be reliably notified to certificate-using systems until all currently issued CRLs are updated, which may take hours, days, or even weeks. Online revocation checking can reduce the latency between a revocation report and the distribution of the information to relying parties.

If CRL checking is enabled, Steel-Belted Radius uses the information contained in a client certificate to connect to the certificate's CRL Distribution Point (CDP). Steel-Belted Radius then uses HTTP, LDAP, or a network file system to retrieve the appropriate CRLs. Steel-Belted Radius stores these retrieved CRLs in the `CRLCache` directory under the *`radiusdir`* server directory.

When a client certificate is presented during EAP-TLS authentication, Steel-Belted Radius evaluates the client's certificate chain against its set of stored CRLs to verify none of the certificates in the chain have been revoked.

You can configure the following settings for CRL checking:

- CRL expiration – The CRL checking feature can be configured to operate in *strict* or *lax* mode.
  - In strict mode, a cached CRL that has expired will be immediately discarded; if Steel-Belted Radius cannot acquire a new CRL in the allotted time during a CRL check on a chain, the user is rejected.
  - In lax mode, you can configure Steel-Belted Radius to accept an expired CRL for a a period past its expiration.

Note that Steel-Belted Radius attempts to obtain a current CRL whether it is running in strict or lax mode.

- Missing CDP attribute – When a CRL check is performed on a certificate chain, Steel-Belted Radius reads the contents of the CDP attribute for each certificate past the root certificate and uses the CDP information to retrieve the appropriate CRL. If a non-root certificate in the chain does not contain a CDP attribute, no CRL checking will be performed for that certificate. You can configure EAP-TLS to reject the user if it encounters a non-root certificate that is missing a CDP attribute.
- Incomplete LDAP CDP – Some CAs may create certificates that contain an LDAP-style CDP (`//ldap:\\...`) that does not specify the identity of the LDAP server to be queried. You can designate a default LDAP server that will be used when such CDPs are encountered. If you do not designate a default LDAP server and an LDAP-style CDP is encountered, the CRL retrieval will fail.

EAP Types

EAP-TTLS

EAP-TTLS is a protocol devised by Funk Software and Certicom. It is designed to provide authentication that is as strong as EAP-TLS, but it does not require that each user be issued a certificate. Instead, only the authentication servers are issued certificates. User authentication is performed by password; but the password credentials are transported in a securely encrypted "tunnel" established based upon the server certificates.

1 After the authentication server determines that the user has made an authentication request, it sends its certificate to the user's system.

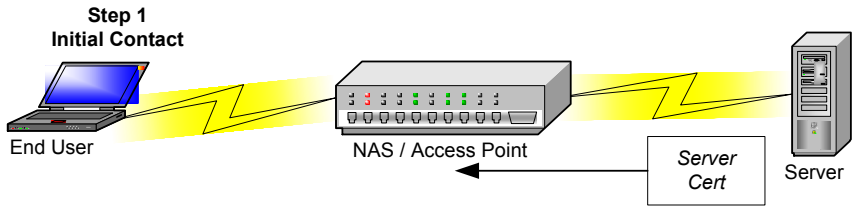


Figure 59 *Server Certificate Sent to RAS*

2 The authentication server's certificate is used to establish a tunnel between the user and the server.

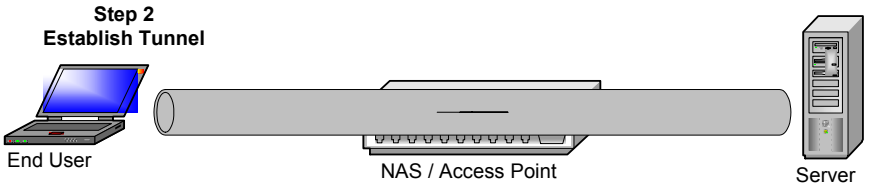


Figure 60 *Tunnel Established*

3 Once the tunnel is established, credentials can be exchanged safely between the server and the user since tunnels encrypt all data in a secure fashion. This stage is called *inner authentication*.

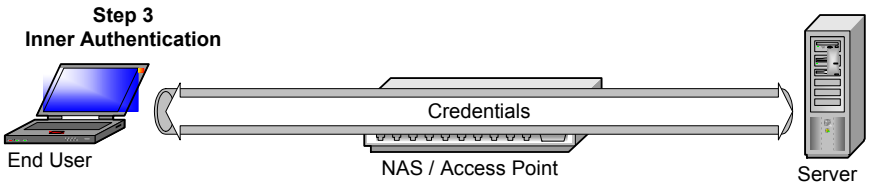


Figure 61 *Inner Authentication*

With EAP-TTLS, it is not necessary to create a new infrastructure of user certificates. User authentication is performed against the same security database that is already in use on the corporate LAN; for example, Windows Domain Controllers, SQL or LDAP databases, or token systems.

The routing of the inner authentication request can be handled via standard Steel-Belted Radius authentication request routing. If your EAP-TTLS tunnel ends at a dedicated server and all the inner authentication requests are to be performed by other servers, you should use standard request routing so the proxy realm target can be determined in a standard fashion (that is, the decoration of the username revealed by inner authentication).

EAP-PEAP

The EAP-PEAP protocol is similar to EAP-TTLS. Unlike EAP-TTLS, which can tunnel any kind of authentication request (such as PAP or CHAP) and extended attributes, PEAP can tunnel only other EAP protocols inside its connection.

EAP-PEAP works in two phases:

- In Phase 1, the client authenticates the server and uses a TLS handshake to create an encrypted tunnel.
- In Phase 2, the server authenticates the user or machine credentials using an EAP authentication protocol. The EAP authentication is protected by the encrypted tunnel created in Phase 1. The authentication type negotiated during Phase 2 can be any valid EAP type, such as GTC (Generic Token Card) or MS-CHAPv2.

Microsoft's implementation of PEAP and Cisco's implementation of PEAP supports different methods of client authentication through the TLS tunnel.

- The Microsoft PEAP implementation requires MS-CHAP-V2 for client authentication.
- The Cisco PEAP implementation supports client authentication by EAP-Generic Token, which Cisco uses both for authenticating token cards and for authenticating users against Windows domain/Active Directory accounts.

The Cisco PEAP implementation supports the ability to hide username identities until the TLS encrypted tunnel is established and authentication phase is complete. The Microsoft PEAP implementation sends the username in clear text in Phase 1 of PEAP authentication.

Steel-Belted Radius supports both Microsoft PEAP and Cisco PEAP.

EAP-FAST

The EAP-FAST protocol enables secure communication between a user and a server by using a TLS handshake to establish a mutually authenticated tunnel. Unlike EAP-TTLS and EAP-PEAP, EAP-FAST does not use a server certificate to secure the exchange. Instead, EAP-FAST uses a high-entropy secret called a Protected Access Credential (PAC) that is known to both the client and the AAA server to secure the TLS handshake.

Steel-Belted Radius generates PACs in a way that allows it to verify a PAC's validity without having to store the individual PACs. This is accomplished by encrypting PACs with a server secret. By default, the EAP-FAST module creates and stores a new server secret every 30 days, though it retains retired server secrets until the PACs encrypted with those secrets expire.

EAP-FAST works in three phases:

- In Phase 0, a client without a valid PAC opens a connection through a RAS to the Steel-Belted Radius server. Steel-Belted Radius uses EAP-MS-CHAPv2 to authenticate the user. If authentication is successful, Steel-Belted Radius generates a PAC for the user. and returns an Access-Reject message that contains the PAC.

  If a client already has a valid PAC, Phase 0 is omitted.

- In Phase 1, the client and the Steel-Belted Radius server establish a TLS tunnel based on the PAC presented by the user. During Phase 1, the server verifies that the PAC presented by the client was generated by its current or retired server secret.

- In Phase 2, the server authenticates the user or machine credentials using EAP-GTC inside the protected TLS tunnel. If the PAC is valid and the user's credentials are correct, the authentication succeeds and Steel-Belted Radius returns an Access-Accept message. If the user's PAC is due to expire soon, Steel-Belted Radius provisions the client with a new PAC over the secure network connection at the end of Phase 2.

***NOTE:** Steel-Belted Radius converts the EAP-GTC inner authentication to Password Authentication Protocol (PAP) format, and it converts the inner EAP-MS-CHAPv2 authentication to MS-CHAPv2.*

If the user's password has expired and if the user's password is stored in a backend server that supports password updates, such as Active Directory, then the PAC provisioning exchange can be used to change the user's password. The password change concludes in the same exchange as the PAP provisioning.

The default lifetime for a user's PAC is 90 days. By default, if Steel-Belted Radius encounters a PAC that will expire within the next 30 days, it provisions a new PAC as part of Phase 2 of the user authentication process. Provisioning a user with a new PAC while the user has a valid (unexpired) PAC is more efficient than issuing a new PAC, since it does not require a failed authentication and Access-Reject message to deliver a provisioned PAC.

If a user presents an expired PAC to Steel-Belted Radius, it ignores the expired PAC and initiates the PAC provisioning sequence starting at Phase 0.

Refer to the *Steel-Belted Radius Reference Guide* for information on how to modify EAP-FAST settings in the `fastauth.aut` file.

EAP-TLS

The EAP-TLS protocol requires that both user and authentication server have certificates for mutual authentication. While the mechanism is very strong, it requires that the corporation that deploys it maintain a certificate infrastructure for all of its users.

EAP-TLS can be deployed as an *authentication method* or as an *automatic EAP helper*.

- EAP-TLS appears in the Authentication Policies panel in SBR Administrator after it is deployed as an authentication method. You can use the Authentication Policies panel to enable the EAP-TLS method and specify its sequence relative to other authentication methods Steel-Belted Radius uses.

  When EAP-TLS is deployed as an authentication method, you can configure it to perform certificate revocation list (CRL) checking. When CRL checking is enabled, EAP-TLS confirms that the client's certificate chain traces back to one of the trusted root certificates installed at initialization and checks the serial number of each certificate in the chain against the contents of CRLs to verify that none of the certificates in the chain have been revoked.

  You can configure the `tlsauth.aut` file to call a fixed profile when TLS-EAP is used. This profile specifies the attributes that are sent back in response to a successful authentication.

  You cannot use secondary authorization when EAP-TLS is deployed as an authentication method.

- When EAP-TLS is deployed as an automatic EAP helper, you must list TLS in the EAP-Type list of an authentication method. When EAP-TLS is triggered, the `tlsauth` authentication goes through the TLS handshake required by the EAP-TLS specification. Assuming the user provides a certificate that the server can verify against a list of trusted root certificates, the EAP-TLS part of the exchange concludes successfully.

  You may not want to grant access to your network to every user with a trusted certificate. By enabling the optional secondary authorization feature of the `tlsauth` plugin, you can have Steel-Belted Radius authorize users with valid certificates on a case-by-case basis. Secondary authorization also allows you to include user-specific attributes in an Access-Accept response; these attributes can be used to communicate options that are to be active for a user's connection to the RAS or Access Point. Without secondary authorization, the only attributes returned on an Access-Accept are those generated by the `tlsauth` plug-in itself (`termination-action` and `session-limit`).

  If the `tlsauth` authentication plug-in is enabled, secondary authorizations must be performed by local authentication methods (they cannot be proxied). The authentication method you select for secondary authorizations must be able to authenticate users in a single pass; it cannot challenge the authorization request and request additional information. The username employed during secondary authorization is derived from a field in the user's certificate. Since a user's certificate

does not include a password, you must configure `tlsauth` to make the secondary authorization request with no password or with a fixed password.

If you configure secondary authorization with no password, your selected authentication method must be capable of handling requests that do not include passwords; the only authentication methods that support this style of authentication and ship with Steel-Belted Radius are Native User, LDAP and SQL. If you configure secondary authorization with a fixed password, you can use any authentication method that supports PAP authentication. In this configuration all user records must have the same fixed password.

LEAP

LEAP is an EAP protocol devised by Cisco that allows a client and server to mutually authenticate each other. Each party does this by confirming that the other has the MD4 hash of the user's password.

LEAP supports the generation of keying material for use in link layer encryption through protocols such as WEP.

EAP Generic-Token

EAP Generic-Token is an EAP protocol that is defined as part of the base EAP specification. It allows for an open-ended exchange between the owner of a security token, such as RSA SecurID and an authentication server.

EAP MD5-Challenge

EAP MD5-Challenge is an EAP protocol that is defined as part of the base EAP specification. It has security characteristics similar to those provided by CHAP credentials, except that in the case of EAP MD5-Challenge, a RADIUS server rather than a RAS generates the challenge bytes.

EAP MS-CHAP-v2

EAP MS-CHAP-v2 is an EAP encapsulation of MS-CHAP-v2 used by Microsoft PEAP and Cisco PEAP as an inner authentication method. It allows authentication to Windows Domains and Active Directory.

EAP Configuration Files

The following files establish settings for configuring the Extensible Authentication Protocol. For more information about these files, refer to the *Steel-Belted Radius Reference Guide*.

Table 20. *EAP Files*

File Name	Function
authtype.aut	Specifies settings for authentication methods, where authtype specifies the name of an authentication method (for example, fastauth.aut for EAP-FAST, ttlsauth.aut for TTLS, and peapauth.aut for EAP-PEAP
authtype.eap	Specifies settings for authentication helpers, where authtype specifies the name of an authentication helper (for example, tlsauth.eap for TLS).
eap.ini	Configures what EAP authentication types are attempted for authenticating users against the different Steel-Belted Radius authentication methods. Each authentication method that you want EAP authentication to be performed against must be configured within this eap.ini file. ***NOTE:*** *You should use the Authentication Polices panel in SBR Administrator to configure the* eap.ini *file.*
radius.ini	Controls (among other things) settings identifying the location of a server's certificate and private key.

Configuring EAP-TTLS and EAP-PEAP

The EAP-TTLS protocol is supported in Steel-Belted Radius by means of the ttlsauth plug-in and EAP-PEAP by means of the peapauth plug-in. EAP-TTLS and EAP-PEAP appear in the list of authentication methods in SBR Administrator. They are configured with similar settings because they operate by means of similar mechanisms. The ttlsauth and peapauth plug-ins require that the server be configured with a server certificate and the accompanying private key.

NOTE: *You can download a time-limited evaluation certificate tool from* http://www.funk.com/RegFiles/sbr30conf.asp.

The inner authentication forwarded by the plug-ins is processed by an attribute filter that can be used to embellish the list of forwarded attributes. Any Access-Accept response may be processed by another attribute filter that can decide which attributes are forwarded on the final RADIUS Access-Accept for the EAP-TTLS or EAP-PEAP exchange.

For EAP-TTLS to operate correctly, the [EAP-TTLS] section of the eap.ini file should set EAP-Type to TTLS and First-Handle-Via-Auto-EAP to 0.

Likewise for EAP-PEAP, the [EAP-PEAP] section of the eap.ini file should set EAP-Type to PEAP and First-Handle-Via-Auto-EAP to 0.

NOTE: *Steel-Belted Radius treats an inner authentication as an authentication request that is separate and distinct from the authentication request that constructs the tunnel. If, for instance, the inner authentication in an EAP-TTLS exchange includes PAP credentials, the authentication is not processed by any authentication method that is marked with EAP-Only=1 in the* eap.ini *file.*

ttlsauth.aut and peapauth.aut Files

The EAP-TTLS and EAP-PEAP plug-ins are configured by modifying the `ttlsauth.aut` and `peapauth.aut` files.

Note that you must configure the [Certificate] section of `radius.ini` to specify the path to the file that describes the server's certificate. For information on `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.

Configuring TTLS for Two-Factor Authentication

Two-factor authentication, described in "Two-Factor Authentication" on page 15, specifies that a client must have both a private key for a valid certificate and the password for an active account to obtain network access.

To configure Steel-Belted Radius to support client certificate checking for EAP-TTLS:

1. Set the Enable parameter in the [Bootstrap] section of the `ttlsauth.aut` file to 1 to enable EAP-TTLS authentication.
2. Set the Require_Client_Certificate parameter in the [Server_Settings] section of the `ttlsauth.aut` file to 1 to specify that clients must provide a certificate as part of the TTLS exchange.
3. If you want Steel-Belted Radius to perform certificate revocation list processing, set the Enable parameter in the [CRL_Checking] section of the `ttlsauth.aut` file to 1.
4. Save the revised `ttlsauth.aut` file.
5. Restart Steel-Belted Radius.

NOTE: *To configure Odyssey Client to use two-factor authentication, you must install a client certificate on the computer running Odyssey Client, and you must configure a profile that tells Odyssey Client to log in using the certificate and to perform inner authentication. For more information, refer to the Odyssey Client Administration Guide and the Odyssey Client User Guide.*

Configuring EAP-TLS as an Automatic EAP Helper

To configure EAP-TLS as an automatic EAP helper, you must set the `First-Handle-Via-Auto-EAP` setting of the authentication method to 1. The server must be configured with a server certificate, the accompanying private key and a list of trusted root certificates from which all client certificates must ultimately derive.

NOTE: *You can use the Odyssey Certificate Authority to configure Steel-Belted Radius with a self-signed root certificate. You can download the Windows-based Odyssey Certificate Authority and its documentation from* `http://www.funk.com/RegFiles/sbr30conf.asp`. *The Odyssey Certificate Authority cannot be used to generate client certificates.*

All trusted root certificates must be DER-encoded, must have a `.der` file extension, and must be placed in the `ROOT` directory. The `ROOT` directory must be created immediately below the directory containing the `radius` daemon and the `radius.ini` file.

tlsauth.eap File

The EAP-TLS automatic EAP helper is configured by modifying the `tlsauth.eap` file.

Note that you must configure the [Certificate] section of `radius.ini` to specify the path to the file that describes the server's certificate. For information on `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.

Configuring Secondary Authorization

The EAP-TLS plug-in may be configured to perform a secondary authorization check that typically requires a traditional authentication method that can be configured to authenticate users without the presence of credentials.

Examples for the Oracle SQL plug-in, the LDAP plug-in, and Native User authentication are provided below.

SQL Authentication

The `.aut` file below shows an example of how the Oracle SQL plug-in on Solaris/Linux can be configured so that password information is not required as input or output.

To configure these two plug-ins to cooperate properly no password has been given in the `SQL= string` entry in the [Settings] section, and the `Password=` entry in the [Results] section has been similarly left empty.

```
[Bootstrap]
LibraryName=radsql_auth_ora.so
Enable=1
InitializationString=Oracle SQL Auth

[Settings]
; OracleInstance is database instance, ($ORACLE_SID from shell)
Connect=OracleUser/OraclePassword@OracleInstance

; Other than procedures, non-interactive SQL statements are not
; terminated
SQL=SELECT FullName FROM orasqlauth WHERE username = %Name/50s
ParameterMarker=?
MaxConcurrent=1
ConcurrentTimeout=10
ConnectTimeout=10
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=15

; LogLevel and TraceLevel for this plugin, regardless of
radius.ini
LogLevel=0
TraceLevel=0
```

```
[Results]
; Empty definition of Password= indicates password to be
ignored,
; since EAP-TLS is assumed to have already authenticated the
user.
Password=
FullName=1/255s
;Profile=2/48
;Alias=3/48

[Failure]
;Accept=0
;Profile=xyz
;FullName=Remote User
```

If the SQL authentication method used for secondary authorization is intended to be used only in conjunction with EAP-TLS, you should set `EAP-Only=1` and `EAP-Type=TLS` in the appropriate section of the `eap.ini` file to prevent unintended use of this SQL authentication method for traditional authentication requests.

LDAP Authentication

The `.aut` file below shows an example of how the LDAP plug-in can be configured so that password information is not required as input or output.

To configure the EAP-TLS and LDAP plug-ins to cooperate properly, the `BindName=` option has been utilized in the [Settings] section to log into the LDAP server and no `%password=` setting has been specified in the [Response] section.

```
[Settings]
MaxConcurrent=2
Timeout=20
ConnectTimeout=5
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=30
BindName=uid=admin,ou=administrators,o=bigco.com
Bindpassword=adminPassword
LogLevel=2
UpperCaseName=0
PasswordCase=original
Search=DoLdapSearch
SSL=0

[Server]
s1=

[Server/s1]
Host=199.185.162.147
Port=389

[Request]
%Username=User-Name
```

```
[Response]
%profile=TheUserProfile

[Search/DoLdapSearch]
Base=ou=Special Users,o=bigco.com
Scope=2
Filter=(uid=<User-Name>)
Attributes=AttrList
Timeout=20
%DN=dn

[Attributes/AttrList]
userprofile
```

If the LDAP authentication method used for secondary authorization is intended to be used only in conjunction with EAP-TLS, you should set `EAP-Only=1` and `EAP-Type=TLS` in the appropriate section of the `eap.ini` file to prevent unintended use of this LDAP authentication method for traditional authentication requests.

Native User Authentication

The only requirements for using EAP-TLS in conjunction with Native User authentication is that appropriate values are given to the `First-Handle-Via-Auto-EAP` and `EAP-Type` settings in the `eap.ini` file.

Configuring EAP-TLS as an Authentication Method

To configure EAP-TLS as an authentication method:

1. Enable loading of the EAP-TLS module by editing the tlsauth.aut file.

   The `tlsauth.aut` file is stored in the Steel-Belted Radius directory.

2. Re-start Steel-Belted Radius.

   Restarting the server causes the EAP-TLS module to be loaded.

3. Verify that EAP-TLS in activated in the Authentication Policies panel.

   In Steel-Belted Radius, authentication methods can be enabled (loaded) without being activated (in use).

4. Position EAP-TLS in the list of authentication methods.

All trusted root certificates must be DER-encoded, must have a `.der` file extension, and must be placed in the `ROOT` directory. This directory must be created immediately below the directory containing the `radius` daemon and the `radius.ini` file.

tlsauth.aut File

The EAP-TLS authentication method is configured by modifying the `tlsauth.aut` file.

Note that you must configure the [Certificate] section of `radius.ini` to specify the path to the file that describes the server's certificate. For information on `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.

Examples of EAP Usage

This section contains examples of the way in which EAP can be configured in Steel-Belted Radius.

EAP-TTLS Example

A customer that uses Steel-Belted Radius wants to extend its application by deploying wireless LANs (WLANs) as part of its internal network. Before deploying the WLAN technology, Steel-Belted Radius was used to authenticate all end-user remote access coming through a VPN gateway and all firewall traversals. User credentials from the VPN gateway used MS-CHAP-v1 while the credentials originating from the firewall used PAP. Users are expected to provide Windows Domain passwords to be granted access through Steel-Belted Radius.

The software being deployed for WLANs on all clients supports EAP-TTLS. The type of user credentials used in the EAP-TTLS inner authentication is MS-CHAP-v1. The passwords that users are expected to provide to their WLAN client software are their Windows Domain passwords.

Perform the following steps to enable this configuration:

1. Enable loading of the EAP-TTLS module.

   This is done by editing the `ttlsauth.aut` file in the Steel-Belted Radius directory and changing the value of `Enable` in the [Bootstrap] section to 1.

2. Re-start Steel-Belted Radius .

   Restarting Steel-Belted Radius causes the EAP-TTLS module to be loaded.

3. Verify that EAP-TTLS in activated in the Authentication Policies panel.

   In Steel-Belted Radius, authentication methods can be enabled (loaded) without being activated (in use).

4. Move EAP-TTLS to the top of the list of authentication methods.

   While this is, strictly speaking, not necessary, it is a bit more efficient than leaving the Windows Domain authentication method at the top of the list.

When authentication requests containing MS-CHAP-v1 or PAP credentials are received, Steel-Belted Radius skips the EAP-TTLS method (since it is marked with `EAP-Type=TTLS` and `EAP-Only=1`) and passes the request to the same authentication method (Windows Domain) as before.

When the request does contain EAP user credentials, the EAP-TTLS method receives the request, performs the TLS handshake (this requires multiple round-trips) and, upon completion of the handshake, creates a new request containing MS-CHAP-v1 user credentials. This new request skips past the EAP-TTLS method (no new request created

by the EAP-TTLS method is passed back to it) and is authenticated by the Windows Domain authentication method.

At the conclusion of a successful inner authentication, the default configuration for EAP-TTLS creates one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

LEAP Example

A customer that uses Steel-Belted Radius wants to extend its application by deploying wireless LANs (WLANs) as part of its internal network. Before deploying the WLAN technology, Steel-Belted Radius was used to authenticate all end-user remote access coming through a VPN gateway and all firewall traversals. User credentials from the VPN gateway used MS-CHAP-v1 while the credentials originating from the firewall used PAP. Users are expected to provide Windows Domain passwords to be granted access through Steel-Belted Radius.

In this example, the software being deployed for WLANs on all clients supports LEAP. Users are expected to provide their Windows Domain passwords to their WLAN client software.

Nothing needs to be done to load LEAP support. LEAP is implemented as an automatic EAP helper that is part of the core Steel-Belted Radius server.

Perform the following steps to enable this configuration:

1. Edit the `eap.ini` file to specify use of LEAP with the Windows Domain authentication method.

   The following configuration needs to appear in the `eap.ini` file:

   ```
   [Windows-Domain]
   EAP-Only = 0
   EAP-Type = LEAP
   First-Handle-Via-Auto-EAP = 1
   ```

   The `First-Handle-Via-Auto-EAP` setting may be 0 or 1 in this case with no significant difference in the outcome.

2. Restart the Steel-Belted Radius service/process.

   Restarting Steel-Belted Radius causes the `eap.ini` file to be re-read.

When authentication requests containing MS-CHAP-v1 or PAP credentials are received, Steel-Belted Radius passes the request to the same authentication method as before.

When the request does contain EAP user credentials, the automatic EAP helper method that supports LEAP receives the request, performs the required operations (this requires multiple round-trips) and, half-way through these operations, calls the Windows Domain authentication with MS-CHAP-v1 user credentials. If the Windows Domain authentication succeeds, the LEAP method continues (in the remaining phase, Steel-Belted Radius must authenticate itself to the LEAP software running on the client).

When the LEAP automatic EAP helper successfully authenticates a user, it creates one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept

response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

LEAP and EAP MD5-Challenge Example

In this example, we assume that you are planning to deploy a WLAN. The software being deployed for WLANs on some of the clients supports EAP MD5-Challenge (with static WEP keys) while on other clients, LEAP is the only supported protocol. All user accounts and passwords are stored in a SQL database, with the passwords having been stored in clear text or reversibly encrypted form. Users must provide a correct username and password to be admitted to the WLAN.

Nothing needs to be done to load LEAP support or EAP MD5-Challenge support. LEAP and EAP MD5-Challenge are implemented as automatic EAP helpers.

Perform the following steps to enable this configuration:

1. Enable loading of the SQL authentication module.

   This is done by editing the `sqlauth.aut` file in the Steel-Belted Radius directory to reflect your SQL setup and by changing the value of `Enable` in the [Bootstrap] section to 1.

1. Edit the `eap.ini` file to specify use of LEAP and EAP MD5-Challenge with the SQL authentication method.

   The following configuration needs to appear in the `eap.ini` file:

   ```
   [SQL]
   EAP-Type = LEAP, MD5-Challenge
   First-Handle-Via-Auto-EAP = 0

   [SQL-ORACLE]
   EAP-Type = LEAP, MD5-Challenge
   First-Handle-Via-Auto-EAP = 0
   ```

   The `EAP-Only` setting may be 0 or 1 in this case. Set it to 0 if you also want to support traditional (non-EAP) authentication via the SQL authentication method.

   If more than half of your clients support LEAP, make it the first EAP type in the list. Otherwise, you may want to make EAP MD5-Challenge the first EAP type in the list. Any clients for whom the EAP protocol selected by Steel-Belted Radius proves incorrect sends back an EAP-NAK requesting a different EAP protocol.

2. Restart Steel-Belted Radius.

   Restarting Steel-Belted Radius causes the SQL authentication module to be loaded and the `eap.ini` file to be re-read.

3. Verify that the SQL authentication method is activated in the Authentication Policies panel.

In Steel-Belted Radius, authentication methods can be enabled (loaded) without being activated (in use).

Assuming that LEAP was ordered ahead of EAP MD5-Challenge, when a request does contain EAP user credentials, the automatic EAP helper that implements LEAP gets to handle the request first. It sends the client an invitation to begin a LEAP sequence. If the

client supports LEAP, the exchange proceeds and (after several round-trips) the LEAP module generates MS-CHAP-v1 user credentials for the user. These user credentials are then passed to the SQL authentication module, which determines its ultimate fate.

Should it succeed, the LEAP module generates one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

If the client supports EAP MD5-Challenge, it responds to the invitation to begin an LEAP exchange with an EAP-NAK indicating that it would prefer to use EAP MD5-Challenge. Steel-Belted Radius looks for the first authentication method in its list that identifies EAP MD5-Challenge as a supported protocol.

The SQL authentication method is also configured for use with EAP MD5-Challenge. Since `First-Handle-Via-Auto-EAP` is set to 1, the request is first passed to the automatic EAP helper that implements EAP MD5-Challenge. It performs the required operations (this requires two round-trips) and generates CHAP user credentials for the user. These user credentials are then passed to the SQL authentication module, which again determines its ultimate fate.

For clients that are authenticated via EAP MD5-Challenge, keying material is transmitted back to the RAS only if a profile containing the `MS-MPPE-Send-Key` and `MS-MPPE-Recv-Key` attributes is specified by the SQL authentication method.

EAP-TTLS and LEAP Example

Let us again assume that authentication requests containing MS-CHAP-v1 or PAP credentials are being handled by the Windows Domain authentication method of Steel-Belted Radius.

In this example, the software being deployed for WLANs on some of the clients supports EAP-TTLS while on other clients, LEAP is the only supported protocol. Users are once again be expected to provide their Windows Domain passwords to their WLAN client software.

Nothing needs to be done to load LEAP support. LEAP is implemented as an automatic EAP helper that is part of the core Steel-Belted Radius server.

Perform the following steps to enable this configuration:

1. Enable loading of the EAP-TTLS module.

   This is done by editing the `ttlsauth.aut` file in the Steel-Belted Radius directory and changing the value of Enable in the [Bootstrap] section to 1.

2. Edit the `eap.ini` file to specify use of LEAP with the Windows Domain authentication method.

   The following configuration needs to appear in the `eap.ini` file:

   ```
   [NT-Domain]
   EAP-Only = 0
   EAP-Type = LEAP
   First-Handle-Via-Auto-EAP = 1
   ```

   The `First-Handle-Via-Auto-EAP` setting may be 0 or 1 in this case with no significant difference in the outcome.

3 Restart Steel-Belted Radius.

Restarting Steel-Belted Radius causes the EAP-TTLS module to be loaded and the `eap.ini` file to be re-read.

4 Verify that EAP-TTLS is activated in the Authentication Policies panel.

In Steel-Belted Radius, authentication methods can be enabled (loaded) without being activated (in use).

5 Set the order of authentication method in the Authentication Policies panel.

If more than half of your clients support EAP-TTLS, make it the first authentication method in the list. If more than half your clients support LEAP, make Windows Domain authentication the first method in your list. Any clients for whom the EAP protocol selected by Steel-Belted Radius proves incorrect send back an EAP-NAK requesting a different EAP protocol.

When authentication requests containing MS-CHAP-v1 or PAP credentials are received, Steel-Belted Radius passes the request to the same authentication method (Windows Domain) as before.

Assuming that EAP-TTLS was ordered ahead of Windows Domain authentication, when the request does contain EAP user credentials, the EAP-TTLS module gets to handle the request first. It sends the client an invitation to begin an EAP-TTLS handshake. If the client supports EAP-TTLS, the handshake proceeds and (after several round-trips) a new authentication request results from the inner authentication sequence. Assuming this request contains MS-CHAP-v1 credentials and EAP-TTLS is set up to perform standard routing, the new request is passed to the Windows Domain authentication method, which determines its ultimate fate.

Should it succeed, the EAP-TTLS module may (based on its configuration information) generate one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

If the client supports LEAP, it responds to the invitation to begin an EAP-TTLS handshake with an EAP-NAK indicating that it would prefer to use LEAP. The Steel-Belted Radius looks for the first authentication method in its list that identifies LEAP as a supported protocol. The Windows Domain authentication method is configured for use with LEAP, but since `First-Handle-Via-Auto-EAP` is set to 1, the request is first passed to the automatic EAP helper that implements LEAP. It performs the required operations (this requires multiple round-trips) and, halfway through these operations, calls the Windows Domain authentication with MS-CHAP-v1 user credentials. Should the Windows Domain authentication succeed, the LEAP method continues (in the remaining phase, the Steel-Belted Radius server must authenticate itself to the LEAP software running on the client).

When the LEAP automatic EAP helper successfully authenticates a user, it creates one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

EAP-TTLS and EAP-TLS Example

In this example, we assume that a customer is looking to deploy Steel-Belted Radius for the sole purpose of supporting a new Wireless LAN deployment within the company's enterprise network.

In this example, the software being deployed for WLANs on some of the clients supports EAP-TTLS while on other clients, EAP-TLS is the only supported protocol. The users of EAP-TTLS clients are expected to enter their Windows Domain user password to gain access to the WLAN, while the EAP-TLS clients must have been configured with a user certificate, matching private key and a list of root certificates from which server certificates must derive.

The customer has already deployed a PKI and end-users have already been supplied with certificates independent of the latest WLAN deployment, though only a subset of users that have certificates are to be allowed access to the WLAN. The identities of the users that are to be allowed access via the WLAN are stored in a SQL database.

Perform the following steps to enable this configuration:

1. Enable loading of the EAP-TTLS module.

   This is done by editing the `ttlsauth.aut` file in the Steel-Belted Radius directory and changing the value of `Enable` in the [Bootstrap] section to 1.

2. Enable loading of the EAP-TLS module.

   This is done by editing the `tlsauth.eap` file in the Steel-Belted Radius directory and changing the value of `Enable` in the [Bootstrap] section to 1.

3. Enable loading of the SQL authentication module.

   This is done by editing the `sqlauth.aut` file in the Steel-Belted Radius directory and changing the value of `Enable` in the [Bootstrap] section to 1. You must also configure the SQL authentication method to be compatible with EAP-TLS.

Tip: See "Configuring Secondary Authorization" on page 125.

4. Edit the `eap.ini` file to specify use of EAP-TLS with the SQL authentication method and use of no EAP protocols with Windows Domain authentication.

   The following configuration needs to appear in the `eap.ini` file:

   ```
   [NT-Domain]
   EAP-Only = 0

   [sqlauth]
   EAP-Only = 1
   EAP-Type = TLS
   First-Handle-Via-Auto-EAP = 1
   ```

   The `First-Handle-Via-Auto-EAP` setting must be set to 1 for the EAP-TLS to operate properly.

5. Re-start Steel-Belted Radius.

   Restarting Steel-Belted Radius causes the EAP-TTLS and EAP-TLS modules to be loaded and the `eap.ini` file to be reread.

6. Verify that EAP-TTLS and SQL authentication is activated in the Authentication Policies panel.

In Steel-Belted Radius, authentication methods can be enabled (loaded) without being activated (in use).

7 Set the order of authentication method in the Authentication Policies panel.

If more than half of your clients support EAP-TTLS, make it the first authentication method in the list. If more than half your clients support EAP-TLS, make SQL authentication the first method in your list. Any clients for whom the EAP protocol selected by Steel-Belted Radius proves incorrect send back an EAP-NAK requesting a different EAP protocol. The placement of Windows Domain authentication in the list does not affect overall operation.

Assuming that EAP-TTLS was ordered ahead of SQL authentication, when the request does contain EAP user credentials, the EAP-TTLS module gets to handle the request first. It sends the client an invitation to begin an EAP-TTLS handshake. If the client supports EAP-TTLS, the handshake proceeds and (after several round-trips) a new authentication request results from the inner authentication sequence.

Assuming this request contains MS-CHAP-v1 credentials and EAP-TTLS is set up to perform standard routing, the new request is passed to the Windows Domain authentication method, which determines its ultimate fate. Should it succeed, the EAP-TTLS module may (based on its configuration information) generate one MS-`MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

If the client supports only EAP-TLS, it responds to the invitation to begin an EAP-TTLS handshake with an EAP-NAK, indicating that it would prefer to use EAP-TLS. Steel-Belted Radius looks for the first authentication method in its list that identifies EAP-TLS as a supported protocol. The SQL authentication method is configured for use with EAP-TLS, but since `First-Handle-Via-Auto-EAP` is set to 1, the request is first passed to the automatic EAP helper that implements EAP-TLS.

The helper performs the EAP-TLS handshake (this requires multiple round-trips) and, upon successful conclusion of the handshake retrieves a specific piece of information from the client certificate presented to it during the handshake (this can be the `Principal-Name` or the least-significant CN portion of the `Subject-Name` attribute of the certificate). The EAP-TLS module then passes a secondary authorization check back to Steel-Belted Radius, which asks the SQL authentication method to verify that the user exists in the SQL database. Should the SQL authentication concur, the authentication request is considered successful.

The EAP-TLS module may (based on its configuration information) generate one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

EAP-PEAP Example

In this example, we assume that you already use Steel-Belted Radius but are now planning to extend its application by deploying Wireless LANs within your company's enterprise network.

Let us say that prior to deployment of the WLAN technology, Steel-Belted Radius was being used to authenticate all end-user remote access (via a VPN gateway) and all firewall traversals. User credentials from the VPN gateway used MS-CHAP-v1 while the credentials originating from the firewall used PAP. Users are expected to provide Windows Domain passwords to be granted access through Steel-Belted Radius.

The software being deployed for WLANs on all clients supports EAP-PEAP. The type of user credentials used in the EAP-PEAP inner authentication is EAP-MS-CHAP-v2. Users are expected to provide their Windows Domain passwords as their passwords to their WLAN client software.

Perform the following steps to enable this configuration:

1. Enable loading of the EAP-PEAP module.

   This is done by editing the `peapauth.aut` file in the Steel-Belted Radius directory and changing the value of `Enable` in the [Bootstrap] section to 1.

2. Restart Steel-Belted Radius.

   Restarting Steel-Belted Radius causes the EAP-PEAP module to be loaded.

3. Verify that EAP-PEAP in activated in the Authentication Policies panel.

   In Steel-Belted Radius, authentication methods can be enabled (loaded) without being activated (in use).

4. Move EAP-PEAP to the top of the list of authentication methods.

When authentication requests containing MS-CHAP-v1 or PAP credentials are received, Steel-Belted Radius skips the EAP-PEAP method (since it is marked with `EAP-Type=PEAP` and `EAP-Only=1`) and passes the request to the same authentication method (Windows Domain) as before.

When the request does contain EAP user credentials, the EAP-PEAP method receives the request, performs the TLS handshake (this requires multiple round-trips) and, upon completion of the handshake, creates a new request containing EAP-MS-CHAP-v2 user credentials. This new request skips past the EAP-PEAP method (no new request created by the EAP-PEAP method is passed back to it) and is authenticated by the Windows Domain authentication method.

At the conclusion of a successful inner authentication, the default configuration for EAP-PEAP creates one `MS-MPPE-Send-Key` and one `MS-MPPE-Recv-Key` attribute in the final Accept response to transmit keying material back to the RAS (or Access Point) that sent it the authentication request.

Configuring the Server

Depending on your authentication requirements, you may need to configure Steel-Belted Radius to work with an external SQL or LDAP database, RSA SecurID service, or TACACS+.

Configuring External Databases (Solaris/Linux)

If you run Solaris or Linux and want to use external databases for authentication or accounting purposes (and you did not configure this feature when prompted by the Steel-Belted Radius installation script), you can set up external database configuration settings.

To configure Steel-Belted Radius to work with an external database:

1. Optionally, perform the instructions in Chapter 14, "Configuring SQL Authentication" on page 181 and/or Chapter 15, "Configuring SQL Accounting" on page 193.
2. If you want to use Steel-Belted Radius with an LDAP database, review your LDAP database vendor's documentation.
3. Perform the instructions in "Configuring LDAP Authentication" on page 208.

Configuring SecurID Authentication

If you want to use SecurID authentication, you must configure Steel-Belted Radius to communicate with the RSA SecurID server.

Perform the following steps to configure a Steel-Belted Radius server to work with an RSA SecurID server. If you are not familiar with the RSA SecurID server, contact your RSA SecurID server administrator for assistance.

1. Verify that the Steel-Belted Radius server has an entry on the RSA SecurID server.

   Start the RSA SecurID server administration program and display the list of clients. If the list of clients does not include the Steel-Belted Radius server, select **Client > Add Client** and complete the Client window, giving the Steel-Belted Radius server a Client type of `Net OS Client`.

2. Copy the `sdconf.rec` file from the `\ACE\data` directory on the RSA SecurID server to the appropriate directory on the Steel-Belted Radius server:

   - **Windows**: `C:\winnt\system32`
   - **Solaris/Linux**: the directory that contains the `radius` daemon on the Steel-Belted Radius server.

3. Edit the [SecurID] section of `radius.ini`. The `radius.ini` file is found in the same directory as the Steel-Belted Radius service or daemon.

   Verify that the `CachePasscodes` field is set to `yes` and the `SecondsToCachePasscodes` field is set to an appropriate number of seconds. These settings ensure that authenticated SecurID users can open a second B-channel during an ISDN connection.

4. Edit the [SecurID] section of the `eap.ini` file, which is found in the same directory as the Steel-Belted Radius service or daemon.

   Verify that the EAP settings in this section are enabled (remove the semi-colon from the start of each line) if you plan to use RSA SecurID authentication with EAP Generic-Token protocol support. The client system must support this protocol as well for this combination to work.

5 If you copy the `sdconf.rec` file after the Steel-Belted Radius service (daemon) has been started, or if you edit the `radius.ini` or `eap.ini` files after Steel-Belted Radius has been started, stop and restart Steel-Belted Radius.

6 Verify connectivity between the Steel-Belted Radius server and the RSA SecurID server.

The RSA SecurID server offers a monitoring window on which it logs every authentication transaction, complete with the reason for the accept or reject decision. You can verify that pass-through to RSA SecurID is working, by creating a SecurID User called `<ANY>` and then attempting to access the network. Look for your request on the RSA SecurID monitor screen. If access is denied, you'll know that there's a configuration problem. Try these steps again, or contact your RSA SecurID administrator for assistance.

These steps complete initial setup of the two servers. To fully enable pass-through authentication to the RSA SecurID server, you must also set up the SecurID authentication method.

Configuring the Location of the sdconf.rec File

The `VAR_ACE` variable in the `sbrd` script file (Solaris/Linux) lets you specify the directory holding the `sdconf.rec` file. The `VAR_ACE` variable must be exported so that Steel-Belted Radius can use it.

For example:

```
VAR_ACE="radiusdir/ace"
    export VAR_ACE
```

This variable is set by default in the file to point to the *`radiusdir`* directory. If the variable is not set at all in the file, the server sets the value of this variable to `/var/ace`.

Configuring TACACS+ Authentication

If you want to use TACACS+ authentication, you must configure Steel-Belted Radius to communicate with the TACACS+ server.

Perform the following steps to configure a Steel-Belted Radius server to work with a TACACS+ server.

1 Verify the `tacplus.ini` file is present in the Steel-Belted Radius directory.

The `tacplus.ini` file must be present in the same directory as the Steel-Belted Radius service (in the case of Windows, usually `C:\RADIUS\Service`), or daemon (in the case of UNIX). This happens automatically following installation.

2 Edit `tacplus.ini` to identify the shared secret and host machine that you use for TACACS+. For more information on the `tacplus.ini` file, refer to the *Steel-Belted Radius Reference Guide*.

3 If you edit `tacplus.ini` after Steel-Belted Radius has been started, then you must stop and restart it before your changes take effect.

To enable pass-through authentication to the TACACS+ server, you must also set up the TACACS+ authentication method. For more information, see "Configuring TACACS+ Authentication" on page 137.

Activating Authentication Methods

The Authentication Methods tab in the Authentication Policies panel (Figure 62) permits you to activate authentication methods and define the order in which different authentication methods are attempted.

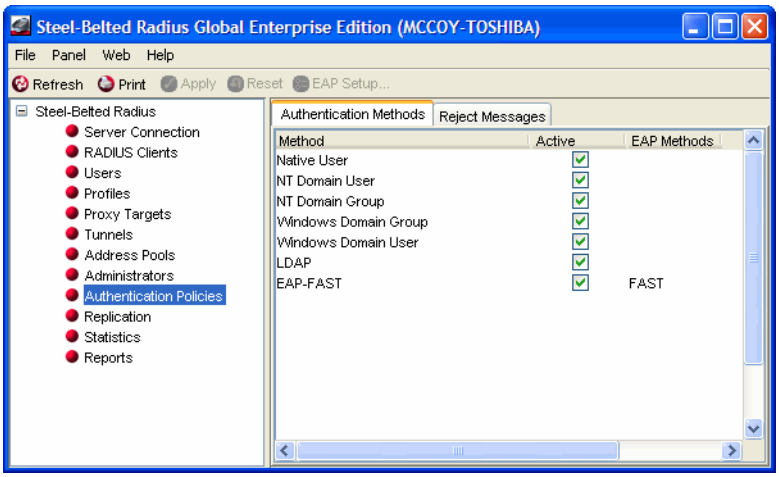


***Figure 62** Authentication Policies Panel: Authentication Methods Tab*

To use the Authentication Methods tab:

1. Open the Authentication Policies panel.

   By default, the Authentication Methods tab is selected. The Authentication Methods tab lists the authentication methods that you have enabled (by editing the appropriate `*.aut` file).

2. Click the **Active** checkbox to enable and disable the authentication methods you want Steel-Belted Radius to use.

3. Click the **Up** and **Down** arrows to change the order of the displayed authentication methods.

   Authentication methods are tried in the order in which they are listed.

To revert to the previous settings, click **Reset**.

Configuring EAP Settings

When Steel-Belted Radius receives a username, it does not know in advance to which authentication category this user belongs. It must try each method that it currently has configured and enabled. The authentication methods list allows you to fine-tune the sequence of authentication attempts.

***NOTE:** The EAP Setup window displays the authentication methods that have been enabled (by editing the Enabled setting in the appropriate `*.aut` file).*

To set up EAP settings for an authentication method:

1. Select the authentication method you want to set up in the Authentication Methods tab.
2. Click the **EAP Setup** button.

   The Setup EAP window (Figure 63) opens.

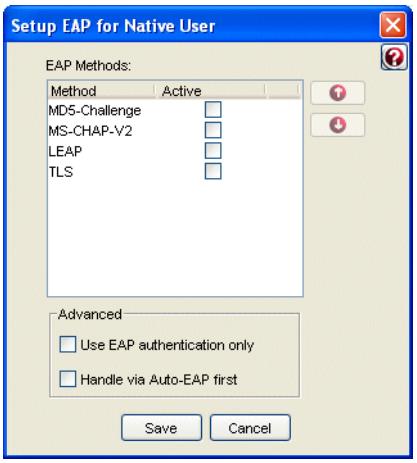


***Figure 63** Setup EAP Window*

3. Optionally, change the order in which the methods are tried by highlighting a method and clicking the **Up** or **Down** buttons.
4. To activate a method, (so that it can be used for authentication), click the **Active** checkbox.

   If you want to deactivate a method (so that it is not used for authentication), unclick the applicable **Active** checkbox.
5. If you want restrict use of this authentication method to requests that contain EAP credentials, click the **Use EAP authentication only** checkbox.

   When this option is enabled, Steel-Belted Radius prevents the authentication method from being called for any request that does not contain EAP credentials, and bypasses the authentication method if an authentication request specifically requests an EAP protocol that is not listed in the authentication method's `EAP-Type` list in the `eap.ini` file.
6. If you want Steel-Belted Radius to use an automatic EAP helper to generate credentials for a user, click the **Handle via Auto-EAP first** checkbox.

   You should unclick the checkbox if an authentication method is capable of handling EAP credentials on its own (without an EAP helper).

   Refer to "First-Handle-Via-Auto-EAP Setting" on page 114 for more information.
7. Click **Save** to return to the Authentication Methods tab.

Configuring Authentication Rejection Messages

When Steel-Belted Radius issues an Access-Reject message in response to a failed authentication request, it can identify the reason why the request was rejected. You can configure the message text returned to the RADIUS client (and possibly to the user, if the RADIUS client forwards the message) when a particular type of error occurs. This text is inserted into the standard RADIUS attribute Reply-Message within the Access-Reject response.

To configure the text for authentication rejection messages:

1. Open the Authentication Policies panel.
2. Click the **Reject Messages** tab (Figure 64).

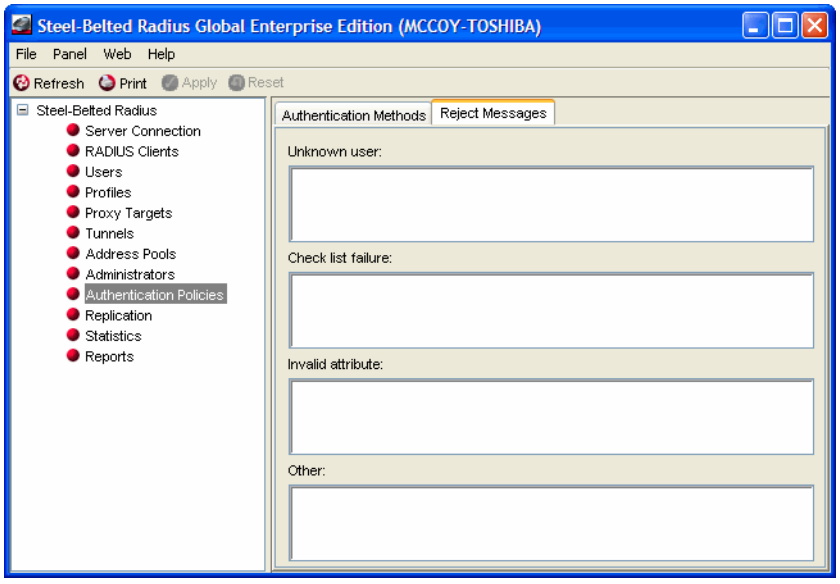


***Figure 64** Authentication Policies Panel: Reject Messages Tab*

3. Use the **Unknown User** field to specify the message Steel-Belted Radius returns when the username and password authentication failed.
4. Use the **Checklist failure** field to specify the message Steel-Belted Radius returns when the user was authenticated but is being rejected because the RADIUS request did not fulfill the requirements of the checklist.
5. Use the **Invalid attribute** field to specify the message Steel-Belted Radius returns when the request contained an attribute in violation of the RADIUS specification.
6. Use the **Other** field to specify the message Steel-Belted Radius returns when some other error, such as a resource failure, occurred.
7. When you are asked to confirm that you want to save your changes, click **Yes**.

Chapter 12
Configuring Replication

This chapter describes how to configure and use the centralized configuration management (CCM) feature to coordinate Steel-Belted Radius server settings in a replication environment.

About Replication

Steel-Belted Radius supports the replication of RADIUS configuration data from a primary server to one or more replica servers within a replication realm. Replication provides administrators with an easy way to configure multiple servers that require the same information. Depending on network configuration, you can use replication to increase AAA capacity, balance AAA traffic across RADIUS servers, or ensure that authentication services are not interrupted if access to a primary or replica server is interrupted (redundancy).

Figure 65 illustrates an environment where RADIUS traffic from each RAS device is directed to its own RADIUS server (solid line) by default. If a RADIUS server becomes unavailable, the RAS can fail over to its backup RADIUS server (dotted line).

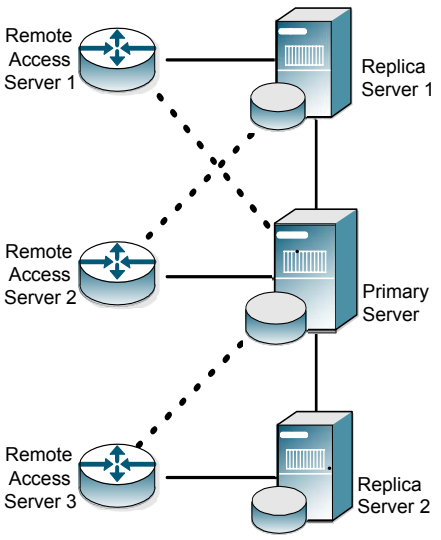


***Figure 65** Using Replication for Load Balancing and Redundancy*

All the servers within a realm reflect the current configuration specified by the network administrator: the network administrator modifies the configuration on the primary server, and the primary server propagates the new configuration to its replica servers. For example, after a network administrator configures a new RADIUS client or profile on the primary server, the network administrator tells the primary server to publish a configuration package file (`replica.ccmpkg`) that contains the updated configuration information. After publication, the primary server notifies each replica server that a new configuration package is ready. Each replica server then downloads and installs the configuration package to update its settings.

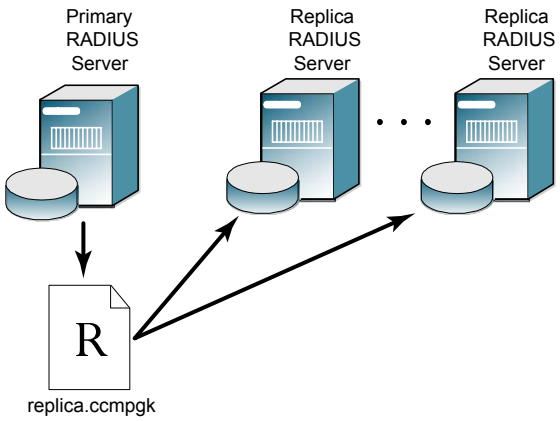


***Figure 66** Publication and Distribute of Replication Packages*

The primary server maintains a list of the replica servers that have registered with it. The primary server uses this list to track which servers to notify after it publishes an updated configuration package to resynchronize the configuration of replica servers.

Important: *You should limit access to the directory in which you store configuration packages on Windows servers to the CREATOR OWNER, SYSTEM, and Administrators. To set file access permissions for the \Radius\Service directory, right-click the directory icon, click the* **Security** *tab, click the* **Allow** *and* **Deny** *checkboxes to limit access to authorized users.*

By default, file permissions for configuration packages on Solaris/Linux servers is set to rw-rw----, which excludes users other than the file owner and the owner's group from displaying the contents of file packages.

If the primary server needs to be taken out of service for an extended period, the network administrator promotes one of the replica servers to be the new primary server. Thereafter, the other replica servers copy the configuration package from the promoted primary server.

The following types of information are included in a replication package.

- Server information
- RADIUS client information
- User information
- Profile information
- Proxy target information
- RADIUS tunnel information
- Name parsing information
- Authentication method information
- Authentication realm information
- Rejection messages

You administer this information by connecting the SBR Administrator to the primary server: the information is propagated to the replica servers in the domain. (If you connect the SBR Administrator to a replica server, you can view this information, but you cannot modify it.)

The following types of information are not included in a replication package:

- Address pool information – You administer address pools for a server by connecting the SBR Administrator to that server. Because an address must not be assigned to two users at the same time, each server in a realm must have its own address pools, and these pools must not overlap.
- Administrator information – Administrator information must be configured for each primary and replica server separately.
- Statistics information – Server statistics are not replicated. You can view statistics for replica servers when you connect SBR Administrator to the primary server.
- Report information – Report information is not replicated. To obtain report information for a primary or replica server, connect SBR Administrator to the applicable server.
- Steel-Belted Radius configuration files (`*.ini` files (other than `eap.ini`), `*.aut` files, `*.dir` files) – When you change configuration files on the primary server, you must copy the modified files to the appropriate directory on each replica server.

NOTE: *Configuration packages are retained until they are replaced. An old configuration package is automatically deleted 24 hours after a new configuration package is published.*

Replication Requirements

Servers in a replication realm must comply with the following requirements.

- All servers in a replication realm be running the same operating system (Windows, Solaris, or Linux).
- All servers in a replication realm must be running the same version and edition (Global Enterprise Edition, Service Provider Edition, or Enterprise Edition) of Steel-Belted Radius.
- All servers in a replication realm must be configured to support the same types of users (domain, host, RSA SecurID, TACACS+, UNIX).
- If RSA SecurID is enabled on the primary server, RSA SecurID must be enabled on the replica servers, and all servers in the realm must have consistent `sdconf.rec` files.
- The system clocks on servers in a replication realm must by synchronized to within 10 minutes of one another and their time zones must be configured correctly. Steel-Belted Radius uses the system clock value and time zone setting to convert local time to Universal Time Coordinated (also known as Greenwich Mean Time) when evaluating synchronization. If possible, you should use a Network Time Protocol (NTP) server to set the system clock on all servers automatically.
- If a firewall stands between servers in a replication realm, the firewall must be configured to pass traffic on the port used for replication communication. The default port for replication communication is TCP 1812, though you can specify another port for replication traffic by modifying the `radius.ini` file.

Configuring Replica Servers

The Replication panel (Figure 67) lets you add servers to a replication realm, initiate publication of a replication package (`replica.ccmpkg`) by a primary server, and notify replica servers that they should download and install a new replication package.

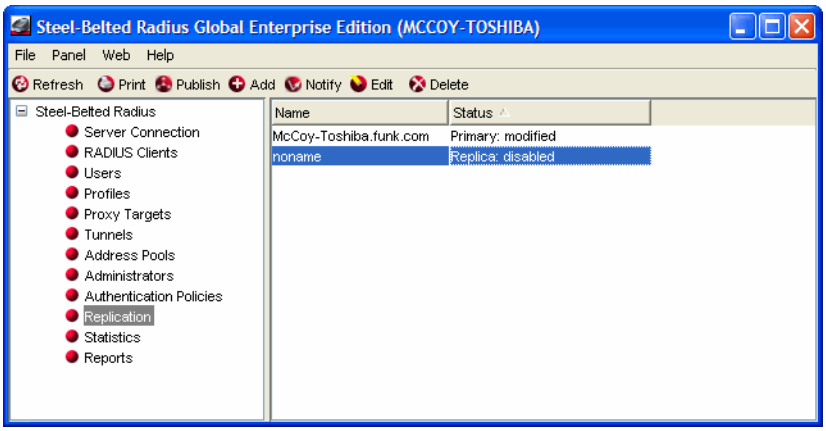


***Figure 67** Replication Panel*

Adding a Replica Server

In most situations, you add a replica server to a realm as follows:

1. Copy the `replica.ccmpkg` configuration package file from the primary server to a directory on the host you want to add as a replica server.

   Note that the replica.ccmpkg file contains sensitive information, and should not be stored in a publicly accessible location, such as a file server or shared directory.

2. Install the Steel-Belted Radius server software on the host you want to add as a replica server.

3. When the installer (Windows) or configuration script (Solaris/Linux) asks what kind of server you are installing, choose **Replica** and, when prompted, enter the path to the `replica.ccmpkg` file.

4. Restart the host you want to add as a replica server.

   The replica server registers itself with the primary server automatically after it is restarted. Thereafter, the replica server automatically connects to the primary server once an hour to check whether an updated configuration package is available.

In some circumstances, however, you may want to add a replica server to the server list on the primary server manually so that it shows up immediately. To register a replica server manually:

1. Run SBR Administrator and connect to the primary server.

2. Open the Replication panel.

3. Click the **Add** button.

   The Add Server window (Figure 68) opens.

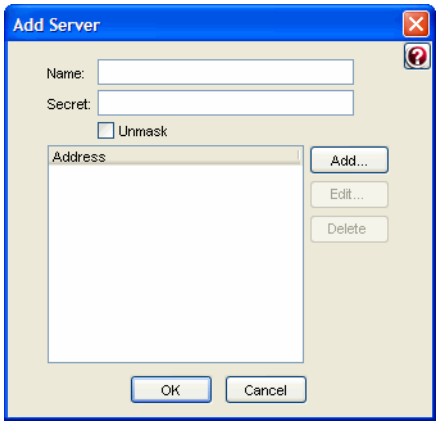


Figure 68 Add Server Window

4. Enter the name of the RADIUS server in the **Name** field.

   Although you can assign any name to a RADIUS server, you should use the device's hostname to avoid confusion.

5. Enter the replication secret for the RADIUS server in the **Secret** field.

For privacy, asterisks are echoed as you type. You can click the **Unmask** checkbox to display the characters in the shared secret.

6 Enter one or more IP addresses for your server.

a Click the **Add** button.

b When the Add IP Address window (Figure 69) opens, enter an IP address you want to associate with the server in the **Address** field and click **Add**.

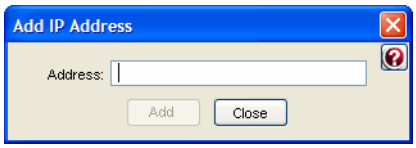


***Figure 69** Add IP Address Window*

c Repeat Step 5b until you have finished adding IP addresses for the server.

d Click **Close**.

7 Click **OK**.

Enabling a RADIUS Server

To enable a RADIUS server:

1 Open the Replication panel.

2 Select the RADIUS server you want to enable and click the **Edit** button (or double-click the RADIUS server entry).

The Edit Server window (Figure 70) opens.

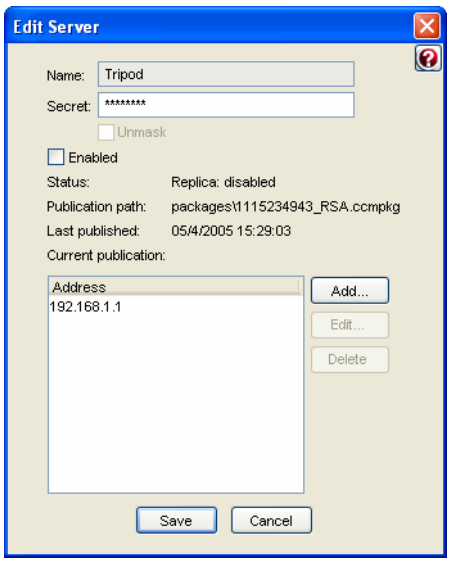


***Figure 70** Edit Server Window*

3 Click the **Enabled** checkbox.

4 Click the **Save** button.

Deleting a RADIUS Server

To delete a replica server from a realm:

1 Open the Replication panel.

2 Select the RADIUS server entry you want to delete.

3 Click the **Delete** button on the SBR Administrator toolbar.

4 When you are prompted to confirm the deletion request, click **Yes**.

Publishing Server Configuration Information

If you change the configuration of your primary server, you must publish the modified configuration so that your replica servers can download the modified settings.

To publish server configuration information:

1 Open the Replication panel.

2 Click the **Publish** button on the toolbar.

This creates a file called `/radius/packages/`*`timestamp`*`_RSA.ccmpkg` (Solaris/Linux) or `\Radius\Service\packages\`*`timestamp`*`_RSA.ccmpkg` (Windows), where *`timestamp`* reflects the date and time the package was created.

Notifying Replica RADIUS Servers

Under normal circumstances, a replica server connects to its primary server every hour to check whether a new `replica.ccmpkg` file has been published. If necessary, a network administrator can manually notify a replica server to download and install the current configuration package from the primary server. Manual notification is useful when network issues prevent the automatic download and installation of a configuration package when it is first published, and the configuration on the replica no longer matches the configuration on the primary server.

***NOTE:** You can display the Replication panel to determine the status of your replica servers.*

To notify replica servers that new configuration information has been published:

1 Open the Replication panel.

2 Select the replica server you want to notify.

3 Click the **Notify** button on the toolbar.

The replica server downloads and installs its configuration package from the primary server. After the package is installed, the replica server is resynchronized with the primary server.

Designating a New Primary RADIUS Server

You can change which server within a realm is designated as the primary server for that realm.

To designate a new primary server:

1. Stop the RADIUS service on the replica server.
2. Log into the replica server as `root`.
3. Open a command window and navigate to the `\Radius\Service` directory (Windows) or `/opt/funk/radius` directory (Solaris/Linux).
4. Run the `sbrsetuptool` utility with the `promote` option.

   ```
   # sbrsetuptool -promote
   ```

   The utility creates a configuration package to change this server to the primary server.
5. Restart the updated replica server to make it the new primary server.
6. Publish a new configuration package administratively to configure all replica servers to use the new primary server.

After you designate a new primary server for a realm, you can configure the old primary server as a replica server by downloading a configuration package published by the new primary server.

Recovering a Replica After a Failed Download

If a replica server fails during the download of a configuration package, its configuration may be corrupted or it may have a stale secret.

To recover after a failed download:

1. Stop the RADIUS service on the replica server.
2. Log into the replica server as `root`.
3. Open a command window and navigate to the `\Radius\Service` directory (Windows) or `/opt/funk/radius` directory (Solaris/Linux).
4. Run the `sbrsetuptool` utility with the `identity` option and information on where to download configuration information.

   To obtain configuration from a configuration package, issue the following command:

   ```
   # sbrsetuptool -identity REPLICA -reppkg pathname
   ```

where *pathname* specifies the path to a `replica.ccmpkg` package.

To obtain configuration from the primary server for the realm, issue the following command:

```
# sbrsetuptool -identity REPLICA -primary name address secret
```

where *name* specifies the DNS name of the primary server, *address* specifies the IP address of the primary server, and *secret* specifies the shared secret used to authenticate configuration downloads.

5 Restart the updated replica server so that it can load its new configuration.

After the replica server is restarted, it will be re-synchronized with the current primary server.

Changing the Name or IP Address of a Server

You may need to change the DNS name or IP address assigned to a primary or replica server if your network changes.

To change the DNS name or IP address of a primary or replica server:

1 Stop the RADIUS service on the RADIUS server you want to change.

2 Log into the RADIUS server as `root`.

3 Open a command window and navigate to the `\Radius\Service` directory (Windows) or `/opt/funk/radius` directory (Solaris/Linux).

4 Run the `sbrsetuptool` (Solaris/Linux) utility with the `identity` option.

To rename a primary server, enter the following command:

```
# sbrsetuptool -identity PRIMARY
```

To rename a replica server, enter the following command:

```
# sbrsetuptool -identity REPLICA
```

5 Restart the updated server so that it can load its new configuration.

6 Run the SBR Administrator and modify the DNS name or IP address for the server you want to rename. Verify that the secret on the renamed server is correct.

You may need to use the Replication panel to delete the old server name from the list of servers in the realm.

7 Publish the modified configuration to propagate the name change to the replica servers.

Replication Error Messages

The following tables list possible causes for error messages caused by replication issues.

Error Messages on Replica Servers

Table 21 lists possible causes for error messages on replica servers in a replication realm.

Table 21. *Replication Error Messages on Replica Servers*

Error Type	Error Message	Possible Cause
Post Errors (Errors with Notification from Primary)	CRadManagedServerNotifyPost:: ExecutePost invalid signature!	Mismatched replication secret.
	CRadManagedServerNotifyPost:: ExecutePost invalid sequence number	Two posts have the same sequence number. The clocks on the primary and replica are more then 10 minutes apart.
	CRadManagedServerNotifyPost:: ExecutePost decrypt failed	Shared secret failed to decrypt. Bad Replication Secret secret.
	CRadManagedServerNotifyPost:: ExecutePost invalid <body> missing parameters	Post had an invalid xml request.
Update Errors (Errors with Published package from Primary)	CRadManagedServerUpdate:: DoStart Failed to open '*file_name*' for writing	Temp directory does not exist. Temp directory or file have incorrect permissions.
	CRadManagedServerUpdate:: StartUpdates has already started update	Update is already in progress.
	CRadManagedServerUpdate:: DownloadPackage HTTP POST error:*errCode Primary ID*	Error in transmitting request to Primary (timeout during transmit).
	CRadManagedServerUpdate:: DownloadPackage HTTP headers parsing error	Error in receiving package. Typically caused by a timeout during receive resulting from in invalid package.
	CRadManagedServerUpdate:: DownloadPackage connection *primary IP Addr* error: *errCode Primary ID*	Replica failed to connect with Primary. Primary not running.
	CRadManagedServerUpdate:: DownloadPackage exceeded iterations limit while communicating with CCM	Update failed after three attempts.
	CRadManagedServerUpdate:: ProcessPackage signature mismatch	Secrets on Replica and Primary do not match.
	CRadManagedServerUpdate:: ProcessPackage CCM error: '*Error String*' '*Pararmeter*'	Error parsing downloaded packages.

Table 21. *Replication Error Messages on Replica Servers (Continued)*

	CRadManagedServerUpdate:: ProcessPackage Failed to open \'" << *file_name* << "\' for writing	Temp directory does not exist. Temp directory or file has incorrect permissions.
	CRadManagedServerUpdate:: ProcessPackage thumbprint mismatch	Invalid package. Republish the package.
Proxy Errors (Statistics retrieve errors)	CRadProxyPost:: ExecutePost invalid signature!	Mismatched replication secret.
	CRadProxyPost:: ExecutePost invalid sequence number	Two posts have the same sequence number. The clocks on the primary and replica are more then 10 minutes apart.
	CRadProxyPost:: ExecutePost decrypt failed	Shared secret failed to decrypt. Bad Replication Secret secret.
	CRadProxyPost:: ExecutePost invalid <body> missing parameters	Post had an invalid XML request.

Error Messages on Primary Servers

Table 22 lists possible causes for error messages on primary servers in a replication realm.

Table 22. *Replication Error Messages*

Error Type	Error Message	Possible Cause
Notify Target (Both Notify and Publish send a notification)	CRadConfigManagedServerHTTP Notification::NotifyTarget failed to fetch	Replica does not exist in database (This is possible if running multiple GUIs and you delete a replica from one then try to publish from the other.)
	CRadConfigManagedServerHTTP Notification::NotifyTarget failed *replicaId*	Notify Failed to Communicate with Replica, Replica is not running or check Replica DCF log for more info
Publication Provider (requests from GUI to Notify or Publish)	CRadConfigPublicationProvider:: UpdateResource notify invoked when not Primary	Attempted to Notify as a Replica, this is only allowed from a Primary
	CRadConfigPublicationProvider:: UpdateResource publish invoked when not Primary	Attempted to Publish as a Replica, this is only allowed from a Primary

Table 22. *Replication Error Messages (Continued)*

Publication Post (parsing of Post from replica to get data)	CRadConfigServerProviderPost:: ExecutePost signature mismatch with server:	Replica and Primary have a Replication Secret Mismatch
	CRadConfigServerProvider:: GetResource invoked when not Primary	Another Replica is requesting a download from this server which is a replica. Replica that is requesting a download needs to be reconfigured (see disaster recovery in docs).
Proxy Errors (Statistics retrieve errors)	CRadProxyClient:: Send failed to fetch	Replica does not exist in database (This is possible if running multiple GUIs, and you delete a replica from one then try to publish from the other.)
	CRadProxyClient:: SendData HTTP POST error:	Connection error with Replica.

Chapter 13
LDAP Configuration Interface

This chapter describes:

- The file used to enable and configure the LDAP configuration interface (LCI)
- An overview of the LCI and LDAP utilities
- A description of the LDAP virtual schema
- Information about how to use LDAP utilities to configure the Steel-Belted Radius database
- Sample `LDIF` files that control the execution of LDAP utilities
- Information about how to view rate statistics variables with LDAP utilities

LDAP Configuration Interface File

The `radius.ini` file establishes settings for the LDAP configuration interface. For more information about `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.

Table 23. *LDAP Configuration Interface Files*

File Name	Function
`radius.ini`	Specifies (among other things) whether the LCI is enabled, the port used for LCI communication, and the interfaces on which Steel-Belted Radius listens for LCI requests.

About the LDAP Configuration Interface

NOTE: *The LDAP Configuration Interface is an optional add-on for the Enterprise edition of Steel-Belted Radius. You must license the LDAP Configuration Interface before you can configure or use it.*

The LDAP Configuration Interface (LCI) provided by Steel-Belted Radius consists of an LDAP interface in the Steel-Belted Radius server and an LDAP virtual schema. The

LDAP virtual schema presents the structure of the Steel-Belted Radius database in a manner that can be understood by the LDAP client utilities. The LCI uses the virtual schema to retrieve, modify and delete entries in the database.

Figure 71 illustrates the relationship between LDAP components.

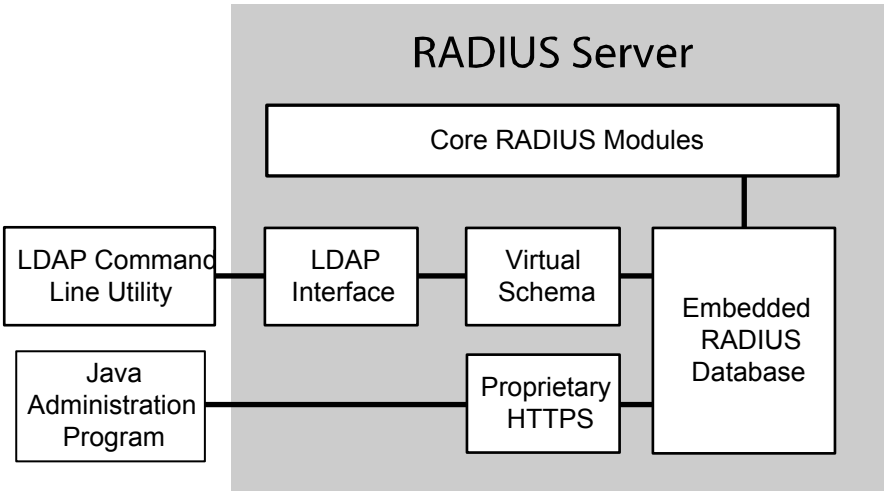


***Figure 71** LDAP Components*

LDAP Utilities

Freeware LDAP utilities, such as `ldapsearch`, `ldapdelete`, and `ldapmodify`, act as clients of the LDAP interface. LDAP utilities let you read and modify an LDAP database.

- `ldapsearch` – The `ldapsearch` utility locates and retrieves LDAP directory entries. The `ldapsearch` utility opens a connection to an LDAP interface using the specified distinguished name and password, binds, and locates entries based on the specified search filter. A search can return a single entry, an entry's immediate subentries, or an entire tree or subtree. Search results are returned in LDIF format.
- `ldapdelete` – The `ldapdelete` utility deletes entries from an existing LDAP directory. `ldapdelete` opens a connection to the specified server using the distinguished name and password you provide, binds, and deletes the entry or entries.
- `ldapmodify` – The `ldapmodify` utility adds or modifies entries in an existing LDAP directory. `ldapmodify` opens a connection to an LDAP interface using the distinguished name and password you supply, binds, and adds or modifies the entries based on the LDIF update statements contained in a specified file. Because `ldapmodify` uses LDIF update statements, `ldapmodify` can do everything `ldapdelete` can do.

LDAP Requests

LDAP requests are submitted in two ways:

- By specifying options on the LDAP configuration interface command line.

- By placing instructions and data into an LDAP Data Interchange Format (LDIF) file, which you then process by invoking an LDAP command line utility using the `-f` option.

Because communication between the LDAP client and server is unencrypted, the LDAP utilities should be run on the same computer as Steel-Belted Radius.

Downloading the LDAP Utilities (Windows)

To use the LCI, you need the `ldapsearch`, `ldapmodify`, and `ldapdelete` utilities. You can download freeware Windows LDAP utilities as follows:

1. Use a browser to navigate to `http://www.sun.com/download/products.xml?id=/3ec28dbd`.
2. When the Sun ONE Directory SDK (software development kit) download page appears, click the **Download** link at the bottom of the page.
3. If you are prompted to register yourself, complete the registration form.
4. When you are prompted to accept the license agreement, click the **Accept** button and then click **Continue**.
5. Download the SDK by clicking the link for the version of the SDK that is appropriate for your computer.

   Versions of the SDK are available for Solaris, Linux, and Windows.
6. When the download is completed, extract the files from the compressed image to a directory on your computer.

To run the LDAP utilities, execute them from this directory. Note that, if you set the path environment variable to point to this directory, you can run them any location on the system.

NOTE: *The examples that follow assume you are using the LDAP utilities provided as part of the Sun ONE Directory SDK. If you are using LDAP utilities from another source, the command options you use may be different. Consult the documentation for your LDAP utilities for more information.*

LDAP Version Compliance

The LDAP interface in Steel-Belted Radius complies with version 2 of the LDAP specification. You should use the `-V 2` command option to direct the utilities to use version 2 features. For example:

```
ldapmodify -c -V 2 -p 354 -D "cn=admin,o=radius"
-w radius -f filename
```

Configuring the LDAP TCP Port

To avoid conflicts with LDAP services that may already be installed, the default port number for communication between Steel-Belted Radius and the LDAP client is 667. You can configure Steel-Belted Radius to use a different TCP port to communicate with an LDAP client. For example, you can change this port number to 389, the standard

LDAP TCP port, if you are certain doing so will not create port number conflicts with other applications.

The following example configures Steel-Belted Radius to use TCP port 354.

1 In the `radius.ini` file, uncomment the [LDAP] section, set `Enable` to 1, and set the `TCPPort` field to the port number you want to use. For example:

```
[LDAP]
Enable = 1
TCPPort = 354
```

You must specify the port number (by means of the `-p` option) when you run the LDAP utilities. For example:

```
ldapsearch -V 2 -p 354 -D "cn=admin,o=radius" -w radius -s
sub -T -b "radiusclass=Client,o=radius" radiusname=*
```

Configuring the LCI Password

After you enable the LCI, you should change the default LCI password to prevent unauthorized LDAP clients from accessing your database. After you install the LDAP utilities and verify that they work, perform the following steps:

1 Create a text file called `temp.ldif` with the following contents:

```
dn: radiusclass=server,o=radius
changetype: modify
replace: server-password
server-password: new-password
```

where *`new-password`* is the LCI password you want to use.

2 Change the `radius.ini` [LDAP] setting to `Enable=1`.

3 Restart Steel-Belted Radius.

4 Execute the following command:

```
ldapmodify -V 2 -h ip-address -p port -D "cn=admin,o=radius"
-w oldpassword -f temp.ldif
```

where:

`-h` *`ip-address`* specifies the IP address of the Steel-Belted Radius server

`-p` *`port`* specifies the port number specified in the [LDAP] section of the `radius.ini` file

`-w` *`oldpassword`* specifies the current password (which is `radius` by default).

5 Verify that the password change was successful by executing the following command:

```
ldapsearch -V 2 -h ip-address -p port -D "cn=admin,o=radius"
-w newpassword -s sub -T -b "o=radius" radiusclass=server
```

where:

`-h` *`ip-address`* specifies the IP address of the Steel-Belted Radius server

`-p port` specifies the port number specified in the [LDAP] section of the `radius.ini` file

`-w newpassword` specifies the password configured in the `temp.ldif` file

After you verify that the password change has been successful, delete the `temp.ldif` file and any other file that contains a cleartext copy of the modified LCI password.

LDAP Virtual Schema

The LDAP interface uses the virtual schema (illustrated in Figures 72–76) to represent the structure of the Steel-Belted Radius database. LDAP clients use the virtual schema to exchange configuration data over the LDAP configuration interface.

***NOTE:** Your edition of Steel-Belted Radius may not support all branches of the schema illustrated in Figures 72–76.*

Many of the top-level items in the LDAP virtual schema correspond to windows and panels in the SBR Administrator.

***Table 24.** LDAP Schema and SBR Administrator Windows*

Item	See
`radiusclass=client`	Chapter 4, “Administering RADIUS Clients” on page 53
`radiusclass=native-user, securid-user, …`	Chapter 5, “Administering Users” on page 59
`radiusclass=profile`	Chapter 6, “Administering Profiles” on page 81
`radiusclass=proxy`	Chapter 7, “Administering Proxy Targets” on page 85
`radiusclass=tunnel`	Chapter 8, “Administering RADIUS Tunnels” on page 91
`radiusclass=server`	Chapter 11, “Setting Up Authentication Policies” on page 111
`radiusclass=ip-addr-pool`	“Setting Up IP Address Pools” on page 99
`radiusclass=ipx-addr-pool`	“Setting Up IPX Address Pools” on page 103
`radiusstatus=statistics`	Chapter 17, “Displaying Statistics” on page 217
`radiusstatus=sessions`	“Displaying the Current Sessions List” on page 226

***NOTE:** `radiusstatus` items can be read, but they cannot be modified.*

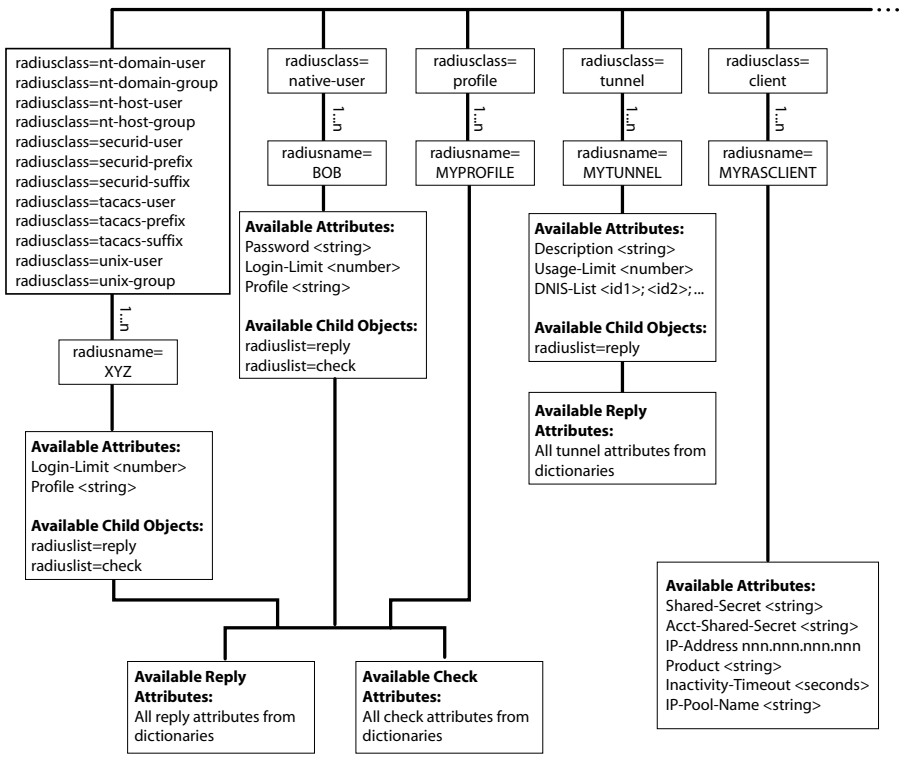


Figure 72 *LDAP Schema (Slide 1 of 5)*

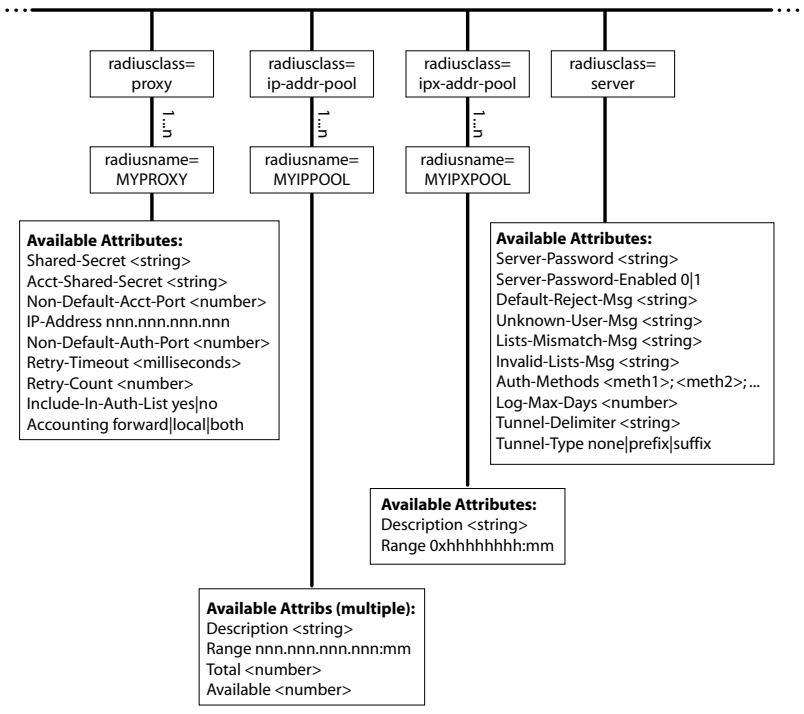


Figure 73 *LDAP Schema (Slide 2 of 5)*

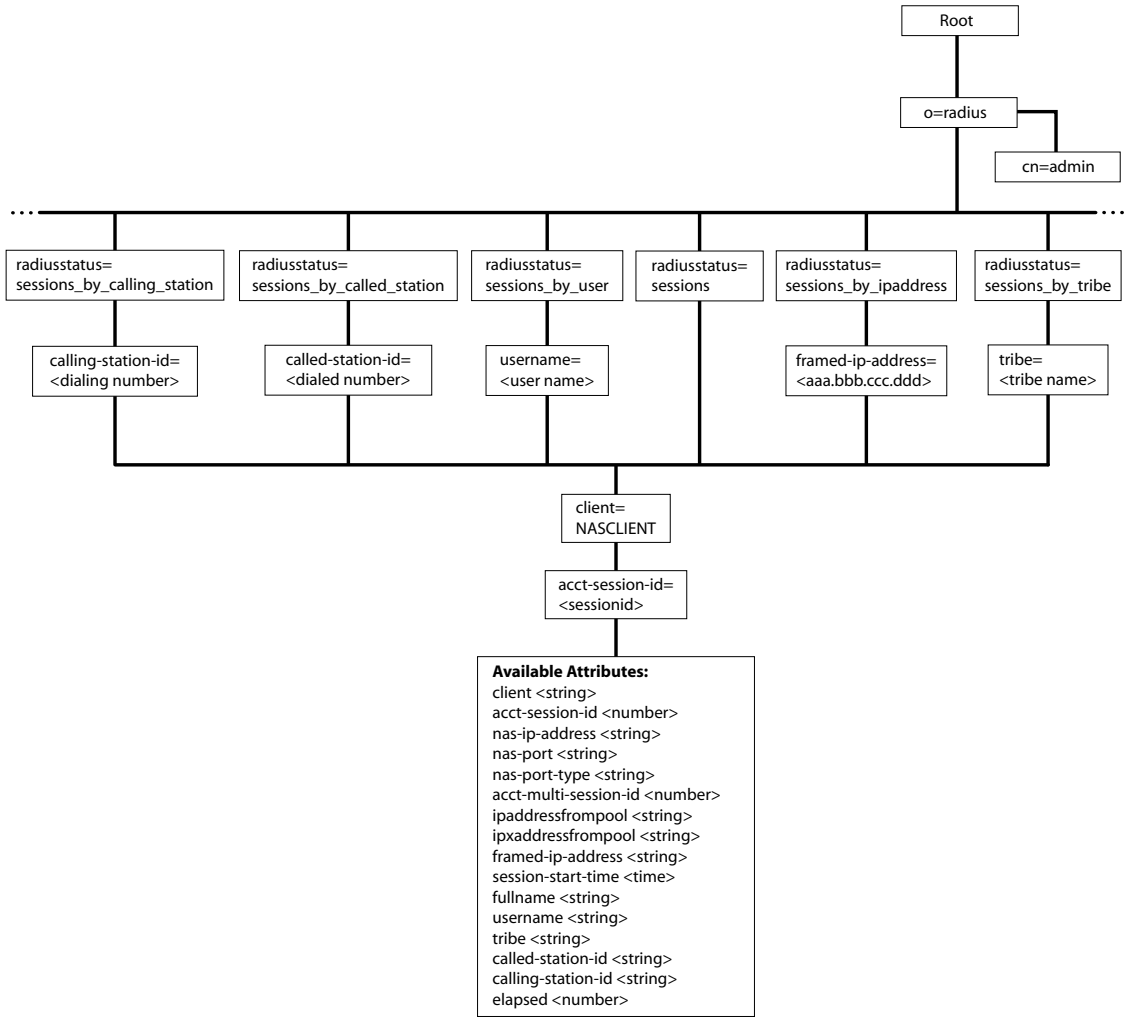


Figure 74 *LDAP Schema (Slide 3 of 5)*

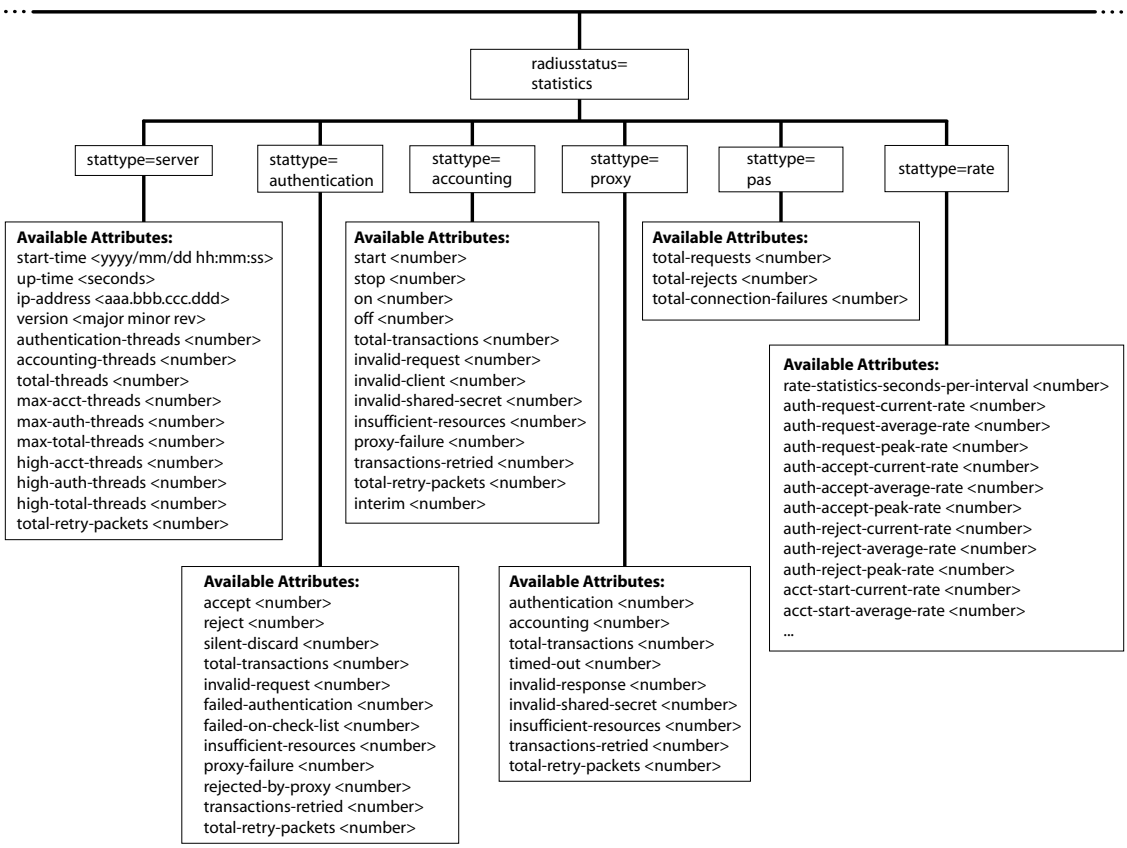


Figure 75 *LDAP Schema (Slide 4 of 5)*

NOTE: *The Enterprise Edition of Steel-Belted Radius with the optional LCI add-on does not support the Statistics items.*

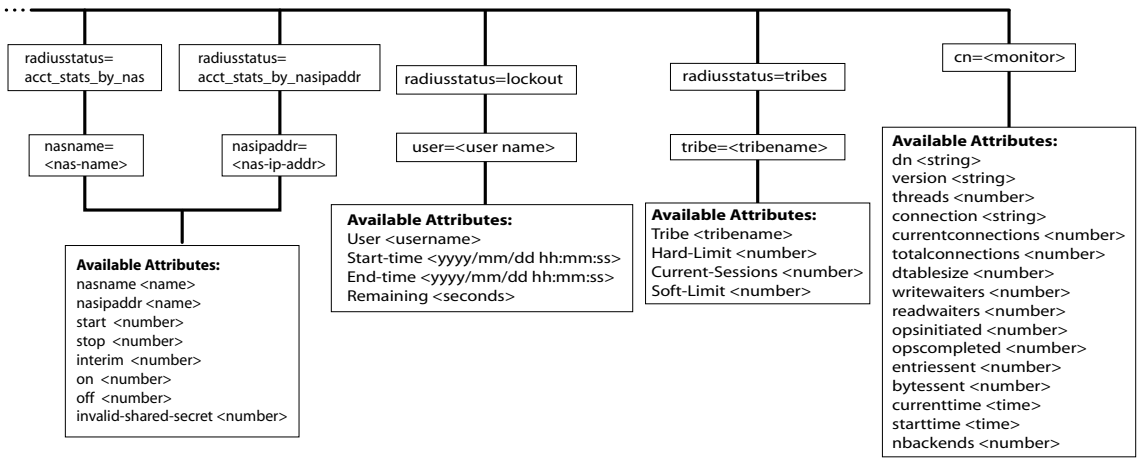


Figure 76 LDAP Schema (Slide 5 of 5)

LDAP Rules and Limitations

While the LDAP virtual schema diagram shows as much of the detail of the LDAP virtual schema as possible, the following rules and limitations should be considered.

- **Bind request** – All attempts to perform operations on the virtual schema must be preceded by an LDAP Bind request that authenticates the administrator to the Steel-Belted Radius server. The Bind request must reference a Steel-Belted Radius administrative account and must provide the password that authenticates that account. This translates into the following command line options for each invocation of the LDAP utilities:

  ```
  -D "cn=AdminName,o=radius" -w AdminPassword
  ```

  where *AdminName* is the administrative account name and *AdminPassword* is its password.

- **Uppercase and lowercase** – The uppercase/lowercase rules for object names are the same as in the SBR Administrator; that is, almost all object names are stored in the database in uppercase format. The exception to this rule is that UNIX User/Group, SecurID User/Prefix/Suffix and TACACS+ User/Prefix/Suffix names are maintained in the case specified in the LDIF files.

- **Attributes** – When you enter attributes, make sure that the attribute name matches the name found in the dictionary and that the attribute's value is consistent with the syntax type for the attribute. Note that the LDAP virtual schema does not list all the dictionary attributes that are available in Steel-Belted Radius.

- **IP addresses** – The `ipaddr-pool` type in the dictionary can represent an IP address or a pool name. If the value specified begins with the marker string [*pool*], the token that follows the marker string is assumed to be an IP pool name; otherwise, it must be a valid IP address. If it is neither, the operation fails.

Address ranges in IP address pool objects are specified in the form *IPaddress*:*NumberOfAddresses*. An example of a valid range is 128.22.12.45:34.

- **IPX addresses** – The `ipxaddr-pool` syntax type in the dictionary lets you enter an IPX network address (up to 8 hexadecimal digits using the format `0xhhhhhhhh`) or choose a pool name. If the value specified begins with the string [*pool*], the token that follows the marker is assumed to be an IPX pool name; otherwise, it must be a valid IPX address. If it is neither, the operation fails.

  Address ranges in IPX address pool objects are specified in the form *IPXNetAddress*:*NumberOfAddresses*. An example of a valid range is 0xa020443b:34.

- **Substrings** – Some attribute may have a value that consists of a list of strings. For example, the DNIS list in a tunnel entry and the authentication method list may consist of multiple substrings. The rule for specifying the data portion for these lists is that semicolons must delimit substrings. For example, a DNIS list for a tunnel entry might be specified as `555-1212;5551212`. If a semicolon needs to appear inside a substring, it must be escaped by placing a backslash character (\) before it.

- **Hexadecimal values** – Hexadecimal numbers (for attributes of syntax type `hex1`, `hex2` or `hex4`) require a `0x` prefix before the hexadecimal digits; for example `0x0000149a`.

- **Password syntax** – Passwords that are retrieved from the database may consist of one of the following:

  - A clear-text password of the form `{x-clear}`*`clear-text-password-string`* if the password is weakly encrypted in the database
  - A string of the form `{x-md5}`*`xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`* if the password is stored as a one-way md5 hash
  - A string of the form `{x-md5}[encrypt]`*`clear-text-password-string`* indicates that, although the password is specified in clear-text form, it is to be stored as a hash.

  White space in a password is treated as follows:

  - When clear-text passwords are specified, the password is assumed to begin immediately after the right brace or right bracket. Adding a white space character, such as a space or tab, after the right brace or right bracket causes the white space to be considered part of the password.
  - White space entered at the beginning of the attribute (before the left brace or left bracket) is ignored.
  - White space entered between the right brace of `{x-md5}` and the left bracket of `[encrypt]` is ignored.
  - All white space specified in the hexadecimal sequence describing a password hash is ignored.

- **Profiles, checklists, and return lists** – Steel-Belted Radius supports user definitions that include attribute subtractions of profile entries. To specify that a

user attribute is to be considered a subtraction of a profile attribute, preface the attribute value with the string `%subtract%`.

Steel-Belted Radius permits user and profile checklists to include default values for attributes. Configuring a default value for an attribute means that, if a RADIUS request does not include this attribute, the request should not be rejected. Instead, the value supplied as the default should be used as if it were received as part of the request. To specify that a checklist attribute is to be considered a default attribute, preface the attribute value with the string `%default%`.

Steel-Belted Radius permits user and profile return lists to include attributes whose values are set by copying the contents of received attributes. This feature is referred to as "attribute echoing." To specify that a return list attribute is to be treated as an echo attribute, enter `%echo%` for the attribute value.

- **Unspecified or 0.0.0.0 RAS IP address** – When you display `acct_stats_by_nasipaddr` information, any RAS entries with an unspecified IP address or an IP address of 0.0.0.0 are omitted. Similarly, when you display `acct_stats_by_nas` information, any RAS entries with an unspecified IP address or an IP address of 0.0.0.0 will have their `nasipaddr` attribute omitted.
- **Duplicate RAS IP addresses** – When displaying `acct_stats_by_nasipaddr` information, two RAS entries that contain the same (non-zero) IP address cause information about one of the entries to be displayed twice. This is the result of the ambiguity of the query and is not a bug.
- **RADIUS client information displayed after deletion** – If you define a RADIUS client entry, send some accounting traffic to it, and then delete the entry, the output of `ldapsearch` queries will continue to list the deleted RADIUS client so that the per-RAS statistics add up to the total RAS statistics.

LDAP Command Examples

This section explains how to use the `ldapdelete`, `ldapmodify`, and `ldapsearch` utilities to configure the server.

Searching for Records

You can use the `ldapsearch` command to extract information from the LDAP tree. The following command lets you extract information about all RADIUS Native Users:

```
ldapsearch -V 2 -p 354 -h 192.168.45.12
-D "cn=oper,o=radius" -w radadmin -s sub -T -b
"radiusclass=Native-User,o=radius" radiusname=*
```

Note there must be a blank space between each option (for example, `-p`) and its value (for example, `354`). Command syntax is case-sensitive.

Table 25. *Searching for Records Using the ldapsearch Command*

ldapsearch Option	Meaning
-V 2	Use LDAP Version 2 to communicate with the server. This option is not required, but it improves the performance of the transaction.
-p 354	Use TCP port 354 to communicate with the LDAP interface of the server. The `-p` value must match the TCPPort setting in the [LDAP] section of `radius.ini`. If the `-p` option is not specified, the LDAP utilities contact Steel-Belted Radius on the default port number (TCP port 389).
-h 192.168.45.12	Contact a remote host at the specified address or name. By default, ldapsearch tries to connect to the local host.
-D "cn=oper,o=radius"	Use the oper administrative account to authenticate the command. ***NOTE:*** *You can use any administrative account name in place of* `oper` *in this example. Do not change the* `o=radius` *argument.*
-w radadmin	Use an authentication password of `radadmin`. ***NOTE:*** *The* `-w` *parameter value (in this case,* `radadmin`*) must match the password of the account named by the* `-D` *parameter.*
-s sub	Perform a recursive subtree search from the base.
-T	Do not wrap long output lines to the next line.
-b "radiusclass=Client,o=radius"	Specifies the base from which the search operation starts.
radiusname=*	Specifies the selection criteria for the search.

If you execute the `ldapsearch` command shown above against a Steel-Belted Radius server containing two Native User definitions, your output LDIF file may look like the output shown in Figure 77.

```
dn: radiusname=KEVIN,radiusclass=Native-User,o=radius
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: KEVIN
password: {x-clear}secret1
profile: ISDN
login-limit: 2

dn: radiusname=MICHAEL,radiusclass=Native-User,o=radius
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: MICHAEL
password: {x-clear}secret99
profile: ISDN
login-limit: 2
```

***Figure 77** Search Results*

Modifying Records

You can use the `ldapmodify` utility to update the Steel-Belted Radius configuration.

```
ldapmodify -c -V 2 -h example.host.com -p 354
-D "cn=oper,o=radius" -w radadmin -f filename
```

Note there must be a blank space between each option (for example, `-p`) and its value (for example, `354`). Command syntax is case-sensitive

***Table 26.** Modifying Records Using the ldapmodify Command*

ldapmodify Option	Meaning
-c	Run the command in continuous mode; do not stop on errors.
-V 2	Use LDAP Version 2 to communicate with the server. This option is not required, but it improves the performance of the transaction.
-h *example.host.com*	Contact a remote host at the specified address or name. If the `-h` option is not used, `ldapsearch` connects to the local database.
-p 354	Use TCP port 354 to communicate with the LDAP interface of the server. The `-p` value must match the TCPPort setting in the [LDAP] section of `radius.ini`. If the `-p` option is not specified, the LDAP utilities contact Steel-Belted Radius on the default port number (TCP port 389).

Table 26. *Modifying Records Using the ldapmodify Command (Continued)*

ldapmodify Option	Meaning
-D "cn=oper,o=radius"	Use the oper administrative account to authenticate the command. ***NOTE:*** *You can use any administrative account name in place of* `oper` *in this example. Do not change the* `o=radius` *argument.*
-w radadmin	Use an authentication password of `radadmin`. ***NOTE:*** *The* `-w` *parameter value (in this case,* `radadmin`*) must match the password of the account named by the* `-D` *parameter.*
-f *filename*	Specifies the input LDIF file to process.

The LDIF files generated by `ldapsearch` differ from those required for input to `ldapmodify`. The `ldapmodify` input files must contain a `changetype` entry immediately after each `dn` entry. The `changetype` entry specifies how to use the data to change the LDAP database.

The full syntax for `changetype` within each transaction is as follows:

```
dn: distinguished-name-of-entry
changetype: keyword
subkeyword: attribute
attribute: value
changetype: keyword
subkeyword: attribute
attribute: value
.
.
.
```

where:

`keyword` can be `add`, `modify`, or `delete`;
`subkeyword` can be (respectively): `add`, `replace`, or `delete`;
`attribute` can be any LDAP attribute in the entry
`value` is the value to assign to the attribute.

Repeated `changetype: keyword` entries are not required within a transaction unless you change the keyword. From top to bottom within the transaction, the latest keyword applies until another `changetype: keyword` entry is provided. The following syntax is valid if the same keyword applies throughout the transaction:

```
dn: distinguished-name-of-entry
changetype: keyword
subkeyword: attribute
attribute: value
subkeyword: attribute
attribute: value
subkeyword: attribute
attribute: value
.
```

.
.

subkeyword: attribute entries are optional and indicate that you want to apply the change to a specific attribute within the entry. If no *subkeyword: attribute* entries in the transaction are found, the change applies to the entire entry. For example, it is faster to delete an entire entry:

```
dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: delete
```

but if you want to delete only a few attributes from the entry, you can do so:

```
dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: delete
delete: retry-count
-
delete: include-in-auth-list
```

If the *subkeyword* is `add` or `replace`, an *attribute: value* entry must appear immediately following the *subkeyword: attribute* entry. If the subkeyword is `delete`, the *attribute: value* entry does not apply and should be omitted.

The following sample LDIF file could be used with an `ldapmodify` command.

```
dn: radiusname=BIGCO.COM,radiusclass=Proxy,o=radius
changetype: add
radiusname: BIGCO.COM
ip-address: 194.132.5.89
accounting: both
retry-count: 3
retry-timeout: 5000
shared-secret: testing123
include-in-auth-list: no

dn: radiusname=BIGGERCO.COM,radiusclass=Proxy,o=radius
changetype: modify
replace: shared-secret
shared-secret: hereisthesecret
-
replace: ip-address
ip-address: 192.7.2.121

dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: modify
delete: include-in-auth-list
```

***Figure 78** Sample LDIF File*

***NOTE:** To delete the proxy entry for TINYCO.COM, issue the following command:*

```
dn: radiusname=TINYCO.COM,radiusclass=Proxy,o=radius
changetype: delete
```

Importing Records From Another LDAP Database

To import entries from one LDAP database into another, run the `ldapsearch` command on the first database. Request only the attributes you want for the new database. When `ldapsearch` completes processing, edit the output LDIF file. After each line that begins with `dn:`, add a single line containing the text `changetype: add`. Once your editing is complete, run an `ldapmodify -f` command that references the new LDIF file. After the `ldapmodify` command is executed, your new database is populated with the records you extracted from the old database.

The LDIF file shown in Figure 79 is derived from the output of the ldapsearch command. When specified as the input to an `ldapmodify -f` command, the contents of the file are added to the target database.

```
dn: radiusname=KEVIN,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: KEVIN
password: {x-clear}secret1
profile: ISDN
login-limit: 2

dn: radiusname=MICHAEL,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: MICHAEL
password: {x-clear}secret99
profile: ISDN
login-limit: 2
```

***Figure 79** Adding Records With an LDIF File*

Deleting Records

The `ldapdelete` command allows you to remove records from the LDAP database. For example, to delete entries names USER1 through USER5, add the information shown in Figure 80 to a file called `deletedemo.ldf`.

```
radiusname=USER1,radiusclass=Native-User,o=radius
radiusname=USER2,radiusclass=Native-User,o=radius
radiusname=USER3,radiusclass=Native-User,o=radius
radiusname=USER4,radiusclass=Native-User,o=radius
radiusname=USER5,radiusclass=Native-User,o=radius
```

***Figure 80** Deleting Records With an LDIF File*

Now, pass the `deletedemo.ldf` file to the `ldapdelete` command.

```
ldapdelete -V2 -h hostname -p 667
-D "cn=admin,o=radius" -w password -f deletedemo.ldf
```

Warning: *Verify that the `dn:` values that usually appear in these entries are not a part of the entries in your file, because this will cause the command to fail.*

You can use `ldapdelete` to remove records from the LDAP database without having to supply a file. For example, to delete the native user record identified as USER1, you would enter the following:

```
ldapdelete -V2 -h hostname -p 667
-D "cn=admin,o=radius" -w password
"radiusname=USER1,radiusclass=native-user,o=radius"
```

You can cause records to be deleted by means of the `ldapmodify` command, if the entries in the text file contain the line `changetype: delete`. Consider the sample LDIF file named `deletemodify.ldf` shown in Figure 81.

```
dn: radiusname=barry,radiusclass=Native-User,o=radius
changetype: delete
dn: radiusname=maurice,radiusclass=Native-User,o=radius
changetype: delete
dn: radiusname=robin,radiusclass=Native-User,o=radius
changetype: delete
```

***Figure 81** deletemodify.ldf Example*

The `deletemodify.ldf` file can be passed to the `ldapmodify` command as follows:

```
ldapmodify -V2 -h hostname -p 667 -D"cn=admi,o=radius"
-w password -f deletemodify.ldf
```

Warning: *On some LDAP servers, an error could cause the deletion of a container without prompting for confirmation. This could, in turn, cause the entire directory server to fail.*

LDIF File Examples

This section explains how to construct LDIF files that, when input to the `ldapmodify` command, add entries to the Steel-Belted Radius database.

Adding RADIUS Clients with LDIF

The sample LDIF entry shown in Figure 82 adds a RADIUS client named `ANNEX105` to the Steel-Belted Radius database.

```
dn: radiusname=ANNEX105,radiusclass=Client,o=radius
changetype: add
objectclass: top
objectclass: Client
radiusname: ANNEX105
ip-address: 193.162.45.12
product: Nortel Networks Remote Annex
shared-secret: testing123
```

***Figure 82** Adding RADIUS Clients*

The syntax in this LDIF entry is shown in Figure 83.

```
dn: radiusname=String,radiusclass=Client,o=radius
changetype: add
objectclass: top
objectclass: Client
radiusname: String
ip-address: IPAddressOfTheClientDevice
product: Make&ModelChoiceFromVendor.IniFile | …
shared-secret:
SharedSecretThatWasConfiguredOnTheClientDevice
RASClientField: RASClientFieldValue
RASClientField: RASClientFieldValue
.
.
.
```

***Figure 83** LDIF Syntax*

Adding Users with LDIF

The sample LDIF entry shown in Figure 84 adds a Native User named `KEVIN` to the Steel-Belted Radius database.

```
dn: radiusname=KEVIN,radiusclass=Native-User,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: KEVIN
password: {x-clear}secret1
profile: ISDN
login-limit: 2
```

***Figure 84** Adding Users*

The syntax in this LDIF entry is shown in Figure 85.

```
dn: radiusname=String,radiusclass=Native-User |
      Solaris-User |…, o=radius
changetype: add
objectclass: top
objectclass: Native-User | Solaris-User | …
objectclass: user
radiusname: String
password: {x-clear}PString | {x-md5}Hash |
      {x-md5}{encrypt}PString |…
profile: NameOfProfileEntryInTheServerDatabase
login-limit: IntegerGivingConcurrentConnectionLimit
UserField: UserFieldValue
UserField: UserFieldValue
.
.
.
```

***Figure 85** LDIF Syntax*

The LDIF file shown in Figure 86 add a native user named `CHRISTIAN`, who has various attribute/value pairs assigned to his checklist and return list.

```
dn:
radiusname=christian,radiusclass=native-user,o=radius
changetype: add
objectclass: top
objectclass: Native-User
objectclass: user
radiusname: CHRISTIAN
password: {x-clear}password
login-limit: 2

dn:
radiuslist=check,radiusname=CHRISTIAN,radiusclass=Native
-User,o=radius
changetype: add
objectclass: top
objectclass: check
radiuslist: check
NAS-IP-Address: 50.50.50.50
Framed-protocol: PPP

dn:
radiuslist=reply,radiusname=CHRISTIAN,radiusclass=Native
-User,o=radius
changetype: add
objectclass: top
objectclass: reply
radiuslist: reply
framed-ip-address: 100.100.100.100
framed-IP-Netmask: 255.255.255.224
```

***Figure 86** Adding a Native User*

Checklists and return lists are objects in the LDAP virtual schema, but the individual RADIUS attributes are not. Therefore, you must use a separate LDIF entry for each checklist and return list object, but each LDIF entry can name multiple attribute/value pairs.

To indicate that a transaction applies to the user's checklist (rather than to the user entry itself), use the keyword `check` as the value for `radiuslist` and `objectclass` within the transaction. You must assign this value to `radiuslist` in the distinguished name, and again before the list of attributes. You must also assign the value to `objectclass`, above the second `radiuslist` entry.

To indicate the return list, use the keyword `reply`.

The LDIF syntax to add a user entry, complete with a checklist and return list, is shown in Figure 87. Note that the `radiusname` and `radiusclass` values for all of the transactions that apply to the same User entry must be the same.

```
dn: radiusname=String,radiusclass=Native-User |
…,o=radius
changetype: add
objectclass: top
objectclass: Native-User | Solaris-User| …
objectclass: user
radiusname: String
password: {x-clear}PString | {x-md5}Hash |
{x-md5}{encrypt}PString |…
profile: NameOfProfileEntryInTheServerDatabase
login-limit: IntegerGivingConcurrentConnectionLimit
UserField: UserFieldValue
UserField: UserFieldValue

dn:
radiuslist=check,radiusname=String,radiusclass=Native-Us
er | …,o=radius
changetype: add
objectclass: top
objectclass: check
radiuslist: check
AttributeName: AttributeValue
AttributeName: AttributeValue
.
.
.
dn: radiuslist=reply,radiusname=String,radiusclass=
Native-User | …,o=radius
changetype: add
objectclass: top
objectclass: reply
radiuslist: reply
AttributeName: AttributeValue
AttributeName: AttributeValue
.
.
.
```

Figure 87 *Adding a User With Checklist and Return List Attributes*

Adding Proxy Targets with LDIF

The sample LDIF entry shown in Figure 88 adds the proxy RADIUS target `BIGCO.COM` to the Steel-Belted Radius database.

```
dn: radiusname=BIGCO.COM,radiusclass=Proxy,o=radius
changetype: add
objectclass: top
objectclass: Proxy
radiusname: BIGCO.COM
ip-address: 194.132.5.89
accounting: both
retry-count: 3
retry-timeout: 5000
shared-secret: testing123
include-in-auth-list: no
```

***Figure 88** Adding Proxy Targets*

The syntax in this LDIF entry is shown in Figure 89.

```
dn: radiusname=StringToParseAsProxyName,
radiusclass=Proxy, o=radius
changetype: add
objectclass: top
objectclass: Proxy
radiusname: StringToParseAsProxyName
ip-address: IPAddressOfTheTargetServer
accounting: Both | …
retry-count: Integer
retry-timeout: Integer
shared-secret:
SharedSecretThatWasConfiguredOnTheTargetServer
include-in-auth-list: Yes | No
ProxyField: ProxyFieldValue
ProxyField: ProxyFieldValue
.
.
.
```

***Figure 89** LDIF Syntax*

Adding Tunnels with LDIF

The sample LDIF entry shown in Figure 90 adds the tunnel `ACME.COM` to the Steel-Belted Radius database.

```
dn: radiusname=ACME.COM,radiusclass=Tunnel,o=radius
changetype: add
objectclass: top
objectclass: Tunnel
radiusname: ACME.COM
dnis-list: 8005551212;6171231234;12343210
description: Tunnel configuration for Acme Corp.
usage-limit: 24
```

***Figure 90** Adding Tunnels*

The syntax in this LDIF entry is shown in Figure 91.

```
dn:
radiusname=StringToParseAsTunnelName,radiusclass=Tunnel,
o=radius
changetype: add
objectclass: top
objectclass: Tunnel
radiusname: StringToParseAsTunnelName
dnis-list: PhoneNumber;PhoneNumber;etc
description: StringDescribingTunnel
usage-limit: IntegerGivingConcurrentConnectionLimit
TunnelField: TunnelFieldValue
TunnelField: TunnelFieldValue
.
.
.
```

***Figure 91** LDIF Syntax*

Adding IP Address Pools with LDIF

The sample LDIF entry shown in Figure 92 adds an IP address pool named `POOL1` to the Steel-Belted Radius database.

```
dn: radiusname=POOL1,radiusclass=IP-Addr-Pool,o=radius
changetype: add
objectclass: top
objectclass: IP-Addr-Pool
radiusname: POOL1
description: Address pool for common users
range: 198.187.100.1:50
range: 198.187.101.1:50
```

***Figure 92** Adding IP Address Pools*

The syntax in this LDIF entry is shown in Figure 93.

```
dn: radiusname=String,radiusclass=IP-Addr-Pool,o=radius
changetype: add
objectclass: top
objectclass: IP-Addr-Pool
radiusname: String
description: StringDescribingPool
range: IPAddress:Range
range: IPAddress:Range
.
.
.
```

Figure 93 *LDIF Syntax*

Adding IPX Address Pools with LDIF

The sample LDIF entry shown in Figure 94 adds an IPX address pool named `NETWARE1` to the Steel-Belted Radius database.

```
dn:
radiusname=NETWARE1,radiusclass=IPX-Addr-Pool,o=radius
changetype: add
objectclass: top
objectclass: IPX-Addr-Pool
radiusname: NETWARE1
description: IPX network numbers for dial in users
range: 0xffff0a00:500
```

Figure 94 *Adding IPX Address Pools*

The syntax in this LDIF entry is shown in Figure 95. You may provide multiple IPX address ranges using the `range` field.

```
dn: radiusname=String,radiusclass=IPX-Addr-Pool,o=radius
changetype: add
objectclass: top
objectclass: IPX-Addr-Pool
radiusname: String
description: StringDescribingPool
range: IPXAddress:Range
range: IPXAddress:Range
.
.
.
```

Figure 95 *LDIF Syntax*

Configuring a RADIUS Server with LDIF

The sample LDIF entry shown in configure your Steel-Belted Radius server by adding the Native User authentication method and defining conventions for tunnel name parsing.

```
dn: radiusclass=Server, o=radius
changetype: add
objectclass: top
objectclass: RadiusClass
radiusclass: Server
auth-methods: Native User
tunnel-delimiter: $
tunnel-type: prefix
```

***Figure 96** Adding a RADIUS Server*

The syntax in this LDIF entry is shown in Figure 97.

```
dn: radiusclass=Server, o=radius
changetype: add
objectclass: top
objectclass: RadiusClass
radiusclass: Server
auth-methods: Native User | Solaris User | SecurID
Prefix | …
tunnel-delimiter: Character
tunnel-type: Prefix | Suffix | Neither
ConfigurationField: ConfigurationFieldValue
ConfigurationField: ConfigurationFieldValue
.
.
.
```

***Figure 97** LDIF Syntax*

Statistics Variables

Server statistics counters record the number of certain types of events. The LCI allows you to read these statistics to monitor the performance of your Steel-Belted Radius server.

***NOTE:** The Enterprise Edition of Steel-Belted Radius with the optional LCI add-on does not support the Statistics items.*

Counter Statistics

The statistics counters can be accessed via the LCI by executing the following one line command:

```
ldapsearch -V 2 -h 127.0.0.1 -p 667 -D "cn=admin,o=radius"
-w radius -s sub -T  -b "radiusstatus=statistics,o=radius"
stattype=typeofstatus
```

The following sections illustrate the variables displayed for different settings of the `stattype` parameter.

stattype: server

```
dn: stattype=server,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: server
start-time: 2002/05/08 13:29:08
up-time: 26188
ip-address: 192.168.21.142
version: v 2.20.33
authentication-threads: 0
accounting-threads: 0
total-threads: 0
max-auth-threads: 100
max-acct-threads: 100
max-total-threads: 200
high-auth-threads: 2
high-acct-threads: 0
high-total-threads: 2
```

stattype: authentication

```
dn:
stattype=authentication,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: authentication
accept: 1
reject: 0
silent-discard: 0
total-transactions: 8
invalid-request: 0
failed-authentication: 0
failed-on-check-list: 0
insufficient-resources: 0
proxy-failure: 0
rejected-by-proxy: 0
transactions-retried: 0
total-retry-packets: 0
```

stattype: accounting

```
dn: stattype=accounting,radiusstatus=statistics,o=radius
objectclass: top
```

```
objectclass: radiusstatus
radiusstatus: statistics
stattype: accounting
start: 0
stop: 0
on: 0
off: 0
total-transactions: 0
invalid-request: 0
invalid-client: 0
invalid-shared-secret: 0
insufficient-resources: 0
proxy-failure: 0
transactions-retried: 0
total-retry-packets: 0
```

stattype: proxy

```
dn: stattype=proxy,radiusstatus=statistics,o=radius
objectclass: top
objectclass: radiusstatus
radiusstatus: statistics
stattype: proxy
authentication: 0
accounting: 0
total-transactions: 0
timed-out: 0
invalid-response: 0
invalid-shared-secret: 0
insufficient-resources: 0
transactions-retried: 0
total-retry-packets: 0
```

Rate Statistics

Rate statistics are derived from existing counter statistics by taking time into consideration. Rate values calculated for each of these counter statistics consist of the following:

- Current rate – The rate measured over the most recent rate interval.
- Average rate – The rate measured since the Steel-Belted Radius server was started or since the last time statistics were reset to zero.
- Peak rate – The highest rate observed since the Steel-Belted Radius server was started or since the last time statistics were reset to zero.

Additionally, there is a (read-only) time value used in calculations:

- Rate statistics seconds-per interval – The duration (in seconds) of the interval over which the rate statistics are gathered.

To read rate statistics from the LCI, you must set `stattype: rate`. This results in output such as the following:

```
rate-statistics-seconds-per-interval: 1
auth-request-current-rate: 0
auth-request-average-rate: 0
auth-request-peak-rate: 7
auth-accept-current-rate: 0
auth-accept-average-rate: 0
auth-accept-peak-rate: 1
auth-reject-current-rate: 0
auth-reject-average-rate: 0
auth-reject-peak-rate: 0
acct-start-current-rate: 0
acct-start-average-rate: 0
acct-start-peak-rate: 0
acct-stop-current-rate: 0
acct-stop-average-rate: 0
acct-stop-peak-rate: 0
proxy-auth-request-current-rate: 0
proxy-auth-request-average-rate: 0
proxy-auth-request-peak-rate: 0
proxy-acct-request-current-rate: 0
proxy-acct-request-average-rate: 0
proxy-acct-request-peak-rate: 0
proxy-fail-timeout-current-rate: 0
proxy-fail-timeout-average-rate: 0
proxy-fail-timeout-peak-rate: 0
proxy-fail-badresp-current-rate: 0
proxy-fail-badresp-average-rate: 0
proxy-fail-badresp-peak-rate: 0
proxy-fail-badsecret-current-rate: 0
proxy-fail-badsecret-average-rate: 0
proxy-fail-badsecret-peak-rate: 0
proxy-fail-missingresr-current-rate: 0
proxy-fail-missingresr-average-rate: 0
proxy-fail-missingresr-peak-rate: 0
proxy-retries-current-rate: 0
proxy-retries-average-rate: 0
proxy-retries-peak-rate: 0
proxy-auth-rej-proxy-current-rate: 0
proxy-auth-rej-proxy-average-rate: 0
proxy-auth-rej-proxy-peak-rate: 0
proxy-acct-fail-prox-current-rate: 0
proxy-acct-fail-prox-average-rate: 0
proxy-acct-fail-prox-peak-rate: 0
proxy-auth-rej-proxy-error-current-rate: 0
proxy-auth-rej-proxy-error-average-rate: 0
proxy-auth-rej-proxy-error-peak-rate: 0
```

Chapter 14
Configuring SQL Authentication

This chapter presents an overview of SQL authentication and describes how to configure SQL authentication in Steel-Belted Radius.

About SQL Authentication

Steel-Belted Radius can authenticate against records stored in an external SQL database. Any attribute or set of attributes, such as username and password, can be used to query the database.

External database authentication is typically used when an organization already has a large amount of user information stored in a SQL database, and this information is to be used to authenticate these users using RADIUS. Authentication against an existing database extends authentication services to user accounts without requiring an administrator to enter user information into the Steel-Belted Radius database.

Steel-Belted Radius offers the SQL authentication feature as a plug-in software module. Key features of the SQL plug-in include:

- The SQL statement is completely user-specified, allowing support of existing tables with existing field names and formats.
- The SQL statement supports a wide range of arithmetic and string expressions as part of the statement.
- The SQL statement is parameterized, so it is compiled once, and each execution uses variable data without need for recompilation.
- Multiple authentications may be overlapped at the same time.
- The SQL authentication method, which appears in the Authentication Policies panel in SBR Administrator, can be activated/deactivated and ordered with respect to other authentication methods.
- Multiple instances of the SQL authentication module can operate simultaneously, allowing authentication to multiple databases.
- If the database connection drops, it is automatically reestablished after a configurable timeout without Steel-Belted Radius being restarted.

- Data from the database can be returned as attributes in the Access-Accept message.

Warning: *While Steel-Belted Radius does its best to provide uniformity in the operation of databases from different vendors, differences occur, particularly in the way SQL statements are interpreted. The capabilities of the SQL authentication module depend on the capabilities of the underlying databases and their clients; things that work with one database may not work with another.*

SQL Authentication Process

Any RADIUS attribute (or Steel-Belted Radius request variable) from the request can be used in an SQL SELECT statement. Any return list attribute (that is, a Steel-Belted Radius response variable) can be retrieved from a SQL database and returned in a RADIUS access response message.

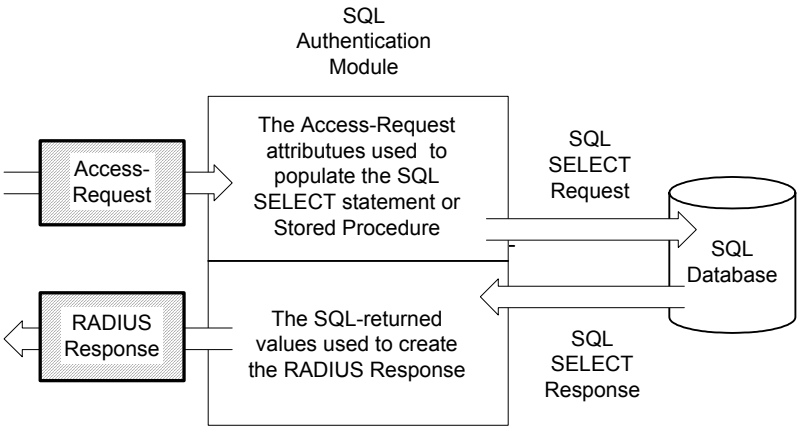


Figure 98 *SQL Authentication Process*

Stored Procedures

A stored procedure is a sequence of SQL statements that form a logical unit and perform a particular task. You can use stored procedures to encapsulate a set of queries or operations that can be executed repeatedly on a database server. For example, you can code operations on an employee database, such as password lookup, as stored procedures that can be executed by application code.

Stored procedures can be compiled and executed with different parameters and results. Stored procedures can use any combination of input parameters (the values passed to the stored procedure at execution time) and output parameters (the values set or returned by the stored procedure to the calling application or environment).

You can write stored procedures for SQL that communicate with Steel-Belted Radius via input and output parameters to implement custom functions. Stored procedures let you use server-side processing on the SQL server to manipulate the information specified by variables. How you use these stored procedures depends on details specific to the implementation of SQL that you are using.

For information on using stored procedures with the Oracle SQL database, see "Working With Stored Procedures in Oracle" on page 190. For information on using stored procedures with the Microsoft SQL database, see "Working With Stored Procedures in MS-SQL" on page 191.

Connectivity Issues

Steel-Belted Radius may encounter serious problems if the connection between Oracle and Steel-Belted Radius becomes unstable The most common reasons for a connection becoming unstable are:

- Slow or unreliable network response times
- Interruptions in connectivity caused by intervening network devices, such as a firewall timing out the connection

To prevent connectivity problems, consider implementation of one of the following solutions:

- To minimize problems caused by intervening firewalls, configure your firewall to pass traffic on the Oracle communications ports between the Steel-Belted Radius server and the Oracle server without restriction.
- To minimize network latency and firewall-related problems, move the Steel-Belted Radius server to the same network segment as the Oracle server.
- If moving your Steel-Belted Radius server is not feasible, locate a second Steel-Belted Radius server on the same network segment as your Oracle server, and configure your current Steel-Belted Radius server to proxy all authentication requests to this new device. This configuration will allow you to open RADIUS ports on the firewall only for the Steel-Belted Radius server (instead of opening RADIUS ports for all RAS devices). Because proxy functions in Steel-Belted Radius do not require an uninterrupted connection to process requests, this solution allows you to retain your current firewall timeout settings.

Configuring SQL Authentication

You must configure both Steel-Belted Radius and the SQL database to support SQL authentication. The configuration procedure must be tailored to the database that you use. However, all procedures must give the following results:

- The required transport must be in place between SQL client software and the SQL server.
- The SQL server must be configured via a plug-in to coordinate with SQL client software.
- The Steel-Belted Radius server must be configured to communicate with the SQL client software in order to interact with the back-end SQL server to perform stored procedures or SQL queries.

Files

The following files establish settings for configuring SQL authentication in Steel-Belted Radius. For more information about these files, refer to the *Steel-Belted Radius Reference Guide*.

Table 27. *SQL Authentication Files*

File Name	Function
`radsql.aut`	Configures settings for SQL authentication (Solaris/Linux).
`sqlauth.aut`	Configures settings for SQL authentication (Windows).
`radsqljdbc.aut`	Configures settings for SQL authentication using JDBC (Solaris/Linux).

Using the SQL Authentication Header File

To configure SQL authentication, you must edit the authentication header files, `radsql.aut` (Solaris/Linux with Oracle), `radsqljdbc.aut` (Solaris/Linux with JDBC), or `sqlauth.aut` (Windows/ODBC), which are located in the same directory that contains the Steel-Belted Radius service or daemon. Most of these options may be left at their original settings; however, you must modify certain options to accommodate your own database.

Tip: See "Configuring the Authentication Sequence" on page 13.

After you complete your changes to the authentication header files and restart Steel-Belted Radius, the `InitializationString` value that you entered in the [Bootstrap] section of the header file appears in the Authentication Methods tab in the Authentication Policies panel. You can then enable, disable, or prioritize your SQL database like any other authentication method in the list.

Using Multiple SQL Authentication Methods

You can configure Steel-Belted Radius to authenticate users against more than one SQL database. Each database that you set up in this way becomes a separate selection in the Authentication Methods tab in the Authentication Policies panel.

To add an additional database, create a new header file with extension `.aut` in the same directory as `radsql.aut` (Solaris/Linux with Oracle), `radsqljdbc.aut` (Solaris/Linux with JDBC), or `sqlauth.aut` (Windows/ODBC). You can give this file any name you like, provided its extension is `.aut`. At startup, Steel-Belted Radius enumerates all `.aut` files to create its list of authentication methods.

When creating the new file, start by copying the original `.aut` file. Be sure to change its `InitializationString` entry to a unique authentication method name; otherwise, Steel-Belted Radius has no way of distinguishing between the different methods in the authentication methods list.

Connecting to the SQL Database

Upon startup, the SQL authentication module connects to the database, using settings specified by a connect string specified in the header file. The connect string contains information such as the name and location of the database, and the password required to connect. The connect string is passed to the database client to establish the connection.

While a sample connect string is provided in the original header file, you must configure the `Connect` entry of the header file with a connect string appropriate to your database.

The password for database access must be provided as part of the connect string. If it is not:

- Solaris/Linux: The connection fails.
- Windows: A pop-up window prompts you to enter the password before making the connection at startup and each time a reconnect is required.

If the initial attempt to connect to the database fails, or if a processing error occurs that the SQL authentication module interprets as a database connection failure, the SQL authentication module drops the connection and attempts to establish a new connection after a period of time. In the interim, all authentication requests are ignored.

The SQL authentication module uses an exponential back-off strategy in determining how long to wait before attempting a new connection, as well as how frequently this attempt should be made. After the first dropped connection, it waits a certain amount of time before attempting to reconnect. If this attempt to reconnect also fails, it waits for twice the amount of time before trying again; and so on, up to some maximum wait time. The initial and maximum wait times are configurable.

Warning: *(Solaris/Linux only): Detailed error information may not be available if there is an error processing the database logon at connect time. A numeric result code is displayed in the log. You may need to refer to product-specific documentation to decode this result code. With Oracle on Solaris/Linux, you can use the* `oerr facility-code error-number` *command with a facility code of* `ora` *from the command shell.*

SQL Statement Construction

The authentication transaction is based on an SQL query that returns a password (and possibly other information) based on the name entered by the user attempting to log in.

While a sample SQL query is provided in the original header file, you must configure the SQL entry of the header file with a query appropriate to your database. The query you enter must be either a `SQL SELECT` or `SQL EXECUTE` statement that contains additional syntax elements which are preprocessed by the SQL authentication module.

The SQL authentication module executes SQL statements in parameterized form. This means that the SQL statement is compiled once, with parameter markers (usually question marks) as placeholders for data items that vary from one execution to the next. Only upon execution of the statement are the actual data values supplied.

The SQL statement you compose must not include parameter markers directly. Instead, the names of the parameters should be included where parameter markers would appear, in a format described below. The SQL authentication module translates the SQL statement provided, replacing parameter names with parameter markers prior to passing the SQL statement to the database engine.

The SQL statement can be very simple. Basically, all that is required is to look up a password and possibly some optional information based on a username. The SQL statement can also be quite complex; it can include inner joins, and it can contain expressions. The underlying database engine is responsible for handling the SQL statement; the SQL authentication module performs no interpretation of the SQL statement other than to translate parameter names to parameter markers.

Example:

```
SELECT password, profile, fullname FROM usertable WHERE
username = %name/63s
```

As shown in the example above, a parameter consists of a percent sign (%), the name of the parameter and a format specifier. Table 28 lists SQL statement parameter names.

***Table 28.** SQL Statement Parameters*

Item	Meaning for SQL Authentication
%OriginalUserName	The original full identification of the user, prior to any processing (that is, `user@realm`).
%UserName	The full user identification (user and realm strings) after all stripping and processing has been performed.
%Name	Synonym for UserName.
%NASName	The name of the RAS device that originated the request. This may be the name of the RADIUS clients entry in the database or the value of the NAS-Identifier or NAS-IP-Address attribute.
%NASAddress	The address of the RAS device, in dotted notation.
%NASModel	The make/model of the RAS device, as specified in the Steel-Belted Radius database.
%Password	The PAP password.
%RADIUSClientName	The name of the RAS device, as specified in a RADIUS clients entry in the Steel-Belted Radius database.

Along with these parameters, any RADIUS attribute received in the Access-Request can be referred to by using an at-sign ('@') followed by the name of the attribute. If you need to specify a literal at-sign character in an SQL statement, such as in a User-Name, you must use two at-signs in a row. For example:

```
SELECT foo FROM bar WHERE field = 'abc@@xyz'
```

Likewise, if you need to specify a literal percent character ('%') in an SQL statement you must use a two percent characters in a row.

The format specifier should describe the database storage format of the column that corresponds to the parameter. It consists of a slash ('/'), a length, and a type, which for

SQL authentication is always 's' for string. For example, if the user's name is stored in the database as a string of up to 63 bytes, you would enter:

```
%name/63s
```

Warning: *Be sure to specify a length no greater than the actual field size in the database. The compilation of the SQL statement may fail if a parameter size greater than the actual field size is specified.*

Password Parameters

Normally, the only parameter you'd include in the SQL statement is `%name`. The `%password` parameter is available to support databases containing non-unique usernames. For example, your database might allow two people named "`George`"; one with password "`swordfish`", and the other with password "`martha`". You can authenticate them correctly with the following query:

```
SELECT password, profile, fullname FROM usertable WHERE
username = %name/63s and password = %password/63s
```

You must return the password as the first column of the result to perform authentication. If the password is not returned in a password column or as an output parameter, no password authentication is performed.

In the following statement, `%name` is an input parameter used to look up a record.

```
SELECT profile FROM database WHERE username = %name
```

Since there's no password output parameter, no password authentication is performed. The [Results] section of the `.aut` file should look something like the following to work with the above `SELECT` statement:

```
[Results]
Password=0
Profile=1/50
Alias=0
```

If the record cannot be found in the database, the authentication attempt fails.

NOTE: *If you are not using password checking for authentication, the Password parameter must be set to 0 in the [Results] section.*

Overlapped Execution of SQL Statements

The SQL authentication module is multi-threaded. SQL authentication can be configured with a maximum number of simultaneous executions of any SQL statement, using the `MaxConcurrent` entry in the `.aut` file's [Settings] section.

If `MaxConcurrent` is set to 1, SQL execution occurs serially, and the SQL execution for each authentication request must complete before execution for the next request may begin.

By increasing `MaxConcurrent`, it may be possible to increase throughput by overlapping operations, especially if the database server is remote and a large part of the time to complete a statement execution is taken up by network latency. If the database

server is local, the point of diminishing returns may be reached at a small value of `MaxConcurrent`, possibly even at 1 or 2. The optimum value is a matter of experiment.

Warning: *A setting of `MaxConcurrent = 1` should be sufficient for all but the most demanding environments. Increase this value only slowly and conservatively.*

You might expect that databases that are licensed by number of connections would debit a single connection regardless of how many SQL statements are active. This is not necessarily the case; some databases count each open compiled SQL statement against the licensed number of connections. So another factor that determines how `MaxConcurrent` should be set might be the database license.

%result Parameter

The `%result` parameter is a string value that can be returned as a column or stored procedure output parameter. The `%result` parameter can be used with or without password authentication.

The value expected to be returned in this parameter when authenticating a user can be specified in the `SuccessResult` entry of the [Settings] section. For example, if a user is successfully authenticated by the SQL authentication method, the result signifying success is the text string "okay". This can be automatically checked by the following setting.

```
[Settings]
SuccessResult = okay
```

NOTE: *The string comparison is case insensitive.*

If the SQL statement succeeds but the SuccessResult value does not match the expected value returned from the database, Steel-Belted Radius issues a reject response, which can include any attributes and values configured in the [FailedSuccessResultAttributes] section of the `*.aut` file.

If `PerformSuccessResultCheckAfterPasswordCheck=1` is specified and the SQL statement performs a password check that fails, Steel-Belted Radius does not process the SuccessResult and does not return the attributes from the [FailedSuccessResultAttributes] section in the reject response. If `PerformSuccessResultCheckAfterPasswordCheck=1` is specified and the SQL statement performs a password check that succeeds but the SuccessResult value does not match the value returned from the database, Steel-Belted Radius issues a reject response that contains the attributes from the [FailedSuccessResultAttributes] section.

In the following statement, `%password` is passed to a stored procedure, which returns a `%result` of either "okay" or something else (that signifies a rejection):

```
BEGIN CheckUser(%name, %password, %result!o); END;
```

Another example might be a database of usernames, passwords, and account status. The administrator can enable a user by setting account status to "okay", disable by setting to some other value, without having to delete the record. In the following statement, both password and result columns are checked:

```
SELECT password, result FROM database WHERE username = %name

[Results]
Password=1/50
%Result=2/50
Profile=0
Alias=0
```

SQL Authentication and Password Format

Steel-Belted Radius supports the authentication of users residing in a SQL database, in which password values for the users are stored in one of the following formats: clear text, UNIXcrypt, Secured Hash Algorithm (SHA1+Base64 hash), MD4 hash, or enc-md5 reversibly-encoded password.

Hashed Passwords

Values in the Password column include a prefix that indicates how the password has been processed. The prefix is in clear text between curly braces { } and is immediately followed by a hash value computed from the password. If no prefix is present in the value retrieved from the table Password column, the entire password is assumed to be in clear text format. In summary:

- *PasswordText* indicates clear text format (no encryption)
- {crypt}*HashHash* indicates UNIXcrypt format
- {SHA}*HashHashHash* indicates SHA1+Base64 hash
- {SSHA}*HashHashHashSalt* indicates salted SHA1+Base64 hash
- {md4}*HashHash* indicates MD4 hash of the Unicode form of password
- {enc-md5}*EncryptedEncrypted* indicates a reversibly encrypted password

***NOTE:** Refer to RFC 2759 for details about how MS-CHAP-V2 produces an MD4 hash value.*

***NOTE:** Although Steel-Belted Radius reads passwords encoded in enc-md5 format, you must purchase the Software Developer's Kit to convert clear-text passwords to this format.*

UNIXcrypt is the standard hash algorithm that is used for the /etc/passwd file on Solaris/Linux systems. This may be necessary if, for example, the standard user database on a Solaris or Linux machine (the /etc/passwd file) is migrated to a SQL database, so that the values in the Password column of the SQL table are processed with UNIXcrypt.

Steel-Belted Radius may be configured to expect that the values retrieved from the SQL table Password column during authentication have been run through UNIXcrypt by adding the following entry into the [Settings] section of the SQL authentication header file:

```
PasswordFormat=3
```

Automatic Parsing

If `PasswordFormat` is set to 0, Steel-Belted Radius attempts to determine the password format automatically by parsing it. This is the recommended setting. Automatic parsing expects the password to be stored in one of the formats above.

***NOTE:** The setting for automatic password parsing in older versions of Steel-Belted Radius (`auto`) has been deprecated.*

Working With Stored Procedures in Oracle

The following notes discuss some considerations specific to Oracle, which uses the term *package* and *package body* when referring to stored procedures.

Assume you have a `SELECT` statement that extracts a user's name, password, and profile from the table `usertable` when it receives the user's name as an input parameter:

```
SELECT fullname, password, profile FROM usertable WHERE
username = %name/63s
```

To write a package called `myPack1` that performs the equivalent function, you would enter the following sequence of commands:

```
Package myPack1
        is
PROCEDURE myProc
(
name  IN  VARCHAR2,
pass  OUT VARCHAR2,
prof  OUT VARCHAR2,
fName OUT VARCHAR2
);
End myPack1;
```

When referencing the package from `sqlauth.aut`, you would point to the package name `myPack1` (not the procedure name `myProc`):

```
Package Body myPack1
            is
PROCEDURE myProc
(
name  IN  VARCHAR2,
pass  OUT VARCHAR2,
prof  OUT VARCHAR2,
fName OUT VARCHAR2
)

IS

BEGIN

SELECT fullname INTO fName, password INTO pass, profile INTO
prof FROM usertable WHERE username = name;
END myProc;
```

```
End myPack1;
```

When you invoke the stored procedure, you would delineate each parameter as an input (`!i`), output (`!o`), or input/output (`!io`) variable. The presence of a `!io` or `!o` keyword indicates that the value returned from the database is to be included in the Access-Accept response as if it had been coded in the [Results] section. If a `r` value is included in the suffix (for example, `!ir`, `!r`, or `!or`), the parameter is expected to be an output parameter, and the attribute is to be treated as if it were included in the [FailedSuccessResultAttributes] section. Variables that are not specifically marked are considered input parameters by default.

You could replace the `SELECT` statement by invoking `myProc` as follows:

```
SQL=BEGIN myPack1.myProc(%name!i, %password!o, %profile!o,
%fullname!o ); END;
```

When using input-output parameters with Oracle, you must set the `DefaultResults` setting to 0. Any other variables that need to be returned (such as `Reply-Message`) must be identified by the "`!o`" marker within the SQL statement.

Working With Stored Procedures in MS-SQL

A simple example of a stored procedure returns a result set in the same way as a `Select` statement. For example, assume you have a table with the following fields: `username`, `password`, `Alias`, and `active`, where all fields have the datatype `varchar`. You want a stored procedure that will return a password and alias when the username and password received in the request match entries in the database, provided that `active` field has a value of yes'.

Example 1

To create a simple stored procedure, run the following command sequence from MS Query Analyzer to create a stored procedure called `rsp_getpword`.

```
CREATE PROCEDURE rsp_getpword
@Uname varchar(21),
@pword varchar(21)

AS
SELECT password, alias FROM authentication WHERE username =
@Uname
AND password = @pword AND active = 'yes'
GO
```

This stored procedure can then be executed from a `*.aut` file as follows:

```
SQL= Execute rsp_getpword %username, %password

[results]
Password=1
Alias=2
```

Example 2

More complex stored procedures take input and output parameters in a manner similar to that used by Oracle. For example, assume you have a table with the following fields: `username`, `password`, `profile`, and `active`, where all fields have a datatype of

varchar. You want a stored procedure that returns a password and profile when the username and password received in the request match a username and password in the database, provided that the active field has a value of yes.

First, to create the stored procedure, run the following command from MS query analyzer:

```
CREATE PROCEDURE rsp_authuser
@uname as varchar(20),
@pword as varchar(21) OUTPUT,
@profile as varchar(21)OUTPUT
AS
SELECT @pword =password,@profile=profile  FROM
authentication WHERE username = @uname AND active = 'yes'
GO
```

This stored procedure can then be executed from a *.aut file as follows:

```
SQL= {call rsp_authuser (%username!i, %password!o,
%profile!o)}
[results]
; No entries should be specified in results, everything but
the header should be commented out.
```

Chapter 15
Configuring SQL Accounting

This chapter presents an overview of SQL accounting and describes how to configure SQL accounting in Steel-Belted Radius.

About SQL Accounting

Steel-Belted Radius can write RADIUS accounting information to an external SQL database, independently of the Steel-Belted Radius accounting log.

To set up an external database for use as a repository for RADIUS accounting data, you must place an `.acc` database configuration file in the same directory that contains the Steel-Belted Radius service (normally `C:\RADIUS\Service`) or daemon. This file must be modified to contain specialized information about your enterprise database.

Steel-Belted Radius offers the SQL accounting feature as a plug-in software module. Key features of the SQL plug-in include:

- The SQL statement is completely user-specified, allowing support of existing tables with existing field names and formats.
- The SQL statement can include a wide variety of arithmetic and string expressions.
- The SQL statement is parameterized, so it is compiled once, and each execution uses variable data without need for recompilation.
- Attribute and other data from the accounting request can be mapped to any parameter of the SQL statement (and hence to any field in the table) by means of a simple syntax.
- Different request types can be mapped to different SQL statements that may operate against distinct tables within the database.
- Multiple instances of a SQL statement can be overlapped for simultaneous execution.
- Multiple instances of the SQL accounting module can operate simultaneously, allowing logging to multiple databases.
- If the database connection drops, it is automatically reestablished after a configurable timeout without restarting Steel-Belted Radius.

- SQL accounting responses can return information.
- Stored procedures invoked by SQL accounting can make use of input parameters, record results, and return output parameters.

Warning: *While Steel-Belted Radius tries to provide uniformity in the operation of databases from different vendors, differences exist, particularly in the way SQL statements are interpreted. The capabilities of the SQL Authentication module depend on the capabilities of the underlying databases and their clients; things that work with one database may not work with another.*

Stored Procedures

A stored procedure is a sequence of SQL statements that form a logical unit and perform a particular task. You can use stored procedures to encapsulate a set of queries or operations that can be executed repeatedly on a database server. For example, you can code operations on an employee database, such as password lookup, as stored procedures that can be executed by application code.

Stored procedures can be compiled and executed with different parameters and results. Stored procedures can use any combination of input parameters (the values passed to the stored procedure at execution time) and output parameters (the values set or returned by the stored procedure to the calling application or environment).

You can write stored procedures for SQL that communicate with Steel-Belted Radius via input and output parameters to implement custom functions. Stored procedures let you use server-side processing on the SQL server to manipulate the information specified by variables. How you use these stored procedures depends on details specific to the implementation of SQL that you are using.

For information on using stored procedures with the Oracle SQL database, see "Working With Stored Procedures in Oracle" on page 190. For information on using stored procedures with the Microsoft SQL database, see "Working With Stored Procedures in MS-SQL" on page 191.

Connectivity Issues

Steel-Belted Radius may encounter serious problems if the connection between Oracle and Steel-Belted Radius becomes unstable The most common reasons for a connection becoming unstable are:

- Slow or unreliable network response times
- Interruptions in connectivity caused by intervening network devices, such as a firewall timing out the connection

To prevent connectivity problems, consider implementation of one of the following solutions:

- To minimize problems caused by intervening firewalls, configure your firewall to pass traffic on the Oracle communications ports between the Steel-Belted Radius server and the Oracle server without restriction.

- To minimize network latency and firewall-related problems, move the Steel-Belted Radius server to the same network segment as the Oracle server.
- If moving your Steel-Belted Radius server is not feasible, locate a second Steel-Belted Radius server on the same network segment as your Oracle server, and configure your current Steel-Belted Radius server to proxy all authentication requests to this new device. This configuration will allow you to open RADIUS ports on the firewall only for the Steel-Belted Radius server (instead of opening RADIUS ports for all RAS devices). Because proxy functions in Steel-Belted Radius do not require an uninterrupted connection to process requests, this solution allows you to retain your current firewall timeout settings.

Configuring SQL Accounting

You must configure both Steel-Belted Radius and the SQL database to support SQL accounting. The configuration procedure must be tailored to the database that you use. However, all procedures must give the following results:

- The SQL server must be configured to be listening for client requests. Note that for SQL purposes, the Steel-Belted Radius server must be a client of the SQL server.
- The Steel-Belted Radius server must know the machine where the SQL server software runs, and it must know the protocol and port used in communicating with that machine.
- The required transport must be in place between SQL client and server.

Files

The following files establish settings for configuring SQL accounting in Steel-Belted Radius. For more information about these files, refer to the *Steel-Belted Radius Reference Guide*.

***Table 29.** SQL Accounting Files*

File Name	Function
`radsql.acc`	Configures settings for SQL accounting (Solaris/Linux and Oracle).
`radsqljdbc.acc`	Configures settings for SQL accounting (Solaris/Linux and JDBC).
`sqlacct.acc`	Configures settings for SQL accounting (Windows/ODBC).

Using the SQL Accounting Header File

To configure SQL accounting, you must edit the accounting header file (`radsql.acc` (Solaris/Linux and ODBC), `radsqljdbc.acc` (Solaris/Linux and JDBC) or `sqlacct.acc` (Windows)), located in the same directory that contains the Steel-Belted Radius service (normally `C:\RADIUS\Service`) or daemon. A reference listing of all header file options appears below.

You must modify certain options in the accounting header file to accommodate your own database. After you update your accounting header file and restart Steel-Belted Radius, accounting proceeds as you have configured it.

Using Multiple SQL Databases

You can configure Steel-Belted Radius to log accounting transactions against more than one SQL database.

To add an additional database, create a new header file with extension `.acc` in the same directory as `radsql.acc` (Solaris/Linux and ODBC), `radsqljdbc.acc` (Solaris/Linux and JDBC), or `sqlacct.acc` (Windows). You can give this file any name you like, provided its extension is `.acc`. At startup, Steel-Belted Radius enumerates all `.acc` files to create its list of accounting modules.

***NOTE:** When creating the new file, start by duplicating the original `.acc` file, then make whatever modifications are necessary.*

Connecting to the SQL Database

Upon startup, the SQL accounting module connects to the database, based on a connect string specified in your accounting header file. The connect string contains information such as the name and location of the database, and the password required to connect. The connect string is passed to the database client to establish the connection.

While a sample connect string is provided in the original header file, you must configure the Connect entry of the header file with a connect string appropriate to your database.

The password for database access must be provided as part of the connect string or the following results occur:

- Solaris/Linux: The connection fails.
- Windows: A pop-up window prompts the user to enter a password before making the connection at startup and each time a reconnect is required.

If the initial attempt to connect to the database fails, or if a processing error occurs that the SQL accounting module interprets as a database connection failure, the SQL accounting module drops the connection and attempts to establish a new connection after a period of time. In the interim, all authentication requests are ignored.

The SQL accounting module uses an exponential back-off strategy in determining how long to wait before attempting a new connection, as well as how frequently this attempt should be made. After the first dropped connection, it waits a certain amount of time before attempting to reconnect. If this attempt to reconnect also fails, it waits for twice the amount of time before trying again; and so on, up to some maximum wait time. The initial and maximum wait times are configurable.

Warning: *(Solaris/Linux only): Detailed error information may not be available if there is an error processing the database logon at connect time. A numeric result code appears in the log. You may need to refer to product-specific documentation to decode this result code. With Oracle on Solaris/Linux, you can use the* `oerr facility-code error-number` *command with a facility code of ora from the command shell.*

SQL Statement Construction

For each accounting request whose Acct-Status-Type is mapped to a SQL statement, that accounting request is logged to the backend database by executing the associated SQL statement.

While a sample SQL statement is provided in the original header file, you must configure one or more SQL entries of the header file with a statement appropriate to your database. Each SQL statement is typically an `INSERT INTO` statement and may contain additional syntax elements that are preprocessed by the SQL accounting module.

The SQL accounting module executes SQL statements in parameterized form. This means that the SQL statement is compiled once, with parameter markers (usually question marks) as placeholders for data items that vary from one execution to the next. Only upon execution of the statement are the actual data values supplied.

The SQL statement you compose must not include parameter markers directly. Instead, the names of the parameters should be included where parameter markers would appear, in a format described below. The SQL authentication module translates the SQL statement provided, replacing parameter names with parameter markers prior to passing the SQL statement to the database engine.

A SQL statement can be very simple. Basically, all that is required is to set fields of the database record with values from the request. The SQL statement can also be quite complex; it can include inner joins, and it can contain expressions. The underlying database engine is responsible for handling the SQL statement; The SQL accounting module performs no interpretation of the SQL statement other than to translate parameter names to parameter markers.

INSERT Statement and VALUES Section

The following is an example of a SQL `INSERT` statement that might be found in a Steel-Belted Radius `.acc` file:

```
INSERT INTO usagelog (Time, NASAddress, SessionID, Type,
Name, BytesIn, BytesOut) VALUES (%TransactionTime,
%NASAddress, @Acct-Session-Id, @Acct-Status-Type,
%FullName/40s, @Acct-Input-Octets, @Acct-Output-Octets)
```

In the `VALUES` section, the names (between parentheses) represent the values inserted into the SQL table columns. To support the SQL accounting module, each item in the VALUES section must be prefixed with a @ sign or a % sign.

- @ indicates a RADIUS accounting attribute. The attribute name must also be listed in the `account.ini` file. This remains true even if the `account.ini` file is disabled.
- % indicates an item associated with the `INSERT` request that is not a RADIUS accounting attribute. Table 30 lists the Steel-Belted Radius items that may be provided.

***Table 30.** Insert Statement Syntax*

Item	Data Type	Meaning
%TransactionTime	Time	The date/time that the event occurred that is the subject of the request.
%Time	Time	The date/time when the request is being processed. (This is later than %TransactionTime if the request is a retry.)
%Type	String	The RADIUS accounting request type.
%NASAddress	IP address	The IP address of the requesting RAS.
%NASName	String	The name of the RAS device that originated the request. This may be the name of the RADIUS client entry in the database or the value of the NAS-Identifier or NAS-IP-Address attribute.
%NASModel	String	The RAS make/model.
%FullName	String	The full name of the logged in user.
%AuthType	String	The method by which the user was authenticated.
%RADIUSClientName	String	The name of the RAS device, as specified in a RADIUS client entry in the Steel-Belted Radius database.

A format specifier may appear immediately following each parameter. The format specifier should describe the database storage format of the column that corresponds to the parameter. It consists of a slash ('/'), possibly a length, and a data type. The following data types are available:

***Table 31.** Data Types*

Format Specifier	Meaning
/*x*s	A text string of length *x*. /s indicates a string with the default length of 256.
/*x*b	A binary data string of length *x*. A binary string is different from an text string in that it is not NULL-terminated and is not restricted to ASCII characters. /b indicates a binary data string with the default length of 256.
/n	32-bit integer
/n8	8-bit integer
/n16	16-bit integer
/n32	32-bit integer (same as /n)

Table 31. *Data Types (Continued)*

Format Specifier	Meaning
/n*xx*	Integer xx bits in length. For example, /n64 indicates a number with a length of 64 bits.
/t	Timestamp

NOTE: *Steel-Belted Radius supports integers larger than 32 bits by manipulating them as binary data strings. The Solaris Oracle 8 plug-ins are able to convert binary data strings between Oracle VARRAW types (/xb) and Oracle NUMBER types (/n). Oracle types must be declared with enough precision to avoid truncation when inserting into the database, and care must also be taken to avoid truncation when retrieving from the database. In particular, avoid retrieving Oracle VARRAW types larger than 256 bytes. Other database/operating-system combinations may not allow for integers larger than 32 bits.*

If a format specifier is not present in the SQL statement syntax, Steel-Belted Radius automatically defaults to an appropriate specifier based on the actual parameter type. For example, `@Acct-Input-Octets` is a number, and defaults to `/n`.

NOTE: *For strings, always include a format specifier, and be sure to specify a length no greater than the actual field size in the database. The compilation of the SQL statement may fail if a length greater than the actual field size is specified. If no format specifier is present, the length defaults to 256 characters, which may cause the compilation to fail.*

Steel-Belted Radius automatically attempts to convert between the internal format of a parameter and its format in the database, as described by the format specifier. In most cases, the formats are equivalent; if not, Steel-Belted Radius performs reasonable conversions.

Table 32 lists the internal formats and their compatible database formats:

Table 32.

Internal Format	Compatible Database Formats
Binary data string	/b, /xb, /n, /n8, /n16, /n32
Number	/n, /n8, /n16, /n32, /xs, /s
String	/xs, /s
Time (seconds since 1/1/70)	/t, /n, /n32, /xs, /s
IP address	/n, /n32, /xs, /s

As you write the INSERT statement for your SQL accounting header file (`.acc`), we recommend the following syntax checklist:

- The column names and their corresponding attributes in the VALUES section are order-dependent. In the example above, the %TransactionTime value would be inserted into the Time column, the %NASAddress value would be inserted into the NASAddress column, and so forth. The ordering of these settings is critical to proper RADIUS accounting data insertion, since each column in the SQL table may be a unique data type (`varchar`, `int`, and so forth).

- The use of left and right parentheses '(' ')', the backslash '\', the forward slash '/' and even blank spaces are all extremely important and must be exact. You can add as many columns and attributes as you want for your RADIUS accounting needs; however, be sure to model your `INSERT` statement syntax on the example above.
- An attribute listed incorrectly in the VALUES section, such as `@Acct_Session-Id` rather than `@Acct-Session-Id`, causes the SQL statement to fail during a RADIUS accounting transaction. The attribute's syntax must match its corresponding attribute name in the `account.ini` file, which in turn matches the attribute's name in the appropriate dictionary file, which allows Steel-Belted Radius to process the attribute correctly when it is received from the RAS (the RADIUS client).
- An attribute listed in the `VALUES` section that is missing its prefix of '@' or '%' causes the SQL statement to fail during a RADIUS accounting transaction.
- If a carriage return is present within the `INSERT` statement without the backslash '\' to indicate the end of the line, the SQL statement fails during a RADIUS accounting transaction.
- Do not make the lines in the `.acc` file too long. There is a line length limit of 255 characters. Use the backslash '\' to indicate the end of the line before that limit is reached. If a line exceeds this limit, the SQL statement fails during a RADIUS accounting transaction.

Using Multiple SQL Statements

The most common use of accounting is to track user sessions. However, accounting requests are generated when the RAS starts up and shuts down; and, vendor-specific uses of accounting are used to track other RAS phenomena. Clearly, it might be advisable to log different types of accounting events to different tables.

The Acct-Status-Type attribute of an accounting request indicates the request type. You may, if you like, create multiple SQL statements, and map each Acct-Status-Type to one of these SQL statements. The different statements may update different tables in the database, but they all share the single database connection.

Overlapped Execution of SQL Statements

The SQL accounting module is multi-threaded. SQL accounting can be configured with a maximum number of simultaneous executions of any SQL statement, using the `MaxConcurrent` entry in the `.acc` file's [Settings] section.

If `MaxConcurrent` is set to 1, SQL execution occurs serially, and the SQL execution for each accounting request must complete before execution for the next request may begin.

By increasing `MaxConcurrent`, it may be possible to increase throughput by overlapping operations, especially if the database server is remote and a large part of the time to complete a statement execution is taken up by network latency. If the database server is local, the point of diminishing returns may be reached at a small value of `MaxConcurrent`, possibly even at 1 or 2. You can find the optimum value for your system by experimentation.

Warning: *A setting of* `MaxConcurrent = 1` *should be sufficient for all but the most demanding environments. Increase this value only slowly and conservatively.*

`MaxConcurrent` determines the maximum overlap for executing any single SQL statement. Multiple SQL statements for different request types are not interdependent, and executions of one statement do not affect executions of a different statement.

You might expect that databases that are licensed by number of connections would debit a single connection regardless of how many SQL statements are active. This is not necessarily the case; some databases count each open compiled SQL statement against the licensed number of connections. The database license may also have an influence on the optimum setting for `MaxConcurrent`.

SQL Accounting Return Values

SQL accounting statements can return information in RADIUS attributes in an accounting response. This is useful only if you are using a client that expects and supports attributes embedded in a RADIUS accounting response message.

Stored procedures can also return output parameters. The way in which these stored procedures are called depends on your operating system:

- To call an Oracle stored procedure in a Solaris/Linux environment:

  ```
  BEGIN storedProcedure(parameters…); END;
  ```

- To call an Oracle stored procedure in a Windows environment:

  ```
  call(storedProcedure(parameters…))
  ```

Accounting Stored Procedure Example

A simple stored procedure can return a result set in the same way as a `Select` statement. For example, assume you have a table with the following fields: `username`, `password`, `Alias`, and `active`, where all fields have the datatype `varchar`. You want a stored procedure that will return a password and alias when the username and password received in the request match entries in the database, provided that `active` field has a value of yes'.

The following example executes a stored procedure to update an accounting table in Steel-Belted Radius.

1. Create an accounting table by executing the following command:

   ```
   create table accounting
   ( TransactionDate varchar(20), Username varchar(21),
   SessionID varchar(12), NASIPAddr varchar(15), NASPort
   varchar(5), UserIPAddr varchar(15), CallingNum varchar(12),
   CalledNum varchar(12),
   ```

```
type varchar(4), Sessiontime varchar(14), Disconnect
varchar(12) )
```

2 Create a `rsp_account` stored procedure that can be called by a `*.acc` file.

```
create procedure rsp_account

@transactiontime varchar(21),
@username varchar(21),
@AcctSessionID varchar(21),
@NASIPAddress varchar(21),
@NASPORTTYPE varchar(21),
@FRAMEDIPADDRESS varchar(21),
@callingstationid varchar(21),
@calledstationid varchar(21),
@TYPE varchar(21),
@ACCTSESSIONTIME varchar(21),
@ACCTTERMINATIONCAUSE varchar(21)
AS
INSERT INTO Accounting (TransactionDate, username,
SessionID, NASIPAddr, NASPort, UserIPAddr, CallingNum,
CalledNum, type, Sessiontime, Disconnect)
VALUES (@transactiontime, @username, @AcctSessionID,
@NASIPAddress, @NASPORTTYPE, @FRAMEDIPADDRESS,
@callingstationid, @calledstationid, @TYPE,
@ACCTSESSIONTIME, @ACCTTERMINATIONCAUSE)
```

3 Create the `mysqlacct.acc` file to call the `rsp_account` stored procedure.

Note that the `mysqlacct.acc` file uses an `SQL=EXECUTE` *`procedure_name value1,...valueN`* statement instead of an `SQL=INSERT into` *`table`* `(`*`column1, ...columnN`*`) Values (`*`value1,...valueN`*`)`, since the stored procedure will do the INSERT action. You would configure the CONNECT statement to reflect your operating environment.

```
[Bootstrap]
LibraryName=sqlacct.dll
Enable=1
InitializationString=

[Settings]
Connect=DSN=<dsn_name_here>;UID=<username_for_dB>;PWD=<pass
word_for_dB>
ConnectTimeout=25
WaitReconnect=2
MaxWaitReconnect=360
ParameterMarker=?
loglevel=2

[Type]
1=User
2=User
3=User

[Type/User]
SQL=Execute rsp_account   %transactiontime/20s, \
```

```
@user-name/21s, \
@Acct-Session-ID/12s, \
@NAS-IP-Address/15s, \
@NAS-PORT-TYPE/5s, \
@FRAMED-IP-ADDRESS/15s, \
@calling-station-id/12s, \
@called-station-id/12s, \
%TYPE/4s, \
@ACCT-SESSION-TIME/14s, \
@ACCT-TERMINATION-CAUSE/12s
ConcurrentTimeout=30
MaxConcurrent=2
```

Chapter 16
Configuring LDAP Authentication

This chapter presents an overview of LDAP authentication and describes how to configure LDAP authentication in Steel-Belted Radius.

About LDAP Authentication

Steel-Belted Radius can authenticate against records stored in an external LDAP database. Any attribute(s), such as username and password, can be used to query the database.

External database authentication is typically used when an organization has a large amount of user information stored in an LDAP database, and wants to authenticate these users using RADIUS. Authentication against an existing LDAP database extends authentication services to user accounts without requiring an administrator to enter user information into the Steel-Belted Radius database.

Steel-Belted Radius offers LDAP authentication as a plug-in software module. Key features of the LDAP plug-in include the following:

- LDAP Version 3 is supported.
- SSL is supported if you have Netscape certificates.
- You can authenticate via LDAP `Bind` or via a password returned from an LDAP Search request (`BindName`).
- A single Search request or a sequence of Search requests can be specified.
- Bind, Base, and Search strings can include variables.
- New Bind parameters can be specified during a sequence of searches.
- Other authentication credentials can be specified in a string that can include variable values.
- Variables may be set from the RADIUS request packet and from LDAP Search results.
- Variables may be used to specify RADIUS response attributes and other response information.

- The RADIUS response can include RADIUS attributes found in the LDAP database, or it can reference a Steel-Belted Radius profile or user entry.
- Several features similar to SQL authentication are supported, such as round-robin load balancing, the "server of last resort," activation targets, and so on.
- The variable table allows both attributes and `%Profile` in the [Response] section.

LDAP Variable Table

The LDAP Variable Table lets you translate a RADIUS request into an LDAP lookup. At the beginning of each LDAP authentication request, Steel-Belted Radius creates a Variable Table. Attributes and other information from the RADIUS request are entered in the Variable Table for use in LDAP Bind, Base, and Search strings. When attributes are returned by LDAP requests, they too are entered in the Variable Table. Finally, selected information from the Variable Table is returned to the RADIUS client in the RADIUS response packet.

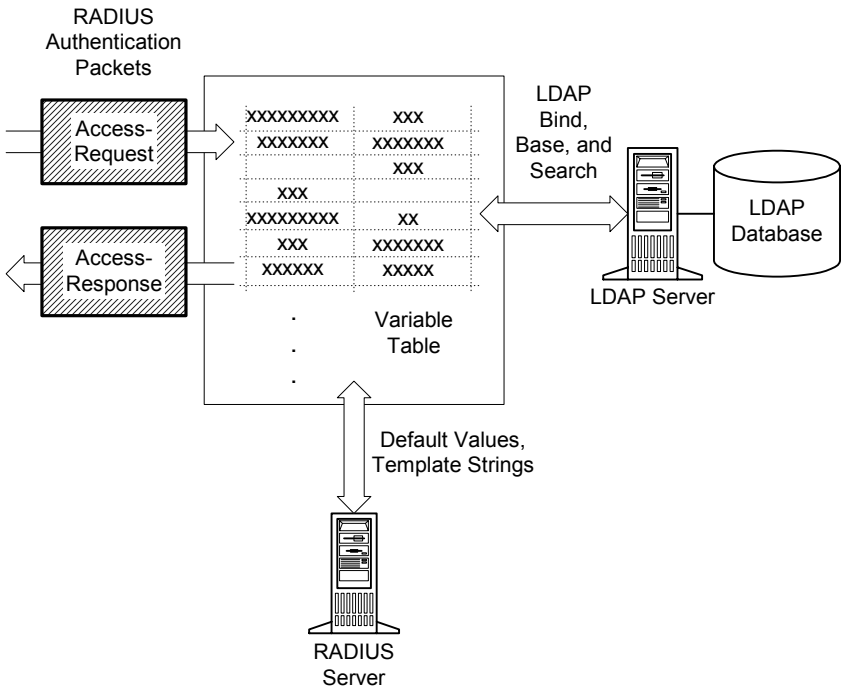


Figure 99 Role of the Variable Table in LDAP Authentication

Types of LDAP Authentication

To design an LDAP authentication method, consider how you want to validate the username and password.

The LDAP plug-in offers two techniques for validating the username and password. Each header file that you write to control LDAP authentication must employ `Bind` or `BindName`. The differences between the two techniques have to do with how

Steel-Belted Radius connects to the LDAP server and whether the username/password validation is performed by the LDAP server or by Steel-Belted Radius.

BindName Authentication

When you use `BindName` authentication, your LDAP header file provides Steel-Belted Radius with the username and password of an account on the LDAP server. This must be an account that has privileges to access all of the information that you require to authenticate users. In the LDAP header file, you provide the username in the `BindName` parameter, and the password in the `BindPassword` parameter.

After you complete the LDAP header file, each time Steel-Belted Radius starts up, it executes a `Bind` request to the LDAP server using the `BindName` and BindPassword parameters as its credentials. If the LDAP server can validate these credentials, a connection is established between the two servers. This connection remains "up" all the time. It is disconnected only if the Steel-Belted Radius server or the LDAP server goes down, and it's re-established as soon as possible after the "down" server comes back up. The LDAP header file offers a number of connection and re-connection timeouts and other parameters that regulate this relationship.

Any time authentication via LDAP is required, Steel-Belted Radius consults the corresponding LDAP header file. When you use `BindName` authentication, this file must contain a Search command that maps the username from the Access-Request to a password attribute in the LDAP database. The Search may retrieve other LDAP attributes as well. When the Search returns its results, Steel-Belted Radius compares the value of the password returned from the LDAP database with the password from the incoming Access-Request. If the two values are the same, the password is considered validated.

When the connection to the LDAP server is established using `BindName`, multiple authentications can be performed at the same time over the same connection.

Bind Authentication

When you use `Bind` authentication, Steel-Belted Radius authenticates connection requests by attempting to `Bind` to the LDAP server using the username and password from the incoming Access-Request or from a configured username and password. If this `Bind` request succeeds, the password is validated. This is essentially "pass-through" authentication; Steel-Belted Radius presents an LDAP user's credentials to the LDAP server and asks to have them validated.

In the simplest case, a single connection is established for each Access-Request and is kept open only long enough for the LDAP server to validate the password and respond to any Search requests. Then Steel-Belted Radius closes the connection and completes any processing that remains to generate an Access-Response.

A more sophisticated search technique can take advantage of flexible `Bind`, which allows you to allocate a sequence of connections for each Access-Request. Each in turn is kept open only long enough for the server to process each search criterion. Then Steel-Belted Radius closes the connection and completes any processing that remains to generate an Access-Response.

Attributes and LDAP Authentication

Tip: See "User Attribute Lists" on page 24.

A username and password may be all the information that you require to authenticate users. However, the LDAP plug-in offers a number of techniques for working with checklist and/or return list attributes, should you need them.

Configuring LDAP Authentication

To configure an LDAP authentication method, you must edit the header file that controls the LDAP authentication sequence.

Table 33 summarizes the process of configuring an LDAP authentication method for Steel-Belted Radius. It lists the sections that you must edit in the header file to accomplish each step. No step may be omitted. You must at least consider the entries that you want to put in each section of the header file, even if you decide to leave most of that section blank.

Table 33. *LDAP Authentication Header File Topics*

Step	LDAP Configuration Task	.aut File Sections
1	Decide how you want Steel-Belted Radius to validate RADIUS access requests. Two major areas of choice are described above: (1) `Bind` or `BindName`; and (2) Profile, Alias, or attribute list.	All sections
2	Determine which incoming RADIUS attributes are required to perform the LDAP search.	[Response]
3	Determine which LDAP attributes support are required to perform the LDAP search.	[Attribute/*name*]
4	Design Search template(s) that can find the necessary data in your LDAP database schema.	[Search/*name*]
5	Extract the data from the incoming RADIUS packet that Steel-Belted Radius will use to perform the LDAP `Bind` and `Search` requests.	[Request]
6	Select defaults that you want Steel-Belted Radius to use when corresponding values are not provided.	[Defaults]
7	Enable connections between the Steel-Belted Radius server and LDAP server(s).	[Server] [Server/*name*] [Settings] [Failure]
8	Enable the LDAP plug-in and name the authentication method.	[Bootstrap]

The order in which you should edit header file sections is the reverse of the order in which Steel-Belted Radius processes them. The processing sequence is described in "LDAP Authentication Sequence" on page 212.

Supporting Secure Sockets Layer

You must follow the instructions below for SSL to be supported by the LDAP plug-in:

1. Set `SSL` in the [Settings] (or [Server/*name*]) section to `1`.
2. Set the `Certificates` field in the [Settings] (or [Server/*name*]) section to the path where the `cert7.db` and `key3.db` files are located. The name of the files (that is `cert7.db` or `key3.db`) should not be included.
3. Set the port in the [Server] section to the SSL port of the LDAP server.

Files

The following file establishes settings for LDAP authentication. For more information about this file, refer to the *Steel-Belted Radius Reference Guide*.

***Table 34.** LDAP Authentication Files*

File Name	Function
`ldapauth.aut`	Specifies settings for LDAP authentication in Steel-Belted Radius.

LDAP Database Schema

The most important factor in the success of your LDAP authentication methods is the design of your LDAP database schema. It's assumed that you already have a schema in place.

Often, you can use the LDAP plug-in *without* changing the LDAP database schema at all. In Figure 100, the user record already provides an LDAP attribute called Organization. If you intend to grant connection privileges according to the organization to which each user belongs, you can create profiles in the Steel-Belted Radius database whose names match the strings you are already using for the Organization attribute. You can then create an LDAP authentication header file that retrieves the value of the Organization attribute from the LDAP database and returns it to Steel-Belted Radius as the name of the profile to use.

***NOTE:** If you are using `BindName` authentication, you need to be able to identify which LDAP attribute contains the user's password. In the schema below, this attribute is called User-Secret.*

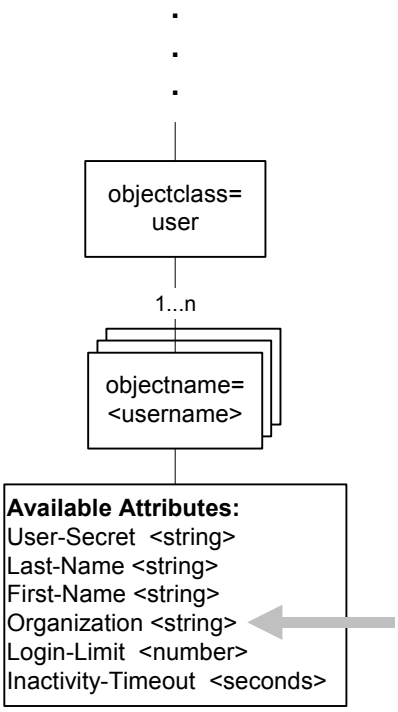


Figure 100 *Capitalizing on an Existing Schema for LDAP Authentication*

In some situations, such as when the authentication strategy you have chosen requires data that is not currently in the schema, you might need to modify the schema.

The name of a Steel-Belted Radius profile is a typical example. Consider the example above. If you want to assign connection privileges to users in some way other than by Organization, and no other LDAP attribute seems appropriate, you can add an LDAP attribute that names a profile. In Figure 101, this attribute is called RADIUS-Profile. This attribute contains a string value that can be set to the name of a profile defined in the Steel-Belted Radius database.

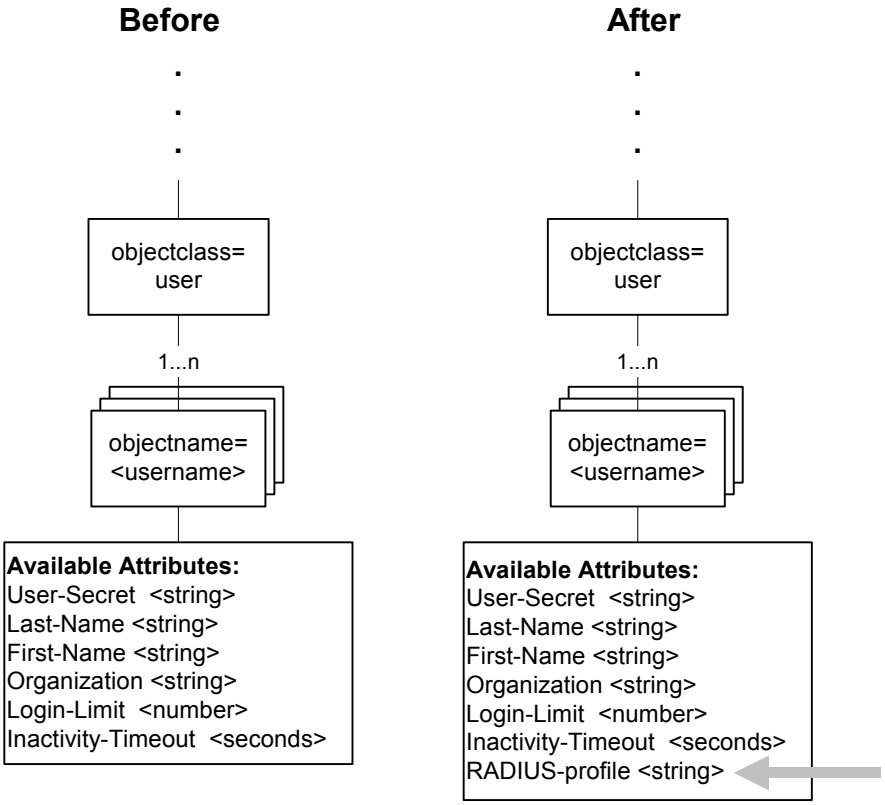


Figure 101 *Modifying a Schema to Enhance LDAP Authentication*

NOTE: *LDAP concepts and the details of your own LDAP schema are entirely outside the scope of this chapter. The instructions in this chapter are provided to help you to make the LDAP plug-in work with an existing LDAP database or databases to provide a Steel-Belted Radius authentication method. The instructions assume that you already have a working knowledge of LDAP syntax and conventions. For details about LDAP, please refer to your usual LDAP information source.*

LDAP Authentication and Password Format

Steel-Belted Radius supports authentication of users whose records reside in an LDAP table in which password values are stored in one of the following formats: clear text, UNIXcrypt, Secured Hash Algorithm (SHA1+Base64 hash), MD4 hash, or enc-md5 reversibly-encoded password.

Hashed Passwords

Encoded values include a prefix that indicates how the password has been processed. The prefix is in clear text between curly braces '**{**' '**}**' and is immediately followed by a hash value computed from the password. If no prefix is present in the value retrieved, the entire password is assumed to be in clear text format. In summary:

- `PasswordText` indicates clear text format (no encryption)
- `{crypt}HashHash` indicates UNIXcrypt format

- `{SHA}`*`HashHashHash`* indicates SHA1+Base64 hash
- `{SSHA}`*`HashHashHashSalt`* indicates salted SHA1+Base64 hash
- `{md4}`*`HashHash`* indicates MD4 hash of the Unicode form of password
- `{enc-md5}`*`EncryptedEncrypted`* indicates a reversibly encrypted password. (Note that, although Steel-Belted Radius reads passwords encoded in this format, you must purchase the Software Developer's Kit to convert clear-text passwords to this format.)

UNIXcrypt is the standard hash algorithm used for the `/etc/passwd` file on Solaris systems. This may be necessary if, for example, the standard user database on a Solaris machine (the `/etc/passwd` file) is migrated to a SQL database, so that the values in the `Password` column of the SQL table are processed with UNIXcrypt.

You can configure Steel-Belted Radius to expect that the values retrieved from a table have been run through UNIXcrypt by adding the following entry into the [Settings] section of the LDAP authentication header file:

```
PasswordFormat=3
```

Automatic Parsing

If `PasswordFormat` is set to 0, Steel-Belted Radius attempts to determine the password format automatically by parsing it. This is the recommended setting. Automatic parsing expects the password to be stored in one of the formats above.

This technique is useful if clear text passwords are available to Steel-Belted Radius (that is, if PAP is used). If you set `PasswordFormat` to `0`, the stored password can be returned to Steel-Belted Radius still encrypted, and the comparison with the password received from the RADIUS client can be done on the Steel-Belted Radius side.

NOTE: *The setting for automatic password parsing in previous versions of Steel-Belted Radius (auto) has been deprecated.*

LDAP Authentication Sequence

The sequence of an LDAP authentication transaction is controlled by the LDAP authentication header file as follows:

1. The Variable Table is initialized to default values as specified in the [Defaults] section. All variables that are not listed in the [Defaults] section are initialized to null values.
2. The values of RADIUS attributes in the Access-Request are copied to the Variable Table, as specified in the [Request] section.
3. If a `Bind` entry was specified in the [Settings] section, authentication via LDAP `Bind` is now performed. The `Bind` entry is used as a template to construct a bind string, using replacement values from the Variable Table. An LDAP `Bind` is then performed to authenticate the user.

4 An LDAP Search request is performed for each [Search/*name*] section specified. You may specify zero or more separate Search requests.

For each Search request, LDAP Base and Filter strings are constructed from templates, using replacement values from the Variable Table. These Base and Filter strings are then transmitted to the LDAP server in a Search request.

Each attribute/value pair returned by the LDAP Search is used to set the value of the corresponding entry in the Variable Table. Also, the DN returned by the search may be used to set a variable.

5 If a %Password entry appears in the [Response] section, authentication is now performed. The password entered by the user is validated against the value that appears in the %Password variable, and the user is rejected if the passwords don't match.

6 If a %Profile entry appears in the [Response] section, the value of the %Profile variable is used to look up a Profile entry in the Steel-Belted Radius database. The checklist and return list attributes in that Profile are used to validate the request and return an appropriate response.

7 If a %Alias entry appears in the [Response] section, the value of the %Alias variable is used to look up a Native User entry in the Steel-Belted Radius database. The current transaction is treated as if it came from the "alias" user; that is, the checklist and return list attributes of the alias user are used to validate the request and return an appropriate response.

8 If neither a %Profile nor a %Alias entry appears in the [Response] section, then RADIUS attributes for the response packet are created from the Variable Table, based on attribute entries in the [Response] section.

LDAP Authentication Examples

This topic provides examples of LDAP authentication header file syntax. The examples illustrate how you might:

- Authenticate passwords (`Bind` or `BindName`).
- Specify checklist and return list attributes (list the attributes or name a profile entry in the Steel-Belted Radius database).

Bind Authentication with Default Profile

The following example is a simple LDAP authentication header file. Every user is authenticated using a `Bind` request to the LDAP database. The same Steel-Belted Radius attribute profile is applied to every Access-Request.

```
[Settings]
MaxConcurrent=1
Timeout=20
ConnectTimeout=25
QueryTimeout=10
WaitReconnect=2
```

```
MaxWaitReconnect=360
Bind=uid=<User-Name>, ou=Special Users, o=bigco.com
LogLevel = 2
UpperCaseName = 0
PasswordCase=original
SSL = 0

[Server]
s1=

[Server/s1]
Host=199.185.162.147
Port = 389

[Defaults]
TheUserProfile = Sample

[Request]
%User-Name = User-Name

[Response]
%Profile = TheUserProfile

[Search/DoLdapSearch]
Base = ou=Special Users, o=bigco.com
Scope = 2
Filter = uid=<dialup>
Attributes = AttrList
Timeout = 20
%DN = dn

[Attributes/AttrList]
```

If the [Response] section was empty, Steel-Belted Radius would pass the `Bind` results (accept or reject) directly to its client and no additional RADIUS attributes would be returned in the Access-Response.

BindName Authentication with Callback Number Returned

In the following example, requests are authenticated using Search. `BindName` and BindPassword values are supplied to permit a connection to the LDAP database. return list attributes for authentication are listed in the [Response] section. In this example, the RAS device needs a callback number to complete the connection. The value of the incoming DNIS attribute Calling-Station-ID is used to ensure that the callback number is the number from which the user's request originated.

***NOTE:** This example is incomplete; it omits the [Bootstrap] and [Settings] sections to save space.*

```
[Server]
s1=

[Server/s1]
Host = 67.186.4.3
```

```
Port = 389
BindName=uid=admin, ou=Administrators,
ou=TopologyManagement, o=NetscapeRoot
BindPassword=ourlittlesecret
Search = DoLdapSearch

[Defaults]
SendThis = DidLDAPAuthSearch

[Request]
%UserName = dialup
Calling-Station-ID = thenumbertocall

[Search/DoLdapSearch]
Base = ou=Special Users, o=bigco.com
Scope = 2
Filter = uid=<dialup>
Attributes = AttrList
Timeout = 20
%DN = dn

[Attributes/AttrList]
dialuppassword

[Response]
%Password = dialuppassword
Reply-Message = SendThis
Ascend-Callback-No = thenumbertocall
```

LDAP Bind with Profile Based on RAS Device

In the following example, requests are authenticated using `Bind`. Checklist and return list attributes for authentication are provided by referencing a profile entry in the Steel-Belted Radius database. The profile to be used depends on the specific RAS device from which the user's request originates. Steel-Belted Radius retrieves the profile name by the LDAP database for an IP address that matches the address of the requesting RAS. If this search fails, a profile called `limited` is used. If a profile name is successfully retrieved from the LDAP database, but no profile by that name can be found in the Steel-Belted Radius database, authentication fails due to "lack of resources" and the user is rejected.

***NOTE:** This example is incomplete; it omits the [Bootstrap] section and many [Settings] entries to save space.*

```
[Settings]
Bind=uid=<loginID>, ou=Special Users, o=bigco.com
Search = DoLdapSearch

[Server]
s1=

[Server/s1]
Host = 67.186.4.3
```

```
Port = 389

[Request]
%UserName = loginID
%NASAddress = deviceIP

[Defaults]
%Profile = limited

[Search/DoLdapSearch]
Base = ou=CommServers, o=bigco.com
Scope = 1
Filter = ipaddr=<deviceIP>
Attributes = AttrList
Timeout = 20
%DN = dn

[Attributes/AttrList]
profile

[Response]
%Profile = profile
```

Chapter 17
Displaying Statistics

The Statistics panel lets you display summary statistics for authentication, accounting, and proxy forwarding transactions. You can also use the Statistics panel to see how long Steel-Belted Radius has been running and to display a list of the users currently connected through a RAS or tunnel.

Displaying Authentication Statistics

Authentication statistics (Figure 102) summarize the number of authentication acceptances and rejections, with summary totals for each type of rejection or retry.

To display authentication statistics, open the Statistics panel, click the **System** tab, pull down the **View** list, and choose **Authentication**.

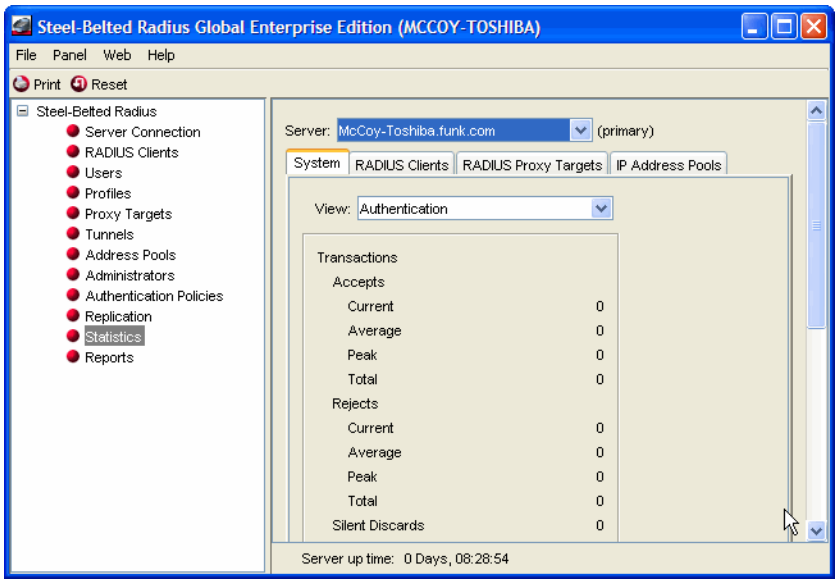


***Figure 102** Statistics Panel: Authentication Statistics*

Table 35 explains the authentication statistics fields and describes possible causes for authentication rejections.

Table 35. *Authentication Statistics*

Authentication Statistic	Meaning
Transactions	
Accepts	The current, average, and peak number of RADIUS transactions that resulted in an accept response.
Rejects	The current, average, and peak number of RADIUS transactions that resulted in a reject response. These are broken out in Reject Details below.
Silent Discards	The number of requests in which the client could not be identified. This might occur if a RADIUS client entry cannot be found for a device with the name and/or IP address of a device requesting authentication services.
Total Transactions	The sum of the accept, reject, and silent discard totals.
Reject Details	
Dropped Packet	The number of RADIUS authentication packets dropped by Steel-Belted Radius because the server was flooded with more packets than it could handle.
Invalid Request	The number of invalid RADIUS requests made. *A device is sending incorrectly formed packets to Steel-Belted Radius; either there is a configuration error or the device does not conform to the RADIUS standard*.
Failed Authentication	The number of failed authentication requests, where the failure is due to invalid username or password. *If all transactions are failing authentication, the problem might be that the shared secret entered into Steel-Belted Radius does not match the shared secret entered on the client device*.
Failed on Checklist	The number of requests that were authenticated but failed to meet the checklist requirements.
Insufficient Resources	The number of rejects due to a server resource problem.
Proxy Failure	The number of rejects that had to be issued because Proxy forwarding to another RADIUS server failed.
Rejected by Proxy	The number of rejects due to receiving a reject response from a proxy RADIUS target server.
Retries Received	
Transactions Retried	The number of requests for which one or more duplicates was received.
Total Retry Packets	The number of duplicate packets received.
Challenges	The number of challenges received.

Displaying Accounting Statistics

Accounting statistics provide information such as the number of transaction starts and stops and the reasons for rejecting attempted transactions. The start and stop numbers rarely match, as many transactions can be in progress at any given time.

To display authentication statistics, open the Statistics panel, click the **System** tab, pull down the **View** list, and choose **Accounting**.

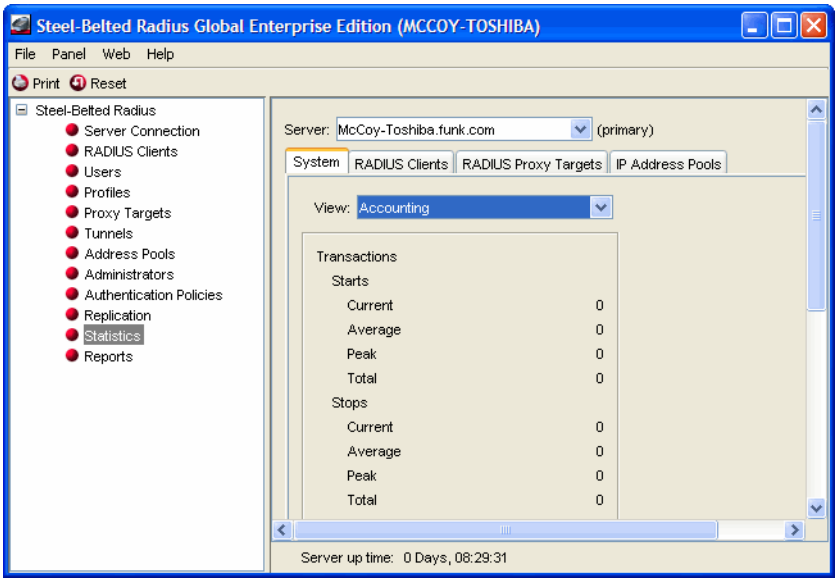


***Figure 103** Statistics Panel: Accounting Statistics*

Figure 36 describes the accounting statistics and describes possible causes for accounting errors.

***Table 36.** Accounting Statistics*

Statistic	Meaning
Transactions	
Starts	The current, average, and peak number of transactions in which a dial-in connection was started following a successful authentication.
Stops	The current, average, and peak number of transactions in which a dial-in connection was terminated.
Ons	The number of Accounting-On messages received, indicating that a RADIUS client has restarted.
Offs	The number of Accounting-Off messages received, indicating that a RADIUS client has shut down.
Total	The sum of the start, stop, on and off totals.
Failure Details	
Dropped Packet	The number of RADIUS accounting packets dropped by Steel-Belted Radius because the server was flooded with more packets than it could handle.
Invalid Request	The number of invalid RADIUS requests made. *A device is sending incorrectly formed packets to Steel-Belted Radius; either there is a configuration error or the device does not conform to the RADIUS standard.*

Table 36. *Accounting Statistics (Continued)*

Statistic	Meaning
Invalid Client	The number of requests in which the RADIUS client could not be identified. *A device might be configured to use Steel-Belted Radius but no RADIUS client entry has been created with the name and/or IP address of the client; or the RADIUS client entry might be configured with an incorrect name or IP address; or some rogue device is attempting to compromise RADIUS security.*
Invalid Shared Secret	The number of packets for which an incorrect digital signature was received. *The shared secret does not match between Steel-Belted Radius and the client device; or some rogue device is attempting to compromise RADIUS security.*
Insufficient Resources	The number of rejects due to a server resource problem.
Proxy Failure	The number of times that proxy RADIUS forwarding failed.
Retries Received	
Transactions Retried	The number of requests for which one or more duplicates was received.
Total Retry Packets	The number of duplicate packets received.
Interim Requests	The number of interim accounting packets received.

Displaying Proxied Request Statistics

Proxied request statistics provide information such as the number of proxy authentication or accounting requests and the reasons for any transaction failures that occur.

To display proxied request statistics, open the Statistics panel, click the **System** tab, pull down the **View** list, and choose **Proxied Requests**.

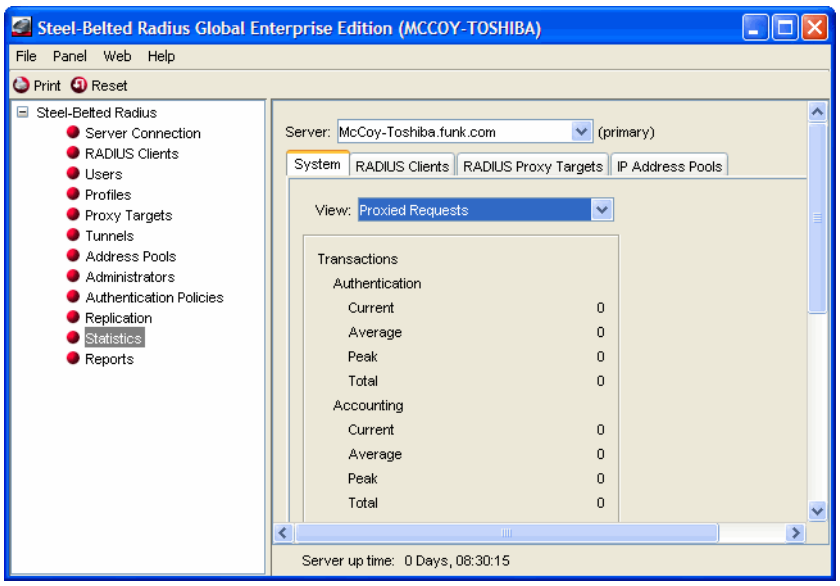


***Figure 104** Statistics Panel: Proxied Request Statistics*

Table 37 describes the proxy request statistics, with possible interpretations in italics.

***Table 37.** Proxy Statistics*

Proxy Statistic	Meaning
Transactions	
Authentication	The number of authentication transactions between the proxy and target RADIUS servers.
Accounting	The number of accounting transactions between the proxy and target RADIUS servers.
Total Transactions	The sum of the authentication and accounting transaction totals.
Failure Details	
Timed Out	The number of RADIUS transactions that timed out. This means that after all retry attempts were made, the transaction still timed out.
Invalid Response	The number of invalid RADIUS responses received. *A target is sending incorrectly formed packets to Steel-Belted Radius; there is a configuration error, the target RADIUS server does not conform to the RADIUS standard, or Steel-Belted Radius did not receive a proxy state echo in the received packet.*
Invalid Shared Secret	The number of packets for which an incorrect digital signature was received. *The shared secret does not match between Steel-Belted Radius and the target; or some unauthorized rogue device is attempting to compromise RADIUS security.*
Insufficient Resources	The number of rejects due to a server resource problem.
Retries Sent	
Transactions Retried	The number of requests for which one or more retried transmissions was performed.
Total Retry Packets	The number of duplicate packets received.

Displaying RADIUS Client Statistics

RADIUS client statistics provide information about the number of authentication and accounting requests by client.

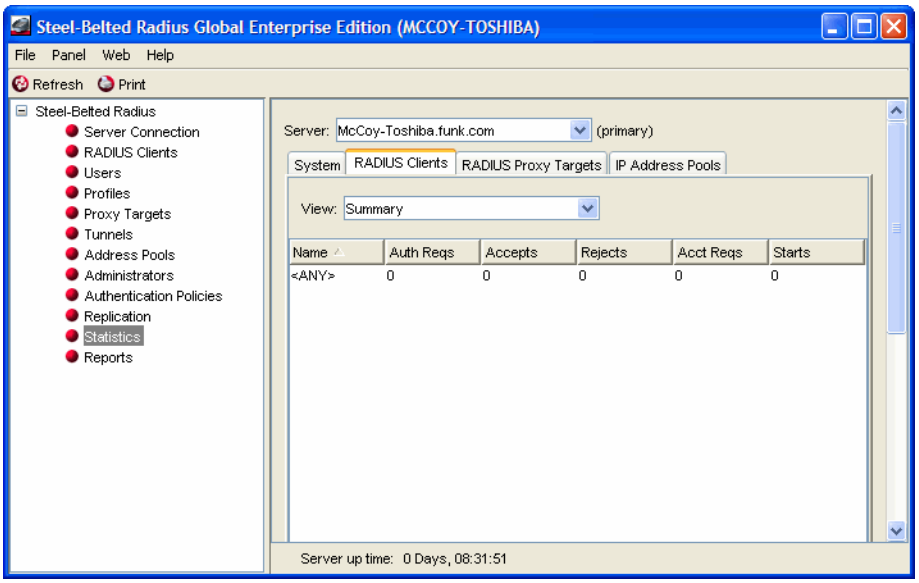


***Figure 105** Statistics Panel: RADIUS Clients Tab*

To display statistics for RADIUS clients:

1. Open the Statistics panel and click the **RADIUS Clients** tab.
2. Use the **View** list to display the type of statistics you want to display.
   - **Summary** – Displays the number of authentication requests, Access-Accepts, and Accept-Reject messages and the total number of accounting requests, starts, and stops for each RADIUS client.
   - **Authentication Request Details** – Displays the number of duplicate messages, challenges, messages containing invalid authentication information, bad authentication requests, bad types, and dropped requests for each RADIUS client.
   - **Accounting Request Types** – Displays the number of accounting start messages, accounting stop messages, interim messages, Accounting-On messages, Accounting-Off messages, and acknowledgement messages sent for each RADIUS client.
   - **Accounting Request Diagnostics** – Displays the number of duplicate messages, messages with invalid secrets, malformed messages, messages with incorrect types, ignored messages, and dropped requests for each RADIUS client.
3. Optionally, sort the messages by clicking a column header.

Displaying RADIUS Proxy Targets Statistics

RADIUS proxy target statistics provide information about the number of authentication and accounting transactions associated with each proxy target.

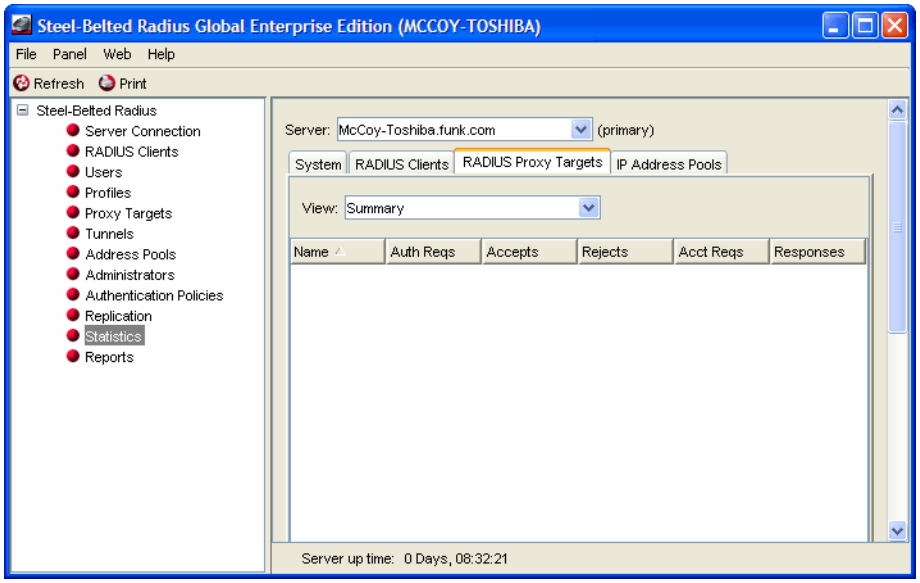


***Figure 106** Statistics Tab: RADIUS Proxy Targets Tab*

To display statistics for RADIUS proxy targets:

1. Open the Statistics panel and click the **RADIUS Proxy Targets** tab.
2. Use the **View** list to display the type of statistics you want to display.
   - **Summary** – Displays the number of authentication requests, accepts and reject messages, and the number of accounting requests and responses for each RADIUS proxy target.
   - **Authentication Request Details** – Displays the number of outstanding messages, retransmitted messages, and challenges, along with the most recent response time for the proxy target.
   - **Authentication Request Diagnostics** – Displays the number of timeouts, invalid secrets, incorrect requests, requests with invalid types, and dropped messages for each proxy target.
   - **Accounting Request Types** – Displays the number of outstanding messages and retransmitted messages, along with the most recent response time for the proxy target.
   - **Accounting Request Diagnostics** – Displays the number of timeouts, invalid secrets, incorrect requests, requests with invalid types, and dropped messages for each proxy target.
3. Optionally, sort the messages by clicking a column header.

Displaying IP Address Pool Statistics

IP address pool statistics provide a summary of the number of addresses allocated from each IPv4 address pool and how many addresses remain available.

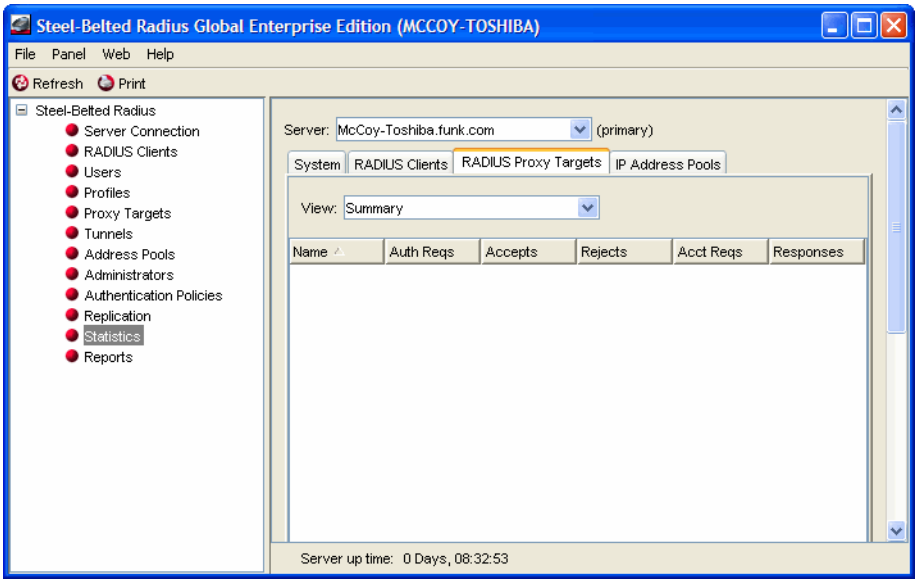


Figure 107 *Statistics Panel: IP Address Pools Tab*

Chapter 18
Logging and Reporting

This chapter describes how to set up and use logging and reporting functions in Steel-Belted Radius.

Logging Files

The following files establish settings for logging and reporting. For more information about these files, refer to the *Steel-Belted Radius Reference Guide*.

Table 38. *Logging and Reporting Files*

File Name	Function
`account.ini`	Controls how RADIUS accounting attributes are logged.
`authlog.ini`	Controls how RADIUS authentication requests are logged by Steel-Belted Radius.
`authReport.ini`	Controls what authentication logs Steel-Belted Radius generates.
`authReportAccept.ini`	Controls options for the acceptance authentication log file.
`authReportBadSharedSecret.ini`	Controls options for the invalid shared secret authentication log file.
`authReportReject.ini`	Controls options for the rejection authentication log file.
`authReportUnknownClient.ini`	Controls options for the unknown client authentication log file.
`events.ini`	Controls dilutions and thresholds for Steel-Belted Radius events used to communicate failures, warnings, and other information.
`radius.ini`	Controls (among other things) the types of messages Steel-Belted Radius records in the server log file and the location of the log directory.

Displaying the Current Sessions List

Steel-Belted Radius tracks the status of the user connections that it authenticates. To obtain a real-time snapshot of currently active connections, display the Current Sessions list. Note that, because the Current Sessions list is based on RADIUS accounting data, the list is accurate only if all of your RAS devices are configured to support RADIUS accounting.

Each server has its own Current Sessions display. Therefore, when you view this display, it typically reflects only the activity on the Steel-Belted Radius server to which you are connected. The Current Users display on a specific server reflects the activity across your entire RADIUS configuration only if (1) all clients in your configuration support RADIUS accounting, and (2) all clients are configured to send accounting messages to the server you are viewing.

***NOTE:** Steel-Belted Radius maintains the Current User list on disk. The information is preserved if you unload and reload the server.*

To display the Current Sessions list (Figure 108), open the Reports panel and click the **Current Sessions** tab.

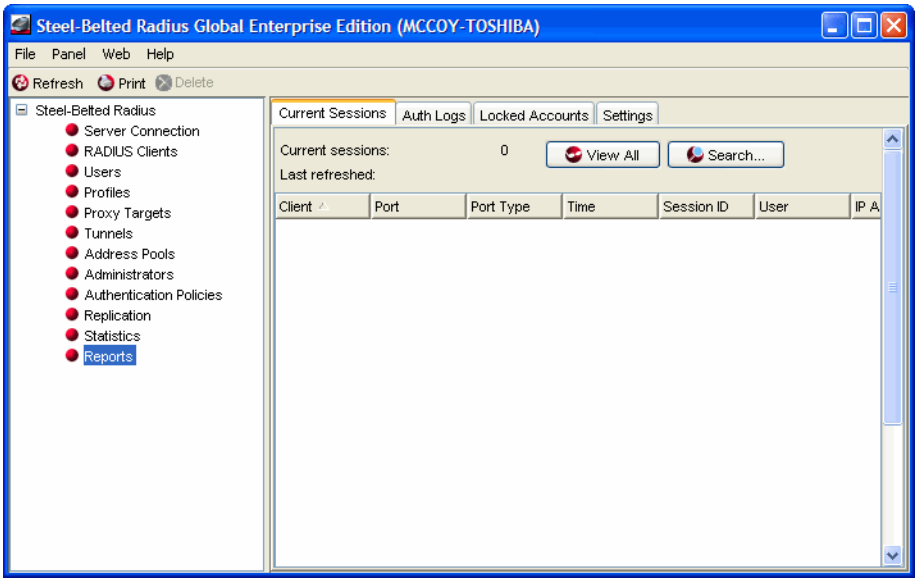


***Figure 108** Reports Panel: Current Sessions List*

Table 39 describes the fields for each active session in the Current Sessions list.

***Table 39.** Sessions List Fields*

Field	Meaning
Client	The identifier for the RADIUS client, which is typically the name or IP address of the device.
Port	The UDP port number on the RADIUS client that has been assigned to the connection. To determine slot number of the physical port on the RADIUS client, consult the device documentation.

Table 39. Sessions List Fields

Field	Meaning
Port Type	Describes how the port is used or configured.
Time	Identifies the date and time at which the connection was opened.
Session ID	Identifies the session key, which is a number generated by the RADIUS client.
User Name	Displays the name of the authenticated user. • If the user is native, the field shows only the username, in the form username. • If the user is non-native, the field shows the remote system name as well as the username, in the form `\\systemname\username`. • If the user is associated with a specific tunnel, the field shows the tunnel name as well as the username, in the form `\\tunnelname\username`.
IP Address	Identifies the IP address that was assigned to the user from an IP address pool. This field will be blank if a static IP address was assigned.

***NOTE:** For tunnel connections, if Steel-Belted Radius was used to authenticate both the user and the tunnel, then two entries are displayed in the Current Sessions window: one entry for the authenticated user, and one for the authenticated tunnel.*

Searching the Current Sessions List

You can search the Current Sessions list to display only those sessions that match user name or client name criteria.

To search the Current Sessions list:

1. Open the Reports panel and click the **Current Sessions** tab.
2. Click the **Search** button.

   The Search Current Sessions window (Figure 109) opens.

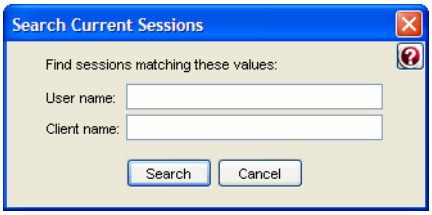


***Figure 109** Search Current Sessions Window*

3. Enter the search criteria you want to use.
   - ▷ To search for a specific username, enter it in the **User name** field.
   - ▷ To search for a specific RADIUS client, enter a client name in the **Client name** field.
4. Click **Search**.

Deleting Entries from the Sessions List

Normally, the system maintains the information in the Sessions list based on accounting information received from the RADIUS client. However, a user who has logged off may still be identified as active in the Current Sessions list if communications between the RADIUS client and Steel-Belted Radius fail or if either the RADIUS client or Steel-Belted Radius is taken down for a period of time.

In most cases, Steel-Belted Radius can correct such anomalies itself. For example, if a new user dials in to the same port on the same RADIUS client, Steel-Belted Radius infers that the previous user must have disconnected and removes the entry.

You can manually correct the Sessions list by highlighting any entry and clicking **Delete**. This removes the user from the list and decrements the user's connection count (if it is being tracked) by one. Any pooled IP or IPX address assigned to the deleted user is returned to the appropriate pool.

Displaying the Authentication Log Files

The Auth Logs tab (Figure 110) on the Reports panel lets you enable and display the following authentication log files:

- Successful request authentication log file – The successful request authentication log file identifies the authentication requests that were approved by Steel-Belted Radius.
- Invalid shared secrets authentication log file – The invalid shared secrets authentication log file identifies the authentication requests that failed because a known RADIUS client supplied an incorrect shared secret. This condition is detectable only if the authentication request contains a Message-Authenticator attribute, which is required if credentials are of an EAP type but optional if credentials are PAP, CHAP, or MS-CHAP.
- Failed request authentication log file – The failed request authentication log file identifies the authentication requests that were rejected because the user supplied incorrect credentials.
- Unknown client request authentication log file – The unknown client request authentication log file identifies authentication requests received from unknown RADIUS clients.

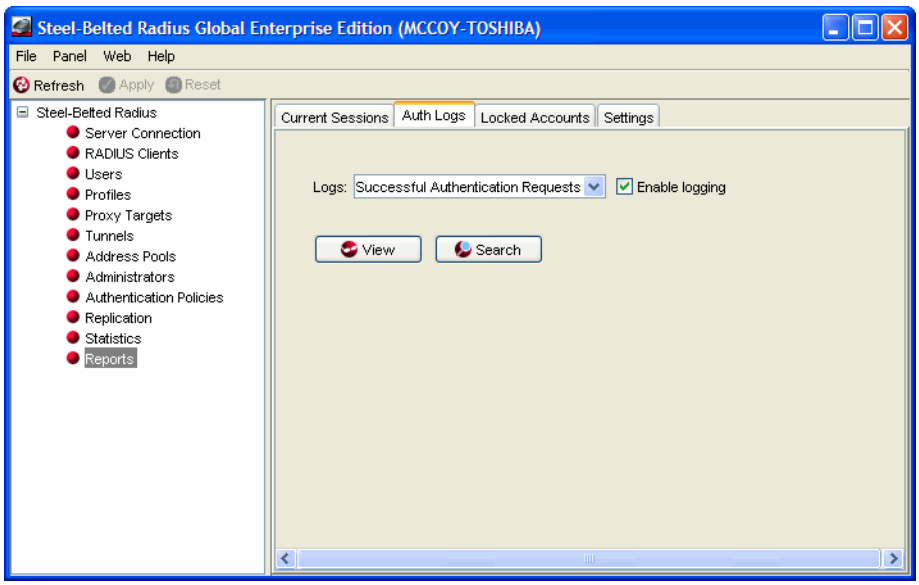


Figure 110 Reports Panel: Auth Logs Tab

Enabling and Disabling the Authentication Log Files

To enable an authentication log file:

1. Open the Reports panel and click the **Auth Logs** tab.
2. Use the **Logs** list to select the authentication log file you want to enable or disable.
3. Click the **Enable logging** checkbox to enable the specified authentication log.

   Unclick the **Enable logging** checkbox to disable the specified authentication log.

Viewing the Authentication Log Files

To display an authentication log file:

1. Open the Reports panel and click the **Auth Logs** tab.
2. Use the **Logs** list to select the log file you want to display.
3. Click the **View** button.

   The Log List window (Figure 111) opens.

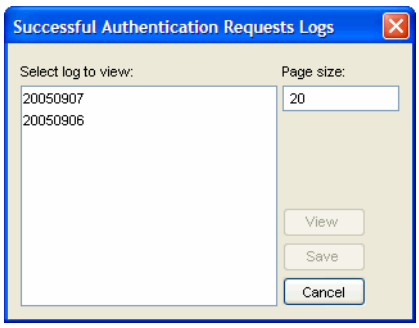


Figure 111 *Log List Window*

4 Select the log you want to display and click **View**.

By default, SBR Administrator displays the authentication log file 20 lines at a time. To change the number of lines displayed, enter a different number in the **Page size** field before you click **View**.

5 When the authentication log file window opens, click the **Up** and **Down** arrows to page through the log file.

To sort the authentication log file, click the appropriate column header.

To print the authentication log file, click the **Print** button.

To refresh the authentication log file display, click the **Refresh** button.

6 When you are finished, click **Close**.

Saving the Log Files

To save an authentication log file to a text file:

1 Open the Reports panel and click the **Auth Logs** tab.

2 Use the **Logs** list to select the authentication log file you want to save.

3 Click the **View** button.

The Log List window (Figure 111) opens.

4 Select the log you want to save and click **Save**.

The Save Log File As window (Figure 112) opens.

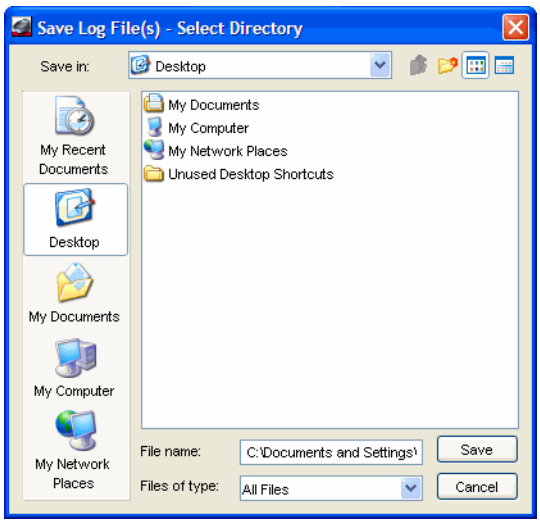


Figure 112 Save Log File As Window

5 Specify the name and destination for the log file and click **Save**.

Searching the Log Files

You can search the Steel-Belted Radius authentication log files to display messages within a specified time range, messages relating to a specific client, or messages relating to a specific user.

To search the authentication log files:

1 Open the Reports panel and click the **Auth Logs** tab.

2 Use the **Logs** list to select the type of authentication log file you want to search.

3 Click the **Search** button.

The Search Logs window (Figure 113) opens.

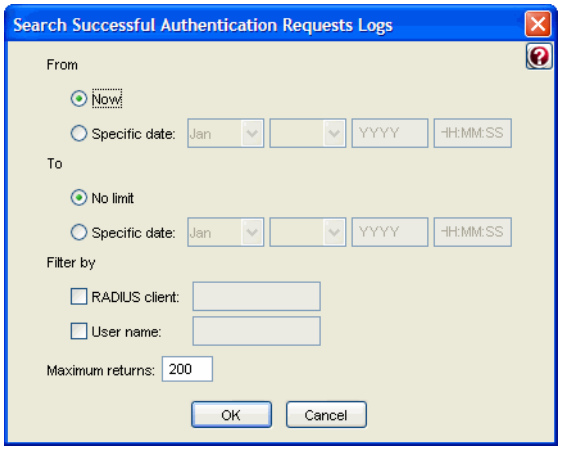


Figure 113 Search Logs Window

4 If you want to search the authentication log file for messages within a specified time range:

   a Specify the starting date/time in the range by clicking the **Now** radio button or by clicking the **From: Specific date** radio button.

   b Specify the ending date/time in the range by clicking the **No limit** radio button or by clicking the **To: Specified date** radio button and entering a date and time.

5 If you want to filter message so that you see only those relating to a specified RADIUS client, click the **RADIUS client** checkbox and enter the name of the RADIUS client in the **RADIUS client** text field.

6 If you want to filter message so that you see only those relating to a specified user, click the **User name** checkbox and enter the name of the user in the **User name** text field.

7 If you want to limit the number of messages you want SBR Administrator to display, enter a number in the **Maximum returns** field.

8 Click **OK**.

Configuring the Log Retention Period

Each day at midnight, the previous day's log files are completed, and new log files are created for the new day's transactions. To prevent the log files from filling up available disk space, you can configure Steel-Belted Radius to discard the log files after a specified number of days.

To configure the log retention period:

1 Open the Reports panel and click the **Settings** tab.

2 When the Settings tab (Figure 114) opens, enter the number of days you want Steel-Belted Radius to retain log file in the **Days to Keep log file** field.

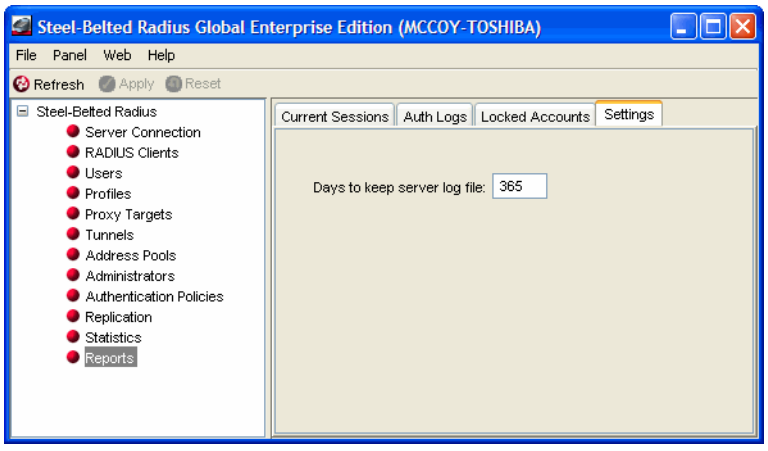


Figure 114 *Reports Panel: Settings Tab*

Using the Server Log File

The server log file records RADIUS events, such as server startup or shutdown or user authentication or rejection, as a series of messages in an ASCII text file. Each line of the server log file identifies the date and time of the RADIUS event, followed by event details. You can open the current log file while Steel-Belted Radius is running.

Server log files are kept for the number of days specified in the **Settings** tab in the Reports panel (described in "Configuring the Log Retention Period" on page 232) and then deleted to conserve disk space.

Optionally, you can specify a maximum size for a server log file by entering a non-zero value for the `LogfileMaxMBytes` setting in the [Configuration] section of the `radius.ini` file.

- If a maximum file size is set, the server log filename identifies the date and time it was opened (*`YYYYMMDD_HHMM`*.log). When the current server log file approaches the specified number of megabytes (1024 x 1024 bytes), the current log file is closed and a new one is opened. The closed file will be slightly smaller than the specified maximum file size.
- If the maximum file size is set to 0 (or if the `LogfileMaxMBytes` setting is absent), the server log file size is ignored and log file names are datestamped to identify when they were opened (*`YYYYMMDD`*.log).

  ***NOTE:** The size of the log file is checked once per minute, and the log file cannot roll over more than once a minute. The log file may exceed the specified maximum file size temporarily (for less than a minute) after it passes the LogfileMaxMBytes threshold between size checks.*

By default, server log files are located in the RADIUS database directory. You can specify an alternate destination directory in the [Configuration] section of the `radius.ini` file.

Level of Logging Detail

You can control the level of detail recorded in server log files by use of the LogLevel, LogAccept, and LogReject settings.

The LogLevel setting determines the level of detail given in the server log file. The LogLevel can be the number 0, 1, or 2, where 0 is the least amount of information, 1 is intermediate, and 2 is the most verbose. The LogLevel setting is specified in the [Configuration] section of `radius.ini` and in the [Settings] sections of `.aut` files.

The LogAccept and LogReject flags allow you to turn on or off the logging of Access-Accept and Access-Reject messages in the server log file. These flags are set in the [Configuration] section of `radius.ini`: a value of 1 causes these messages to be logged, and a value of 0 causes the messages to be omitted. An Accept or Reject is logged only if LogAccept or LogReject, respectively, is enabled and the LogLevel is "verbose" enough for the message to be recorded.

The TraceLevel setting specifies whether packets should be logged when they are received and being processed, and what level of detail should be recorded in the log.

Using the Accounting Log File

RADIUS accounting events are recorded in the accounting log file. Accounting events include START messages, which indicate the beginning of a connection; STOP messages, which indicate the termination of a connection; and INTERIM messages, which indicate a connection is ongoing.

Accounting log files use comma-delimited, ASCII format, and are intended for import into a spreadsheet or database program. Accounting log files are located in the RADIUS database directory area by default, although you can specify an alternate destination directory in the [Configuration] section of the `account.ini` file. Accounting log files are named *yyyymmdd*.ACT, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day on which the log file was created.

Accounting log files are kept for the number of days specified in the Settings tab of the Reports panel, and are deleted after that to conserve disk space.

The current log file can be opened while Steel-Belted Radius is running.

Accounting Log File Format

The first six fields in every accounting log entry are provided by Steel-Belted Radius for your convenience in reading and sorting the file:

- `Date` - the date when the event occurred
- `Time` - the time when the event occurred
- `RAS-Client` - the name or IP address of the RADIUS client sending the accounting record
- `Record-Type` - START, STOP, INTERIM, ON, or OFF, the standard RADIUS accounting packet types
- `Full-Name` - the fully distinguished name of the user, based on the authentication performed by the RADIUS server
- `Auth-Type` - a number that indicates the class of authentication performed:

```
0 - Native
10 - SecurID User
11 - SecurID Prefix
12 - SecurID Suffix
13 - Solaris User
14 - Solaris Group
15 - TACACS+ User
16 - TACACS+ Prefix
17 - TACACS+ Suffix
100 - Tunnel User
200 - External Database
(other) - Proxy
```

By default, the standard RADIUS attributes follow the `Auth-Type` identifier. See "Standard RADIUS Accounting Attributes" on page 236.

You can include vendor-specific attributes if the device sending the accounting packet supports them. For more information, see "Vendor-Specific Attributes" on page 23.

You can edit the `account.ini` initialization file to add, remove or reorder the standard RADIUS or vendor-specific attributes that are logged. For information on `account.ini`, refer to the *Steel-Belted Radius Reference Guide*.

First Line Headings

The first line of the accounting log file is a file header that lists the attributes that have been enabled for logging in the order in which they are logged. The following example of a first line shows required headings in bold italic, standard RADIUS headings in bold, and vendor-specific headings in regular text:

```
"Date","Time", "RAS-Client", "Record-Type", "Full-Name",
"Auth-Type", "User-Name", "NAS-Port", "Acct-Status-Type",
"Acct-Delay-Time", "Acct-Input-Octets", "Acct-Output-Octets",
"Acct-Session-Id", "Acct-Authentic","Acct-Session-Time",
"Acct-Input-Packets", "Acct-Output-Packets",
"Acct-Termination-Cause", "Acct-Multi-Session-Id",
"Acct-Link-Count","Acc-Err-Message",
"Nautica-Acct-SessionId","Nautica-Acct-Direction",
"Nautica-Acct-CauseProtocol","Nautica-Acct-CauseSource",
"Telebit-Accounting-Info","Last-Number-Dialed-Out",
"Last-Number-Dialed-In-DNIS","Last-Callers-Number-ANI",
"Channel","Event-Id","Event-Date-Time",
"Call-Start-Date-Time","Call-End-Date-Time",
"Default-DTE-Data-Rate","Initial-Rx-Link-Data-Rate",
"Final-Rx-Link-Data-Rate","Initial-Tx-Link-Data-Rate",
"Final-Tx-Link-Data-Rate","Sync-Async-Mode",
"Originate-Answer-Mode","Modulation-Type",
"Equalization-Type","Fallback-Enabled","Characters-Sent",
"Characters-Received","Blocks-Sent","Blocks-Received",
"Blocks-Resent","Retrains-Requested","Retrains-Granted",
"Line-Reversals","Number-Of-Characters-Lost",
"Number-of-Blers","Number-of-Link-Timeouts",
"Number-of-Fallbacks","Number-of-Upshifts",
"Number-of-Link-NAKs","Back-Channel-Data-Rate",
"Simplified-MNP-Levels","Simplified-V42bis-Usage",
"PW_VPN_ID"
```

Comma Placeholders

Steel-Belted Radius writes accounting events to the accounting log file, If an event recorded in the accounting log file does not have data for every attribute, a comma "placeholder" marks the empty entry, so that all entries remain correctly aligned with their headings. For example, based on the "first line" of headings described above, the following is a valid accounting log entry, in which the value of the Acct-Status-Type attribute is 7:

```
"12/23/1997","12:11:55","RRAS","Accounting-On",
,,,,7,,,,
```

Standard RADIUS Accounting Attributes

Table 40 lists the standard RADIUS accounting attributes defined in RFC 2866, "RADIUS Accounting."

Table 40. *Standard RADIUS Accounting Attributes*

`User-Name`	The name of the user as received by the client.
`NAS-Port`	The port number on the client device.
`Acct-Status-Type`	A number that indicates the beginning or ending of the user service: 1 - Start 2 - Stop 3 - Interim-Acct 7 - Accounting-On 8 - Accounting-Off
`Acct-Delay-Time`	Indicates how many seconds the client has been trying to send this record, which can be subtracted from the time of arrival on the server to find the approximate time of the event generating this request.
`Acct-Input-Octets`	Number of octets (bytes) received by the port over the connection; present only in STOP records.
`Acct-Output-Octets`	Number of octets (bytes) sent by the port over the connection; present only in STOP records.
`Acct-Session-Id`	Identifier used to match START and STOP records in a log file.
`Acct-Authentic`	indicates how the user was authenticated by RADIUS, the RAS itself, or another remote authentication protocol: 1 - RADIUS 2 - Local 3 - Remote
`Acct-Session-Time`	Elapsed time of connection in seconds; present only in STOP records.
`Acct-Input-Packets`	Number of packets received by the port over the connection; present only in STOP records.
`Acct-Output-Packets`	Number of packets sent by the port over the connection; present only in STOP records.

Table 40. *Standard RADIUS Accounting Attributes (Continued)*

`Acct-Termination-Cause`	Number that indicates how the session was terminated; present only in STOP records: 1 - User Request 2 - Lost Carrier 3 - Lost Service 4 - Idle Timeout 5 - Session Timeout 6 - Admin Reset 7 - Admin Reboot 8 - Port Error 9 - NAS Error 10 - NAS Request 11 - NAS Reboot 12 - Port Unneeded 13 - Port Preempted 14 - Port Suspended 15 - Service Unavailable 16 - Callback 17 - User Error 18 - Host Request
`Acct-Multi-Session-Id`	Unique accounting identifier to make it easy to link together multiple related sessions in a log file.
`Acct-Link-Count`	The count of links that are known to have been in a given multi-link session at the time the accounting record is generated.

Appendix A
Glossary

802.1X	The IEEE 802.1X standard defines a mechanism that allows a supplicant (client) to connect to a wireless access point or wired switch (authenticator) so that the supplicant can provide authentication credentials that can be verified by an authentication server.
AAA	Authentication, authorization, and accounting.
accounting	The process of recording and aggregating resource use statistics and log files for a user, connection session, or function for billing, system diagnosis, and usage planning.
agent	SNMP module on a managed device that responds to requests from a management station and sends traps to one or more recipients (trap sinks) to inform administrators of potential problems.
AP	Access Point. A device that serves as a communication hub to connect 802.1X wireless clients to a wired network.
attribute	RADIUS attributes carry the specific authentication, authorization, and accounting.
authentication	The process of verifying the identity of a person or file system and whether the person is allowed on a protected network.
authentication server	A back-end database server that verifies, from the credentials provided by an access client, whether the access client is authorized to use network resources.
authorization	The process of controlling the access settings, such as privileges and time limits, that the user can exercise on a protected network.
AVP	Attribute-value pair. An attribute and its corresponding value.; for example, `User-Name = admin`.
blacklist	A profile of checklist attributes that cause Steel-Belted Radius to reject an authentication request. For example, a blacklist profile might specify calling station phone numbers or IP addresses that are blocked by Steel-Belted Radius.
CA	Certificate authority. A trusted entity that registers the digital identity of a site or individual and issues a digital certificate that guarantees the binding between the identity and the data items in a certificate.

CCM

Centralized configuration management. The process by which information is shared between a primary RADIUS server and one or more replica RADIUS servers in a multi-server environment.

certificate

A digital file signed by a CA that guarantees the binding between an identity and the contents of the certificate.

CHAP

Challenge Handshake Authentication Protocol. An authentication protocol where a server sends a challenge to a requestor after a link has been established. The requestor responds with a value obtained by executing a hash function. The server verifies the response by calculating its own hash value: if the two hash values match, the authentication is acknowledged.

checklist

A list of attributes that must accompany a request for connection before the connection request can be authenticated.

CIDR

Classless Inter-Domain Routing. In CIDR notation, an IP address is represented as A.B.C.D/n, where /n identifies the IP prefix or network prefix). The IP prefix identifies the number of significant bits used to identify a network. For example, 192.168.1.22/18 means "use the first 18 bits to represent the network and the remaining 14 bits to identify hosts." Common prefixes are /8 (Class A network), /16 (Class B network), /24 (Class C network), and /32.

Table 41. *CIDR Translation*

CIDR Format	First Address	Last Address	Number of Usable IP Addresses[a]	Comparable IP Subnet Mask
10.0.0.0/8	10.0.0.0	10.255.255.255	16,777,214	255.0.0.0
10.0.0.0/16	10.0.0.0	10.0.255.255	65,534	255.255.0.0
192.168.0.0/24	192.168.0.0	192.168.0.255	254	255.255.255.0
192.168.0.0/25	192.168.0.0	192.168.0.127	126	255.255.255.128
192.168.0.0/26	192.168.0.0	192.168.0.63	62	255.255.255.192
192.168.0.0/27	192.168.0.0	192.168.0.31	30	255.255.255.224
192.168.0.0/28	192.168.0.0	192.168.0.15	14	255.255.255.240
192.168.0.0/29	192.168.0.0	192.168.0.7	6	255.255.255.248
192.168.0.9/29	192.168.0.8	192.168.0.15	6	255.255.255.248
192.168.0.10/30	192.168.0.8	192.168.0.11	2	255.255.255.252
192.168.0.10/31	192.168.0.10	192.168.0.11	0	255.255.255.254
192.168.0.10/32	192.168.0.10	192.168.0.10	1	255.255.255.255

a. Excludes the first address (network address) and last address (broadcast address) in an address range.

community

An SNMP community is a group of devices and management stations running SNMP. An SNMP device or agent may belong to more than one SNMP community.

community string

Character string included in SNMP messages to identify valid sources for SNMP requests and to limit access to authorized devices.

- The read community string allows an SNMP management station to issue Get and GetNext messages.

- The write community string allows an SNMP management station to issue Set messages.

credentials	Data that is verified when presented to an authenticator, such as a password or a digital certificate.
CRL	Certificate Revocation List. A data structure that identifies the digital certificates that have been invalidated by the certificates' issuing CA prior to their expiration date.
daemon	A program on a Solaris or Linux host that runs continuously to handle service requests.
dictionary	Text file that maps the attribute/value pairs supported by third-party RADIUS vendors.
DHCP	Dynamic Host Configuration Protocol. Protocol by which a server automatically assigns (leases) a network address and other configuration settings to a client temporarily or permanently.
DNIS	Dialed number identification service. A telephone service that identifies what number was dialed by a caller.
DNS	Domain Name Service. Internet protocol for mapping host names, domain names, and aliases to IP addresses.
EAP	Extensible Authentication Protocol. An industry-standard authentication protocol for network access that acts as a transport for multiple authentication methods or types. Defined by RFC 2284.
EAP-15	
EAP-32	See POTP.
EAP-FAST	Authentication method that uses EAP (Extensible Authentication Protocol) and FAST (Flexible Authentication via Secure Tunneling).
EAP-TLS	Authentication method that uses EAP (Extensible Authentication Protocol) and TLS (Transport Layer Security).
EAP-TTLS	Authentication method that uses EAP (Extensible Authentication Protocol) and TTLS (Tunneled Transport Layer Security).
GTC	Generic Token Card.
IEEE	Institute of Electrical and Electronics Engineers.
IETF	Internet Engineering Task Force. Technical subdivision of the Internet Architecture Board that coordinates the development of Internet standards.
IPv4	Implementation of the TCP/IP suite that uses a 32-bit addressing structure.
IPv6	Implementation of the TCP/IP suite that uses a 128-bit addressing structure.
Java	Programming language designed for use in distributed environments such as the Internet.
JDBC	Java Database Connectivity. Application programming interface for accessing a database from programs written in Java.
LDAP	Lightweight Directory Access Protocol. An IETF standard protocol for updating and searching directories over TCP/IP networks.

LDIF	LDAP Data Interchange Format. The format used to represent directory server entries in text form.
LEAP	Lightweight Extensible Authentication Protocol.
MAC	(1) Message Authentication Code. A MAC function takes a variable-length input and a key to produce a fixed-length output to carry authentication and integrity protection of data. (2) Media Access Control. The unique hardware address associated with a computer network interface.
managed device	A device that runs an SNMP agent.
management station	Host that monitors and controls managed devices running SNMP agents.
MIB	Management Information Base. A database of objects, such as alarm status or statistics counters, that can be monitored or overwritten by an SNMP management station.
MPPE	Microsoft Point-to-Point Encryption. A means of representing point-to-point packets in an RC4 encrypted format. Defined in RFC 3078.
MS-CHAP	Microsoft CHAP. Proprietary version of CHAP.
NAS	Network Access Server. Network device that accepts connection requests from remote users, authenticates users via RADIUS, and routes user onto the network. Identical in meaning to RAS.
NAT	Network Address Translation. Technique that allows an intranet to use IP addresses that are different from what the outside Internet thinks
native user	A user authenticated by Steel-Belted Radius using its internal authentication database.
ODBC	Open Database Connectivity. Standard (open) application programming interface for accessing a database.
OTP token	One-time password token. Hardware or software module that generates one-time passwords that can be used to authenticate a user.
PAC	Protected Access Credential. A high-entropy secret that is known to both the RADIUS client and the RADIUS server to secure the TLS handshake in EAP-FAST authentication.
PAP	Password Authentication Protocol. An authentication protocol where a requestor sends an identifier and password to a server after a link has been established. If the identifier and password match an entry in the server's database, the authentication is acknowledged.
PEAP	Protected Extensible Authentication Protocol. A two-phase authentication protocol where (1) an authentication server is authenticated to a supplicant using a digital certificate and a secure channel is established; and (2) the supplicant is authenticated to the authentication server via the secure channel.
POTP	Protected One-Time Password. EAP method that uses one-time password tokens for unilateral or mutual authentication.
proxy RADIUS	Process of authenticating users whose profiles are on other RADIUS servers by forwarding access-request packets received from a RADIUS client to a remote RADIUS

	server (the proxy target), and then forwarding the response from the remote server back to the RADIUS client.
proxy target	The remote RADIUS server that actually performs authentication in a proxy RADIUS sequence.
RADIUS	Remote Authentication Dial In User Service. A client/server security administration standard that functions as an information clearinghouse, storing authentication information about users and administering multiple security systems across complex networks.
RAS	Remote Access Server. Network device that accepts connection requests from remote users, authenticates users via RADIUS, and routes user onto the network. Identical in meaning to NAS.
return list	A list of attributes that Steel-Belted Radius must return to a RADIUS client after authentication of a user succeeds. The return list usually provides additional parameters that the RADIUS client needs to complete the connection.
roaming	The ability to move from one Access Point coverage area to another without interruption of service or loss of connectivity.
RSA SecurID	Security token system that allows remote-access users to generate a pseudorandom value they can forward as part of an authentication sequence.
session ID	Session Identifier. A string of characters uniquely identifying the session.
SHA-1	Secure Hash Algorithm-1. A one-way cryptographic function that takes a message of any length and produces a 160-bit message digest.
shared secret	An encryption key known only to the sender and receiver of data.
silent discard	The process of discarding a packet without further processing and without notification to the sender.
SNMP	Simple Network Management Protocol.
SSL	Secure Sockets Layer. Program layer that manages the security of messages on a network.
supplicant	The client in an 802.1X-authenticated network.
TACACS+	Terminal Access Controller Access Control System (with enhancements). An authentication protocol that allows a RAS to communicate with an authentication server to determine if a user should have access to a protected network.
TLS	Transport Layer Security.
trap	An SNMP message that reports a significant event, such as a problem, error, or change in state, that occurred within a managed device.
trap sink	The destination for trap messages sent by an SNMP agent on a managed device.
TTLS	Tunneled Transport Layer Security.
user database	A database where a RADIUS server keeps information about users, such as authentication information and network access permissions.
user profile	A record in the user database that describes how a particular user or class of users should be configured during authentication and authorization.

VSA	Vendor Specific Attributes.
WEP	Wired Equivalent Privacy. An encryption method designed to encrypt traffic between a WLAN client and an access point.
WLAN	Wireless Local Area Network.

Appendix B
When to Restart Steel-Belted Radius

This appendix explains the following topics when to stop and restart Steel-Belted Radius.

The **least** drastic action that causes this change to take effect is indicated by `Yes` in this table:

Table 42. *When to Stop and Restart Steel-Belted Radius*

Item changes:	Save the window or file	Issue a HUP signal	Stop/restart the server
Access window or object	Yes	No	(Also works)
`*.acc` files	No	No	Yes
`account.ini` file	No	No	Yes
`*.aut` files	No	(Sometimes)	Yes
Authentication policy	Yes	No	(Also works)
`*.dct` files	No	No	Yes
*.eap files	No	No	Yes
eap.ini file	No	No	Yes
`events.ini` file	No	No	Yes
Import `*.rif` or users file	Yes	No	(Also works)
IP Pools window or object	Yes	No	(Also works)
IPX Pools window or object	Yes	No	(Also works)
Log levels (in radius.ini file)	No	No	(Also works)
Profiles window or object	Yes	No	(Also works)
Proxy window or object	Yes	No	(Also works)
`radius.dct` file (see Notes below)	No	No	Yes
`radius.ini` file	No	No	Yes
RADIUS Clients window or object	Yes	No	(Also works)
Servers window or object	Yes	No	(Also works)
services file	No	No	Yes

Table 42. *When to Stop and Restart Steel-Belted Radius (Continued)*

Item changes:	Save the window or file	Issue a HUP signal	Stop/restart the server
`tacplus.ini` file	No	No	Yes
TLS and TTLS	No	No	(Also works)
Trace levels	No	No	(Also works)
Tunnels panel or object	Yes	No	(Also works)
Users window or object	Yes	No	(Also works)
`vendor.ini` file	No	No	Yes

Appendix C
Technical Notes

This appendix contains the following technical bulletins:

LDAP Support for Novell eDirectory

The Steel-Belted Radius LDAP authentication plug-in contains features to enable greater interoperability with Novell eDirectory.

If you have configured eDirectory to limit the number of grace logins available to a user, Steel-Belted Radius can be configured to coordinate with eDirectory on this feature. Each time a user authenticates, the number of grace logins available is decremented until the account is locked out and needs an administrator to unlock it. A profile is assigned to these users — a profile that overrides their normal profile — when they are being authenticated using a grace login.

The features include:

- **Allowing Expired Accounts**: When enabled, this feature allows users to log in even after their account has expired.

  NOTE: *As eDirectory itself does not check the password, this feature should be used only if the administrator has configured the ProfileForExpiredUsers setting to assign an alternate profile to the user, one that would inform the user of their account status but not allow a usable connection to the network. For example, you can use http redirection to force the connection to a web page with relevant information.*

NOTE: *Administrators should contact their NAS vendors to determine the capabilities of their NAS equipment and what attribute-value pairs would be needed to create such a connection.*

NOTE: Grace Logins*: When grace logins are limited in eDirectory, Steel-Belted Radius can be configured to accept or reject a user whose password has expired. This user is said to be in grace login mode; the user can also be allowed to log in but is provided with an alternate profile.*

NOTE: *The grace login feature requires NetWare 6.0 or later with eDirectory 8.6 or later.*

- **BindName**: You can use the BindName technique to search the eDirectory directory for a matching user and, having retrieved the user's DN, apply the Bind technique to authenticate the user's credentials. This combination allows the user to specify their common name (rather than the more cumbersome DN) when requesting authentication.

  NOTE: *This feature works only if the NetWare server accepts LDAP Bind requests for users who are in grace login state. Earlier versions of the NetWare server did not support this feature.*

Warning: *You must configure the eDirectory server to 'Allow Clear_text passwords' to use these LDAP extensions. You can do this from the ConsoleOne application, in the Properties section of the LDAP group entry for your server.*

Configuration

The `[NDS]` section (Table 43) has been added to the `ldapauth.aut` file to configure these features.

Table 43. *[NDS] Settings*

Field	Usage
Enable	Set to 1 to enable the Novell eDirectory (NDS) extensions. Default value is 0.
AllowExpiredAccountsForUsers	Set to 1 to allow users to authenticate even after their account has expired. Default value is 0. **Important**: This requires that the Netware server notify Steel-Belted Radius that the user's account has expired when the Netware server is attempting to Bind.
ProfileForExpiredUsers	The name of the profile to assign a user (as an override) if authenticated with an expired account. If you do not provide a value for this setting, the user is accepted with the usual profile and attributes. We recommend that you provide the user with a profile with restricted access.
AllowGraceLoginsForUsers	Set to 1 if users should be allowed to be authenticated in grace login mode. This decrements the grace login counter, and rejects the user when it has run out. Default value is 1.

Table 43. [NDS] Settings (Continued)

Field	Usage
ProfileForGraceLoginUsers	The name of the profile to assign a user (as an override) if authenticated in grace login mode. If you do not provide a value for this setting, the user is accepted with the usual profile and attributes.

If Enable is set to 1, Steel-Belted Radius works as follows:

- If the eDirectory directory is configured to operate without grace logins:
  - If `AllowExpiredAccountsForUsers` is set to 0 (the default), users with expired passwords are rejected.
  - If `AllowExpiredAccountsForUsers` is set to 1, users with expired accounts are accepted; the attributes returned in the Access-Accept response are either the attributes normally assigned to the user or, if the `ProfileForExpiredUsers` setting is specified, the attributes specified in that profile. If you enable this feature, we recommend that you configure the `ProfileForExpiredUsers` setting.
- If the eDirectory directory is configured to operate with grace logins:
  - If `AllowGraceLoginsForUsers` is set to 0, users with correct but about-to-expire passwords are rejected.
  - If `AllowGraceLoginsForUsers` is set to 1 (the default), users with correct but about-to-expire passwords are accepted; the attributes returned in the Access-Accept are either the regular attributes assigned to the user or, if `ProfileForGraceLoginUsers` is specified, the attributes specified in that profile.

Sample ldapauth.aut file

```
[Bootstrap]
LibraryName=ldapauth.dll
Enable=1
InitializationString=LDAP

[Settings]
MaxConcurrent=1
Timeout=20
ConnectTimeout=25
QueryTimeout=10
WaitReconnect=2
MaxWaitReconnect=360
BindName=uid=<User-Name>, ou=sales, o=bigco.com
LogLevel = 0
UpperCaseName = 0
PasswordCase=original
PasswordFormat=0
Search = DoLdapSearch
SSL = 0
```

```
[Server]
s1=

[Server/s1]
Host=192.168.5.110
Port = 389

[Request]
%UserName = User-Name

[Response]
%profile= attThatContainsUserProfile

[Search/DoLdapSearch]
;Bind as a privileged user or someone that has the right to
; search the tree and retrieve the DN of users
bind=cn=administrator,o=netware6
Password= support
Base = o=netware6
Scope = 2
Filter = uid=<User-Name>
%DN = dn
;if the user is found perform search getprofile
onfound = GetProfile
;else reject the user
onnotfound=$reject


[Search/GetProfile]
; bind using the DN retrieved in doLdapSearch
Bind = <dn>
;You do not have to supply the password SBR knows to use the
;one received in the auth request.
;Setting base to the DN is most efficient
Base =<DN>
Scope = 2
Filter = uid=<User-Name>
Attributes = AttrList

[Attributes/AttrList]
attThatContainsUserProfile

[NDS]
Enable=1
AllowExpiredAccountsForUsers=1
ProfileForExpiredUsers=Expired
AllowGraceLoginsForUsers=1
ProfileForGraceLoginUsers=Grace

[Attributes/AttrList]
;Filter-Id
;Session-Timeout
;thepasswordis
```

CCA Support for 3COM

Steel-Belted Radius can support the generation of 3Com CCA tunnel attributes.

Configuration

To enable the return of the required CCA tunnel attributes, the `ccagw.ini` file must be modified.

The `ccagw.ini` file contains information about gateways, which are stored in the [*gateway*] sections. A [*gateway*] section must be present for each gateway supported.

The following table describes each field:

Table 44. *ccagw.ini File*

Setting	Meaning
Address	The address of the gateway.
TunnelRefresh	The number of seconds before the tunnel refreshes. The default value is 0.
Description	A text string describing the gateway.
Secret	The shared secret between Steel-Belted Radius and this gateway device.

For example:

```
[Jupiter-Gateway]
Address = 200.47.98.142
TunnelRefresh = 3600
Description = Jersey City facility, East Coast subscribers
Secret = Holland Tunnel
```

Setting User and Profile Attributes

To enable this functionality for a particular user, the return list for the user must contain the following attributes:

```
Tunnel-Authentication
VPN-Gateway
```

Both of these attributes are defined as strings. The value of each attribute must be set to the name of the gateway used in the `ccagw.ini` file. For example, the return list of a user would have to include:

```
Tunnel-Authentication = Jupiter-Gateway
VPN-Gateway = Jupiter-Gateway
```

It is important to make sure that both attributes name the correct gateway. If an unknown gateway is named, the request is rejected.

NOTE: *Steel-Belted Radius is capable of returning multiple pairs of attributes for different gateways. For each gateway named in one of the attributes, a different random session key is generated.*

NOTE: *Please see your 3Com documentation for more information.*

Ascend Filter Translation

Ascend defines two attributes — `Ascend-Data-Filter` (242) and `Ascend-Call-Filter` (243) — that contain structured binary data representing a filter to be applied to the NAS device.

Instead of entering hexadecimal strings to configure these attributes, users can configure these attributes as text strings. Steel-Belted Radius automatically converts the text strings to the proper binary representation. The original filter attributes are still supported, and these attributes still may be configured as hexadecimal strings.

The following attributes allow configuration as text:

```
Ascend-Data-Filter-String
Ascend-Call-Filter-String
```

When Steel-Belted Radius formats a response packet, it translates the string version of the attribute to the appropriate binary value, and returns the attribute in the Access-Accept message.

Configuration

These attributes may be entered as text strings through the SBR Administrator. The attributes may also be returned from an LDAP or SQL database during authentication.

No syntax validation is performed when the attribute is configured. The validation of syntax occurs only when the response packet is formatted. If the syntax is invalid, a reject response is issued and an error is logged.

***NOTE:** These attributes should be tested before configured on a production server.*

Two types of filter are supported: “ip” and “generic”. “ipx” filters are not supported.

Syntax

In the syntax descriptions below, brackets [ ] indicate that the items enclosed are optional.

`ip [direction] [action] [srcip address[/mask]] [dstip address[/mask]] protocol [srcport operator port] [dstport operator port]`

Table 45.

Parameter	Values
direction	May be "in" or "out". The default is "out".
action	May be "forward" or "drop". The default is "drop".
address	An IP address in decimal dotted notation.
mask	The number of bits (decimal) in the network portion, from 0 through 32. The default is based on class of network.

Table 45.

Parameter	Values
protocol	The protocol number (decimal); for example, 6 for TCP or 17 for UDP. The following protocol names are translated to the proper number: • icmp(1) • tcp(6) • udp(17) • ospf(89).
operator	May be = (equal sign), != (exclamation and equal sign), < (less than), or > (greater than).
port	The port number (decimal). In addition, the following service names are translated to the proper port number: • ftp-data(20) • ftp(21) • telnet(23) • smpt(25) • nameserver(42) • domain(53) • tftp(69) • gopher(70) • finger(79) • www(80) • kerberos(88) • hostname(101) • nntp(119) • ntp(123) • exec(512) • login(513) • cmd(514) • talk(517)

Example:

```
ip out forward srcip 10.1.1.0/24 6 dstport = 80 srcport < 1023
```

NOTE: *Please see your Ascend documentation for details about the syntax for these attributes.*

NOTE: *If the ProfileData attribute stores multiple attribute-value pairs and one or more of those attributes appears in the applicable dictionary, then that attribute and its value are returned to the RADIUS client even if the attribute is not enumerated in the [Response] section of the file.*

Ericsson Enhanced Token Caching

This section pertains only to Ericsson equipment that supports enhanced token caching.

Enhanced token caching allows the administrator to specify that particular users are authenticated with both an ordinary password and a SecurID passcode. For such users, the ordinary PAP or CHAP password is checked first. If this first authentication is successful, the user's SecurID passcode is authenticated. Only if both authentications succeed is the user allowed access.

NOTE: *The enhanced token caching feature does not interact in any way with the ordinary token caching feature described in the above section. Enhanced token caching is required to support the newer firmware releases on Ericsson devices.*

Enhanced Token Caching Configuration

To enable enhanced token caching, a file with extension `.aut` must be included in the server directory - typically named `sidalt.aut`. It has the following format.

```
[Bootstrap]
LibraryName = sidalt.dll
Enable = { 0 | 1 }
InitializationString = string

[Settings]
TokenAttr = string
CacheTimeoutAttr = string
MessageID = { 0 | 1 }
ChallengeTokenInPassword = { 0 | 1 }
```

Table 46 describes each field in the `sidalt.aut` file.

Table 46. *sidalt.aut Syntax*

Sidalt.aut Field	Meaning
Enable	Set to 0 to disable. Set to 1 to enable. Default is 0.
LibraryName	Identifies the dynamic link library used to process SecurID authentication. Default is `sidalt.dll`.
InitializationString	Identifies the enhanced token caching software component for logging purposes. A typical value might be `SecurID Alt`. This field is required.
TokenAttr	The name of the Access-Request attribute containing the passcode or other information to passed to the RSA SecurID server. This attribute must match the corresponding dictionary (`.dct` file) entry. This field is required.
CacheTimeoutAttr	The name of the Access-Response attribute containing the number of seconds a passcode remains in the cache from the time it was first validated by the RSA SecurID server. This attribute must match the corresponding dictionary (`.dct` file) entry. This field is required.
MessageID	Controls the format of Reply-Message attribute response packets, which are used to prompt the user for information during a challenge, and to inform the user of results. If set to 0, the Reply-Message attribute consists only of a text message. If set to 1, the Reply-Message attribute consists of a 1-byte message ID followed by a text message. Default is 0.

Table 46. sidalt.aut Syntax (Continued)

Sidalt.aut Field	Meaning
ChallengeTokenInPassword	Allow test clients to interpolate with the enhanced token caching plug-in. If set to 0, passcode or other information entered by the user as a response to a challenge appears in the TokenAttr entry. If set to 1, passcode or other information entered by the user as a response to a challenge appears in the User-attribute, rather than in the attribute specified by the TokenAttr entry.

Enhanced Token Caching Administration

To authenticated a user through the enhanced token caching component, the following must be true:

1 The attribute specified by the TokenAttr entry must be present in the Access-Request;

2 The return list specified for that user, either directly or through a profile, must include the attribute specified by the CacheTimeoutAttr entry.

If either attribute is not supplied, the user is assumed not to require enhanced token caching authentication, and is accepted.

***NOTE:** See your Ericsson product documentation for information about NAS and PPP client operation under the enhanced token caching authentication method.*

Uniport Plug-In

3Com's Uniport project can operate with Steel-Belted Radius via a plug-in.

Operation

Uniport requires RADIUS call-type determination as a back-up for SIP call-type determination. To determine call-type, the HiPerARC system sends Steel-Belted Radius a request containing a Service-Type attribute of Call-Check (10) and a User-Name attribute in which the value is the same as the Called-Station-Id (DNIS) attribute. The type of call is then determined based on the User-Name (DNIS), and the appropriate Service-Type attribute returned in the Access-Accept packet.

A Uniport plug-in method is instantiated for each value of the Service-Type attribute which can be returned in the Access-Accept. The proper method is utilized using proxy mapping to a directed realm which specifies the method instance. The method then sets the configured profile in the response and indicates it was successful.

The Uniport methods return a reject if a Service-Type attribute with a Call-Check value is not present in the request, if User-Name or Called-Station-Id attributes are not present, or if their values are not identical.

Configuration

The attribute(s) to be returned in the Access-Accept to identify the call-type are defined as Steel-Belted Radius profiles. For example, a FAX profile may be created which would return a value of 96 in a Service-Type attribute.

The Uniport methods are configured with `*.AUT` files which specify the profile to return. The method is identified and associated with a directed realm by the initialization string.

For example, a method to return the FAX profile may use a configuration file such as `FAX.AUT`, which would have the following settings:

```
[Bootstrap]
Enable = 1
InitializationString = UNIPORT FAX
Profile = FAX
LibraryName = Uniport.so
```

The corresponding directed realm would then identify the method in its `*.DIR` file. For example, in the `FAX.DIR` file the settings might be:

```
[Auth]
Enable = 1

[AuthMethods]
Uniport fax
```

If you want a default profile, the map may be configured to direct the request to a Uniport method by default. The same result may be obtained by omitting the default from the map and setting the first method in the authentication method chain to the desired default.

If you do not want a default profile, configure the map to direct requests by default to a method which has no profile set; the method returns with a value to indicate a failure to authenticate.

Windows Performance Monitor

The Steel-Belted Radius service has information which can be viewed with the Performance Monitor on a Windows administrative workstation or server. You can start multiple instances of the Windows Performance Monitor to display performance statistics graphs for more than one Steel-Belted Radius server simultaneously.

To view a graph of Steel-Belted Radius performance:

1. Run the Windows performance monitor by choosing **Start > Control Panel > Administrative Tools > Performance**.
2. Click the **+** button in the Performance window (or press CTRL+I).

   The Add Counters window opens.
3. Pull down the **Performance Object** list and choose **Steel-Belted Radius**.

If you are running multiple Steel-Belted Radius servers, select the one you want to monitor from the **Select counters from computer** list.

4 If you want to select the counters that you want to graph, click the **Select counters from list** radio button, choose the counter you want, and click **Add**. Continue choosing counters and clicking Add until all desired counters have been selected.

If you want to display all counters for Steel-Belted Radius, click the **All counters** radio button and click **Add**.

Most perfmon counters relating to Steel-Belted Radius have self-explanatory names, such as `Acct Failures - Insufficient Resources` or `Acct Failures - Invalid Shared Secret`.

Of special interest is the `Failed Auths - n` counter, where *n* is a number between 1 and 16. You can set up as many as 16 `Failed Auths - n` counters, where each counter tracks the number of failed authentication requests that were encountered for all of the RADIUS clients that you have mapped to collection number *n*.

To set up the `Failed Auths - n` counter, you must configure the [FailedAuthOriginStats] section of `radius.ini`. For information on `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.

5 Click **Close** when you are finished adding counters.

The Performance Monitor window displays a graph of the counters you have selected. The graph updates itself at regular intervals until you close the Performance Monitor window.

6 Optionally, specify the color, scale, line width, and line style for one or more Steel-Belted Radius counters.

Right-click the name of a counter and choose **Properties** from the context menu. When the System Monitor Properties window opens, specify the characteristics you want the graph of the counter to use.

Table 47 describes the meaning of each perfmon counter.

***Table 47.** perfmon Counters*

perfmon Counter	Meaning
Acct Failures - Insufficient Resources	The number of accounting requests that were discarded because the RADIUS server was unable to obtain sufficient system resources to process the request.
Acct Failures - Invalid Clients	The number of accounting requests that were discarded because the RADIUS client identified in the request was not defined in the RADIUS server database.
Acct Failures - Invalid Requests	The number of accounting requests that were discarded because the request was malformed or contained invalid attributes.
Acct Failures - Invalid Shared Secret	The number of accounting requests that were discarded because the request contained an invalid digital signature. This is usually due to a mismatch in the shared secrets defined on the RADIUS client and the RADIUS server.

Table 47. perfmon Counters (Continued)

perfmon Counter	Meaning
Acct Proxy Failures	The number of forwarded accounting requests for which failures were encountered.
Acct Requests Forwarded	The number of accounting requests that were forwarded to other RADIUS servers.
Acct Requests Retried	The number of unique accounting requests for which retries were received by the RADIUS server.
Acct Requests Retried/sec	The number of accounting requests per second for which one or more retries has been received by the RADIUS server.
Acct Retry Requests	The number of actual accounting request retries received by the RADIUS server.
Acct Retry Requests/sec	The number of accounting request retries per second received by the RADIUS server.
Acct Service Time	The number of seconds that elapsed from the time the last completed accounting request was received to the time the RADIUS server sent a response. Responses generated by proxies are not reflected in this statistic.
Acct Starts	The number of accounting start requests received by the RADIUS server. An accounting start signifies the granting of a connection to an end-user by the remote access server.
Acct Starts/sec	The number of accounting start requests received by the RADIUS server per second. An accounting start signifies the granting of a connection to an end-user by the remote access server.
Acct Stops	The number of accounting stop requests received by the RADIUS server. An accounting stop signifies that an end-user has disconnected from the remote access server.
Acct Stops/sec	The number of accounting stop requests received by the RADIUS server per second. An accounting stop signifies that an end-user has disconnected from the remote access server.
Auth Failure - Authentication Failures	Number of unique authentication requests to which the RADIUS server replied with a reject because no user specified in the database possessed a matching password. A mismatch in shared secrets would also cause this counter to be incremented.
Auth Failure - Checklist Mismatches	Number of unique authentication requests to which the RADIUS server replied with a reject because the request did not include required checklist information.
Auth Failure - Insufficient Resources	Number of unique authentication requests to which the RADIUS server replied with a reject because the RADIUS server ran into a system resource limitation.
Auth Failure - Invalid Clients	Number of unique authentication requests to which the RADIUS server replied with a reject because the request was from a RADIUS client not identified in the RADIUS server's database.
Auth Failure - Invalid Requests	Number of unique authentication requests to which the RADIUS server replied with a reject because the request was malformed or contained invalid attributes.
Auth Proxy Failures	The number of forwarded authentication requests for which failures were encountered.
Auth Proxy Rejects	The number of forwarded authentication requests for which rejects were received from the target RADIUS server.

Table 47. perfmon Counters (Continued)

perfmon Counter	Meaning
Auth Requests	The number of unique authentication requests that the RADIUS server has received.
Auth Requests Forwarded	The number of authentication requests that were forwarded to another RADIUS server.
Auth Requests Retried	The number of unique authentication requests for which retries were received by the RADIUS server.
Auth Requests Retried/sec	The number of authentication requests per second for which one or more retries has been received by the RADIUS server.
Auth Requests/sec	The number of unique authentication requests that the RADIUS server has received per second.
Auth Retry Requests	The number of actual authentication request retries received by the RADIUS server.
Auth Retry Requests/sec	The number of authentication request retries per second received by the RADIUS server.
Auth Service Time	The number of seconds that elapsed from the time the last completed authentication request was received to the time the RADIUS server sent an Accept response. Accept responses generated for tunnel requests or by proxies are not reflected in this statistic.
Auth SQL Disconnects	The number of times an existing connection to a SQL authentication database failed.
Auth SQL Failures	The number of times an attempt to connect to a SQL authentication database failed.
Auth SQL Records Not Found	The number of times no record was found in a SQL authentication database for the specified username.
Auth SQL Timeouts	The number of times a timeout occurred attempting to execute a SQL authentication request.
Auth Successes	The number of unique authentication requests to which the RADIUS server replied with an Accept.
Auth Successes/sec	The number of unique authentication requests to which the RADIUS server replied with an accept per second.
Concurrency Auth Failures	The number of times an authentication request was forwarded to the concurrency server and the concurrency server returned a reject for reasons other than users being over their port limits.
Concurrency Auth Service Time	The number of seconds elapsed from the last time an authentication request was sent to the concurrency server and a response was received.
Concurrency Auth Timeouts	The number of times an authentication request was forwarded to the concurrency server and no response was received within the configured time for the proxy entry.
Concurrency Over Port Limit	The number of times an authentication request was forwarded to the concurrency server and the concurrency server returned a reject because users were over their port limits.

***Table 47.** perfmon Counters (Continued)*

perfmon Counter	Meaning
Failed Auths - 1	The number of failed authentication requests that were encountered for clients categorized in collection number 1. To set up the Failed Auths - n counter, you must configure the [FailedAuthOriginStats] section of `radius.ini`. For information on `radius.ini`, refer to the *Steel-Belted Radius Reference Guide*.
Failed Auths - 2	The number of failed authentication requests that were encountered for clients categorized in collection number 2.
Failed Auths - 3	The number of failed authentication requests that were encountered for clients categorized in collection number 3.
Failed Auths - 4	The number of failed authentication requests that were encountered for clients categorized in collection number 4.
Failed Auths - 5	The number of failed authentication requests that were encountered for clients categorized in collection number 5.
Failed Auths - 6	The number of failed authentication requests that were encountered for clients categorized in collection number 6.
Failed Auths - 7	The number of failed authentication requests that were encountered for clients categorized in collection number 7.
Failed Auths - 8	The number of failed authentication requests that were encountered for clients categorized in collection number 8.
Failed Auths - 9	The number of failed authentication requests that were encountered for clients categorized in collection number 9.
Failed Auths - 10	The number of failed authentication requests that were encountered for clients categorized in collection number 10.
Failed Auths - 11	The number of failed authentication requests that were encountered for clients categorized in collection number 11.
Failed Auths - 12	The number of failed authentication requests that were encountered for clients categorized in collection number 12.
Failed Auths - 13	The number of failed authentication requests that were encountered for clients categorized in collection number 13.
Failed Auths - 14	The number of failed authentication requests that were encountered for clients categorized in collection number 14.
Failed Auths - 15	The number of failed authentication requests that were encountered for clients categorized in collection number 15.
Failed Auths - 16	The number of failed authentication requests that were encountered for clients categorized in collection number 16.
Forwarded Requests Retried	The number of unique forwarded accounting and authentication requests for which retries were transmitted by the RADIUS server.
Forwarded Requests Retried/sec	The number of unique forwarded accounting and authentication requests for which retries were transmitted per second by the RADIUS server.
Forwarded Retry Requests	The number of actual retransmissions of forwarded accounting and authentication request performed by the RADIUS server.
Forwarded Retry Requests/sec	The number of actual retransmissions of forwarded accounting and authentication request per second performed by the RADIUS server.

Table 47. *perfmon Counters (Continued)*

perfmon Counter	Meaning
Proxy Failures - Insufficient Resources	The number of authentication and accounting requests that were forwarded to other RADIUS servers for which the RADIUS server was unable to obtain sufficient system resources to process the request.
Proxy Failures - Invalid Response	The number of authentication and accounting requests that were forwarded to other RADIUS servers for which malformed or invalid responses were received.
Proxy Failures - Invalid Shared Secret	The number of authentication and accounting requests that were forwarded to other RADIUS servers for which responses were discarded because the response contained an invalid digital signature. This is usually due to a mismatch in the shared secrets defined on the RADIUS client and RADIUS server.
Proxy Failures - Time Out	The number of authentication and accounting requests that were forwarded to other RADIUS servers for which no response was received after the specified number of retries.
Seconds since started	Number of seconds Steel-Belted Radius has been running.
Sessions Online	The number of sessions currently active in the RADIUS server's Sessions list.
Static Acct Service Time	The number of seconds elapsed from the last time an accounting request was sent to the static accounting proxy server and a response was received.
Total Acct Failures	The number of unique accounting requests to which the RADIUS server did not reply. Reasons for the failures are identified in other statistics.
Total Acct Failures/sec	The number of unique accounting requests to which the RADIUS server did not reply per second because of an error. Reasons for the failures are identified in other statistics.
Total Acct Offs	The number of accounting off requests received by the RADIUS server. An accounting off signifies that the accounting support in the RADIUS client has been disabled. This request is most often issued when a RADIUS client in being shut down.
Total Acct Offs/sec	The number of accounting off requests received by the RADIUS server per second.
Total Acct Ons	The number of accounting on requests received by the RADIUS server. An accounting on signifies that the accounting support in the RADIUS client has been enabled. This request is most often issued when a RADIUS client is powered on.
Total Acct Ons/sec	The number of accounting on requests received by the RADIUS server per second.
Total Auth Challenges	The number of authentication requests that resulted in a RADIUS challenge response.
Total Auth Failures	The number of unique authentication requests to which the RADIUS server replied with a reject. Reasons for the failures are identified in other statistics.
Total Auth Failures/sec	The number of unique authentication requests to which the RADIUS server replied with a reject, per second. Reasons for the failures are identified in other statistics.
Total Forwarded Request Failures	The number of forwarded authentication and accounting requests that encountered failures.

Table 47. *perfmon Counters (Continued)*

perfmon Counter	Meaning
Total Forwarded Request Failures/sec	The number of forwarded authentication and accounting requests that encountered failures, per second.
Total Forwarded Requests	The number of authentication and accounting requests that were forwarded to other RADIUS servers.
Total Forwarded Requests/sec	The number of authentication and accounting requests per second that were forwarded to other RADIUS servers.
Users Online	The number of unique users represented in the RADIUS server's Sessions List.

Appendix D
Authentication Protocols

This appendix provides a matrix of authentication methods and their supported authentication protocols.

Table 48. *Authentication Protocols*

Method	PAP	CHAP	MS-CHAP	MS-CHAP-V2	LEAP	EAP-MSCHAP-V2	EAP-MD5	PAP/Token card	EAP/Token card
Microsoft PEAP available inner authentication protocols	No	No	No	No	No	Yes	No	No	No
Cisco PEAP available inner authentication protocols	No	No	No	No	Yes	Yes	Yes	No	Yes
TTLS available inner authentication protocols	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Native	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A
Native (password saved as Allow PAP only, {SHA} or {crypt}).	Yes	No	No	No	No	No	No	N/A	N/A
Windows Domain Authentication									
Windows Domain Group	Yes	No	Yes	Yes	Yes	Yes	No	N/A	N/A
Windows Domain User	Yes	No	Yes	Yes	Yes	Yes	No	N/A	N/A
UNIX authentication methods									
UNIX User	Yes	No	No	No	No	No	No	N/A	N/A
UNIX Group	Yes	No	No	No	No	No	No	N/A	N/A
Other authentication plug-ins									
RSA SecurID	Yes	No	No	No	No	No	No	N/A	Yes
TACACS+	Yes	Yes	No	No	No	No	Yes	N/A	N/A
LDAP									
BIND (this includes AD and eDirectory/NDS)	Yes	No	No	No	No	No	No	N/A	N/A

Table 48. *Authentication Protocols (Continued)*

Method	PAP	CHAP	MS-CHAP	MS-CHAP-V2	LEAP	EAP-MSCHAP - V2	EAP-MD5	PAP/ Token card	EAP/ Token card
BINDNAME (password stored in clear text)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A
BINDNAME (password stored in SHA/Solaris Crypt text)	Yes	No	No	No	No	No	No	N/A	N/A
BINDNAME (password Stored as MD4 hash of unicode value text)	Yes	No	No	Yes	No	Yes	No	N/A	N/A
BINDNAME (password Stored as enc-md5)	Yes	Yes	No	No	No	No	No	N/A	N/A
SQL									
Password stored in clear text	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A
Password password stored in SHA/Solaris Crypt text	Yes	No	No	No	No	No	No	N/A	N/A
Password stored as {MD4} hash of unicode value text	Yes	No	No	Yes	No	Yes	No	N/A	N/A
Password stored as {enc-md5}	Yes	Yes	No	No	No	No	No	N/A	N/A

Appendix E
Importing and Exporting Data

This appendix describes how to export database information from one Steel-Belted Radius server to an XML file, and then selectively import database information into another Steel-Belted Radius server. The ability to export and import information facilitates configuration of multiple Steel-Belted Radius servers.

Steel-Belted Radius uses XML files with UTF-8 encoding to store exported information. Export files contain only those attributes that have been assigned values, including defaults. For example, if an exported native user item does not have a profile associated with it, the exported information will not identify a profile name, though it will include default values required by Steel-Belted Radius.

NOTE: *Previous versions of Steel-Belted Radius used a file format called RADIUS Import File (RIF) to export and import information. For backward compatibility, Steel-Belted Radius includes a utility to convert RIF files to XML format. For more information, see the Steel-Belted Radius Getting Started manual.*

Exporting to a RADIUS Information File

To export information from the Steel-Belted Radius database to an XML file:

1. Run the SBR Administrator.
2. Choose **File > Export**.
3. When the Export window (Figure 115) opens, select the information you want to export.

   Each tab in the window lists exportable items of a particular category. For each category, select the appropriate tab and click each item you'd like to export.

   To select a contiguous range of items, select the first item in the range, hold down the SHIFT key, and click the last item in the range.

   To select a non-contiguous set of items, hold down the CONTROL key as you click each item you want.

   To select all items in a category, click **All**.

To select all items in all categories, click **Select All**.

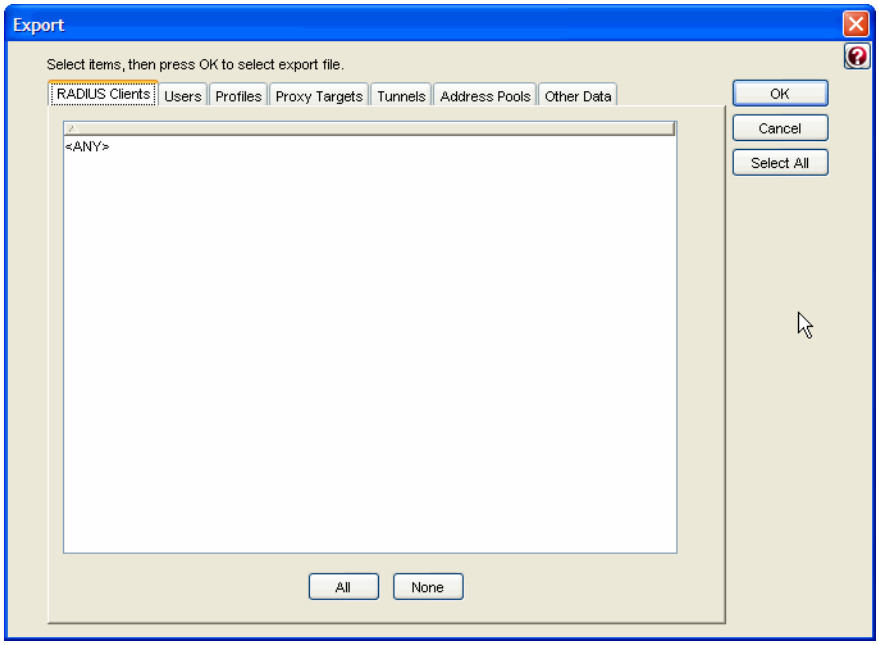


***Figure 115** Export Window*

4 After you have selected the items you want to export, click **OK**.

5 When the Export to XML window (Figure 116) opens, specify a file name and click **Save**.

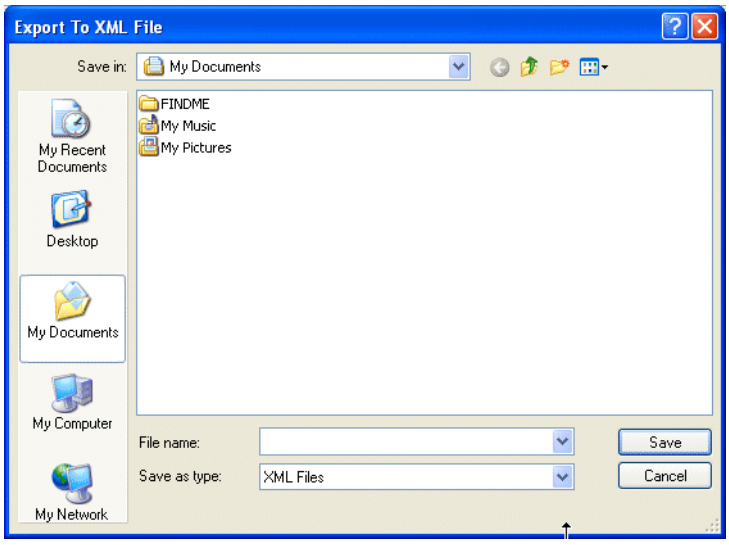


***Figure 116** Export to XML Window*

Importing Into the Steel-Belted Radius Database

To import information from an XML file into the database on your Steel-Belted Radius server:

1. Run the SBR Administrator.
2. Choose **File > Import**.
3. When the Import from XML window (Figure 117) opens, select the XML file containing the information you want to import and click **Open**.

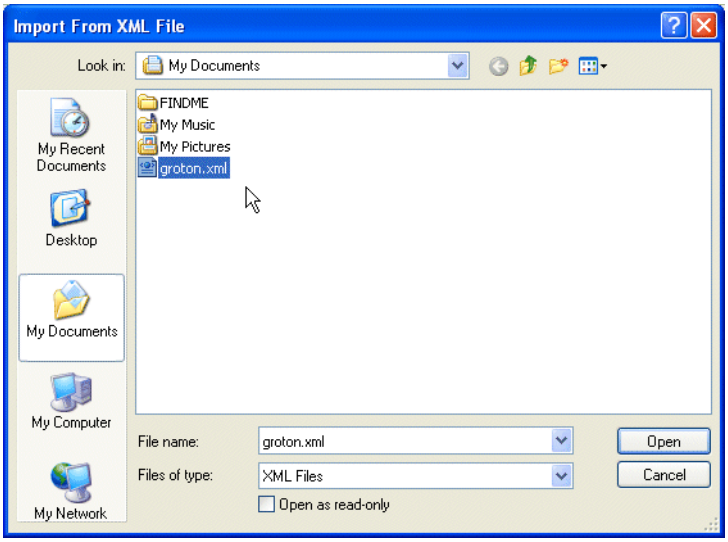


***Figure 117** Import from XML File Window*

4. When the Import window (Figure 118) opens, specify whether what the SBR Administrator should do when it finds an object with the same name already in the Steel-Belted Radius database.
   - Click **Skip** if you want SBR Administrator to leave the item already in the database intact.
   - Click **Replace** if you want SBR Administrator to overwrite the item in the database with the imported information.
5. Select the information you want to import by clicking each tab and selecting items to import.

   To select a contiguous range of items, select the first item in the range, hold down the SHIFT key, and click the last item in the range.

   To select a non-contiguous set of items, hold down the CONTROL key as you click each item you want.

   To select all items in a category, click **All**.

   To select all items in all categories, click **Select All**.

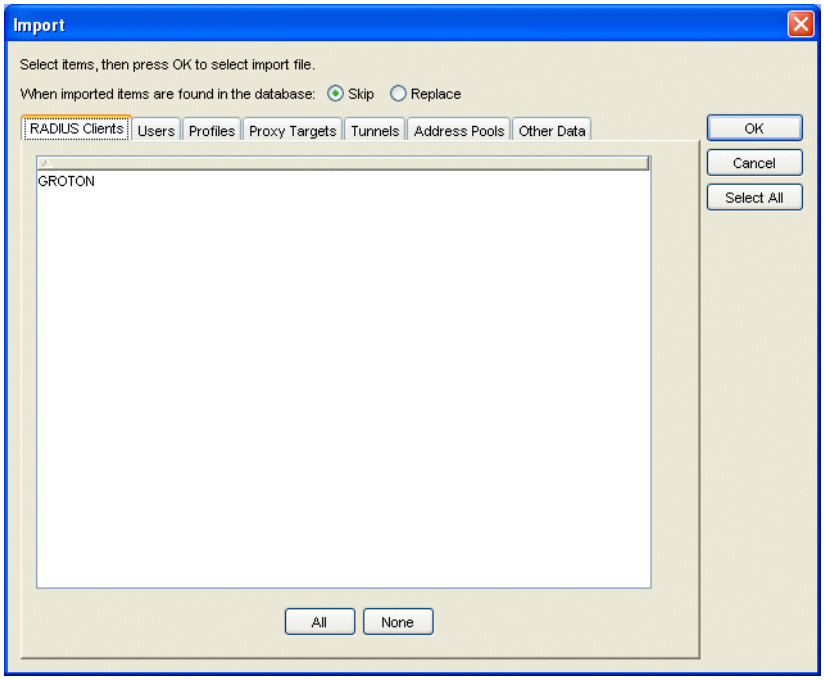


***Figure 118** Import Window*

6 After you select all the items you want, click **OK**. The items you selected are added to the Steel-Belted Radius database.

Appendix F
Configuring IPv6 Support

This appendix presents an overview of current IPv6 functions in Steel-Belted Radius and describes how to configure Steel-Belted Radius to enable IPv6 addressing and attribute processing.

***NOTE:** Version 5.x of Steel-Belted Radius provides early support for the evolving IPv6 protocol suite to meet the needs of early-adopter environments, such as carrier test laboratories. IPv6 functions are intended for users who are actively experimenting with IPv6 networks. Enabling the IPv6 functionality in Steel-Belted Radius server Version 5.x in a production environment is not recommended. Funk Software is committed to providing production-quality IPv6 support in future versions of Steel-Belted Radius. Please check the Funk Software website (`www.funk.com`) for updates.*

About IPv6

IPv6 is the next step in the evolution of the Internet Protocol, currently implemented as Internet Protocol Version 4 (IPv4). IPv6, which has been under development for more than 10 years, provides improvements over IPv4 in addressing, configuration, and security. Although IPv6 is still an evolving standard, many operating system vendors offer production-quality IPv6 implementations for customers interested in experimenting with IPv6 networks.

IPv6 and Steel-Belted Radius

With few exceptions, Steel-Belted Radius supports IPv6 addressing wherever IPv4 addressing is supported. You can perform configuration, authentication, and accounting of RADIUS IPv6 attributes per RFC 3162, *RADIUS and IPv6*. The SBR Administrator (configuration application) and the LDAP configuration interface (LCI) support the configuration of RADIUS IPv6 attributes.

NOTE: *Because many third-party libraries and software development kits (SDKs) do not support IPv6, the Steel-Belted Radius server must support local IPv4 socket connections.*

IPv6 Features

Significant changes from IPv4 to IPv6 include the following:

- Expanded routing and addressing capabilities – IPv6 increases the IP address size from 32 bits to 128 bits. As a consequence, IPv6 supports more levels of addressing hierarchy, provides a much greater number of available addresses, and simplifies auto-configuration of addresses. As a consequence, address conservation techniques such as network address translation, are not necessary.
- Improved multicast routing – Multicast routing, which existed in IPv4, has been redefined and improved in IPv6. Multicast addresses now include a Scope field that limits the scope of multicasts, improving scalability. A new Anycast address type allows you to send a message to the nearest single member of a multicast group.
- Header format simplification – The IPv6 packet header format has been designed to be efficient. The IPv6 header has a fixed length of 40 bytes, divided into eight fields.
- Improved support for extensions and options – IPv6 uses extension headers, which are inserted between the IPv6 header and the transport header and packet payload, to handle special packet processing requirements. Extension headers provide a flexible means to support authentication, encryption, fragmentation, source routing, network management, and other functions. An IPv6 packet can carry any number of extension headers.
- Improved datagram sizing and fragmentation – The maximum transmission unit (MTU) describes the maximum size of a datagram that can be transmitted over a network without fragmentation. IPv6 increases the minimum MTU from 576 bytes to 1280 bytes, which makes IPv6 packets more efficient and reduces the need for packet fragmentation. Path MTU discovery enables source routers to determine the appropriate packet size for a route.
- Quality-of-Service (QoS) functions – Packets belonging to a traffic flow that requires special handling, such as real-time video service, can be labeled by the sender. Because traffic in a particular flow can be identified in the IPv6 header, support for QoS can be implemented even when the payload of a packet is encrypted.
- Improved privacy and security – IPv6 supports extensions for authentication and data integrity to improve security and privacy of network traffic.

IPv6 Addressing

IPv6 addresses are 128 bits in length, which creates a much larger address space than 32-bit IPv4 addresses. IPv6 addresses identify individual interfaces and sets of interfaces. IPv6 addressing architecture is defined in RFC 2373, *IP Version 6 Addressing Architecture.*

Address Notation

In full form, IPv6 addresses are written as eight 16-bit hexadecimal blocks separated by colons:

```
FE80:0000:0000:0000:1232:E4BF:FE1A:8324
```

IPv6 addresses can be interpreted as having two variable-length fields: an IPv6 prefix and an IPv6 interface identifier.

- The IPv6 prefix varies from 0 to 128 bits in length and forms the routable network number portion of the IPv6 address. The trailing CIDR notation that may appear after human-readable IPv6 addresses (for example, /64) indicates the bit length of the IPv6 prefix.
- The IPv6 interface identifier consists of the non-prefix portion of the IPv6 address, if any, and identifies the host interface portion of the address, which identifies an IPv6 interface on the local network. The IPv6 interface identifier is typically generated automatically by the host as a function of a unique hardware identifier, such as an Ethernet MAC address. IPv6 hosts can automatically configure interface addresses by combining the IPv6 prefixes obtained from router advertisements with the IPv6 interface IDs that are determined locally.

For example:

```
IPv6 Prefix: FEC0:0000:0000:0000:0000:0000:0000:0000/64
IPv6 Interface ID:                0260:08FF:FEFF:FFFF
IPv6 Address: FEC0:0000:0000:0000:0260:08FF:FEFF:FFFF
```

To simplify address notation, IPv6 accepts abbreviations in address notation. For example, leading zeros in a 16-bit block may be omitted:

```
FE80:0:0:0:0232:E4BF:FE1A:8324
```

A double colon (::) can replace a series of consecutive zeros within an address:

```
FE80::232:E4BF:FE1A:8324
```

Only one double colon can be used to compress an IPv6 address. If more than one double colon was included in an address, networking devices would not know how many zeros to insert for each double colon when expanding a compressed address to its full 128-bit representation.

In networks that support IPv4 and IPv6 nodes, IPv4 addresses can be embedded in the last four bytes of the address. An IPv4 address of 192.168.1.12 can be represented in IPv6 format as `::192.168.1.12`, where `::` represents a string of zeroes to pad the address to 128 bits.

Address Types

IPv6 introduces a number of special address types. Steel-Belted Radius accepts all equivalent forms of IPv6 addresses and recognizes them as being equivalent. Steel-Belted Radius usually displays an IPv6 address in abbreviated form, though IPv6 addresses should always be written out in full and unabbreviated form when wild cards are used. Embedded IPv4 addresses must also be represented in unabbreviated hexadecimal form when wild cards are used.

NOTE: *The use of wild cards in IPv6 addresses is an experimental Steel-Belted Radius feature, and is not known to be documented or prohibited by any IPv6 standards at this time.*

Among the types of address introduced for IPv6 addresses are the following:

- The **IPv6 unspecified address** (`::`) is a special address consisting entirely of zeros. The IPv6 unspecified address is analogous to the IPv4 unspecified address 0.0.0.0. You can use the unspecified address when any inbound or outbound interface is acceptable for performing a network transport function. If an inbound unspecified address is used, the server listens on all available local interfaces. If an outbound unspecified address is used, the server attempts to select an appropriate interface. The unspecified address is not routable on the larger network; a server must convert it to a routable address.
- The **IPv6 loopback address** (`::1`) consists of a series of zeros except for the last bit, which has a value of 1. The IPv6 loopback address, which is analogous to the IPv4 loopback address (127.0.0.1), is used to create a local network connection in memory (without involving any networking hardware), in cases where a physical network connection is not desired, in cases where the local IP address has not yet been assigned, in cases where the local IP address is not yet known, and in cases involving trouble-shooting. The loopback address is not routable on the larger network.
- **IPv6 link local unicast addresses** are used to enable IPv6 on the local network segment without requiring any IPv6 routers or IPv6 services such as Domain Name Service (DNS) or Dynamic Host Configuration Protocol (DHCP). The first 10 bits of an IPv6 link local unicast address is always 1111 1110 00 (`fe80::/10 in hexadecimal form`). The next 54 bits is usually zeros, followed by an interface identifier (derived from a device's MAC address in the case of Ethernet adapters). Link local addresses can be used to contact IPv6 nodes on the local network segment. Routers do not forward packets that contain link local addresses, so link local addresses cannot be used to contact IPv6 nodes beyond the local network segment.

  Link local addresses are best suited for network discovery mechanisms at OS boot time. The RFCs discourage use of link local applications in other applications, since an automatically generated link local address may not be unique beyond the local network segment.
- **IPv6 site local unicast addresses** are used to communicate with other IPv6 nodes on a local area network. The first 10 bits of an IPv6 site local unicast address is always 1111 1110 11 (`fec::/10 in hexadecimal form`). The next 38 bits is usually zeros, followed by a 16-bit subnet ID field. While site local addresses can be used to contact IPv6 nodes on the local area network (as well as nodes on the local network segment), site local addresses cannot be used to contact IPv6 nodes outside the local area network.

  Site local addresses can be non-unique in situations where a multi-homed host is connected with more than one unrelated network. In such cases, a scope ID must be used to disambiguate non-unique site local addresses. For information on scope IDs, see "Scope IDs" on page 274. Site local addressing is likely to be deprecated as the IPv6 standard evolves.

- **IPv6 aggregatable global unicast addresses** (for example, `2*:*:*:*:*:*:*:*` or `3*:*:*:*:*:*:*:*`) are used to communicate with other IPv6 nodes on the Internet. Global addresses are often derived from a prototypical router address or dynamically assigned using IPv6 services such as DHCP. Global addresses can be used to contact any reachable IPv6 node on the Internet. However, not all IPv6 nodes are reachable, since some nodes use only site local or link local addresses within private networks.
- **IPv6 IPv4 mapped addresses** (for example, `::ffff:192.168.10.12`) are used by IPv6-enabled applications to communicate with IPv4 nodes transparently through the IPv6 stack by means of IPv6 APIs. The IPv4 address of the IPv4 node is embedded in the last 32 bits of the IPv6 IPv4 mapped address. IPv6 IPv4 mapped addresses are routable on the Internet.

  The IPv6 stack should convert the mapped address into an IPv4 address and pass the communication through the IPv4 stack. However, IPv6 IPv4 mapped addresses are not supported on all platforms (for example, Windows XP), and this address type may be deprecated as the IPv6 standard evolves.
- **IPv6 IPv4 compatible addresses** (for example, `::192.168.21.45`) are used by IPv6-enabled applications to tunnel IPv6 packets over an IPv4 routing infrastructure dynamically. The actual IPv4 address of the IPv4 node that supports a tunnel end-point is embedded in the last 32 bits of the IPv6 IPv4 compatible address. IPv6 IPv4 compatible addresses are routable on the Internet.
- **IPv6 multicast addresses** (for example, `ff*:*:*:*:*:*:*:*`) are used to transmit data simultaneously to multiple IPv6 nodes. Receiving nodes subscribe to routers to receive multicast transmissions. In turn, downstream routers subscribe to upstream routers until the transmitting application is reached as the source. This address type is routable on the Internet.
- **IPv6 6to4 addresses** (for example, `2002:*:*:*:*:*:*:*`) are used to provide tunneled unicast connectivity between IPv6 sites and nodes across the IPv6 Internet by treating the wide area IPv4 network as a unicast point-to-point link layer. Hosts using 6-to4 addressing do not require special configuration; rather, 6to4 routers encapsulate and decapsulate IPv6 traffic in IPv4 packets.

  For more information on IPv6 6to4 addresses, see RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds."
- **IPv6 ISATAP (Intra-Site Automatic Tunnel Addressing Protocol) addresses** allow IPv6-in-IPv4 tunnels to be created automatically within a site. ISATAP addressing provides unicast IPv6 connectivity between IPv6 hosts across an IPv4 intranet. ISATAP treats a site's IPv4 infrastructure as a Non-Broadcast Multiple Access (NBMA) link layer. ISATAP addresses use the locally administered interface identifier `::0:5EFE:`*`w.x.y.z`*, where:
  - The `0:5EFE` portion is formed from the combination of the Organizational Unit Identifier (OUI) that is assigned to the Internet Assigned Numbers Authority (IANA) (`00-00-5E`) and a type that indicates an embedded IPv4 address (`FE`).
  - The *`w.x.y.`*`z` portion is any unicast IPv4 address, which includes both public and private addresses.

The ISATAP interface identifier can be combined with any 64-bit prefix (including 6to4 prefixes) for IPv6 unicast addresses.

- **IPv6 Teredo addresses** allow nodes located behind an IPv4 NAT to obtain IPv6 unicast connectivity by tunneling packets over UDP/IPv4. Running the service requires the help of Teredo servers and *Teredo relays*:
  - Teredo servers are stateless, and only have to manage a small fraction of the traffic between Teredo clients.
  - Teredo relays act as IPv6 routers between the Teredo service and the "native" IPv6 Internet; the relays can also provide interoperability with hosts using other transition mechanisms such as "6to4."

  If the NAT supports UDP port translation, the NAT typically supports Teredo addressing.

Scope IDs

Some IPv6 address types are designed to be globally unique, while other types of addresses, such as link local addresses or site local addresses, are limited to a local area network or a local network segment. This creates a potential ambiguity with respect to choice of network interface, especially on multi-homed hosts. For example, a local host might be assigned the same link local address on multiple network interfaces connected to different subnets. In such cases, an extra parameter, called a scope ID or zone ID, must be supplied by the application to disambiguate the addresses.

The scope ID is a non-negative integer that is attached to an IPv6 address to identify a particular network interface on a local host. For link local addresses, the scope ID is the interface number, as displayed in the `ipv6 if` command. For site-local addresses, the scope ID is the site number, as displayed in the `ipv6 if` command. For example, the following address has a scope ID of 4:

```
FE80:0:0:0:260:8FF:FEFF:FFFF%4
```

Note that a percent sign (%) is used as a delimiter between the IPv6 address and the scope ID.

At present, scope IDs are used on Windows hosts, while Solaris and Linux continue to use traditional means of specifying a particular network interface.

Address Prefixes

Like IPv4 addresses, IPv6 addresses are composed of a routable network number, known as the *IPv6 prefix*, and a host identifier, known as the *IPv6 interface ID*. IPv6 does not support address classes; IPv6 uses Classless Inter-Domain Routing with variable length network numbers, or prefixes, meaning that an IPv6 prefix is specified by supplying a bit length in conjunction with the address.

IPv6 prefixes are written as an IPv6 address, followed by a slash and the bit length of the prefix portion of the address. The prefix can be 0–128 bits in length, but the prefix is always written in terms of a 128-bit address. When writing prefixes, the trailing bits of the address comprising the interface ID are sometimes dropped so that the prefix can be

abbreviated. The following prefixes are equivalent, assuming that the interface ID portion of the address may be ignored:

Canonical form: `2001:1c44:820d:eea0:0260:08ff:feff:ffff/64`

Abbreviated form: `2001:1c44:820d:eea0::/64`

In many cases, the interface ID portion of the address contains relevant information. A hierarchy of prefixes may reflect the assignment and reassignment of blocks of addresses to progressively smaller organizations. For example, in a typical hierarchy, the largest service providers are assigned the largest blocks of addresses and hence the shortest prefixes, called *Top Level Aggregator Identifiers (TLA IDs)*. The large service providers reassign blocks of addresses to smaller service providers by adding a few more bits after the TLA ID; these added bits are known as *Next Level Aggregator IDs*. The smaller service providers again reassign smaller blocks of addresses to end-user organizations by again adding a few more bits after the NLA ID. These added bits are known as *Site Level Aggregator IDs*. The hierarchy continues in this way until the prefix is exhausted, leaving only the trailing bits that correspond to the non-routable IPv6 interface ID.

Steel-Belted Radius accepts all equivalent forms of IPv6 prefixes and recognizes them as being equivalent. The following prefixes are related by hierarchy, but only the canonical and abbreviated forms are equivalent:

Canonical form: `2001:1c44:820d:eea0:0260:08ff:feff:ffff/64`

Abbreviated form: `2001:1c44:820d:eea0::/64`

Site level prefix: `2001:1c44:820d::/48`

Next level prefix: `2001:1c44:8200::/40`

Top level prefix: `2001:1c00::/24`

IPv6 prefixes should always be written out in full and unabbreviated form when wild cards are used, as the abbreviations become ambiguous in the presence of wild cards. The following prefixes are equivalent:

Canonical form: `2001:1c44:820d:eea0:0260:08ff:feff:ffff/64`

With wild cards: `2001:1c*:??0d:eea0:*:*:*:*/64`

Address Interface IDs

Because the overall size of the IPv6 address is fixed, a longer address prefix means a shorter interface ID. For example, a 48-bit prefix implies an 80-bit interface ID:

Canonical prefix: `2001:1c44:820d:eea0:0260:08ff:0000:0000/48`

Abbreviated prefix: `2001:1c44:820d::/48`

48-bit interface ID: `eea0:0260:08ff:0000:0000`

Though the interface ID portion of an IPv6 address can be 0–128 bits in length, the RADIUS standard assumes 64-bit interface IDs. Steel-Belted Radius uses a convention that all interface IDs are written as a series of four unabbreviated hexadecimal fields regardless of how they are entered:

64-bit interface ID: `0260:08ff:0000:0000`

IPv6 interface IDs should always be written out in full and unabbreviated form when wild cards are used, as the abbreviations become ambiguous in the presence of wild cards. The following interface IDs are equivalent:

64-bit interface ID: `0260:08ff:0000:0000`

With wild cards: `02??:08ff:*:*`

NOTE: *The use of wild cards in IPv6 interface IDs is a Steel-Belted Radius feature. Wildcards in IPv6 interface IDs are not known to be documented or prohibited by any IPv6 standards at this time.*

IPv6 Network Numbers

In very specific cases, such as checklist processing, Steel-Belted Radius recognizes both IPv4 and IPv6 addresses as representing entire *ranges* of addresses. Steel-Belted Radius extends the concept of IPv4 network numbers to IPv6 as a means of representing a range of network addresses. Note that using this concept of network numbers means you cannot specify a valid network address that also happens to be a network number.

Prior to the adoption of Classless Inter-Domain Routing (CIDR), the IPv4 address space is divided into five address classes, as shown in Table 49.

Table 49. *IPv4 Address Classes*

Class	Address Range	Description
A	0.0.0.0 – 127.255.255.255	1-bit class, 7-bit network, 24-bit host
B	128.0.0.0 – 191.255.255.255	2-bit class, 14-bit network, 16-bit host
C	192.0.0.0 – 223.255.255.255	3-bit class, 21-bit network, 8-bit host
D	224.0.0.0 – 239.255.255.255	4-bit class, 28-bit multicast group
E	240.0.0.0 – 247.255.255.255	5-bit class, 27-bit reserved

Within an IPv4 address class, each network is identified by a network number that consists of the leading class bits and the network bits that follow. Network numbers are typically written as IP addresses with trailing zero bits; for example, . the network number corresponding to the class C address 199.100.10.24 would typically be written as 199.100.10.0.

Each network represents a potential physical interconnection of a maximum number of hosts determined by the number of host bits. Thus, the physical network identified by the network number 199.100.10.0 might connect up to 256 hosts identified by the addresses 199.100.10.0 through 199.100.10.255 inclusive. (In practice, host addresses such as 199.100.10.0 are avoided to prevent confusion between host addresses and network numbers.) Thus, it is reasonable to interpret a network number as the entire range of addresses sharing the same network portion of the address:

Network number: `199.100.10.0`

Start of address range: `199.100.10.0`

End of address range: `199.100.10.255`

As a wild carded address: `199.100.10.*`

To see how the concept of network numbers can be extended to IPv6 addresses, consider that IPv6 addresses can contain embedded IPv4 addresses. The IPv6 address `::ffff:199.100.10.0` can therefore be interpreted as the range `::ffff:199.100.10.0` through `::ffff:199.100.10.255` inclusive.

The IPv6 address space is not divided into classes, because IPv6 was designed with CIDR in mind. Constructing an arbitrary definition of IPv6 network numbers that both resembles IPv4 and scales well across all possible IPv6 addresses is difficult. However, since large portions of the IPv6 address space have not yet been defined and since the RADIUS specification concerns itself only with 64-bit interface IDs, we can consider arbitrarily assigning special meaning to all IPv6 addresses ending in 64 bits of zero. This represents a fraction ($1/2^{64}$) of the IPv6 address space, where the addresses have arbitrarily been assigned special meaning overriding their true meaning. The cases where this arbitrary definition would cause trouble are expected to be extremely rare, and it should be possible to avoid them.

Steel-Belted Radius artificially defines the concept of IPv6 network numbers as IPv6 addresses ending in 64 bits of zero, where the network number is interpreted as the entire range of IPv6 addresses sharing the same 64-bit prefix as the network number:

Network number: `2001:1c44:820d:eea0:0000:0000:0000:0000`

Start of address range: `2001:1c44:820d:eea0:0000:0000:0000:0000`

End of address range: `2001:1c44:820d:eea0:ffff:ffff:ffff:ffff`

As a wild carded address: `2001:1c44:820d:eea0:*:*:*:*`

IPv6 Support in Steel-Belted Radius

In general, IPv6 support in Steel-Belted Radius parallels IPv4 support, both in terms of IPv6 network transport and in terms of RADIUS IPv6 attributes. When IPv6 networking is not supported by an operating system (either because the operating system cannot support IPv6 or because the IPv6 stack for the operating system has not been configured), Steel-Belted Radius recognizes IPv6 addresses and attributes, but does not use IPv6 transport mechanisms.

Socket interfaces in Steel-Belted Radius are both IPv4- and IPv6-capable. By default, IPv4 support is enabled and IPv6 support is disabled in Steel-Belted Radius. You must explicitly enable IPv6 support (by modifying settings in the [IPv6] section of the `radius.ini` file) before you can use IPv6 networking. Steel-Belted Radius recognizes IPv6 attributes whether or not IPv6 networking is enabled.

With few exceptions, IPv6 addresses may be configured wherever you can configure IPv4 addresses in configuration files and in the SBR Administrator.

Similarly, IPv6 RADIUS attributes can be configured wherever IPv4 RADIUS attributes can be configured. All IPv6 attributes are defined in the `radius.dct` file to allow inclusion in all standards-conforming dictionaries. IPv6 attributes are correctly interpreted and fully validated by the LDAP Configuration Interface (LCI) and by the SBR Administrator.

Table 50 presents a summary of IPv6 support in Steel-Belted Radius.

Table 50. *IPv6 Feature Matrix*

Feature	Supported?	Comments
Server networking	Yes	IPv6 networking must be explicitly enabled. IPv6 attributes can be processed even if IPv6 networking is not enabled.
Server DNSv6	Yes	DNSv6 must be explicitly enabled. Both IPv6 and IPv4 network connections are supported with remote DNSv6 servers. Only IPv4 network connections are supported with remote DNS servers.
Server logs	Yes	The diagnostic logging and tracing of IPv6 network connections and IPv6 attributes are fully supported.
LCI networking	Yes	If IPv6 networking is enabled, IPv6 addresses can be configured in the [LDAPAddresses] section of the `radius.ini` file.
LCI inputs	Yes	In most cases, IPv6 values can be supplied wherever IPv4 inputs can be specified.
Attributes	Yes	Basic IPv6 attributes defined in RFC-3162 and listed in `radius.dct` are supported as native types or as regular text strings, as appropriate.
Checklists	Partial	IPv6 attributes can appear in checklists, and IPv6 address values can contain network numbers similar to IPv4 address values. IPv6 prefix values and IPv6 interface values cannot be masked or wildcarded.
Return lists	Yes	IPv6 attributes can appear within return lists, and IPv6 values can be assigned.
Attribute value pools	Yes	IPv6 attributes can appear in attribute value pools, and IPv6 values can be assigned to implement round-robin return list processing.
Attribute filtering	Yes	IPv6 attributes can appear in filter rules, and IPv6 values can be assigned.
Service type mapping	Yes	IPv6 attributes can appear in a service type mapping, and IPv6 values can be wildcarded similar to IPv4 values. Reliable string comparison of regular expressions requires all values to be expressed in canonical form.
Attribute mapping	Yes	IPv6 attributes can appear in an attribute mapping, and IPv6 values can be wildcarded similar to IPv4 values. Reliable string comparison of regular expressions requires all values to be expressed in canonical form.
Class attribute	Partial	The RADIUS Class attribute cannot contain any IPv6 attributes. You can configure IPv6 addresses in the [Hosts] section of the `spi.ini` file to process class attributes originating from IPv6 network connections.
DHCP	No	The use of IPv6 networking to communicate with any DHCP server is not supported. The allocation of IPv6 addresses obtained from any DHCP server is not supported.

Table 50. *IPv6 Feature Matrix (Continued)*

Feature	Supported?	Comments
IP address pools	No	The allocation of IPv6 addresses from an SBR-managed IP address pool is not supported. However, RFC 3162 provides an attribute, Framed-IPv6-Pool, that allows the RAS to implement an IPv6 address pool.
Networking for authentication	Yes	IPv6 addresses can be configured in the [Addresses] section of the `radius.ini` file.
Authentication logs	Yes	IPv6 attributes are fully supported.
Native User authentication	Yes	IPv6 attributes are fully supported.
Authenticate-Only requests	Yes	IPv6 attributes are fully supported.
Pass-through authentication	Partial	IPv6 attributes are fully supported. However, because many third-party libraries do not support IPv6, IPv6 networking is not necessarily supported with external services such as RSA SecurID and TACACS+.
External authentication (for example, LDAP or SQL)	Partial	IPv6 attributes are fully supported. However, because many third-party libraries do not support IPv6, IPv6 networking is not necessarily supported with external services such as LDAP and SQL.
EAP-TTLS authentication	Yes	IPv6 attributes are fully supported.
Directed authentication	Yes	IPv6 attributes are fully supported.
Networking for accounting	Yes	IPv6 addresses can be configured in the [Addresses] section of the `radius.ini` file.
Accounting logs	Yes	IPv6 attributes are fully supported.
External accounting (for example, SQL)	Partial	IPv6 attributes are fully supported. However, because many third-party libraries do not support IPv6, IPv6 networking is not necessarily supported with external services such as SQL.
Directed accounting	Yes	IPv6 attributes are fully supported.
Spooled accounting	Yes	IPv6 attributes are fully supported.
Networking for proxy	Yes	IPv6 addresses can be configured in the [Interfaces] section of the `proxy.ini` file. Both classic and extended proxy support IPv6.
Proxy authentication	Yes	IPv6 attributes are fully supported. Both classic and extended proxy support IPv6.
Proxy accounting	Yes	IPv6 attributes are fully supported. Both classic and extended proxy support IPv6.
3GPP2	Partial	RFC standards are not sufficiently evolved to enable full support of IPv6. Although IPv6 attributes can be processed, they are not meaningful in the context of 3GPP2.

Table 50. *IPv6 Feature Matrix (Continued)*

Feature	Supported?	Comments
Master SNMP agent	No	(Solaris/Linux only) The use of IPv6 networking to communicate with an IPv6 capable SNMP management station and/or SNMP subagent is not supported.
SNMP subagent	No	(Solaris/Linux only) IPv6 networking, IPv6 trap variables, and IPv6 MIB objects are not supported. IPv6 addresses are reported as IPv4 MIB objects possessing the value 255.255.255.255.
Windows events	No	(Windows only) Neither IPv6 networking nor IPv6 event variables are supported at this time.
Networking for plug-Ins	Yes	Steel-Belted Radius does not control the networking of back end servers with its plug-ins. IPv6 networking is generally a hidden detail of third-party back end server configuration.
Plug-In APIs	Yes	IPv6 features and parameters are exposed in the new plug-in APIs. The older plug-in APIs are deprecated but still functional. You should upgrade to the new plug-in APIs to gain access to IPv6 features. IPv6 APIs can be invoked even if IPv6 networking is not enabled.
Plug-In attributes	Yes	Basic IPv6 attributes defined in RFC-3162 and listed in `radius.dct` are supported as native types or as regular text strings, as appropriate.
Oracle plug-Ins (Solaris)	Partial	(Solaris only) IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
ODBC plug-Ins	Partial	(Windows/Linux only) IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
JDBC plug-Ins	Partial	(Windows/Linux only) IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
LDAP plug-In	Partial	IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
RSA SecurID plug-Ins	Partial	IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
PEAP Plug-In	Partial	IPv6 attributes are fully supported, but the required third party software either does not currently support IPv6 or we have not tested it.
TLS plug-In	Partial	IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
TTLS plug-In	Partial	IPv6 attributes are fully supported, but the required third-party software may not support IPv6.
PAS plug-Ins	Yes	(SPE only) IPv6 attributes are fully supported.
Concurrency plug-Ins	Yes	(SPE only) IPv6 attributes are fully supported.
Uniport Plug-In	Yes	IPv6 attributes are fully supported.

Table 50. IPv6 Feature Matrix (Continued)

Feature	Supported?	Comments
3COM CCA Tunnels (deprecated)	Partial	The use of IPv6 networking is not supported. IPv6 attributes can be processed even if IPv6 networking is not enabled.

RADIUS IPv6 Attributes

All RADIUS attributes defined in RFC 3162, *RADIUS and IPv6*, are supported in Steel-Belted Radius as native types or as regular text strings. All forms of attribute processing, such checklist processing, return list processing, attribute echoing/deleting/merging, are supported. However, IPv6 prefix values and IPv6 interface values cannot be masked or wildcarded in checklist processing.

Third-party plug-ins that have not been upgraded to support IPv6 should be able to process IPv6 attributes as opaque hexadecimal strings.

Table 51 lists the attribute number, and the number of times an attribute can appear in an Access-Request, Access-Accept, Access-Reject, Access-Challenge, and Accounting-Request packets for each type of IPv6 RADIUS attribute.

Table 51. IPv6-Specific RADIUS Attributes

Attribute	Attr Num	Acc-Req	Acc-Accept	Acc-Rej	Acc-Chall	Acct-Req
NAS-IPv6-Address	95	0-1	0	0	0	0-1
Framed-Interface-Id	96	0-1	0-1	0	0	0-1
Framed-IPv6-Prefix	97	0+	0+	0	0	0+
Login-IPv6-Host	98	0+	0+	0	0	0+
Framed-IPv6-Route	99	0	0+	0	0	0+
Framed-IPv6-Pool	100	0	0-1	0	0	0-1

NAS-IPv6-Address

The NAS-IPv6-Address attribute is similar in function to the NAS-IP-Address attribute. Either attribute is sufficient to identify the IP address of the requesting RAS to the RADIUS server. If both attributes appear in the same RADIUS Access-Request packet, Steel-Belted Radius processes the NAS-IPv6-Address attribute for the purpose of identifying the RAS.

The NAS-IPv6-Address attribute may be specified by the RAS in access and accounting request packets. Zero or one NAS-IPv6-Address attributes may be specified. If present, the fixed length NAS-IPv6-Address attribute contains the complete 128-bit IPv6 address of the requesting RAS.

Steel-Belted Radius allows zero or one 128-bit IPv6 address to be specified for each RAS. The server authentication logic validates these addresses on extraction from the database and compares them with NAS-IPv6-Address attributes when they are received

in access and accounting request packets. The server accounting logic writes these addresses to the accounting logs in a human readable format.

Example

```
Human readable:fe80::260:8ff:feff:ffff
RADIUS attribute:5f 12 fe 80 00 00 00 00 00 00 02 60
08 ff fe ff ff ff
```

Framed-Interface-Id

The Framed-Interface-Id attribute specifies the IPv6 interface ID to be assigned to a user. When combined with a Framed-IPv6-Prefix attribute, a single Framed-Interface-Id attribute forms one or more complete IPv6 addresses to be assigned to the user.

In general, the user is assigned the number of addresses equal to the number of Framed-IPv6-Prefix attributes, where the addresses have the same Framed-Interface-Id value and different Framed-IPv6-Prefix values

It is possible to assign complete IPv6 addresses using only Framed-IPv6-Prefix attributes (i.e. without specifying any Framed-Interface-Id attribute). For example, in the case of PPP, it can be quite difficult to automatically generate a unique IPv6 interface ID for a given network segment, so it is recommended that the RADIUS server honor the hint if this attribute is suggested by the RAS. This is typically accomplished with echo attributes).

The Framed-Interface-Id attribute may be specified by the RAS in Access- and Accounting-Request packets, and by the RADIUS server in Access-Accept packets. Zero or one Frame-Interface-Id attributes may be specified. If present, the fixed length Framed-Interface-Id attribute contains the 64-bit interface ID to be assigned to the user.

Steel-Belted Radius allows zero or one 64-bit interface ID to be specified for each native user. The server authentication logic validates this interface ID on extraction from the database and includes the Framed-Interface-Id attribute in Access-Accept packets if none is received in the Access-Request packets.

Example

```
Human readable:260:8ff:feff:ffff
RADIUS attribute:60 0a 02 60 08 ff fe ff ff ff
```

Framed-IPv6-Prefix

The Framed-IPv6-Prefix attribute specifies the IPv6 networks to be assigned to a user. When combined with a Framed-Interface-ID attribute, a single Framed-IPv6-Prefix attribute forms one or more complete IPv6 addresses to be assigned to the user.

In general, the user is assigned the number of addresses equal to the number of Framed-IPv6-Prefix attributes, where the addresses have the same Framed-Interface-Id value, but different Framed-IPv6-Prefix values.

It is possible to assign complete IPv6 addresses using only Framed-IPv6 Prefix attributes (that is, without specifying any Framed-Interface-Id attribute). For example, the Framed-IPv6-Prefix attributes may be suggested by the RAS and overridden by the RADIUS server. In any case, the RAS is expected to be able to plumb the routes implied by the Framed-IPv6-Prefix attributes and these need not be repeated in separate Framed-IPv6-Route attributes.

The Framed-IPv6-Prefix attribute may be specified by the RAS in Access- and Accounting-Request packets, and by the RADIUS server in Access-Accept packets. Zero or more Framed-IPv6-Prefix attributes may be specified. If present, the variable-length Framed-IPv6-Prefix attribute contains an IPv6 prefix from 0 to 128 bits in length. Extra bits beyond the actual prefix length must be set to 0.

Steel-Belted Radius allows zero or more variable-length IPv6 prefixes to be specified for each native user or attribute profile. The server authentication logic validates these prefixes on extraction from the database and include the proper number of Framed-IPv6-Prefix attributes in Access-Accept packets if none are received in the access request packets. The server accounting logic writes these prefixes to the accounting logs in a human readable format.

Example

```
Human readable:fe80::260:8ff:feff:ffff/10
RADIUS attribute:61 14 00 0a fe 80 00 00 00 00 00 00 00
00 00 00 00 00 00 00
(8-bit type) (8-bit length)
(8-bit zero) (8-bit prefix length) (128-bit IPv6 prefix)
```

Login-IPv6-Host

The Login-IPv6-Host attributes specify the IPv6 addresses of the systems with which the user is connected when the Login-Service attribute is also included. The Login-IPv6-Host attribute may be suggested by the RAS and overridden by the RADIUS server.

The Login-IPv6-Host attribute may be specified by the RAS in access and accounting request packets, and by the RADIUS server in Access-Accept packets. Zero or more Login-IPv6-Host attributes may be specified. If present, the fixed length Login-IPv6-Host attribute contains the complete 128-bit IPv6 address of the login host, or a special value:

- 128-bits set to 0 indicates that the RAS should select the login host for the user.
- 128-bits set to 1 indicates that the RAS should allow the user to select the login host.
- Other values indicate the actual 128-bit IPv6 address of the login host.

Steel-Belted Radius allows zero or more 128-bit IPv6 addresses (including special values) to be specified for each native user or attribute profile. The server authentication logic validates these addresses (including special values) on extraction from the database and includes the proper number of Login-IPv6-Host attributes in Access-Accept packets if none are received in the access request packets. The server accounting logic writes these addresses to the accounting logs in a human readable format.

Example

```
Human readable:fe80::260:8ff:feff:ffff
RADIUS attribute:62 12 fe 80 00 00 00 00 00 00 02 60 08 ff fe
ff ff ff
```

Framed-IPv6-Pool

The Framed-IPv6-Pool attribute specifies the name of a RAS managed pool (as opposed to a RADIUS server managed pool) from which the RAS should assign an IPv6 prefix

to the user. The Framed-IPv6-Pool attribute may not be suggested by the RAS and is always determined by the RADIUS server.

The Framed-IPv6-Pool attribute may be specified by the RAS in Accounting-Request packets, and by the RADIUS server in Access-Accept packets. Zero or one Framed-IPv6-Pool attributes may be specified. If present, the variable-length Framed-IPv6-Pool attribute contains the name of a RAS managed pool in human readable text. The text is not NULL terminated.

Steel-Belted Radius allows zero or one variable length pool name to be specified for each native user. The server authentication logic validates the pool name on extraction from the database and includes the proper number of Framed-IPv6-Pool attributes in Access-Accept packets. The server accounting logic writes these pool names to the accounting logs in a human readable format.

Example

```
Human readable:ipv6-pool
RADIUS attribute:64 0b 69 70 76 36 2d 70 6f 6f 6c
```

Framed-IPv6-Route

The Framed-IPv6-Route attribute specifies the IPv6 routing information to be configured for the user on the RAS. The RAS is expected to be able to plumb the routes specified by the Framed-IPv6-Route attributes in addition to those that may already be implied by separate Framed-IPv6-Prefix attributes. The Framed-IPv6-Route attribute may not be suggested by the RAS and is always determined by the RADIUS server.

The Framed-IPv6-Route attribute may be specified by the RAS in accounting request packets, and by the RADIUS server in Access-Accept packets. Zero or more Framed-IPv6-Route attributes may be specified. If present, the variable-length Framed-IPv6-Route attribute contains IPv6 routing information in human readable text. The text is not NULL terminated. The format of the text (destination prefix, gateway address, metrics) is described in RFC-3162.

Steel-Belted Radius allows zero or more variable-length IPv6 routes to be specified for each native user or attribute profile. The server authentication logic validates these routes on extraction from the database and includes the proper number of Framed-IPv6-Route attributes in Access-Accept packets. The server accounting logic writes these routes to the accounting logs in a human readable format.

Example

```
Human readable:2000:0:0:106::/64 2000::106:a00:20ff:fe99:a998
1
RADIUS attribute:63 32 32 30 30 30 3a 30 3a 30 ... 39 3a 61 39
39 38 20 31
```

Configuring IPv6

This section describes how to configure Steel-Belted Radius to use IPv6 networking.

Enabling IPv6 Networking

To enable IPv6 networking in Steel-Belted Radius, you must modify the `radius.ini` file and then restart your Steel-Belted Radius server. For information on the settings in the `radius.ini` file, refer to the *Steel-Belted Radius Reference Guide*.

Note that Steel-Belted Radius can process IPv6 attributes even if IPv6 networking is not enabled, provided that the IPv6 attributes are described in the RADIUS dictionary files.

Configuring IPv6 Scope IDs

Some types of IPv6 addresses requires an IPv6 scope ID to avoid address ambiguity. In some cases, the Steel-Belted Radius server can select a scope ID automatically. You can specify the scope ID explicitly for IPv6 link local and site local addresses.

The [IPv6] section of the `radius.ini` file can specify how scope IDs are selected for each IPv6 address type that is recognized by the server. If the parameter value is 0, the Steel-Belted Radius server selects a scope ID automatically. If the parameter value is not 0, then the Steel-Belted Radius server uses that value as the scope ID when establishing network connections involving that IPv6 address type.

***NOTE:** You can use the output of the `ifconfig -a` shell command on Solaris/Linux platforms, and the output of the `ipconfig /all` shell command on Windows platforms (`ipv6 if` on Windows XP platforms) to determine the proper host specific scope ID for an address type. The scope ID is identical to the interface index on which the address type is supported and on which the desired destinations are reachable. On Solaris/Linux platforms, the server accepts traditional interface names, such as `hme0`, instead of numeric scope IDs.*

Configuring IPv6 Addresses for RADIUS Client Connections

You can configure the [Addresses] section of the `radius.ini` file if you want to specify the local address or addresses on which Steel-Belted Radius listens for RADIUS client connections. By default, Steel-Belted Radius automatically discovers and configures all available IPv4 interfaces on the local host. If IPv6 is enabled, Steel-Belted Radius discovers and configures both IPv4 and IPv6 interfaces.

You can configure Steel-Belted Radius to configure IPv4 automatically by entering `AutoConfigureIPv4` in the [Addresses] section. Similarly, you can configure Steel-Belted Radius to configure IPv6 automatically by entering `AutoConfigureIPv6` in the [Addresses] section. If you configure specific IPv4 or IPv6 addresses, Steel-Belted Radius listens for RADIUS traffic on only those interfaces.

SBR automatically uses site local addresses if autoconfiguration is enabled, and allows site local addresses to be configured manually. Unless a single shared scope ID is also configured for these addresses (that is, unless all site local addresses exist on the same subnet), SBR tries to determine the appropriate scope ID for any given site local address automatically.

The IPv6 unspecified address `::` allows connections on any IPv6 address or IPv4 address, with IPv4 connections represented as IPv6 IPv4 mapped addresses. Because IPv6 IPv4 mapped addresses are not currently supported by the Windows IPv6 protocol

stack, you must enter the IPv4 unspecified address `0.0.0.0` with the IPv6 unspecified address `::` to approximate the desired behavior on Windows platforms.

Steel-Belted Radius does not use link local addresses unless they are configured explicitly. To use link local addresses, you must configure a nonzero IPv6LinkLocalUnicastScopeId in the [IPv6] section of the `radius.ini` file. If configured, all link local addresses must exist on the same network interface and use the same scope ID.

Configuring DNSv6 Support

Enabling Domain Name Service Version 6 (DNSv6) support lets Steel-Belted Radius communicate with a DNSv6 server to resolve host names. By default, Steel-Belted Radius uses DNS unless IPv6 is enabled and DNSv6 support is configured by means of the DynamicNameResolution parameter in the [IPv6] section of the `radius.ini` file.

- If DynamicNameResolution is set to 0, Steel-Belted Radius uses IPv4 DNS, which means it does not query DNSv6 services.
- If DynamicNameResolution is set to 1, Steel-Belted Radius uses IPv6 DNS (DNSv6), which means it does not query IPv4 DNS services and ignores IPv4-specific information returned by DNSv6 services.
- If DynamicNameResolution is set to 2, Steel-Belted Radius queries DNSv6 services for IPv6-specific information, and then queries IPv4 DNS services for IPv4 specific information if DNSv6 fails to resolve a host name.

Appendix G
Stopping and Starting Steel-Belted Radius

This appendix describes how to stop and restart the Steel-Belted Radius server.

Stopping the Steel-Belted Radius Server

After the Steel-Belted Radius service (Windows) or RADIUS daemon (Solaris/Linux) is installed, it stops and starts automatically each time you shut down or restart the server. If you modify the settings in the Steel-Belted Radius configuration files, you may need to restart the Steel-Belted Radius server manually before the server recognizes its new settings.

Windows

You can stop the Steel-Belted Radius service at any time by performing the following steps:

1. Choose **Start > Control Panel > Administrative Tools > Services**.
2. When the Services window (Figure 119) opens, click the Steel-Belted Radius entry.

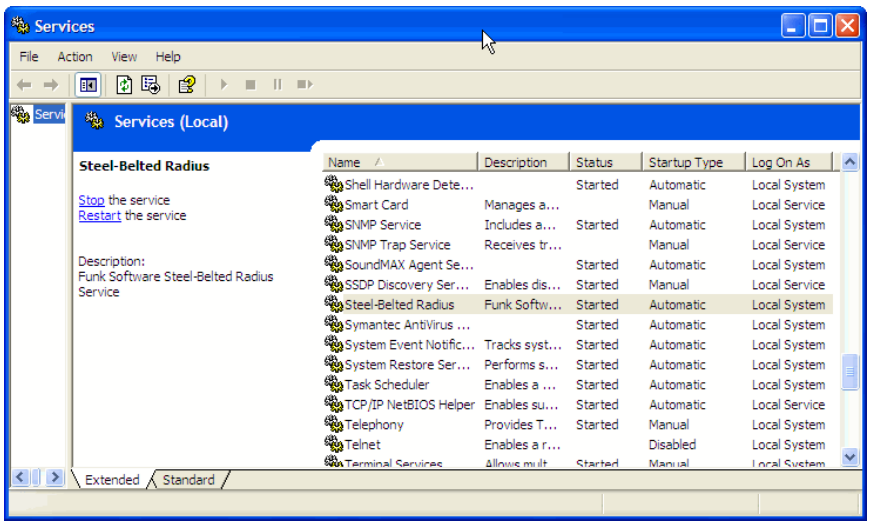


Figure 119 *Services Window*

3 Click the **Stop the service** button.

Solaris

You can stop the RADIUS daemon on a Solaris server at any time by issuing the following commands:

```
cd server-directory
./sbrd stop
```

Linux

After the RADIUS daemon is installed on the server, it stops and starts automatically each time you shut down or restart the server. You can stop the RADIUS daemon on a Linux server at any time by issuing the following commands:

```
cd server-directory
./sbrd stop
```

When you execute the `sbrd stop` command, Steel-Belted Radius allows its subsystems to complete outstanding work, release resources, and then stops the `mkded` (btrieve) daemon and the `radius` service gracefully.

If Steel-Belted Radius fails to stop after you issue an `sbrd stop` command, you can use the optional `force` argument to terminate all subsystems immediately.

```
cd server-directory
./sbrd stop force
```

Starting the Steel-Belted Radius Server

You must restart the Steel-Belted Radius service (Windows) or RADIUS daemon (Solaris/Linux) after you modify the Steel-Belted Radius configuration files.

Windows

To start the Steel-Belted Radius service on a Windows server after it has been stopped:

1. Choose **Start > Control Panel > Administrative Tools > Services**.
2. When the Services window (Figure 119, "Services Window," on page 288) opens, click the Steel-Belted Radius entry.
3. Click the **Start the service** button.

To restart the Steel-Belted Radius service without stopping it:

1. Choose **Start > Control Panel > Administrative Tools > Services**.
2. When the Services window (Figure 119, "Services Window," on page 288) opens, click the Steel-Belted Radius entry.
3. Click the **Restart the service** button.

Solaris

Use the following commands to start the RADIUS daemon on a Solaris server:

```
cd server-directory
./sbrd start
```

Linux

Use the following command to start the RADIUS daemon after you have issued an `sbrd stop` command on a Linux server:

```
cd server-directory
./sbrd start
```

When you execute the `sbrd start` command, Steel-Belted Radius starts the `mkded` (btrieve) daemon to allow database access, and then starts the radius service.

If you change configuration settings for your Steel-Belted Radius server, you may need to restart the server to make the changes effective. As an alternative to issuing an `sbrd stop` command immediately followed by an `sbrd start` command, you can use the `sbrd restart` command when you want to restart Steel-Belted Radius. When you issue the `sbrd restart` command, the system shuts down the `mkded` (btrieve) daemon and the `radius` service (if they are running), and then immediately starts the `mkded` (btrieve) daemon and the `radius` service.

```
cd server-directory
./sbrd restart
```

Displaying RADIUS Status Information (Linux)

You can use the `sbrd status` command to display status information for the RADIUS daemon.

```
cd server-directory
./sbrd status
```

Figure 120 illustrates the output of the `sbrd status` command.

```
> sbrd status
ecarter 25927 .mkded start
btrieve processes are active

------- Shared Memory Segments ------
keyshmidownerpermsbytesnattchstatus
0x42545256891968ecarter60080000002

------- Semaphore Arrays ------
keysemidownerpermsnsems
0x42545256167116ecarter660250

btrieve shared IPC objects exist
btrieve state is running
btrieve status 1101

ecarter 26066 radius
-d/home/ecarter/sbr/5.0.5.1553/funk/radius sbr.xml
radius processes are running
radius state is running
radius status 1101

Aggregate state is running
```

***Figure 120** Output of `sbrd status` Command*

Index

Symbols

A

B

C

D

E

F

H

I

L

M

N

O

P

R

S

T

U

V