JUNIPER NETWORKS

Volume Usage Control with the SRC VTA

Published: 2012-07-18

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

Volume Usage Control with the SRC VTA

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

List of Figures

List of Tables

About the Documentation

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at http://www.juniper.net/techpubs/.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at http://www.juniper.net/books .

Supported Platforms

For the features described in this document, the following platforms are supported:

- C Series

Documentation Conventions

Table 1 on page xii defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Documentation Conventions

Table 1 on page xii defines the notice icons used in this guide. Table 3 on page xii defines text conventions used throughout this documentation.

Table 2: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Table 3: Text Conventions

Convention	Description	Examples
Bold text like this	• Represents keywords, scripts, and tools in text. • Represents a GUI element that the user selects, clicks, checks, or clears.	• Specify the keyword **exp-msg**. • Run the **install.sh** script. • Use the **pkgadd** tool. • To cancel the configuration, click **Cancel**.

Table 3: Text Conventions *(continued)*

Bold text like this	Represents text that the user must type.	**user@host# set cache-entry-age** *cache-entry-age*

`Fixed-width text like this`	Represents information as displayed on your terminal's screen, such as CLI commands in output displays.	`nic-locators {`
		`  login {`
		`  resolution {`
		`  resolver-name /realms/`
		`  login/A1;`
		`  key-type LoginName;`
		`  value-type SaeId;`
		`  }`

Regular sans serif typeface	• Represents configuration statements. • Indicates SRC CLI commands and options in text. • Represents examples in procedures. • Represents URLs.	• **system ldap server{ stand-alone;** • Use the **request sae modify device failover** command with the **force** option • **user@host# . . .** • http://www.juniper.net/techpubs/software/management/sdx/api-index.html
Italic sans serif typeface	Represents variables in SRC CLI commands.	**user@host# set local-address** ***local-address***
Angle brackets	In text descriptions, indicate optional keywords or variables.	Another runtime variable is <gfwif>.
Key name	Indicates the name of a key on the keyboard.	Press Enter.
Key names linked with a plus sign (+)	Indicates that you must press two or more keys simultaneously.	Press Ctrl + b.
Italic typeface	• Emphasizes words. • Identifies book names. • Identifies distinguished names. • Identifies files, directories, and paths in text but not in command examples.	• There are two levels of access: *user* and *privileged*. • *SRC-PE Getting Started Guide.* • *o=Users, o=UMC* • The */etc/default.properties* file.
Backslash	At the end of a line, indicates that the text wraps to the next line.	Plugin.radiusAcct-1.class=\ net.juniper.smgt.sae.plugin\ RadiusTrackingPluginEvent
Words separated by the | symbol	Represent a choice to select one keyword or variable to the left or right of this symbol. (The keyword or variable may be either optional or required.)	diagnostic | line

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to **techpubs-comments@juniper.net**, or fill out the documentation feedback form at

https://www.juniper.net/cgi-bin/docbugreport/ . If you are using e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable)

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf .
- Product warranties—For product warranty information, visit http://www.juniper.net/support/warranty/ .
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: http://www.juniper.net/customers/support/
- Search for known bugs: http://www2.juniper.net/kb/
- Find product documentation: http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base: http://kb.juniper.net/
- Download the latest versions of software and review release notes: http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://www.juniper.net/alerts/
- Join and participate in the Juniper Networks Community Forum: http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Management tool: http://www.juniper.net/cm/

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://tools.juniper.net/SerialNumberEntitlementSearch/

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at http://www.juniper.net/cm/ .
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see http://www.juniper.net/support/requesting-support.html .

PART 1

Overview

CHAPTER 1

Software Features Overview

SRC Component Overview

The SRC software is a dynamic system. It contains many components that you use to build a subscriber management environment. You can use these tools to customize and extend the SRC software for your use and to integrate the SRC software with other systems. The SRC software also provides the operating system and management tools for C Series Controllers.

Table 4 on page 3 gives a brief description of the components that make up the SRC software.

Table 4: Descriptions of SRC Components

Component	Description
Server Components	
Service activation engine (SAE)	• Authorizes, activates, and deactivates subscriber and service sessions by interacting with systems such as Juniper Networks routers, cable modem termination system (CMTS) devices, RADIUS servers, and directories. • Collects accounting information about subscribers and services from routers, and stores the information in RADIUS accounting servers, flat files, and other accounting databases. • Provides plug-ins and application programming interfaces (APIs) for starting and stopping subscriber and service sessions and for integrating with systems that authorize subscriber actions and track resource usage.
Subscriber Information Collector (SIC)	Used in conjunction with the MX Series router running the packet-triggered subscribers and policy control (PTSP) solution, the SIC listens for RADIUS accounting events from IP edge devices (accounting clients) and stores them in the Session State Registrar (SSR), or forwards them to a remote AAA server, allowing the SRC software to gain increased subscriber awareness. Additionally, the SIC can optionally edit accounting events before routing them.
Juniper Policy Server (JPS)	Acts as a policy decision point (PDP) and policy enforcement point (PEP) that manages the relationships between application managers and CMTS devices in a PCMM environment.

Table 4: Descriptions of SRC Components *(continued)*

Component	Description
Network information collector (NIC)	Collects information about the state of the network and can provide a mapping from a given type of network data to another type of network data.
Redirect Server	Redirects HTTP requests received from IP Filter to a captive portal page.
3GPP Gateway	The SRC Third-Generation Partnership Project (3GPP) gateway is a Diameter-based component in the SRC software, which provides integration with 3GPP Policy and Charging Control environments, to provide fixed-mobile convergence (FMC). The SRC 3GPP gateway provides Gx-based integration with the Policy and Charging Rules Function (PCRF). The SRC 3GPP gateway uses the Gx interface to mediate between the PCRF and Juniper Networks routers like the E Series Broadband Services routers and MX Series routers. The Gx interface on the SRC 3GPP gateway communicates with the PCRF using the Diameter protocol.
Web Application Service	The SRC software includes a Web application server that hosts the Web Services Gateway and the Volume Tracking Application (SRC VTA). In production environments, this application server is designed to host only these applications. However, you can load your own applications into this server for testing or demonstration purposes.
Web Services Gateway	Allows a gateway client—an application that is not part of the SRC network—to interact with SRC components through a Simple Object Access Protocol (SOAP) interface. The Web Services Gateway provides the Dynamic Service Activator which allows a gateway client to dynamically activate and deactivate SRC services for subscribers and to run scripts that manage the SAE.
Repository	
Directory	The SRC software includes the Juniper Networks database, which is a built-in Lightweight Directory Access Protocol (LDAP) directory for storing all SRC data including services, policies, and small subscriber databases. For large subscriber databases, you must supply your own directory.
Session State Registrar (SSR)	The SSR is a stateless, highly reliable and highly available database cluster. When used in conjunction with an MX Series router running the packet-triggered subscribers and policy control (PTSP) solution, the SSR stores the IP edge attachment subscriber sessions data learned from IP edge devices in the centralized SSR database.
SRC Configuration and Management Tools	

Table 4: Descriptions of SRC Components *(continued)*

Component	Description
SRC command line interface (CLI)	Provides a way to configure the SRC software on a C Series Controller from a Junos OS–like CLI. The SRC CLI includes the policies, services, and subscribers CLI, which has separate access privileges.
C-Web interface	Provides a way to configure, monitor, and manage the SRC software on a C Series Controller through a Web browser. The C-Web interface includes a policies, services, and subscribers component, which has separate access privileges.
Simple Network Management Protocol (SNMP) agent	Monitors system performance and availability. It runs on all the SRC hosts and makes management information available through SNMP tables and sends notifications by means of SNMP traps.
Service Management Applications (Run on external system)	
IMS Services Gateway	Integrates into an IP multimedia system (IMS) environment. The SRC software provides a Diameter protocol-based interface that allows the SRC software to integrate with services found on the application layer of IMS.
SRC Programming Interfaces	
NETCONF API	Allows you to configure or request information from the NETCONF server on a C Series Controller that runs the SRC software. Applications developed with the NETCONF API run on a system other than a C Series Controller.
CORBA plug-in service provider interface (SPI)	Tracks sessions and enables linking the rest of the service provider's operations support system (OSS) with the SRC software so that the OSS can be notified of events in the life cycle of SAE sessions. Hosted plug-ins only.
CORBA remote API	Provides remote access to the SAE core API. Applications that use these extensions to the SRC software run on a system other than a C Series Controller.
NIC access API	Performs NIC resolutions. Applications that use these extensions to the SRC software run on a system other than a C Series Controller.
SAE core API	Controls the behavior of the SRC software. Applications that use these extensions to the SRC software run on a system other than a C Series Controller.

Table 4: Descriptions of SRC Components *(continued)*

Component	Description
Script services	Provides an interface to call scripts that supply custom services such as provisioning policies on a number of systems across a network.
VTA API	The Volume Tracking Application (VTA) API is a Simple Object Access Protocol (SOAP) interface that allows developers to create gateway clients and that administrators use to manage VTA subscribers and sessions. The SRC Web Services Gateway allows a gateway client—an application that is not part of the SRC network—to interact with SRC components, such as the VTA, through a SOAP interface.
Authorization and Accounting Applications	
AAA RADIUS servers	Authenticates subscribers and authorizes their access to the requested system or service. Accepts accounting data—time active and volume of data sent—about subscriber and service sessions. RADIUS servers run on a system other than a C Series Controller.
SRC Admission Control Plug-In (SRC ACP)	Authorizes and tracks subscribers' use of network resources associated with services that the SRC application manages.
Flat file accounting	Stores tracking data to accounting flat files that can be made available to external systems that send the data to a rating and billing system.
Volume Tracking Application	The SRC Volume Tracking Application (SRC VTA) is an SRC component that allows service providers to track and control the network usage of subscribers and services. You can control volume and time usage on a per-subscriber or per-service basis. This level of control means that service providers can offer tiered services that use volume as a metric, while also controlling abusive subscribers and applications. When a subscriber or service exceeds bandwidth limits (or quotas), the SRC VTA can take actions including imposing rate limits on traffic, sending an e-mail notification, or charging extra for additional bandwidth consumed.
Demonstration Applications (available on the Juniper Networks Web site)	
Enterprise Audit Plug-In	Defines a callback interface, which receives events when IT managers complete specified operations.
Enterprise Manager Portal	Allows service providers to provision services for enterprise subscribers on routers running JunosE or Junos OS and allows IT managers to manage services. Enterprise Manager Portal can be used with NAT Address Management Portal to allow service providers to manage public IP addresses for use with NAT services on routers running Junos OS and to all IT managers to make requests about public IP addresses through the Enterprise Manager Portal.

Table 4: Descriptions of SRC Components *(continued)*

Component	Description
Monitoring Agent application	Integrates IP address managers, such as a DHCP server or a RADIUS server, into an SRC-managed network so that the SAE is notified about subscriber events. The Monitoring Agent application runs on a Solaris platform.
Residential service selection portals	Provides a framework for building Web applications that allow residential and enterprise subscribers to manage their own network services. It comes with several full-featured sample Web applications that are easy to customize and suitable for deployment. The Residential service selection portals run on a Solaris platform.
Sample enterprise service portal	Lets service providers supply an interface to their business customers for managing and provisioning services.

Related Documentation

- SRC Product Description

CHAPTER 2

Overview of Controlling Volume Usage with the SRC VTA

Overview of the SRC VTA

The SRC Volume-Tracking Application (SRC VTA) allows service providers to track and control the network usage of subscribers and services. You can control volume and time usage on a per-subscriber or per-service basis. This level of control means that service providers can offer tiered services that use volume as a metric, while also controlling abusive subscribers and applications.

When a subscriber or service exceeds bandwidth limits (or quotas), the SRC VTA can take actions, including imposing rate limits on traffic, sending an e-mail notification, or charging extra for additional bandwidth consumed. You can configure multiple SRC VTAs.

If you use the SRC VTA with the deep packet inspection (DPI) feature, you can control the volume of traffic for specific applications, such as peer-to-peer file sharing.

Terminology

Table 5 on page 10 defines terms that are used in the SRC VTA documentation and sample data.

For information about loading the sample data for the SRC VTA, see Loading Sample Data into a Juniper Networks Database (SRC CLI).

Table 5: SRC VTA Terms

Term	Definition
Bought quota	Allowance of data volume that subscribers purchase and can transfer (upload or download) at any time. (This term is used in sample and typical SRC VTA configurations and is not inherent in the SRC VTA itself.)
Bought account	Record that details a subscriber's use of bought quota. (This term is used in sample and typical SRC VTA configurations and is not inherent in the SRC VTA itself.)
Periodic quota	Allowance of data volume that a service provider allocates to subscribers on a recurrent basis. Subscribers use this allowance to upload or download data. (This term is used in sample and typical SRC VTA configurations and is not inherent in the SRC VTA itself.)
Periodic account	Record that tracks a subscriber's use of periodic quota. (This term is used in sample and typical SRC VTA configurations and is not inherent in the SRC VTA itself.)
Quota service	Service for which the SRC VTA monitors usage. The SRC VTA activates the service for subscribers when they have a positive balance in their SRC VTA accounts, and deactivates the service when the SRC VTA account has a negative balance. (This term is used in sample and typical SRC VTA configurations and is not inherent in the SRC VTA itself.)
VTA account	Record of credit and debit entries that track a subscriber's use of a particular network resource.
VTA session	Period of activity between a subscriber and an SRC VTA.

SRC VTA Service and Subscriber Accounts

SRC VTA accounts represent the resources available to a service or a subscriber. You can configure SRC VTA accounts and then charge a particular service or subscriber's usage against the account. Each subscriber or service can have a different quota, or allowance of data volume.

You can set up the way the SRC VTA charges accounts and how account balances are updated.

You can also configure actions in response to changes in account balances. Available actions include stopping a service, starting a service, updating an account balance, sending an e-mail, and running a script. For example, if account A is emptied, the action might be to stop services X and Y, and start service Z.

The SRC VTA requires a relational database to store information about accounts. The SRC VTA installation includes sample schemas for the MySQL and Oracle databases.

SRC VTA Sessions

The SRC VTA tracks subscriber activity through VTA sessions. SRC VTA sessions do not necessarily correspond to an individual subscriber session or service session. For example, a single service session can correspond to multiple SRC VTA sessions if the service session covers multiple billing periods.

The SRC VTA can track more than just the volume and time of a service session, it can track any state of a subscriber derived from SAE plug-in events and respond to the state change.

The SRC VTA requires a relational database to store information about sessions. The SRC VTA installation includes sample schemas for the MySQL and Oracle databases.

Volume-Based Services

The SRC VTA lets you set triggers at multiple levels to provide flexible and extensive volume-based services. For example:

- When the volume remaining for the account is 300 MB, turn on the internet-256 service, turn off the internet-512 service, and send an e-mail to the subscriber.
- When the volume level reaches 100 MB, send an e-mail warning to the subscriber.
- When the volume level is 0 MB, turn on the continue-TCP-only service, turn off the internet-256 service, send an e-mail to the subscriber, and notify the accounting server.
- When the volume level is –100 MB, turn off the continue-TCP-only service, send an e-mail to subscriber, and notify the accounting server.

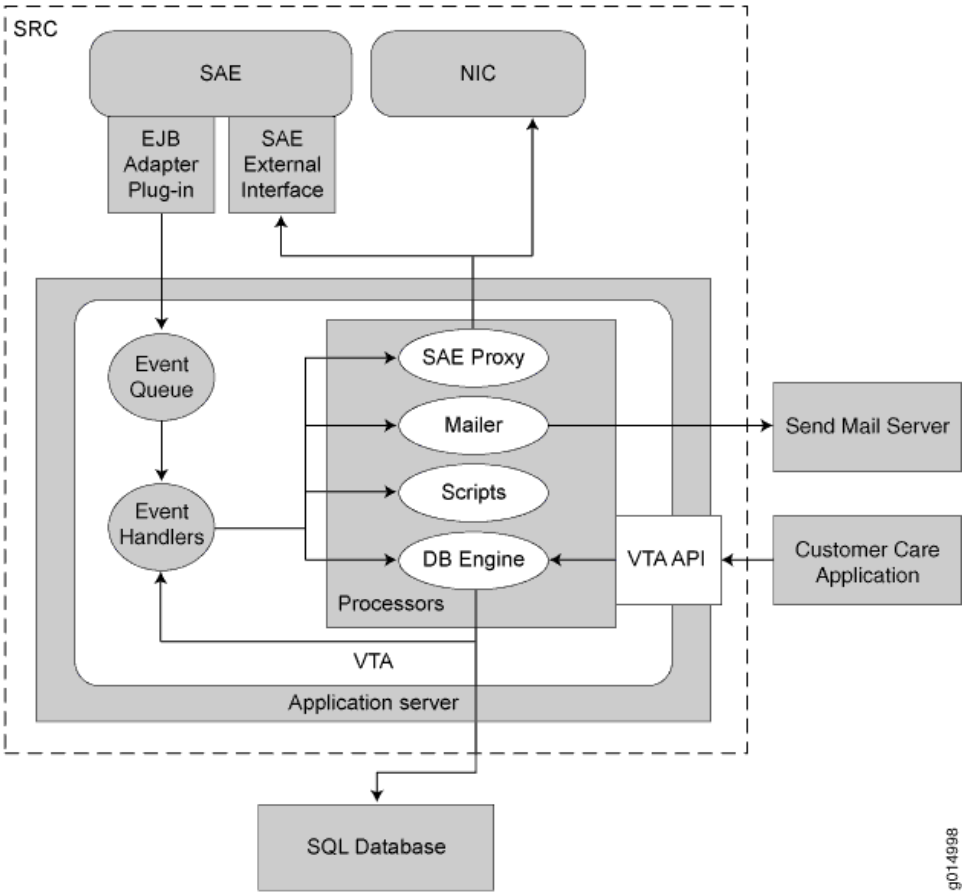
Related Documentation

- How the SRC VTA Works on page 12
- SRC VTA Operation on page 17
- Before You Configure the SRC VTA on page 37
- SRC VTA Architecture and Connections to Other SRC Components on page 11

SRC VTA Architecture and Connections to Other SRC Components

Figure 1 on page 12 shows the SRC VTA architecture and the position of the SRC VTA in the SRC network.

Figure 1: SRC VTA Architecture and Position in the SRC Network



Related Documentation

- Overview of the SRC VTA on page 9
- How the SRC VTA Works on page 12
- SRC VTA Operation on page 17

How the SRC VTA Works

The SRC VTA manages subscriber accounts using a rule-driven event-processing system that can prioritize the actions taken for certain conditions. The SRC VTA is triggered by events, such as the logging in of subscribers, the use of network services, or the changing of account balances. These events can cause actions, such as updating account balances, starting or stopping network services, or running scripts to perform external actions.

The SRC VTA processes external events based on its configuration. The SRC VTA configuration is made up of:

- Event handlers
- Actions
- Processors

Events

Each SRC VTA event corresponds to one subscriber and contains some attributes. The SRC VTA supports the following types of events:

- Service and subscriber-tracking events from the SAE; for example, start or stop tracking events.
- Account update events triggered by updating database accounts.
- Callback events triggered by an SRC VTA API call.

Event Handlers

An event handler defines how the SRC VTA processes an event. SRC VTA event handlers consist of:

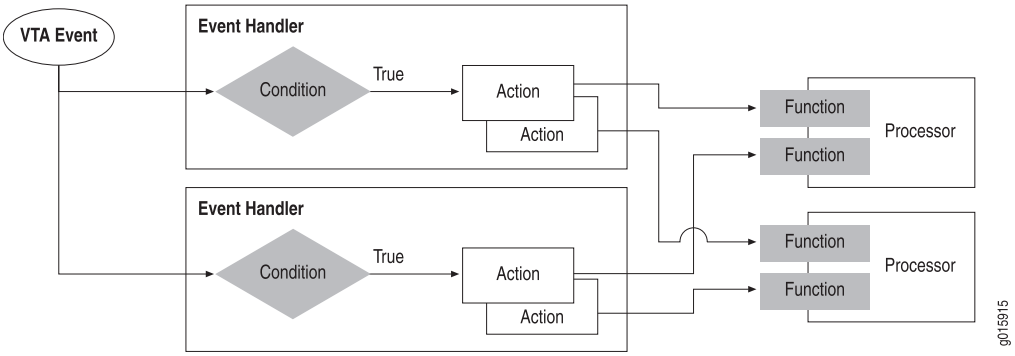
- Type of event—Tracking, update, or callback event; for example, a subscriber start–tracking event or a service start–tracking event.
- Priority—Priority at which event handlers are evaluated and executed. Event handlers are evaluated and executed from high to low priority.
- Condition—Condition that the event handler evaluates to determine whether the event handler should handle the event.
- Actions—List of actions to be performed by the event handler.

You can set up multiple event handlers to process events. For example, the first event handler could retrieve the balance for a quota account, and the next event handler could refill the quota account, depending on whether the condition of the second event handler is met.

Figure 2 on page 14 shows the event handler model. The SRC VTA processes an event as follows:

1. The event handler with the highest priority receives the event, determines whether the event's type is the same as the event type of the event handler, and determines whether the event satisfies the condition of the event handler.
2. If the condition is met, the SRC VTA performs the corresponding actions based on the event attributes. An action invokes a function and provides the parameters required by that function to the processor.
3. When an event handler finishes processing an event, the next applicable event handler according to the priority of the event handler processes the event.

Figure 2: SRC VTA Event Handler Model



Actions

You specify actions that the SRC VTA takes in response to events; for example, updating an account balance, starting a service, or stopping a service. An action is modeled as a call to a function. If an event matches the type and condition requirements of the event handler, then all actions defined for the event handler process the event, one after another, in the order the actions are configured.

An action can update event attributes or add new attributes to an event for subsequent processing of the same event by another action in the same event handler, or by actions in subsequent event handlers. The updated attributes can also be used to change whether the event satisfies a subsequent event handler's condition.

An action configuration includes the following:

- Function—Function that the action invokes
- Parameter—Parameters and corresponding values to be passed to the function

Processors

Processors implement the functions that receive and process events. The SRC VTA has four processors:

- Db-engine processor—Acts as a proxy to a database
- Mailer processor—Sends e-mail notifications when certain events occur
- SAE proxy processor—Acts as a proxy to the SAE. (Requires no configuration)
- Scripts processor—Runs external scripts or JavaScript programs

Db-Engine Processor

The db-engine processor acts as a proxy to the external database. You can use the functions provided by the db-engine processor to:

- Calculate the use of network resources for a service.
- Calculate the interim accounting interval for each service based on a subscriber's remaining resources and use of the service.
- Update SRC VTA accounts with a JavaScript program.
- Terminate an SRC VTA session. This feature is usually used at the end of a billing period so that you can finish collecting data for the current billing period and start a new SRC VTA session for the new billing period. Depending on what practices you use for terminating SRC VTA sessions, you may choose to optimize the way the SRC VTA terminates sessions.

If an SRC VTA session is terminated while the SAE is in the middle of a service session, the SAE service session can split into two or more SRC VTA sessions. This process works fine if you are not frequently terminating a large number of SRC VTA sessions. However, in some cases, you may choose to terminate SRC VTA sessions on a regular basis—for example, every 24 hours. In this case, the SAE session can last a long time and can result in hundreds of SRC VTA sessions. When this occurs, processing an interim accounting event can be slow because the SRC VTA needs to find and read many records. To improve the processing time in these environments, you can optimize the SRC VTA database.

To optimize SRC VTA database operations when you are terminating large numbers of SRC VTA sessions on a regular basis, you can set the **sessions-terminated-frequently** option under the **[edit shared vta group *name* processor db-engine]** hierarchy level. When this option is set, the SRC VTA creates a special VTA session. Instead of finding and reading all session records, the SRC VTA needs to read from and write to only the special VTA session record.

NOTE: Setting the sessions-terminated-frequently option is an irreversible act. You can set this option to true, but you cannot change it back to false.

To use the sessions-terminated-frequently optimization feature with a set of SRC VTAs, you must follow this specific process:

1. All C Series Controllers running SRC VTAs must be running SRC software release 4.3.x or later; otherwise, you must upgrade the system.
2. Send all SAE events to only one SRC VTA.
3. Wait until the removed SRC VTAs (from Step 2) have finished processing events in their event queues.
4. Set the sessions-terminated-frequently option to true.
5. Wait until all SRC VTAs have reacted to the option being set. The SRC VTA information log displays the following message: "DBEngine will assume VTA sessions are terminated frequently."
6. Resume distributing SAE events to all SRC VTAs.

SAE Proxy Processor

The SAE proxy processor is a proxy to the SAE external interface that resolves the subscriber interface based on the event types to which functions are applied.

- If a function is applied to SAE subscriber-tracking or service-tracking events, the processor finds the SAE reference in the event message.

  There is an exception if the **current-subscriber-only** parameter is set to false. In this case, the function finds subscribers in all SAEs with the NIC.

- If a function is applied to other events, the processor uses the subscriber's ID in the event as the key for the NIC to find the SAE reference. The SRC VTA uses the SAE reference and the **subscriber-id-solution** option that you specify under the VTA group to look up the SAE and the subscriber.

You can use the functions provided by the SAE proxy processor to:

- Set an interim interval for a service.
- Set a service session timeout for a subscription.
- Set a session timeout for a subscriber.
- Start a subscription to a service. You can specify the parameter substitutions to use when the service is started.
- Stop a subscription to a service. You can include a reason for stopping the subscription. When the service is stopped, the reason is sent to the billing system so it can differentiate between service stops.

The SAE proxy processor does not require configuration. To use the functions provided by the SAE proxy processor, you configure an action that calls the respective function for the event handler.

Mailer Processor

You can use the functions provided by the mailer processor to set up the SRC VTA to send e-mail notifications when certain events occur. You can specify that e-mail notifications be sent to subscribers, system administrators, or an automated billing system.

Scripts Processor

The scripts processor can invoke external executable scripts or JavaScript scripts. We recommend using JavaScript, where possible, for better performance.

- External scripts are executable programs, such as shell scripts, that are available on the SRC VTA's host. Each external script can perform a task and return a value. If the script returns a value, the value can be added to the current event as an event attribute.
- JavaScript programs are used to process SRC VTA event attributes and can also be used for any arbitrary purpose, just like external scripts. For example, a JavaScript program can convert an SRC VTA event attribute in a timestamp to a date string and add it to the event as a new attribute. The attribute can then be used for subsequent

actions, such as sending an e-mail notification to the subscriber. The JavaScript program can refer to any attributes of the event being processed, and it must return a value.

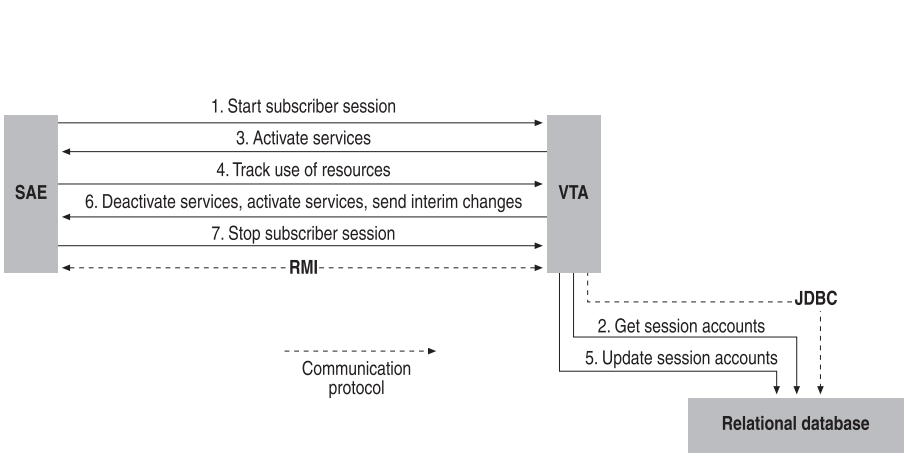
Related Documentation

- Overview of the SRC VTA on page 9
- SRC VTA Operation on page 17
- SRC VTA Architecture and Connections to Other SRC Components on page 11
- Overview of Configuring Event Handlers on page 18
- Overview of Configuring Actions on page 22
- Configuring the DB-Engine Processor for the SRC VTA Group (SRC-CLI) on page 68
- Configuring the Mailer Processor for the SRC VTA Group (SRC CLI) on page 78

SRC VTA Operation

Figure 3 on page 17 illustrates the SRC VTA operation process.

Figure 3: Operation of the SRC VTA



The SRC VTA operates as follows:

1. When an event that activates a service occurs (for example, a subscriber logs in), the SAE sends a session start event to the SRC VTA through the ejb- adaptor plug-in you configure on the SAE.
2. Optionally, the SRC VTA queries a database for the subscriber's current set of sessions and accounts.
3. Depending on the configuration of the SRC VTA, it may activate or deactivate services based on the subscriber's use of resources.
4. The SRC VTA tracks the subscriber's use of resources for a period of time.
5. The SRC VTA updates sessions and account balances in the database.

6. The SRC VTA sends to the SAE changes in the interim accounting interval and takes action to limit excessive bandwidth use or to allow increased bandwidth use.
7. When an event that deactivates a service occurs (for example, the subscriber logs out), the SRC VTA updates the VTA session, closes the VTA session, and may update the account balances.

Events received through the SRC VTA API can also cause the SRC VTA to activate services, deactivate services, or change the accounting interval.

Related Documentation

- Overview of the SRC VTA on page 9
- How the SRC VTA Works on page 12
- SRC VTA Architecture and Connections to Other SRC Components on page 11

Overview of Configuring Event Handlers

Event handlers define how the SRC VTA processes events. An event handler configuration includes the following options:

- Events—Type of event including tracking, update, or callback event; for example, a service-start tracking event.
- Priority—Priority at which event handlers are evaluated and executed. Event handlers are evaluated and executed from low to high.
- Condition—Condition that the event handler evaluates to determine whether the event handler should process the event.
- Actions—List of actions to be performed by the event handler.

You can set up multiple event handlers to process events. For example, the first event handler could retrieve the balance for a quota account, and the next event handler could refill the quota account depending on whether the condition of the second event handler is met.

When an event is received, the corresponding event type and condition configured for the event handler are evaluated based on the priority. When a condition is met, the corresponding actions are performed according to the event attributes. The action can update or add event attributes to the events for subsequent processing of the same event. An action can invoke a function provided by any processor. After an event handler has completed its processing, event processing continues with the next applicable event handler.

The SRC VTA communicates with the SAE through the Enterprise JavaBean (EJB) adapter plug-in. When an event is sent by the ejb-adapter plug-in on the SAE, it is received by the SAEEventListener, which places the event in the event queue. Because you can have multiple VTA instances, each SRC VTA has a separate SAEEventListener. The SRC VTA takes events one by one and presents them to an ordered sequence of event handlers. Each event handler includes a list of actions. If the event handler is configured to handle the event, it acts on the event based on the configured actions—for example, updating a balance or starting a service. Each action can specify a function.

The ejb-adapter plug-in can use either the round-robin algorithm or the primary/backup algorithm to send events from the SAE to the SRC VTAs. When you configure the ejb-adapter plug-in, you specify a comma-separated list of SRC VTA IP addresses or hostnames. When you use the round-robin algorithm, an outgoing event from the SAE is sent to each of the SRC VTAs specified in the comma-separated list until it is handled by one of the SRC VTAs. When you use the primary/backup algorithm, the first SRC VTA in the comma-separated list is the primary SRC VTA; all others are secondary SRC VTAs. All outgoing events from the SAE are forwarded to the primary SRC VTA until it becomes full or inaccessible. When the handling of an event fails in the primary SRC VTA, the event is sent to the secondary VTAs in a round-robin fashion until one of the secondary SRC VTAs handles the event. On successful handling of the event, the secondary SRC VTA is promoted and becomes the new primary SRC VTA.

Priority

When you configure an event handler, you need to specify the priority for evaluating and running the event handler. The priority is an integer where the smaller number has the higher priority. The event handler with the highest priority receives the event, determines whether the event's type is the same as the event type of the event handler, and determines whether the event satisfies the condition of the event handler. When an event handler finishes processing an event, the next applicable event handler according to the priority of the event handler processes the event.

Events

Each SRC VTA event corresponds to one subscriber and contains some attributes. The SRC VTA supports the following event types:

- Service and subscriber tracking events from the SAE; for example, start or stop tracking events.
- Account update events triggered by updating database accounts.
- Callback events triggered by an SRC VTA API call.

Various events can be received by the SRC VTA from the SAE. Table 6 on page 19 describes the events supported by the SRC VTA.

Table 6: SRC VTA Event Types

Event	Description
account-update	Database update event
callback:*callid*	External callback event for the specified call
service-interim:*service name*	Service interim–tracking event for the specified service
service-start:*service name*	Service start–tracking event for the specified service
service-stop:*service name*	Service stop–tracking event for the specified service
user-interim	User interim–tracking event

Table 6: SRC VTA Event Types *(continued)*

user-start	User start–tracking event
user-stop	User stop–tracking event

The *service-name* is the name of any configured SRC service.

To process an event, the event handler must be configured to handle the particular event type. You configure which events are handled by the event handler with the **shared vta group *name* event-handler *event-handler-name* events *events*** statement. Separate events with a comma. You can configure an event handler to handle multiple event types. If an event handler is configured to handle an event, it can pass that event to a sequence of actions.

An example of an event is:

service-start: *QuotaInternet*, callback: *TerminateSession*

Event Attributes

Each event carries attributes. Table 7 on page 20 describes the types of attributes that are available for each type of event.

Table 7: Event Attributes

Event Type	Available Attributes
Service and subscriber tracking events from the SAE	• Plug-in attributes, such as PA_SERVICE_NAME or PA_LOGIN_NAME, associated with an SAE plug-in event. For a list of SAE plug-in events, see "Overview of Testing the SRC VTA Configuration" on page 116. • currentTime attribute—Time since January 1, 1970 UTC when the SRC VTA begins passing the event to the event handlers (events are queued in the event queue and then passed to the event handlers). The value is the number of milliseconds in the range 0–9223372036854775807. • subscriberId—Subscriber ID based on attributes of a service or a subscriber-tracking event. The subscriberId event attribute is a result of the calculation. It identifies the subscriber of the corresponding SRC VTA event.
Account update events	• old_status_*accountname*—The old status of the account. • new_status_*accountname*—Returns the new status of the specified account. • old_lastUpdateTime_*accountname*—Returns the old last update time of the account. • new_lastUpdateTime_*accountname*—Returns the new last update time of the account. • old_balance_*accountname*—Returns the old balance of the account. • new_balance_*accountname*—Returns the new balance of the specified account • currentTime—Time since January 1, 1970 UTC, when the SRC VTA begins passing the event to the event handlers (events are queued in the event queue and then passed to the event handlers). The value is the number of milliseconds in the range 0–9223372036854775807. • subscriberId—Subscriber ID based on attributes of a service or a subscriber-tracking event. The subscriberId event attribute is a result of the calculation. It identifies the subscriber of the corresponding SRC VTA event.

Table 7: Event Attributes *(continued)*

Event Type	Available Attributes
Callback events	• callId—When the callback event type is invoked the *callId* value is specified. The *callId* value is placed in to the event using this event attribute and then processed. If an event handler is configured to handle this event type, the actions specified for the event handler can invoke functions, which have access to the *callId* in this event attribute. • currentTime—Time since January 1, 1970 UTC, when the SRC VTA begins passing the event to the event handlers (events are queued in the event queue and then passed to the event handlers). The value is the number of milliseconds in the range 0–9223372036854775807. • subscriberId—Subscriber ID based on attributes of a service or a subscriber-tracking event. The subscriberId event attribute is a result of the calculation. It identifies the subscriber of the corresponding SRC VTA event.

Condition

Each event handler evaluates conditions to determine whether it should handle the event. You specify the condition as a script written in the JavaScript programming language that must return one of the following values:

- True—Event handler should handle the event.
- False—Event handler should not handle the event.

If the condition is met, the SRC VTA performs the corresponding actions based on the event attributes. An action invokes a function and provides the parameters required by that function to the db-engine processor. If no condition is specified, true is returned as the value. If a referenced attribute does not exist in the event, the referenced attribute's value is null. Following is an example condition:

```
var newBalance=<balance_BoughtQuota>+<balance_PeriodicQuota>;
if(<old_balance_PeriodicQuota>==null)
   <old_balance_PeriodicQuota>=<balance_PeriodicQuota>;
if(<old_balance_BoughtQuota>==null)
   <old_balance_BoughtQuota>=<balance_BoughtQuota>;
return <old_balance_PeriodicQuota>+<old_balance_BoughtQuota><=0&&newBalance>0;
```

NOTE: In any condition or other JavaScript script, event attributes are referred to by enclosing them in angle < > brackets.

Actions

When you configure an event handler, you specify actions that the event handler executes in response to an event; for example, updating an account balance, starting a service, or stopping a service. An action is performed only if the event matches the specified event type and condition you configure for the event handler. An action can invoke functions provided by any processor.

NOTE: We recommend that when you configure event handlers and their actions, you ensure that for any given event, all database operations are performed before any other operations that have permanent effects. This is because if a database error occurs—for example, due to normal contention for database records between different event threads—the SRC VTA rolls back the current database transaction (no changes are made to the database) and then restarts processing the event. If the event performs some other operation other than database operations before such an error, such as start a service, then that other operation is performed again when the event is reprocessed following the error.

Related Documentation

- Overview of Testing the SRC VTA Configuration on page 116
- How the SRC VTA Works on page 12
- Configuring Event Handlers (SRC CLI) on page 65
- Overview of Configuring Actions on page 22
- Configuring Actions for SRC VTA Event Handlers (SRC CLI) on page 64

Overview of Configuring Actions

Actions are executed by event handlers in response to an event. For example, the action may update an account balance, start a service, or stop a service. You configure what action the event handler takes in response to an event when you configure an event handler. An action is performed only if an event matches the event type and condition specified in the event handler configuration. An action can invoke functions provided by any processor.

An action can update event attributes or add new attributes to an event for subsequent processing of the same event by subsequent actions in the same event handler, or in subsequent event handlers.

You define an action with the **edit shared vta group** *name* **action** *action-name* configuration statement. You refer to a previously defined action when you configure an event handler. Event handlers can refer to multiple actions. To configure an event handler to use a particular action, specify it as follows:

```
[edit shared vta group name event-handler name]
user@host# set actions actions
```

NOTE: We recommend that when you configure event handlers and their actions, you ensure that for any given event, all database operations are performed before any other operations that have permanent effects. This is because if a database error occurs—for example, due to normal contention for database records between different event threads—the SRC VTA rolls back the current database transaction (no changes are made to the database) and then restarts processing the event. If the event performs some other operation other than database operations before such an error, such as start a service, then that other operation is performed again when the event is reprocessed following the error.

When you configure an action, you specify the following options:

- Function—The function that the action invokes
- Parameter—The parameters and corresponding values to be passed to the function
- On-error—What the event handler does in response to an error

Functions are variables that an action invokes, for example, to update database accounts. The function takes as input the configured parameters, the event, and the event context (which may include values created by previously executed actions). The SRC VTA provides a set of available functions. Each function takes a different set of input parameters. Table 8 on page 23 describes the available SRC VTA functions and their associated input parameters.

Table 8: SRC VTA Functions and Input Parameters

Function Name	Description	Input Parameters
db-engine-calculate-interim	Calculates the interim interval in the service-tracking event by using the interim interval function for the service, as defined with the **shared vta group name processor db-engine service *name* interim-interval-function** configuration statement . This function adds the following attribute to the event after the function is executed: • *interimInterval*—Interim interval of the service	None

Table 8: SRC VTA Functions and Input Parameters *(continued)*

db-engine-calculate-usage	Calculates usage in the service-tracking event by using the usage metric function for the service, as defined with the **shared vta group name processor db-engine service *name* usage-metric-function** configuration statement . This function adds the following attributes to the event after the function is executed: • *currentUsage*—Usage since the previous usage report • *interimTime*—Session length since the previous usage report	None
db-engine-get-accounts	Retrieves account data for the corresponding subscriber for the event. Data for multiple accounts is retrieved; accounts are identified by the *accountName* suffix. Accounts are defined with the **shared vta group *name* processor db-engine account *name*** configuration statement, where the account name corresponds to the *_accountName* suffix. Subsequent event handlers of the event can use the retrieved data. This function adds the following attributes to the event after the function is executed: • *balance_accountName*—Balance for the account. • *lastUpdateTime_accountName*—Last update time in milliseconds since January 1, 1970 UTC for the account. • *status_accountName*—Status of the account.	None
db-engine-terminate-session	Closes active SRC VTA sessions (for the subscriber for which the event occurred) that have a status of Start or Interim. It does not stop the corresponding services in the SAE. You can use this function to stop a session at the end of a billing period. Usage data collected after the SRC VTA session is stopped is stored in new SRC VTA session records.	None

Table 8: SRC VTA Functions and Input Parameters *(continued)*

db-engine-update-accounts	Runs an account update script that changes the account balances of the corresponding subscriber for the event. Scripts are defined with the **shared vta group *name* processor db-engine account-update-script *name*** configuration statement. The predefined script name is used as input to this function. This function adds the following attributes to the event after the function is executed: • *balance_accountName*—Balance for the account after it is updated. • *lastUpdateTime_accountName*—Last update time in milliseconds since January 1, 1970 UTC for the account after it is updated. • *status_accountName*—Status of the account after it is updated.	**script-name**—Name of the account update script defined in the db-engine processor.
mailer-send	Sends e-mail notifications when certain events occur. You can specify that e-mail notifications be sent to anyone, for example, subscribers, system administrators, or an automated billing system.	• **recipient**—Destination e-mail address • **from**—Source e-mail address • **subject**—Subject of the e-mail • **text**—Content of the e-mail
sae-set-interim-interval	Sets the interim accounting interval for the service session identified in the event to be the value of the *interimInterval* attribute currently found in the event. Before this function is called, the db-engine-calculate-interim function of the db-engine processor must be called in order to place the *interimInterval* attribute into the event.	**current-subscriber-only**—Specifies whether the function is applied only to the subscriber identified in the event or to all subscribers who have the same subscriber ID. • Set to true to apply the function to the current subscriber only. If you do not set this parameter, true is the default behavior. • Set to false to apply the function to all subscribers who have the same subscriber ID.

Table 8: SRC VTA Functions and Input Parameters *(continued)*

sae-set-service-timeout	Sets the service session timeout for the service session identified in the event.	• **subscription-name**—Name of the subscription in the format *serviceName*%*subscriptionId*. Default subscriptions have the same name as the service. This parameter is optional when a service-tracking event is being processed. If %*subscriptionId* is omitted, the default subscription is assumed. • **session-name**—Name of the service session. This parameter is ignored if a service-tracking event is being processed and the subscription name is omitted. In this case, the session name from the service-tracking event is used. If this parameter is omitted, the default service session is used. • **session-timeout**—Length of the service session timeout in seconds. If the session timeout is set to 0, the service session is stopped immediately. When the session timeout expires, the service session is stopped. • **current-subscriber-only**—Specifies whether the function is applied only to the subscriber identified in the event or to all subscribers who have the same subscriber ID. ◦ Set to true to apply the function to the current subscriber only. If you do not set this parameter, true is the default behavior. ◦ Set to false to apply the function to all subscribers who have the same subscriber ID.
sae-set-user-timeout	Sets the subscriber session timeout of the subscriber identified in the event.	• **session-timeout**—Length of the subscriber session timeout in seconds. If the session timeout is set to 0, the subscriber session is stopped immediately. When the session timeout expires, the subscriber is logged out. • **current-subscriber-only**—Specifies whether the function is applied only to the subscriber identified in the event or to all subscribers who have the same subscriber ID. ◦ Set to true to apply the function to the current subscriber only. If you do not set this parameter, true is the default behavior. ◦ Set to false to apply the function to all subscribers who have the same subscriber ID.

Table 8: SRC VTA Functions and Input Parameters *(continued)*

sae-start-service	Starts the specified service subscription.	• **subscription-name**—Name of the subscription in the format *serviceName*%*subscriptionId*. Default subscriptions have the same name as the service. This parameter is optional when a service-tracking event is being processed. If %*subscriptionId* is omitted, the default subscription is assumed.
		• **session-timeout**—Length of the service session timeout in seconds. When the session timeout expires, the service session is stopped.
		• **session-name**—Name of the service session. If this parameter is omitted, the default service session is used.
		• **current-subscriber-only**—Specifies whether the function is applied only to the subscriber identified in the event or to all subscribers who have the same subscriber ID.
		  ◦ Set to true to apply the function to the current subscriber only. If you do not set this parameter, true is the default behavior.
		  ◦ Set to false to apply the function to all subscribers who have the same subscriber ID.
		• **substitution**—Policy parameter substitution to use when starting the service. This is specified as **substitution *name* value *value***. If this parameter is omitted, the service is started without substitutions.
		• **persistent**—Specifies whether a service session is persistent. If you use the SRC VTA to activate a service with the persistent option, this service is subsequently activated by the SAE every time the subscriber connects to the network. This option provides efficiency because the SRC VTA does not need to make a decision to activate the service on subsequent logins and because some applications can more efficiently activate a group of services at login.
		  ◦ Set to true to cause the session to be persistent.
		  ◦ Set to false to specify that the session is not persistent and the service is activated or deactivated only for the current subscriber session.

Table 8: SRC VTA Functions and Input Parameters *(continued)*

sae-stop-service	Stops the specified subscription to the specified service.	• **subscription-name**—Name of the subscription in the format *serviceName*%*subscriptionId*. Default subscriptions have the same name as the service. This parameter is optional when a service-tracking event is being processed. If %*subscriptionId* is omitted, the default subscription is assumed. • **session-name**—Name of the service session. If the subscriptionName parameter is omitted, this parameter is ignored. If this parameter is omitted, the default service session is used. • **reason**—Reason for the termination. When the service is stopped, the termination cause can be sent to the billing system so it can differentiate between service stops. If this parameter is omitted, no termination cause is provided to the billing system. Specify an integer that identifies the termination cause. Possible values are defined in RFC 2866—RADIUS Accounting (June 2000). • **current-subscriber-only**—Specifies whether the function is applied only to the subscriber identified in the event or to all subscribers who have the same subscriber ID. ◦ Set to true to apply the function to the current subscriber only. If you do not set this parameter, true is the default behavior. ◦ Set to false to apply the function to all subscribers who have the same subscriber ID. • **persistent**—Specifies whether a service session is persistent. ◦ Set to true to cause the session to be persistent. ◦ Set to false to specify that the session is not persistent and the service is deactivated only for the subscriber identified in the event.
scripts-run-external-script	Executes an external script that is present on the local SRC file system, as previously defined with the **shared vta group** *name* **processor scripts external-script** *name* configuration statement.	• **script-name**—Name of an external script previously defined with the **shared vta group** *name* **processor scripts external-script** *name* configuration statement. • **name**—Any other parameters the script expects to receive as previously defined under the **shared vta group** *name* **processor scripts external-script** *name* **parameters** configuration statement. These parameters are entered as name/value pairs.

Table 8: SRC VTA Functions and Input Parameters *(continued)*

scripts-run-javascript	Executes a script written in the JavaScript programming language that you previously defined by using the **shared vta group *name* processor scripts javascript *name*** configuration statement.	• **script-name**—Name of a script written in the JavaScript programming language that you previously defined by using the **shared vta group *name* processor scripts javascript *name*** configuration statement. Any parameters referenced in the script itself are taken from the event and passed to the function automatically.

On Error

For each action, you must specify what the event handler does if an error occurs; you specify this with the **on-error** option. If an error occurs, the event handler can do one of the following:

- Abort-event-processing—Stop processing the current event.
- Go-to-next-action—Continue with the next action, if any, in the same event handler.
- Go-to-next-event-handler—Skip any remaining actions in the current event handler and proceed to the next event handler (if any).

Related Documentation

- How the SRC VTA Works on page 12
- Overview of Configuring Event Handlers on page 18
- Configuring Actions for SRC VTA Event Handlers (SRC CLI) on page 64
- Configuring Event Handlers (SRC CLI) on page 65

Overview of Managing SRC VTA Accounts and Sessions

The SRC VTA allows service providers to manage accounts and sessions by:

- Identifying Subscribers, SAEs, and Sessions on page 29
- Managing SRC VTA Accounts and Sessions on page 30

Identifying Subscribers, SAEs, and Sessions

The SRC VTA must be able to identify each subscriber by a unique identifier. The SRC VTA uses the identifier to manage:

- SRC VTA accounts and sessions
- Subscriber and service sessions

You can configure the SRC VTA to use data keys to identify corresponding data values for these management tasks. The data keys depend on the subscriber's identifier and comprise one or more plug-in attributes. Some identifiers are suitable for residential subscribers and some for enterprise subscribers.

Managing SRC VTA Accounts and Sessions

Depending on the information that identifies subscribers in your SRC configuration, you can configure the SRC VTA to use several types of plug-in attributes as data keys to identify accounts and sessions in the SRC VTA database. If you use a NIC with the SRC VTA, the SRC VTA can also use some of these plug-in attributes to construct a data key that the NIC can use to determine which SAE manages a subscriber. When the NIC identifies an SAE, the SRC VTA can also obtain a key to identify the subscriber session that the SAE is managing for the subscriber.

You configure the data keys that identify accounts and sessions with the **shared vta group *name*** statement and specifying the data keys by setting the **subscriber-id-solution** option.

Related Documentation

- Overview of the SRC VTA on page 9
- Configuring a Database to Store Account and Session Data (SRC CLI) on page 42
- Configuring the Initial Balance and Status of a Subscriber Account in the External Database (SRC CLI) on page 69
- Example: Limiting Subscriber Access Based on Account Balances on page 105

Overview of Adjusting the Interim Accounting Interval

When you configure services in the SRC VTA, you can optionally define a formula to dynamically adjust the interim accounting interval for each service based on the subscriber's remaining resources and use of the network for that service. Each service in the SRC VTA can use a different formula. You can configure the SRC VTA software to evaluate the formula to obtain the accounting intervals. Depending on the result, the SRC VTA performs the following functions:

- If the result is zero, the SRC VTA disables interim accounting.
- If the result is a negative number, the SRC VTA does not change the interim accounting interval.
- If the result is a positive number, the SRC VTA changes the interim accounting interval to this value.

The variables used to define the interim accounting interval are categorized as:

- Current service—Provides session data of the service for the current service-tracking event.
- Other service—Provides service session usage information for another subscriber service for the current service-tracking event. For example, if a subscriber has two quota services, QuotaLocal and QuotaInternet, the interim formula for QuotaLocal can provide usage information to QuotaInternet.
- Account balance—Provides the balance in the account.

For details on the variables used to define the interim accounting interval formula, see "Variables Used to Define the Interim Accounting Interval for Services" on page 73.

Related Documentation

- Configuring SRC VTA Services and Policies on page 43
- Variables Used to Define the Interim Accounting Interval for Services on page 73
- Configuring the Interim Account Interval and Usage Metric of a Service in the External Database (SRC CLI) on page 71

Using JavaScript Programs in SRC VTA Configurations

You can use JavaScript programs in your SRC VTA configuration for such tasks as calculating a usage metric or an interim accounting interval, specifying an event condition, updating event attributes in processors, and writing scripts to update accounts.

You can reference a specific set of predefined variables in a JavaScript program, such as plug-in attributes, event attributes, or account balances.

When a variable is referenced or updated in the JavaScript program, enclose it in angle brackets (<>) so that the JavaScript program retrieves only the necessary information. Within the JavaScript program, only one instance of the referenced variable must be enclosed in angle brackets.

You define formulas in the JavaScript scripting language (see http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html).

Related Documentation

- How the SRC VTA Works on page 12
- Configuring JavaScript Programs on page 80

SRC VTA SOAP Interface

The SRC Volume Tracking Application (SRC VTA) API is a Simple Object Access Protocol (SOAP) interface that allows developers to create gateway clients and that administrators use to manage SRC VTA subscribers and sessions. The SRC Web Services Gateway allows a gateway client—an application that is not part of the SRC network—to interact with SRC components, such as the SRC VTA, through a SOAP interface.

SRC VTA SOAP Interface URL

After you have installed the SRC VTA, you can access a Web Services Definition Language (WSDL) file for the application. The WSDL file defines the SOAP properties that you or your customers can use to develop a client. The URL for the SRC VTA WSDL file is:

http://<host-name>:8080:/<vta-group>/api/soap/services/vtaAPI?wsdl

- **<host-name>**—Name of the C Series Controller that is running the SRC VTA
- **<vta-group>**—Name of the SRC VTA group

Related Documentation

- Enabling the SOAP Interface for an SRC VTA Group (SRC CLI) on page 94
- Methods for the Volume Tracking Application SOAP Interface on page 95

- Viewing SRC VTA SOAP API Statistics (SRC CLI)

Overview of the Web Application Server on C Series Controllers

The SRC software on a C Series Controller includes a Web application server that hosts the Web Services Gateway and the Volume Tracking Application (SRC VTA). In production environments, this application server is designed to host only these applications. However, you can load your own applications into this server for testing or demonstration purposes.

By default, the SRC Web application server listens on port 8080 for HTTP connections on the eth0 interface (interface to the trusted network) and on the configured ports for HTTP and HTTPS connections on the **eth1** interface (interface to the untrusted network).

You can control access to applications deployed in the Web application server by configuring virtual hosts. A virtual host contains aliases and lists of the clients that are allowed to access the virtual host.

The aliases are DNS names or IP addresses that appear in the host part of the URLs used by clients to access a Web application. When the Web application server receives a request for an application, it searches for the virtual host with the alias that matches the host in the URL. If a virtual host is found, the Web application server verifies that the application is deployed on this virtual host and the client making the request is allowed to access the virtual host. If no virtual host is found, or if access to the application or client is not allowed by the virtual host, the request is rejected and the client receives an error code.

By default, SRC applications use the virtual host **eth0**. You must configure this virtual host and the following aliases:

- The IP address assigned to **eth0**.
- The name for the SRC host configured at the **[edit system host-name]** and **[edit system domain-name]** hierarchy levels.

For this reason, if you want to access the **eth0** virtual host with URLs containing the DNS name of your SRC host, you must configure your SRC hostname in your DNS server.

You configure the built-in applications, such as Dynamic Service Activator, to deploy the application to a specific virtual host. Other applications that you can load for demonstration purposes are automatically deployed on the built-in virtual host **eth0**.

Clustering

The SRC Web application server supports clustering, which provides reliability through failover and load balancing. The nodes in the cluster automatically discover one another on startup and automatically synchronize their state with the rest of the group. The cluster configuration is part of the shared SRC configuration and is stored in the Juniper Networks database. You can configure several Web application server clusters. However, a single SRC Web application server instance belong to only one cluster; it cannot belong to more than one cluster.

Local and Shared Configuration

If you want a Web application server instance to be part of a cluster, you need to specify the cluster name in the local configuration by using the **[edit slot 0 application-server]** configuration statement. This statement points to the shared configuration stored in the Juniper Networks database. The Web application shared configuration is specified using the **[edit shared application-server cluster *cluster-name*]** configuration statement.

Storing the cluster configuration in the Juniper Networks database ensures that all nodes in the cluster share the same configuration, including the unique identifier of each node, and the shared cluster name. All nodes must be specified within the same Juniper Networks database community name.

The configuration of the application server cluster lists the information about each application server node. When the application server is started, the system retrieves the shared application server cluster configuration and generates the appropriate startup script for the application server node. If no cluster is defined, the application server is started in "all" mode, but without the cluster parameters.

NOTE: If you change the shared-cluster configuration, you must restart the local Web application server.

By default, the intra-cluster communication is done through multicasting and UDP is used as the channel stack protocol. If multicasting is not an option for deployment, you can use TCP as the channel stack. The shared cluster configuration is valid only if the following conditions are fulfilled:

- The multicast-address is configured and either the channel stack is not set (the system uses UDP by default) or the channel stack is set to UDP.
- The channel stack is set to TCP and the multicast-address is not configured.

Related Documentation

- Configuring the Web Application Server (SRC CLI) on page 49
- Configuration Statements for the Web Application Server on page 50

PART 2

Configuration

CHAPTER 3

Prerequisites for Running the SRC VTA

Before You Configure the SRC VTA

Because the SRC VTA relies on other components in the SRC network, you must complete several tasks before you configure the SRC VTA.

Before you configure the SRC, you must complete the following tasks:

1. Deploy a working SRC network.

   To support the SRC VTA, you must install SAEs to manage the routers or other devices through which subscribers connect to the network.

   See *Configuring the SAE (SRC CLI)*.

2. Configure the SRC Web application server.

   See "Configuring the Web Application Server (SRC CLI)" on page 49.

3. (Optional) Configure a NIC that identifies the SAE reference for each subscriber type. You need to complete this task only if your configuration requires a NIC—for example, if an event handler action affects more than just the current subscriber described by an event.

   See "Configuring a NIC for the SRC VTA (SRC CLI)" on page 45.

4. (Optional) Create the Java scripts that the SRC VTA invokes.

5. On a separate host, install a relational database to store the data that the SRC VTA tracks. You must have the database server running, and then you must create the SRC VTA database within your database server.

See, "Configuring a Database to Store Account and Session Data (SRC CLI)" on page 42.

6. To allow the SRC Web application server to connect to your external database, copy the JDBC driver **.jar** file for your database server brand and version to the **/opt/UMC/appsvr/common/lib** directory on *every* system running the SRC VTA.

   See "Installing the JDBC Driver .jar File" on page 43.

7. Configure the associated services and policies.

   See "Configuring SRC VTA Services and Policies" on page 43.

8. Configure your subscribers and subscriptions.

   See "Configuring Subscribers and Subscriptions to SRC VTA Services" on page 44 .

9. Configure the SAE to send tracking events to the SRC VTA.

   See "Configuring the SAE to Send Tracking Events to the SRC VTA (SRC CLI)" on page 38.

Related Documentation

- Overview of the SRC VTA on page 9
- How the SRC VTA Works on page 12
- SRC VTA Operation on page 17
- Overview of the Web Application Server on C Series Controllers on page 32
- Configuration Statements for the Web Application Server on page 50

Configuring the SAE to Send Tracking Events to the SRC VTA (SRC CLI)

The SRC VTA communicates with the SAE through the Enterprise JavaBean (EJB) adapter plug-in. This plug-in is an SAE plug-in and performs the following functions:

- Filters SAE plug-in events for the SRC VTA.
- Adapts internal SAE events to EJB-compatible methods.
- Sends SAE tracking plug-in events to the SRC VTA.

To configure the EJB adapter plug-in:

1. From configuration mode, access the EJB adapter plug-in configuration. In this sample procedure, the EJB adapter plug-in called QuotaVTA is configured in the nw-area SAE group.

   user@host# edit shared sae group nw-area configuration plug-ins name QuotaVTA ejb-adapter

2. Configure a list of all SRC VTAs to which SAE tracking events are sent.

   [edit shared sae group nw-area configuration plug-ins name QuotaVTA ejb-adapter]
   user@host# set application-server-url *application-server-url*

Specify the *application-server-url* as a comma-separated list of SRC VTA IP addresses or hostnames. The prefix jnp:// and the suffix :1099 are allowed, but unnecessary. The default value is 127.0.0.1.

NOTE: The setting of this option changes as of SRC software release 4.3.x. When upgrading from earlier releases of the SRC software to Release 4.3.x, it is mandatory that you configure this option. See the *SRC Release Notes* for Release 4.3 for the complete step-by-step upgrade procedure. It is very important that the upgrade procedure is followed exactly as described in the *SRC Release Notes*.

3. Configure the JNDI name of the SAEEventListener EJB of the peer SRC VTA. Because multiple SRC VTA groups are supported, you must specify the SAEEventListenerBean for each SRC VTA group. Specify the SAEEventListenerBean in the following format:

   vta-*VTA group name*/SAEEventListenerBean

   For example, if you want a particular SAE to send events to a particular SRC VTA instance (group) called “apple”, set the **jndi-sae-event-listener** as follows:

   ```
   [edit shared sae group nw-area configuration plug-ins name QuotaVTA ejb-adapter]
   user@host# set jndi-sae-event-listener  vta-apple/SAEEventListenerBean
   ```

   If you want a particular SAE to send some events to one SRC VTA instance (group) called “apple”, and some events (can be exactly the same events) to another SRC VTA instance called “orange”, configure two separate ejb-adapter plug-ins—one with the **jndi-sae-event-listener** set to “vta-apple/SAEEventListenerBean”, and the other with the **jndi-sae-event-listener** set to “vta-orange/SAEEventListenerBean.”

4. (Optional) Configure the LDAP filter that determines the subscriber and service events that the EJB adapter plug-in sends to the SRC VTA. If you specify plug-in attributes in this field, you must include the same attributes in the attributes option.

   ```
   [edit shared sae group nw-area configuration plug-ins name QuotaVTA ejb-adapter]
   user@host# set event-admitter  event-admitter
   ```

   Table 9 on page 39 lists the values that you can use for LDAP filter strings.

Table 9: Settings for Filter Strings

Filter String	Action
()	Matches no objects
(*)	Matches all objects

Table 9: Settings for Filter Strings *(continued)*

Filter String	Action
List of <attribute>= <value> pairs <attribute>—Name of a property or attribute <ldapAttributeName> <value>—One of the following: • * (asterisk) • Explicit string • String that contains an * **Note:** To define a special character (* & , ! | \ ) in a string, precede it with the backslash symbol (\).	• If <value> is *, checks for any value. • If <value> is an explicit string, checks whether any value of the property matches the string, regardless of case. • If <value> is a string that contains a *, checks whether any value of the property contains the string, regardless of case.
(&<filter><filter>...)	True if all filters match
(|<filter><filter>...)	True if at least one filter matches
(!<filter>)	True if the filter does not match

The variables in the filter include the names of plug-in attributes and a PluginEventType variable. The value of this variable is the name of the type of event, such as PE_START_SERVICE. For names of plug-in attributes and plug-in event types, see the SAE CORBA plug-in documentation on the Juniper Networks Web site at http://www.juniper.net/techpubs/software/management/src/api-index.html or in the **SDK+AppSupport+Demos+Samples.tar.gz** file on the Juniper Web site at https://www.juniper.net/support/products/src/index.html.

5. (Optional) Configure the plug-in attributes that the EJB adapter plug-in sends to the SRC VTA listener. If you do not define a list of attributes, the EJB adapter plug-in sends all plug-in attributes to the SRC VTA. Sending unnecessary plug-in attributes can adversely affect the performance of SRC components.

   [edit shared sae group nw-area configuration plug-ins name QuotaVTA ejb-adapter]
   user@host# set attributes [(host | router-name | interface-name | ...)...]

   Specify at least the following plug-in attributes: router-name, session-id, login-name, user-ip-address, ssp-host, domain, service-name, event-time, session-time, in-octets, out-octets, in-packets, out-packets, session-timeout, downstream-bandwidth, upstream-bandwidth, service-session-name, subscription-name. You may need to add attributes if you use them for the event admitter.

6. Configure how you want the SAE to send events to the SRC VTAs. Events can be sent using either the round-robin algorithm or the primary/backup algorithm.

   [edit shared sae group nw-area configuration plug-ins name QuotaVTA ejb-adapter]
   user@host# set use-primary-vta-if-available

By default, the SAE communicates with the SRC VTAs by using the round-robin algorithm. To use the primary/backup algorithm for SAE-VTA communication, enable the **use-primary-vta-if-available** option.

Related Documentation

- How the SRC VTA Works on page 12
- Installing the JDBC Driver .jar File on page 43

Specifying Tracking Plug-Ins for Enterprise Subscribers on Junos OS Routing Platforms

When user-tracking plug-ins are attached to the retailer on routers running the Junos OS, login names are needed to trigger the user-tracking plug-in and generate user-tracking events. Because enterprise subscribers do not have a login name, the SRC VTA cannot get the required user-tracking events.

To allow enterprise subscribers on routers running the Junos OS to use retailer-attached user-tracking plug-ins, configure the EJB adapter plug-in to filter SAE plug-in events for the SRC VTA and send SAE tracking events to the SRC VTA.

To use the EJB adapter plug-in to send events for a specific retailer:

1. Configure the event admitter of the EJB adapter plug-in (see "Configuring the SAE to Send Tracking Events to the SRC VTA (SRC CLI)" on page 38).

   Specify the PA_USER_DN event attribute with the retailer's relative distinguished name (RDN). For example, the following event admitter matches events from subscribers in the SP-Quota retailer:

   ```
   PA_USER_DN=*SP-Quota*
   ```

2. Configure the EJB adapter plug-in as the global subscriber-tracking plug-in for the SAE. See Configuring Tracking Plug-Ins (SRC CLI).

   In the sample procedure, the EJB adapter plug-in called QuotaVTA is configured as the subscriber-tracking plug-in in the nw-area SAE group.

   ```
   [edit shared sae group nw-area configuration plug-ins event-publishers]
   user@host# set subscriber-tracking QuotaVTA
   ```

Related Documentation

- Configuring the SAE to Send Tracking Events to the SRC VTA (SRC CLI) on page 38
- Configuring Administrative Information for Enterprise Subscribers (SRC CLI)

Configuring the External Database

This section describes how to configure an external database to store SRC VTA account and session data.

1. Configuring a Database to Store Account and Session Data (SRC CLI) on page 42
2. Installing the JDBC Driver .jar File on page 43

Configuring a Database to Store Account and Session Data (SRC CLI)

The SRC VTA requires a relational database to store accounts and session data. You must create this database on a separate, non-SRC machine before you can run the SRC VTA. For information about databases that we have tested for use with the SRC VTA, see the *SRC Release Notes*.

For each SRC VTA instance, you need to create a database that uses the schema for the SRC VTA. To configure a database:

1. From any C Series Controller running SRC software release 4.2.x or later, navigate to the /opt/UMC/vta/database/ directory and copy the appropriate SRC VTA schema file and save it on your external database machine. We provide the following schema files:

   - /opt/UMC/vta/database/vta-database-oracle.sql
   - /opt/UMC/vta/database/vta-database-mysql.sql

   These files contain the SQL statements that you must execute to create a database in either a MySQL or Oracle database server. These files are examples that show the required database schema (in the file for MySQL, you must modify the database username and password before you can execute the SQL statements). If you have a different type of database, you can use these files as a reference on how to create a database with the same schema as is described in these two files.

2. Configure access to the database for an administrator by using the SRC VTA to monitor and manage subscribers. Edit the following options in the file as required for your external database:

   - SRC VTA database name
   - Username
   - Password

3. Use your standard mechanism to execute the SQL statements contained in the file. This creates a database inside your database server, with the schema needed by the SRC VTA.

NOTE: **If the provided file does not contain SQL sufficient to create a database inside your database server, you must create the database manually. Use the provided file as a reference to understand the exact schema required by the SRC VTA**

Installing the JDBC Driver .jar File

To allow the SRC Web application server to connect to your external database, you need to copy the JDBC driver **.jar** file for your database server brand and version to *every* system running the SRC VTA.

1. Obtain the relevant .jar file that contains the JDBC driver for your particular database server brand and version.
2. Copy the file to the **/opt/UMC/appsvr/common/lib** directory on *every* system running the SRC VTA. You can use the SRC CLI "file" commands, FTP, or SCP to copy the file.
3. Restart the Web application server on each SRC system.

   user@host> **restart component appsvr**

Related Documentation

- Configuring the Connection Between the SRC VTA and the External Account and Session Database (SRC CLI) on page 62
- Configuring the Web Application Server (SRC CLI) on page 49
- Overview of the Web Application Server on C Series Controllers on page 32
- Configuring the Web Application Server Shared Cluster Configuration (SRC CLI) on page 52

Configuring SRC VTA Services and Policies

Only the SRC VTA should activate and deactivate services that the SRC VTA controls, and you must ensure that these services are not visible on a portal for subscribers to control manually. You can use other services with the SRC VTA if you design the policies and priorities for those services to work together.

For example, if you manage subscribers with the SRC VTA, you can allow subscribers to manually activate a service that overrides the quota service, and consequently prevents charges in the periodic and bought accounts. You would account for use of this service through RADIUS rather than the SRC VTA, and subscribers would incur an extra cost for using the service. In this case, you configure the overriding service with a higher precedence than the quota service.

To configure services for the SRC VTA:

1. Create services for which the SRC VTA monitors and manages usage.
2. Configure policies that specify ingress and egress accounting rules consistent with the usage formula.

   For information about configuring accounting rules for a policy, see Policy Management Overview.

Related Documentation

- How the SRC VTA Works on page 12
- Configuring Subscribers and Subscriptions to SRC VTA Services on page 44

Configuring Subscribers and Subscriptions to SRC VTA Services

To configure subscribers to SRC VTA services, see the *SRC PE Subscribers and Subscriptions Guide*.

1. Create at least one shared subscriber.
2. For all subscribers managed by the SRC VTA, create an individual or a group subscription to services for which the SRC VTA monitors and manages usage.

Related Documentation

- Overview of the SRC VTA on page 9
- Locating the SAE That Manages a Subscriber for the SRC VTA on page 45
- Configuring SRC VTA Services and Policies on page 43
- Configuring a Database to Store Account and Session Data (SRC CLI) on page 42

CHAPTER 4

Configuration Tasks for Using NIC with the SRC VTA

Locating the SAE That Manages a Subscriber for the SRC VTA

You can use NIC proxies if the SRC VTA software needs to locate the SAE that manages a particular subscriber. For example, if the SRC VTA receives an account update event and determines that it needs to reconfigure the corresponding SAE session, the SRC VTA must find the SAE that is managing the session. The SRC VTA can do this through the NIC.

You can also use the NIC with the SRC VTA to allow the following:

- Immediately activate subscriptions to quota services—The SRC VTA immediately activates a subscriber's quota service when a deposit is made to the subscriber's account. In this case, the NIC maps the subscriber's identifier to the SAE reference. This scenario is for subscribers who connect to the network through routers running JunosE or Junos OS.

  If you do not set up a NIC for this purpose or you use an identifier that the NIC cannot map to an SAE reference, subscribers must log out and log in again before the SRC VTA can activate their quota services when deposits are made to their accounts.

Related Documentation

- Identifying Subscribers, SAEs, and Sessions on page 29
- Configuring Subscribers and Subscriptions to SRC VTA Services on page 44
- Configuring a NIC for the SRC VTA (SRC CLI) on page 45
- Configuring NIC Proxies for the SRC VTA on page 46

Configuring a NIC for the SRC VTA (SRC CLI)

For demonstrations and installations with few subscribers, you can configure the SRC VTA to use a NIC proxy stub, which explicitly defines a set of data mappings. However,

for most standard installation with a significant number of subscribers and multiple SAEs, you need to set up a full NIC configuration. The requirement to set up a NIC depends on your SRC VTA configuration. For example, configure a NIC if you specify an action that affects more than just the current subscriber described by an event.

To configure a NIC for the SRC VTA:

1. Use the OnePopLogin configuration scenario (see NIC Configuration Scenarios).
2. Plan and configure the NIC hosts. See Configuring the NIC (SRC CLI).
3. Add the NIC SAE agents to each SAE configuration as external plug-ins. Specify these plug-in attributes: router-name, session-id, user-type, login-name, user-ip-address.

   For information about configuring SAE plug-ins, see Configuring the SAE for External Plug-Ins (SRC CLI).
4. (Optional) Configure a NIC proxy stub. See Configuring NIC Test Data (SRC CLI) for information about configuring the NIC proxy stub.
5. Configure a NIC proxy for the SRC VTA. See "Configuring NIC Proxies for the SRC VTA" on page 46.

Related Documentation

- Before You Configure the NIC
- Locating the SAE That Manages a Subscriber for the SRC VTA on page 45
- Starting the NIC (SRC CLI)
- Configuration Statements for the NIC

Configuring NIC Proxies for the SRC VTA

For information about NIC proxies, see Overview of NIC Proxy Configuration.

You can configure a NIC proxy that passes the subscriber's identifier to a NIC resolver and receives the corresponding SAE reference. This NIC allows the SRC VTA to immediately activate a subscriber's quota service when a deposit is made to the subscriber's account. This feature is available for subscribers who connect to the network through routers running JunosE or Junos OS.

Use the following statement to configure the NIC proxy for the SRC VTA:

shared vta nic-proxy-configuration *name*

To configure the NIC proxy for the SRC VTA:

1. From configuration mode, access the statement that specifies the NIC proxy for the SRC VTA.

   ```
   [edit]
   user@host# edit shared vta nic-proxy-configuration name
   ```
2. To complete the configuration of the NIC proxy, follow the procedure described in chapter *Configuring SRC Applications to Communicate with an SAE (SRC CLI)* in the *SRC PE Network Guide*.

3. Create a reference between the shared SRC VTA group configuration and the NIC proxy configuration.

   From the **[edit shared vta group *name*]** hierarchy level execute:

   ```
   [edit shared vta group name]
   user@host# set nic-proxy nic-proxy
   ```

   Set the **nic-proxy** option to name you specified in Step 1. For more information, see "Creating and Configuring an SRC VTA Shared Group Configuration (SRC CLI)" on page 59.

Related Documentation

- Before You Configure a NIC Proxy
- Locating the SAE That Manages a Subscriber for the SRC VTA on page 45
- Configuring a NIC for the SRC VTA (SRC CLI) on page 45

CHAPTER 5

Configuration Tasks for the SRC Web Application Server

Configuring the Web Application Server (SRC CLI)

Tasks to configure the Web application server are:

1. Configure the Web application server shared cluster configuration.

   See "Configuring the Web Application Server Shared Cluster Configuration (SRC CLI)" on page 52.

2. Configure the operating properties.

   See "Configuring Local Properties for the Web Application Server (SRC CLI)" on page 51.

3. Configure the nodes of the Web application server cluster.

   See "Configuring the Nodes in the Web Application Server Cluster (SRC CLI)" on page 53.

4. Configure remote access to the application server.

   See "Configuring Remote Access to the Application Server (SRC CLI)" on page 54.

5. Configure the virtual host for the Web application, including whether to allow or deny access by specific remote clients.

   See "Configuring Virtual Hosts for the Web Applications (SRC CLI)" on page 55.

6. Configure the user accounts for the Web application.

   See "Configuring User Accounts for Web Applications (SRC CLI)" on page 56.

Related Documentation

- Overview of the Web Application Server on C Series Controllers on page 32
- Configuring the Web Application Server Shared Cluster Configuration (SRC CLI) on page 52
- Configuring the Nodes in the Web Application Server Cluster (SRC CLI) on page 53

Configuration Statements for the Web Application Server

Use the following configuration statements to configure the operating properties for the Web application server at the **[edit]** hierarchy level.

```
slot number application-server {
   java-garbage-collection-options java-garbage-collection-options;
   java-heap-size  java-heap-size;
   shared-cluster shared-cluster
}

shared application-server cluster name {
   channel-stack (udp|tcp);
   multicast-address multicast-address;
}

shared application-server cluster name nodes node address {
   node-id node-id;
}

slot number application-server web http {
   port port;
   interface interface;
}

slot number application-server web https {
   local-certificate local-certificate;
   port port;
   interface interface;
}

slot number application-server web virtual-host host-name {
   alias alias;
   allow-address allow-address;
   allow-host allow-host;
   deny-address deny-address;
   deny-host deny-host;
}

shared application-server user name

shared application-server user name authentication {
```



```
    encrypted-password encrypted-password;
    plain-text-password;
  }
```

Related Documentation

- Configuring the Web Application Server (SRC CLI) on page 49
- Configuring Remote Access to the Application Server (SRC CLI) on page 54
- Configuring Virtual Hosts for the Web Applications (SRC CLI) on page 55
- Configuring User Accounts for Web Applications (SRC CLI) on page 56
- Overview of the Web Application Server on C Series Controllers on page 32

Configuring Local Properties for the Web Application Server (SRC CLI)

To configure basic local properties:

1. From configuration mode, access the configuration statement that configures the local properties.

   user@host# **edit slot 0 application-server**

2. Configure the garbage collection functionality of the Java Virtual Machine.

   [edit slot 0 application-server]
   user@host# **set java-garbage-collection-options** *java-garbage-collection-options*

3. (Optional) If you encounter problems caused by lack of memory, change the maximum memory size available to the JRE.

   [edit slot 0 application-server]
   user@host# **set java-heap-size** *java-heap-size*

4. (Optional) Configure the cluster name. Specify the shared-cluster as **/application-server/*shared-cluster***.

   [edit slot 0 application-server]
   user@host# **set shared-cluster /application-server/***shared-cluster*

   For example, to configure a shared cluster called cluster-1:

   [edit slot 0 application-server]
   user@host# **set shared-cluster /application-server/***cluster-1*

   NOTE: If you change the shared cluster name, you must restart the local application server for the change to take effect.

5. (Optional) Verify your configuration.

   [edit slot 0 application-server]
   user@host# **show**

```
shared-cluster /application-server/cluster-1;
web {
  http {
    interface eth0;
    port 8080;
```

```
    }
    virtual-host eth0;
}
```

Related Documentation

- Configuring the Web Application Server Shared Cluster Configuration (SRC CLI) on page 52
- Configuring the Nodes in the Web Application Server Cluster (SRC CLI) on page 53
- Overview of the Web Application Server on C Series Controllers on page 32

Configuring the Web Application Server Shared Cluster Configuration (SRC CLI)

Use the following statements to configure a Web application server shared cluster configuration:

```
shared application-server cluster name {
    channel-stack (udp|tcp);
    multicast-address multicast-address;
}
```

To configure the Web application server shared cluster configuration:

1. From configuration mode, access the statement that configures the shared cluster configuration. The name you specify must match the name you configured for the local configuration at the **[edit slot 0 application-server]** hierarchy level.

   ```
   user@host# edit shared application-server cluster name
   ```

   For example, if you have the following local configuration:

   ```
   [edit slot 0 application-server]
   shared-cluster /application-server/cluster-1
   ```

   You need to specify cluster-1 as the cluster name for the shared configuration:

   ```
   user@host# edit shared application-server cluster cluster-1
   ```

2. Configure the channel stack.

   ```
   [edit shared application-server cluster cluster-1]
   user@host# set channel-stack (udp|tcp)
   ```

3. (Optional) Specify the multicast address. The multicast address is required only if UDP is selected as the channel stack.

   ```
   [edit shared application-server cluster cluster-1]
   user@host# set multicast-address multicast-address
   ```

4. (Optional) Verify your configuration.

   ```
   [edit shared application-server cluster cluster-1]
   user@host# show
   ```

   ```
   channel-stack tcp;
   [edit shared application-server cluster cluster-1]
   user@host#
   ```

Related Documentation

- Overview of the Web Application Server on C Series Controllers on page 32
- Configuring the Nodes in the Web Application Server Cluster (SRC CLI) on page 53
- Configuring the Web Application Server (SRC CLI) on page 49
- Configuring Local Properties for the Web Application Server (SRC CLI) on page 51
- Viewing the Web Application Server Cluster Status (SRC CLI)

Configuring the Nodes in the Web Application Server Cluster (SRC CLI)

Use the following statements to configure the nodes in the Web application server cluster:

```
shared application-server cluster name nodes node address {
  node-id node-id;
}
```

To configure the Web application server cluster nodes:

1. From configuration mode, access the statement that configures the cluster nodes and specify the IP address of the node.

   user@host# shared application-server cluster *name* nodes node *address* {

2. Configure the node ID for the node. The node ID is a random number you assign to the node. Each node must have a unique node ID specified as the integer type.

   [edit shared application-server cluster *name* nodes node *address*]
   user@host# **set node-id** *node-id*

3. (Optional) Verify your configuration.

   [edit shared application-server cluster *name* nodes node *address*]
   user@host# **show**

   Following is a sample output of the cluster node configuration:

```
channel-stack udp;
multicast-address 255.255.100.100;
nodes {
  node 10.1.2.3 {
    node-id 2;
  }
  node 10.1.2.4 {
    node-id 1;
  }
  node 10.1.2.5 {
    node-id 4;
  }
}
```

Related Documentation

- Overview of the Web Application Server on C Series Controllers on page 32
- Configuring the Web Application Server Shared Cluster Configuration (SRC CLI) on page 52
- Configuring the Web Application Server (SRC CLI) on page 49

- Configuring Local Properties for the Web Application Server (SRC CLI) on page 51

Configuring Remote Access to the Application Server (SRC CLI)

Before you can start using the application server, you need to configure and enable access to the application server. You can make the application server accessible through secure HTTP (HTTPS) or HTTP.

- Configuring Access to the Application Server Through Secure HTTP on page 54
- Configuring Access to the Application Server Through HTTP on page 54

Configuring Access to the Application Server Through Secure HTTP

Before you configure access to the application server through HTTPS, obtain a digital security certificate on the system.

To make the application server accessible through HTTPS:

1. From configuration mode, access the statement that configures access through HTTPS.

   user@host# **edit slot 0 application-server web https**

2. Specify which TCP port is to receive incoming connection requests for the application server.

   [edit slot 0 application-server web https]
   user@host# **set port** *port*

3. Specify the interface to be used for connections to the application server.

   [edit slot 0 application-server web https]
   user@host# **set interface** *interface*

   On a C Series Controller, use **eth1** for built-in Web applications; you can use **eth0** for demonstration applications.

4. Specify the name of the certificate on the local system.

   [edit slot 0 application-server web https]
   user@host# **set local-certificate** *local-certificate*

5. (Optional) Configure user accounts to allow specified clients to authenticate with the application server.

Configuring Access to the Application Server Through HTTP

To make the application server accessible through HTTP:

1. From configuration mode, access the statement that configures access through HTTP.

   user@host# **edit slot 0 application-server web http**

2. Specify which TCP port is to receive incoming connection requests for the application server.

   [edit slot 0 application-server web http]

```
user@host#  set port port
```

3. Specify the interface to be used for connections to the application server.

```
[edit slot 0 application-server web http]
user@host#  set interface interface
```

On a C Series Controller, use **eth1** for built-in Web applications; you can use **eth0** for demonstration applications.

4. (Optional) Configure user accounts to allow specified clients to authenticate with the application server.

Related Documentation

- Configuring the Web Application Server (SRC CLI) on page 49
- Configuring User Accounts for Web Applications (SRC CLI) on page 56
- Overview of the Web Application Server on C Series Controllers on page 32
- Overview of Digital Certificates

Configuring Virtual Hosts for the Web Applications (SRC CLI)

Use the following configuration statements to configure virtual hosts at the **[edit]** hierarchy level:

```
slot number application-server web virtual-host host-name {
  alias [alias...];
  allow-address [allow-address...];
  allow-host [allow-host...];
  deny-address [deny-address...];
  deny-host [deny-host...];
}
```

To configure virtual hosts for the Web applications:

1. From configuration mode, access the statement that configures the virtual host.

By default, SRC applications run on the virtual host **eth0**. You must configure **eth0** as a virtual host. The hostname must be unique.

```
user@host# edit slot 0 application-server virtual-host eth0
```

2. Specify the alternate DNS names or IP addresses for the virtual host.

```
[edit slot 0 application-server virtual-host eth0]
user@host#  set alias [alias ...]
```

The alias must be unique. Specify the following alias for the **eth0** virtual host:

- The IP address assigned to **eth0**.
- The name for the SRC host configured at the **[edit system host-name]** and **[edit system domain-name]** hierarchy levels.

3. Configure access to the virtual host. Specify the IP addresses for remote clients that are allowed access to the virtual host.

```
[edit slot 0 application-server virtual-host eth0]
```

```
user@host# set allow-address [allow-address...]
```

4. Configure access to the virtual host. Specify the hostnames for remote clients that are allowed access to the virtual host.

```
[edit slot 0 application-server virtual-host eth0]
user@host# set allow-host [allow-host...]
```

5. Deny access to the virtual host. Specify the IP addresses for remote clients that are denied access to the virtual host.

```
[edit slot 0 application-server virtual-host eth0]
user@host# set deny-address [deny-address...]
```

6. Deny access to the virtual host. Specify the hostnames for remote clients that are denied access to the virtual host.

```
[edit slot 0 application-server virtual-host eth0]
user@host# set deny-host [deny-host...]
```

Related Documentation

- Configuring the Web Application Server (SRC CLI) on page 49
- Overview of the Web Application Server on C Series Controllers on page 32

Configuring User Accounts for Web Applications (SRC CLI)

User accounts provide one way for clients to authenticate with the application server. For each account, you define the login name for the user, authentication information, and role. You can configure plain-text password or encrypted password as the type of authentication for user accounts. When you delete user accounts, the software verifies that the user account is not referenced by another configuration.

NOTE: Client profiles can be cached by applications for 30 minutes. If you change the password or role of a client that has been used within the last 30 minutes, it can take up to 30 minutes before these changes take effect.

If you do not want to wait 30 minutes for the changes to take effect, restart the Web application server.

Use the following configuration statements to configure user accounts at the **[edit]** hierarchy level:

```
shared application-server user name

shared application-server user name authentication {
  encrypted-password encrypted-password;
  plain-text-password;
  role [DSA | PCMM | VTA-group name;
}
```

To configure a user account:

1. From configuration mode, access the configuration statement that configures a user account and specify a username that identifies the client.

```
user@host#  edit shared application-server user name
```

The username must be unique within the system. Do not include spaces, colons, or commas in the username.

2. Configure authentication for the user account.

```
[edit shared application-server user name]
user@host#  set authentication (plain-text-password | encrypted-password)
```

where:

- **plain-text-password**—Prompt for a plain-text (unencrypted) password.
- **encrypted-password**—Password encoded with crypt. The format of encrypted passwords is "{crypt}<13-characters in a-zA-Z0-9./>".

We recommend that you not enter the password in encrypted format.

For example:

```
user@host# set authentication plain-text-password
New password: type password here
Retype new password: retype password here
```

3. Configure the role for the user account.

```
[edit shared application-server user name]
user@host# set role VTA-Quota
```

Set the role to one of the following values:

- DSA—Role for clients accessing the DSA services: dsa-service and dsa2-service
- PCMM—Role for clients accessing the DSA service: pcmm-service
- VTA-*group name*—Role for clients accessing the SOAP API for the SRC VTA. The CLI returns all SRC VTA groups configured under the **[edit shared vta group]** hierarchy with the prefix "VTA". For example, set the role to VTA-Quota for clients accessing the SOAP API for the SRC VTA group called Quota.

Related Documentation

- Configuring Remote Access to the Application Server (SRC CLI) on page 54
- SRC VTA SOAP Interface on page 31
- Enabling the SOAP Interface for an SRC VTA Group (SRC CLI) on page 94
- Methods for the Volume Tracking Application SOAP Interface on page 95

Installing Web Applications in the SRC Web Application Server

The SRC software includes a Web application server component for deploying Web applications for lab tests and demonstrations.

Use the following procedure to deploy Web applications in the SRC Web application server.

NOTE: You can deploy a Web application in the Web application server for lab tests and demonstrations. However, running non-SRC Web applications in production environments is not supported.

To deploy a Web application in the SRC Web application server:

1. Start the Web application server.
2. Prepare the Web application archive (WAR) file on a machine other than the C Series Controller.
3. Deploy the WAR file on the C Series Controller. The SRC Web application server automatically starts the Web application when a new WAR file is deployed.

   user@host> **request appsvr deploy file** *name*

   For example:

   user@host> **request appsvr deploy file ftp://host/path/ssportal.war**

Related Documentation

- Removing Web Applications from the Application Server
- Starting the Web Application Server on a C Series Controller on page 58
- Restarting the Web Application Server on a C Series Controller

Starting the Web Application Server on a C Series Controller

To start the Web application server on a C Series Controller:

user@host> **enable component appsvr**

Related Documentation

- Restarting the Web Application Server on a C Series Controller
- Stopping the Web Application Server on a C Series Controller

CHAPTER 6

Configuration Tasks for the SRC VTA

Configuring an SRC VTA Shared Group Configuration (SRC CLI)

Creating and Configuring an SRC VTA Shared Group Configuration (SRC CLI)

You can set up multiple SRC VTAs; each SRC VTA is configured as a separate group with its own shared configuration.

Use the following statements to configure an SRC VTA shared configuration:

```
shared vta group name {
   subscriber-id-solution subscriber-id-solution ;
   nic-proxy nic-proxy;
}
```

To configure an SRC VTA shared configuration:

1. From configuration mode, access the statement that configures an SRC VTA shared group configuration. For example, to configure an SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1
   ```

2. Specify the **subscriber-id-solution** option. This option specifies a data key that uniquely identifies subscriber accounts and sessions in the external database. Some settings also provide information that the NIC and the SAE use to identify subscribers.

   ```
   [edit shared vta group vta1]
   user@host# set subscriber-id-solution subscriber-id-solution
   ```

   For more information about configuring the **subscriber-id-solution** option, see "Keys Used to Specify the Subscriber ID Solution (SRC CLI)" on page 60.

3. Specify the location of the NIC proxy configuration relative to the configuration properties for the SRC VTA.

   ```
   [edit shared vta group vta1]
   user@host# set nic-proxy nic-proxy
   ```

   Set this option to the name you configured under the **[edit shared vta nic-proxy-configuration *name*]** configuration statement. See "Configuring NIC Proxies for the SRC VTA" on page 46.

   If you are using a NIC to map subscriber identifiers to an SAE, and you select an SRC VTA subscriber ID value that provides a data key for the NIC, specify the NIC proxy that uses that data key.

4. Verify your configuration.

   ```
   [edit shared vta group vta1]
   user@host# show
   ```

5. Commit your configuration.

   ```
   [edit shared vta group vta1]
   user@host# commit
   ```

NOTE: You must enable the new SRC VTA instance with the **enable component vta-name** command.

Keys Used to Specify the Subscriber ID Solution (SRC CLI)

You use the **subscriber-id-solution** option under the **shared vta group *name*** configuration statement to specify keys for managing accounts and sessions. Table 10 on page 61 shows the keys that you can specify for the SRC VTA to query the database, NIC, and SAE. For the SRC VTA to use a subscriber identifier, the plug-in event must include the corresponding NIC or SAE attributes that are listed in Table 10 on page 61 (attributes

that start with PA_). For more information about plug-in attributes, see the documentation for the SAE CORBA plug-in on the Juniper Networks Web site at

http://www.juniper.net/techpubs/software/management/src/api-index.html

Table 10: Keys That the SRC VTA Constructs to Manage Accounts and Sessions

Subscriber's Identifier	Database Key	Corresponding NIC Key	Corresponding SAE Key
accounting-id	PA_ACCOUNTING_ID	PA_ACCOUNTING_ID	IP address that is returned from the NIC lookup
user-dn	PA_USER_DN	PA_USER_DN	PA_USER_DN
interface-alias	PA_INTERFACE_ALIAS	None	None
interface-alias-and-router	PA_INTERFACE_ALIAS@ PA_ROUTER_NAME	PA_ROUTER_NAME	None
interface-and-router	PA_INTERFACE_NAME@ PA_ROUTER_NAME	PA_ROUTER_NAME	PA_INTERFACE_NAME
login-name	PA_LOGIN_NAME	PA_LOGIN_NAME	PA_LOGIN_NAME (default)
mac-address	PA_USER_MAC_ ADDRESS	None	None
primary-user-name (PPP login name or public DHCP name)	PA_PRIMARY_USER_ NAME	None	PA_PRIMARY_USER_ NAME
nas-port-id-and-router	PA_PORT_ID@ PA_ROUTER_NAME	None	None

login-name, **user-dn**, and **interface-and-router** also provide data keys for the NIC and the SAE; the other settings do not.

Related Documentation

- Overview of Managing SRC VTA Accounts and Sessions on page 29
- Overview of the SRC VTA on page 9
- Using One SRC VTA Account for Multiple Subscriber Sessions on page 92
- Managing SRC VTA Accounts and Sessions on page 30

Configuring the Connection Between the SRC VTA and the External Account and Session Database (SRC CLI)

The SRC VTA requires a relational database to store accounts and session data. The database connection information specifies how the SRC VTA connects to this database. You must configure the connection to the external database for each SRC VTA group.

NOTE: The values shown in the following sample procedure should work for a MySQL database. The values are different for an Oracle database, or other database servers. The values may also differ for different versions of the same brand of database server. You must understand how to connect to your particular brand and version of database server using JDBC before configuring the database connection.

Following is an example Oracle configuration:

```
user@host> show configuration shared vta group test database
check-valid-connection-sql 'select 1 from dual';
connection-url jdbc:oracle:thin:@//10.10.1.2:1521/vta;
datasource-mapping Oracle9i;
driver-class oracle.jdbc.OracleDriver;
max-pool-size 50;
min-pool-size 5;
password ********;
user-name vta;

user@host>
```

Use the following statements to configure the connection to the external database:

```
shared vta group name database {
    connection-url connection-url ;
    driver-class driver-class;
    user-name user-name;
    password password;
    data-source-mapping data-source-mapping;
    min-pool-size min-pool-size;
    max-pool-size max-pool-size;
    check-valid-connection-sql check-valid-connection-sql;
}
```

To configure the connection to the external database:

1. From configuration mode, access the statement that configures the connection to the external database. For example, to configure the connection for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 database
   ```

2. Configure the URL connection information. For example, to specify the connection url as jdbc:mysql://10.10.10.1:3306/vta1:

```
[edit shared vta group vta1 database]
user@host# set connection-url jdbc:mysql://10.10.10.1:3306/vta1
```

3. Configure the driver class information. For example, to configure a JDBC driver for MySQL:

```
[edit shared vta group vta1 database]
user@host# set driver-class com.mysql.jdbc.Driver
```

4. Configure the username used to access the database. For example to configure the username called admin1:

```
[edit shared vta group vta1 database]
user@host# set user-name admin1
```

5. Configure the password used to access the database. For example, to configure the password as pwd1:

```
[edit shared vta group vta1 database]
user@host# set password pwd1
```

6. Configure the data source mapping information. For example, for MySQL:

```
[edit shared vta group vta1 database]
user@host# set data-source-mapping mySQL
```

7. Specify the minimum number of simultaneous connections to the database.

```
[edit shared vta group vta1 database]
user@host# set min-pool-size min-pool-size ;
```

8. Specify the maximum number of simultaneous connections to the database.

```
[edit shared vta group vta1 database]
user@host# set max-pool-size max-pool-size ;
```

NOTE: The max-concurrency option should be set greater than or equal to the max-pool-size option; otherwise, database connections are likely to be unused, see "Overview of Configuring Event Handlers" on page 18.

9. Specify the SQL used to check whether a database connection is still valid.

```
[edit shared vta group vta1 database]
user@host# check-valid-connection-sql check-valid-connection-sql ;
```

The specified SQL depends on the type of database server. For example, with Oracle, you can use "select 1 from dual," and with MySQL or MS SQL Server, you can use "select 1".

Related Documentation

- Configuring the External Database on page 41

- Configuring the Web Application Server (SRC CLI) on page 49
- Managing SRC VTA Accounts and Sessions on page 30
- Troubleshooting Database Deadlocks on page 131

Configuring Actions for SRC VTA Event Handlers (SRC CLI)

Use the following statement to configure an action that can be invoked by one or more event handlers:

```
shared vta group name action action-name {
  function (db-engine-calculate-interim | db-engine-calculate-usage |
    db-engine-get-accounts | db-engine-terminate-session | db-engine-update-accounts
    | mailer-send | sae-set-interim-interval | sae-set-service-timeout | sae-set-user-timeout
    | sae-start-service | sae-stop-service | scripts-run-external-script |
    scripts-run-javascript);
  on-error (abort-event-processing | go-to-next-action | go-to-next-event-handler);
  parameter;
}
```

NOTE: We recommend that when you configure event handlers and their actions, you ensure that for any given event, all database operations are performed before any other operations that have permanent effects. This is because if a database error occurs—for example, due to normal contention for database records between different event threads—the SRC VTA rolls back the current database transaction (no changes are made to the database) and then restarts processing the event. If the event performs some other operation other than database operations before such an error, such as start a service, then that other operation is performed again when the event is reprocessed following the error.

1. From configuration mode, access the statement that configures an action and specify a name for the action. For example, to configure an action called act1 in the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 action atc1
   ```

2. Specify a function you want the action to call. For example, to have the action call the **db-engine-calculate-usage** function:

   ```
   [edit shared vta group vta1 action atc1]
   user@host# set function db-engine-calculate-usage
   ```

3. (Optional) Specify any input parameters required for the function.

   ```
   [edit shared vta group vta1 action atc1]
   user@host# set parameter
   ```

The parameters that you need to set depend on what function you specified. Some functions require no input parameters. You can use the "?" completer to see what parameters need to be set, or see "Overview of Configuring Actions" on page 22 for a complete list of functions and their associated input parameters.

4. Specify what you want the action to do in response to an error. For example, to configure the action to abort processing if an error occurs:

   ```
   [edit shared vta group vta1 action atc1]
   user@host# set on-error abort-event-processing
   ```

5. Verify your configuration.

   ```
   [edit shared vta group vta1 action atc1]
   user@host# show
   ```

   ```
   user@host# show
   function db-engine-calculate-usage;
   on-error abort-event-processing;

   [edit shared vta group vta1 action act1]
   user@host#
   ```

6. Commit your configuration.

   ```
   [edit shared vta group vta1 action atc1]
   user@host# commit
   ```

Related Documentation

- Overview of Configuring Actions on page 22
- Overview of Configuring Event Handlers on page 18
- Configuring Event Handlers (SRC CLI) on page 65

Configuring Event Handlers (SRC CLI)

Use the following statements to configure event handlers:

```
shared vta group name event-handler event-handler-name {
  events events;
  actions actions;
  condition condition;
  priority priority;
}
```

To configure an event handler:

NOTE: We recommend that when you configure event handlers and their actions, you ensure that for any given event, all database operations are performed before any other operations that have permanent effects. This is because if a database error occurs—for example, due to normal contention for database records between different event threads—the SRC VTA rolls back the current database transaction (no changes are made to the database) and then restarts processing the event. If the event performs some other operation other than database operations before such an error, such as start a service, then that other operation is performed again when the event is reprocessed following the error.

1. From configuration mode, access the statement that configures SRC VTA event handlers. For example, to configure an event handler called evh1 in the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 event-handler evh1
   ```

2. Specify an event. For example, to have the event handler handle start events for the service named QuotaInternet:

   ```
   [edit shared vta group vta1 event-handler evh1]]
   user@host# set events service-start:QuotaInternet
   ```

3. Specify previously configured actions that the event handler invokes in response to the event. For example, to specify the previously configured actions called act1, act2, and act3:

   ```
   [edit shared vta group vta1 event-handler evh1]
   user@host# set actions [act1 act2 act3]
   ```

4. Specify the condition the event handler uses to determine whether it should handle the event.

   ```
   [edit shared vta group vta1 event-handler evh1]
   user@host# set condition condition
   ```

   Specify the condition as script written in the JavaScript programming language that must return one of the following Boolean values:

   - True—Event handler should handle the event.
   - False—Event handler should not handle the event.

5. Specify the priority of the event handler.

   ```
   [edit shared vta group vta1 event-handler evh1]
   user@host# set priority priority
   ```

   Event handlers with the lower number priority have the highest priority and evaluate the event first.

6. Verify your configuration.

```
[edit shared vta group vta1 event-handler evh1]
user@host# show
```

Related Documentation

- How the SRC VTA Works on page 12
- Overview of Configuring Actions on page 22
- Overview of Configuring Event Handlers on page 18
- Configuring Actions for SRC VTA Event Handlers (SRC CLI) on page 64

Configuring Event Queues for SRC VTA Groups (SRC CLI)

Use the following statement to configure the event queue:

```
shared vta group name queue {
  max-concurrency max-concurrency;
  persistent;
}
```

To configure the event queue:

1. From configuration mode, access the statement that configures the event queue. For example, to configure the event queue for the SRC VTA group called vta1:

```
[edit]
user@host# edit shared vta group vta1 queue
```

2. Specify the maximum number of threads consuming events from the queue and passing them to the configured event handlers.

```
[edit shared vta group vta1 queue]
user@host# set max-concurrency max-concurrency
```

NOTE: **max-concurrency should be set greater than or equal to the max-pool-size option; otherwise, database connections are likely to be unused, see** "Configuring the Connection Between the SRC VTA and the External Account and Session Database (SRC CLI)" on page 62.

3. (Optional) Specify whether the event queue is persistent (events are saved to disk) or non-persistent (events are saved in memory). If set, events are saved to disk. By default, the event queue is set to non-persistent.

```
[edit shared vta group vta1 queue]
user@host# set persistent
```

Related Documentation

- Overview of Configuring Event Handlers on page 18

Configuring the DB-Engine Processor for the SRC VTA Group (SRC-CLI)

The SRC VTA uses the db-engine processor to update database and subscriber accounts. The db-engine processor works as a proxy to the external database. It calculates usage, updates account balances, retrieves account and active session data, and sets initial balances of subscriber accounts. A subscriber account is a record of credit and debit entries in the database that track a subscriber's use of a particular network resource. You can also use the db-engine processor to dynamically adjust interim accounting intervals based on a service or based on a subscriber's remaining resources and use of the network for that service. Configuring the db-engine processor involves the following tasks:

Recording Balance Changes and Calculating the Average Usage Rate (SRC CLI)

You can configure the db-engine processor to record account balance changes and calculate the average rate at which the subscriber is consuming volume in units per second, called the averageUsageRate.

The **session-history-depth** option determines the number of historical service session records that are used to calculate the averageUsageRate for a service. When you configure the interim accounting interval, you can specify the variable averageUsageRate_*serviceName*. If this variable is specified, the SRC VTA calculates the averageUsageRate for the specified service by querying the history sessions of the service over the most recent *X* hours, where *X* is defined by the **session-history-depth** option.

For more information, see "Configuring the Interim Account Interval and Usage Metric of a Service in the External Database (SRC CLI)" on page 71 and "Variables Used to Define the Interim Accounting Interval for Services" on page 73.

Use the following statements to configure the SRC VTA to record balance changes and specify the session history depth:

```
shared vta group name processor db-engine {
   record-balance-change;
   session-history-depth;
}
```

NOTE: **The CLI editing level must be set to expert to set the session-history-depth option.**

To configure the SRC VTA to record balance changes and specify the session history depth:

1. From configuration mode, access the statement used to configure the db-engine processor. For example, to configure the db-engine processor for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor db-engine
   ```

2. (Optional) Configure the SRC VTA to record balance changes.

   ```
   [edit shared vta group vta1 processor db-engine]
   user@host# set record-balance-change
   ```

   This option is set or not set. If it is set, every account balance change action is recorded in the database. Not setting this option requires fewer database updates and less database space.

3. (Optional) Specify the session-history-depth (in hours) used to calculate the averageUsageRate for a service.

   ```
   [edit shared vta group vta1 processor db-engine]
   user@host# set session-history-depth
   ```

Configuring the Initial Balance and Status of a Subscriber Account in the External Database (SRC CLI)

Use the following statements to set the initial balance and status of a subscriber account:

```
shared vta group name processor db-engine account account{
   initial-balance initial-balance;
   initial-status initial-status;
}
```

To set the initial balance and status of a subscriber account:

1. From configuration mode, access the statement used to configure database accounts. For example, to configure an account called BoughtQuota in the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor db-engine account BoughtQuota
   ```

2. Specify the initial balance for the account.

   ```
   [edit shared vta group vta1 processor db-engine account BoughtQuota]
   user@host# set initial-balance initial-balance
   ```

Enter the value as an integer in the range -9223372036854775807 through 9223372036854775807.

3. Specify the initial status for the account.

```
[edit shared vta group vta1 processor db-engine account BoughtQuota]
user@host# set initial-status initial-status
```

Enter the initial status as a text string—for example “active.”

Configuring Scripts That Update Accounts (SRC CLI)

You can set up scripts to update balances in the accounts from which the usage of a service is charged and update accounts by assigning values to variables for the account balances.

Use the following statements to configure scripts that update accounts:

```
shared vta group name processor db-engine account-update-script name{
  script script;
}
```

To configure scripts that update accounts:

1. From configuration mode, access the statement used to configure scripts that update accounts. For example, to configure a script called DebtQuotaUsage in the SRC VTA group called vta1:

```
[edit]
user@host# edit shared vta group vta1 processor db-engine account-update-script
  DebtQuotaUsage
```

2. Specify the script parameters.

```
[edit shared vta group vta1 processor db-engine account-update-script
  DebtQuotaUsage]
user@host# set script script
```

Enter a JavaScript program that updates a subscriber’s account. The script can refer to the name of any attributes in the event being processed. An account can be updated by assigning values to the following parameters:

- balance_<accountName>—Values written to this parameter are put in the balance field of the account.
- status_<accountName>—Values written to this parameter are put in the status field of the account.
- lastUpdateTime_<accountName>—Values written to this parameter are put in the last_update_time field of the account.

  For example:

```
[edit shared vta group vta1 processor db-engine account-update-script
  DebtQuotaUsage]
```

```
user@host# set script <balance_PeriodicQuota>=<balance_PeriodicQuota>-\
  <currentUsage>;<lastUpdateTime_PeriodicQuota>=<currentTime>;
```

Configuring the Interim Account Interval and Usage Metric of a Service in the External Database (SRC CLI)

To configure how the db-engine processor manages services, you need to specify the usage metric and the interim accounting interval options. Both of these options are configured with JavaScript variables.

Use the following statement to configure how the SRC VTA manages services:

```
shared vta group name processor db-engine service name {
   interim-interval-function interim-interval-function;
   usage-metric usage-metric;
}
```

NOTE: The CLI editing level must be set to expert for this task.

To configure how the db-engine processor manages services:

1. From configuration mode, access the statement that configures the db-engine processor to manage services and specify the name of the service. For example, to configure a service called QuotaLocal in the SRC VTA group vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor db-engine service QuotaLocal
   ```

2. Specify the JavaScript function that defines the usage metric for the service.

   ```
   [edit shared vta group vta1 processor db-engine service QuotaLocal]
   user@host# set usage-metric usage-metric
   ```

3. Specify the JavaScript variables that define the interim accounting interval for the service.

   ```
   [edit shared vta group vta1 processor db-engine service QuotaLocal]
   user@host# set interim-interval-function interim-interval-function
   ```

Configuring SRC VTA Database Optimization (SRC CLI)

To optimize SRC VTA database operations, you can set the **sessions-terminated-frequently** option.

NOTE: Setting the **sessions-terminated-frequently** option is an irreversible act. You can set this option to true, but you cannot change it back to false.

To use the **sessions-terminated-frequently** optimization feature with a set of SRC VTAs, you must follow this specific process:

1. All C Series Controllers running SRC VTAs must be running SRC software release 4.3 or later; otherwise, you must upgrade the system.
2. Send all SAE events to only one SRC VTA.
3. Wait until the removed SRC VTAs (from Step 2) have finished processing events in their event queues.
4. Set the **sessions-terminated-frequently** option to true.
5. Wait until all SRC VTAs have reacted to the option being set. The SRC VTA information log displays the following message"DBEngine will assume VTA sessions are terminated frequently."
6. Resume distributing SAE events to all SRC VTAs.

Use the following statement to configure SRC VTA database optimization:

```
shared vta group name processor db-engine {
   sessions-terminated-frequently ;
}
```

To configure SRC VTA database optimization:

1. From configuration mode, access the statement used to configure the db-engine processor. For example, to set up SRC VTA database optimization for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor db-engine
   ```

2. Set the **sessions-terminated-frequently** option.

   ```
   [edit shared vta group vta1 processor db-engine]
   user@host# set sessions-terminated-frequently
   ```

Variables Used to Define the Interim Accounting Interval for Services

Current Service Variables

Use the variables described in this section to define a formula for the interim accounting interval.

lastInterimTime

- Last interim time interval.
- Value—Number of seconds in the range 1–2147483647

sessionLength

- Length of the current session.
- Value—Number of seconds in the range 0–2147483647; value is 0 when the SRC VTA is calculating the interim time of start events. For other events, value is set by the PA_SESSION_TIME attribute.

maxUsageRate

- Maximum rate at which the subscriber can use network resources according to the formula described in Table 11 on page 75.
- Value—Integer in the range 0–9223372036854775807
- Guidelines—This formula corresponds to the usage formula for the same service as the interim formula.

The maxUsageRate variable is calculated for a service by means of the following values for the variables in the corresponding usage formula:

- upStreamBytes=PA_UPSTREAM_BANDWIDTH
  - downStreamBytes=PA_DOWNSTREAM_BANDWIDTH
  - interimTime=lastInterimTime
  - upStreamPackets=0
  - downStreamPackets=0

If you use the parameters upStreamPackets (PA_IN_PACKETS) and downStreamPackets (PA_OUT_PACKETS) in the usage formula and at the same time maxUsageRate in the interim interval formula, the maxUsageRate is not accurate, because the values for maximum upStreamPackets and downStreamPackets are unknown.

averageUsageRate

- Average rate at which the subscriber is consuming volume in units per second. The unit can be a value such as dollars, bytes, or packets. The type of unit depends on the value specified in the formula. Measurement begins when the service starts.
- Value—Integer in the range 0–9223372036854775807; the value is 0 when the SRC VTA is calculating the interim time of start events.

For other events, the value is the usage formula divided by PA_SESSION_TIME. The usage formula is calculated from PA_IN_PACKETS, PA_OUT_PACKETS, PA_OUT_OCTETS, PA_IN_OCTETS, and PA_SESSION_TIME.

latestUsageRate

- Rate of service usage since the last usage report.
- Value—Integer in the range 0–9223372036854775807; the value is 0 when the SRC VTA is calculating the interim time of start events.

The value is calculated by using the result of the usage formula divided by the length of the service session since the previous usage report for the same service.

Other Service Variables

Use the variables described in this section to define an interim accounting formula that depends on usage of another service tracked by the SRC VTA.

System requirements to calculate service usage, in the form of the averageUsageRate and the sessionLength variables, can affect system performance. Using a longer interim interval means that there are fewer interim events to process, which requires fewer system resources.

averageUsageRate_<serviceName>

- Average rate at which the service is consuming volume in units per second. The unit can be a value such as dollars, bytes, or packets. The type of unit depends on the value specified in the interim accounting formula. Measurement begins when the service starts.
- Value—Integer in the range 0–9223372036854775807; the value is 0 when the SRC VTA is calculating the interim time of start events.
- Guidelines—Service names can contain alphanumeric characters and dashes (–).

sessionLength_<serviceName>

- Length of a service session for the service.
- Value—Integer in the range 0–2147483647; the value is 0 when the SRC VTA is calculating the interim time of start events.
- Guidelines—Service names can contain alphanumeric characters and dashes (–).

Account Balance Variable

Use the variable described in this section to obtain balance information from each of the subscriber's accounts.

balance_<accountName>

- Balance for the specified account before the new usage value is applied.
- Value—Integer in the range 0–9223372036854775807
- Example—balance_PeriodicQuota refers to the balance for the PeriodicQuota account.

Sample Formulas for Interim Accounting Interval

Table 11 on page 75 provides examples of formulas to dynamically adjust the interim accounting interval for a service.

Table 11: Examples of Interim Accounting Interval

Formula	Description
return 900	Accounting interval is fixed at 900 seconds (15 minutes).
return (<balance_Periodic> + <balance_Bought>) /<maxUsageRate>	Minimum time required for the subscriber to empty the periodic and bought accounts.
return <sessionLength> >= 60*15 ? (<balance_Periodic> + <balance_Bought>) /<averageUsageRate>/2 : (<balance_Periodic> + <balance_Bought>) /<maxUsageRate>	Half the time required for the subscriber to empty the accounts at the current average rate, or the minimum time if the session is shorter than 15 minutes. Because the average rate may not be representative early in the session, check when the account is half empty.

Variables Used to Define the Usage Metric for Services

A usage metric is a formula that calculates usage based on an accounting event for the specified service. The formula is specified in the form of a JavaScript program and it can specify variables. When you configure a service, you need to specify the interim interval and the usage metric.

For the usage metric, you define a formula that determines the use of network resources for a service. Each service in the SRC VTA can use a different formula. You can configure the SRC VTA software to evaluate this formula for every accounting event it receives from the SAE for each quota service. The SRC VTA can then debit the result from the accounts.

Use the variables described in this section to define the formula. See "Sample Formulas for Usage Metrics for the SRC VTA" on page 77 for examples of usage metric formulas.

downStreamBytes

- Amount of data that the subscriber downloaded from the network since the last accounting event.
- Value—Number of bytes in the range 0–9223372036854775807

downStreamPackets

- Number of data packets that the subscriber downloaded from the network since the last accounting event.
- Value—Integer in the range 0–9223372036854775807
- Guidelines—Do not use downStreamPackets in a usage formula and maxUsageRate in the interim interval formula for the same service at the same time.

interimTime

- Time since the last accounting event.
- Value—Number of seconds in the range 0–2147483647
- Guidelines—Generally, this value equals the interim accounting interval; however, it may exceed the interim accounting interval if an accounting event is lost. Similarly, the value may be less than the interim accounting interval if a stop event occurs in the middle of an accounting interval.

upStreamBytes

- Amount of data that the subscriber uploaded to the network since the last accounting event.
- Value—Number of bytes in the range 0–9223372036854775807

upStreamPackets

- Number of data packets that the subscriber uploaded to the network since the last accounting event.
- Value—Integer in the range 0–9223372036854775807
- Guidelines—Do not use upStreamPackets in a usage formula and maxUsageRate in the interim interval formula for the same service at the same time.

Sample Formulas for Usage Metrics for the SRC VTA

Table 12 on page 77 provides examples of usage formulas.

Table 12: Examples of Formulas That Calculate Use of Network Resources

Formula	Description	Function
return <upStreamBytes> + <downStreamBytes>	Number of bytes sent and received by the subscriber.	Tracks volume of data that the subscriber transfers.
return 2*<upStreamBytes> + <downStreamBytes>	Twice the number of sent bytes plus the number of received bytes.	Allows higher charges for subscribers who are operating servers.
return <interimTime>	Time the subscriber is connected.	Tracks time that the subscriber connects rather than volume of data transfer.
return <downStreamBytes>/<interimTime>	Rate of downstream data transfer.	Allows higher charges for higher transfer rates.
QuotaInternet formula: return <upStreamBytes> + <downStreamBytes>– (<upStreamPackets> + <downStreamPackets>)*20 QuotaLocal formula: return (<upStreamBytes> + <downStreamBytes>– (<upStreamPackets> + <downStreamPackets>)*20)/2	Formulas for separate, complementary services in a single SRC VTA. The following expression returns the total number of bytes in the IP headers of packets uploaded and downloaded by the service, and as such is not subscriber data. It is not counted as usage. (<upStreamPackets>+ <downStreamPackets>)*20	Provides support for two services: QuotaInternet for Internet service and QuotaLocal for local service. Allows higher charges for Internet service than for local service. By allocating a fixed usage limit for both services to each subscriber, the formula encourages subscribers to access local resources due to decreased cost.

Related Documentation

- Overview of Adjusting the Interim Accounting Interval on page 30

- Overview of Managing SRC VTA Accounts and Sessions on page 29
- How the SRC VTA Works on page 12

Configuring the Mailer Processor for the SRC VTA Group (SRC CLI)

Configuring the mailer processor involves the following tasks:

- Configuring the SRC VTA Mailer Processor to Send E-Mail Notifications (SRC CLI) on page 78
- Configuring the SRC VTA to Send E-Mail Notifications (SRC CLI) on page 79

Configuring the SRC VTA Mailer Processor to Send E-Mail Notifications (SRC CLI)

Use the SRC VTA mailer processor to specify the SMTP server to use for e-mail messages that the SRC VTA sends to subscribers.

Use the following statement to configure the mailer processor:

```
shared vta group name processor mailer {
  smtp-server smtp-server;
  smtp-server-password smtp-server-password;
  smtp-server-port smtp-server-port ;
  smtp-server-user smtp-server-user;
}
```

1. From configuration mode, access the statement that configures the SRC VTA mailer processor. For example, to configure the mailer processor for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor mailer
   ```

2. Configure the SMTP server used for outgoing e-mail by specifying the host name or IP address of the SMTP server. For example, set the SMTP server to a host called mail.example.com:

   ```
   [edit shared vta group vta1 processor mailer]
   user@host# set smtp-server mail.example.com
   ```

3. Configure the password used to access the SMTP server.

   ```
   [edit shared vta group vta1 processor mailer]
   user@host# set smtp-server-password smtp-server-password
   ```

4. Configure the port used to connect to the SMTP server.

   ```
   [edit shared vta group vta1 processor mailer]
   user@host# set smtp-server-port smtp-server-port
   ```

5. Configure the username for accessing the SMTP server. mail.example.com:

   ```
   [edit shared vta group vta1 processor mailer]
   user@host# set smtp-server-user smtp-server-user
   ```

6. Commit your configuration.

```
[edit shared vta group vta1 processor mailer]
user@host# commit
```

Configuring the SRC VTA to Send E-Mail Notifications (SRC CLI)

To configure the SRC VTA to send e-mail notifications, you need to configure an event handler that specifies an action that calls the *mailer-send* function.

1. From configuration mode, access the statement that configures an action for the SRC VTA event handler. For example, to create an action called act1 for the SRC VTA group called vta1:

```
[edit]
user@host# edit shared vta group vta1 action act1
```

2. Configure the action to call the *mailer-send* function.

```
[edit shared vta group vta1 action act1]
user@host# set function mailer-send
```

3. Configure the parameters for the mailer-send function.

```
[edit shared vta group vta1 action act1]
user@host# set parameter
```

   Specify the following parameters for the mailer-send function:

   - Recipient—Address of the e-mail recipient
   - From—Address of the e-mail sender
   - Subject—Subject of the e-mail
   - Text—Text of the e-mail

4. Configure what to do if an error occurs. For example, to configure the event handler to go to the next action if an error occurs:

```
[edit shared vta group vta1 action act1]
user@host# set on-error go-to-next-action
```

5. Configure the event handler and specify the action you configured in the previous steps of this procedure.

   See "Configuring Event Handlers (SRC CLI)" on page 65.

Configuring the SRC VTA Scripts Processor (SRC CLI)

The scripts processor can invoke external executable scripts or JavaScript programs. We recommend using JavaScript programs for better performance.

- External scripts are executable programs, such as shell scripts, that are available on the SRC VTA's host. Each external script can perform a task and return a value. If the script returns a value, the value can be added to the current event as an event attribute.
- JavaScript programs are used to process SRC VTA event attributes. For example, a JavaScript program can convert SRC VTA event attributes in a timestamp to a date string and add them to the event as a new attribute. The attributes can then be used for subsequent actions, such as sending an e-mail notification to the subscriber. The JavaScript program can refer to any attributes of the event being processed and return a value.

Configuring the SRC VTA scripts processor involves the following tasks:

- Configuring the SRC VTA to Run Scripts on page 80
- Configuring JavaScript Programs on page 80
- Configuring External Scripts on page 81

Configuring the SRC VTA to Run Scripts

To configure the scripts processor:

1. Configure a JavaScript program or an external script.
   - See "Configuring JavaScript Programs" on page 80
   - See "Configuring External Scripts" on page 81
2. Configure an action for the event handler that calls the appropriate script function.

   The action specifies a function that the scripts processor performs on events. You can set up an action to run either an external script or a JavaScript program.
   - Specify the *scripts-run-javascript* function if the script is a JavaScript.
   - Specify the *scripts-run-external-script* function if the script is an external script.

   When you specify these functions, you need to specify the name of a script you previously configured by using the **shared vta group id processor scripts** statement. If the script configuration contains a return-attribute and return-type, an attribute with that name and type are added to the event after the script is executed.

   See "Configuring Actions for SRC VTA Event Handlers (SRC CLI)" on page 64
3. Configure an event handler and specify the action you configured in Step 2.

   See "Configuring Event Handlers (SRC CLI)" on page 65

Configuring JavaScript Programs

Use the following statement to configure a JavaScript program for the SRC VTA:

```
shared vta group name processor scripts javascript name {
  script script;
  return-type return-type;
  return-attribute return-attribute;
}
```

1. From configuration mode, access the statement that configures a JavaScript script and specify the name of the JavaScript program. For example, to configure a JavaScript script called js1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor scripts javascript js1
   ```

2. Specify a function body in JavaScript.

   ```
   [edit shared vta group vta1 processor scripts javascript js1]
   user@host# set script script
   ```

   To refer to the event attributes being processed, include the attribute name delimited by angle brackets (<and>). The JavaScript program can verify whether the event has the referenced attribute. If a referenced attribute does not exist in the event, the attribute's value is null. The JavaScript program must return a value.

3. (Optional) Specify the Java type of the return attribute.

   ```
   [edit shared vta group vta1 processor scripts javascript js1]
   user@host# set return-type return-type
   ```

   Where the **return-type** is one of the following:

   - Integer
   - Long
   - Float
   - Double
   - String
   - Boolean

4. (Optional) Specify the name of the return attribute.

   ```
   [edit shared vta group vta1 processor scripts javascript js1]
   user@host# set return-attribute return-attribute
   ```

   Specify an attribute that provides the return value of the script as a valid Java identifier that subsequent actions and event handlers can refer to. If this attribute is not set, the return value is not added to the event as an event attribute.

   JavaScript scripts must return an attribute value. The name of a return attribute cannot start with an underscore (_) because these event attributes are reserved for internal use.

Configuring External Scripts

Use the following statement to configure an external script for the SRC VTA:

```
shared vta group name processor scripts external-script name {
  full-path full-path;
```

```
    parameters parameters;
    return-type return-type;
    return-attribute return-attribute;
}
```

1. From configuration mode, access the statement that configures an external script and specify the name of the external script. For example, to configure an external script called script1:

   ```
   [edit]
   user@host# edit shared vta group vta1 processor scripts external-script script1
   ```

2. Specify the full path name to the external script.

   ```
   [edit shared vta group vta1 processor scripts external-script script1]
   user@host# set full-path full-path
   ```

   You must specify a fully qualified path to an executable program on the local file system.

3. Specify the parameters required by the script as [a b c]. When an action invokes the script, the action must supply values for these parameters in the same order as they are defined here.

   ```
   [edit shared vta group vta1 processor scripts external-script script1]
   user@host# set parameters parameters
   ```

4. (Optional) Specify the Java type of the return attribute.

   ```
   [edit shared vta group vta1 processor scripts external-script script1]
   user@host# set return-type return-type
   ```

   Where return-type is one of the following:

   - Integer
   - Long
   - Float
   - Double
   - String
   - Boolean

5. (Optional) Specify the name of the return attribute.

   ```
   [edit shared vta group vta1 processor scripts external-script script1]
   user@host# set return-attribute return-attribute
   ```

   Specify an attribute that provides the return value of the script as a valid Java identifier that subsequent actions and event handlers can refer to. If this attribute is not set, the return value is not used by the event. The external script returns the value by printing to standard output.

NOTE: The name of a return attribute cannot start with an underscore (_) because these event attributes are reserved for internal use.

Related Documentation

- Configuring Scripts That Update Accounts (SRC CLI) on page 70
- Configuring the SRC VTA to Run Scripts on page 80

Configuring SRC VTA Logging (SRC CLI)

The SRC VTA generates messages that you can save in logs—either by writing the messages to text files or by using the system log facilities. Configuring SRC VTA logging involves the following tasks:

- Logging SRC VTA Messages to a Text File on page 83
- Logging SRC VTA Messages to a System Logging Server on page 87

Logging SRC VTA Messages to a Text File

Use this procedure to configure the SRC VTA to save messages, such as event messages, SRC VTA startup messages, errors, and so on, in a text file.

Use the following statement to configure the SRC VTA to save messages in text files:

```
shared vta group name logger name file {
    filename filename;
    filter filter;
    maximum-file-size maximum-file-size;
    rollover-filename rollover-filename;
}
```

1. From configuration mode, access the statement that configures logging to a text file. For example, to configure a logger called vta1-logger for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 logger vta1-logger file
   ```

2. Specify the path and filename of the current log file. For example:

   ```
   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set filename pathname/filename.log
   ```

   Make sure you have write access to the folder.

3. (Optional) Specify a filter that determines the type of messages that this log file contains.

   ```
   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set filter filter
   ```

The filter is specified in an expression. The software filters events by evaluating each subexpression from left to right. When the software finds a match, it logs or ignores the message accordingly. You can specify an unlimited number of subexpressions. The order in which you specify the subexpressions affects the result. Expressions have the format:

singlematch [,singlematch]

Where

singlematch—[!] (<category> | ([<category>]/[<severity>] | [<minimumSeverity>]-[<maximumSeverity>]))

- !—Do not log matching events.
- <category>—SRC component that generated the event message. To log only events in a specific category, you can define the category, which is a text string that matches the name of a category. The text string is not case sensitive. For the names of categories, view a log file for a default filter. Juniper Networks Technical Assistance Center (JTAC) can also provide category names.
- [<severity>] | [<minimumSeverity>]-[<maximumSeverity>]—Name or number in the range 1–127. A higher number indicates a higher severity level. Table 13 on page 84 shows common severity levels that you can specify by name.

Table 13: Named Severity Levels

Name	Severity Level
logmin	1
debug	10
info	20
notice	30
warning	40
error	50
crit	60
alert	70
emerg	80
panic	90
logmax	127

You can set up *event trail* logging for the SRC VTA, which is helpful when debugging the SRC VTA configuration. Event trail logging logs messages that show you exactly how the SRC VTA is handling each event as it passes through the SRC VTA. You can follow a particular event as it passes through the SRC VTA and see exactly how each event handler and action handles that event.

To set up event trail logging, set the filter option to **EventTrail/9-**. Enabling event trail logging generates messages such as the following:

```
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA received event:
    Event type = SAEPluginEvent
    Event name = USERSTART
    Attributes:
      PA_ACCT_INTERIM_TIME = 0
      PA_DOMAIN = vtatest.com
      PA_EVENT_TIME = 1321471198
      PA_EVENT_TIME_MILLISECOND = 1321471198995
      PA_INTERFACE_NAME = ip10.227.1.96
      PA_INTERFACE_SPEED = 0
      PA_IN_OCTETS = 0
      PA_IN_PACKETS = 0
      PA_LAC_IP = 0
      PA_LOGIN_NAME = vtatest1@vtatest.com
      PA_NAS_INET_ADDRESS = 0.0.0.0
      PA_NAS_IP = 0.0.0.0
      PA_NAS_PORT = 0
      PA_OUT_OCTETS = 0
      PA_OUT_PACKETS = 0
      PA_PRIMARY_USER_NAME = vtatest1@vtatest.com
      PA_ROUTER_NAME = sim
      PA_ROUTER_TYPE = junos
      PA_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_SESSION_TIME = 0
      PA_SESSION_TIMEOUT = -1
      PA_SSP_HOST = kimberley
      PA_TERMINATE_CAUSE = 0
      PA_TUNNEL_ID = 0
      PA_TUNNEL_SESSION_ID = 0
      PA_USER_DN =
uniqueId=vtatest1,ou=group01,retailerName=vtatest,o=Users,o=UMC
      PA_USER_INET_ADDRESS = 10.227.1.96
      PA_USER_IP_ADDRESS = 10.227.1.96
      PA_USER_SESSION_HANDLE = SAE:KcVGVRM6OAwL0AAU@sim
      PA_USER_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_USER_TYPE = AUTHINTF
      PA_VPN_ID =
      currentTime = 1321471199007
      processingUuid = afe51007-7e35-465b-a515-3be12735afd6
      subscriberId = vtatest1@vtatest.com
    SAE information:
      SAE references =
[IOR:000000000000003549444C3A736D67742E6A756E697065722E6E657
42F7361652F536572766963654163746976174696F6E456E67696E653A312E30000000000000000100
0000000000006C000102000000000D31302E3232372E362E313236000022610000001468696D6265726
C65792F736165504F412F534145000000020000000000000008000000004A414300000000010000002
0000000000010001000000010501000100010109000000020501000100010100]
      NIC proxy namespace = /nicProxies/np1
```

```
      VTA subscriber ID solution = [Subscriber ID = PA_LOGIN_NAME, NIC key
= PA_LOGIN_NAME, SAE key = LOGIN_NAME]
      SAE subscriber ID = LOGIN_NAME
    SAE subscriber references: 0
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA ignoring event.
              .
              .
              .
              .
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  Event handler NoQuota: Received event.
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event handler NoQuota: Ignoring event cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
14:19:59.040 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  VTA completed processing event.
```

This is only an excerpt of the log and does not show all messages. Notice that each message starts with a unique ID generated when the SRC VTA starts processing an event—for example, “Event trail afe51007-7e35-465b-a515-3be12735afd6.”

Enabling debug log messages has a negative effect on system performance. Do not enable debug log messages unless JTAC instructs you to do so.

You can define a severity level as follows:

- Specify an explicit severity. For example:

  warning—Defines only warning messages

- Specify a minimum severity and a maximum severity. For example:

  info-warning—Defines messages of minimum severity level of info and a maximum severity level of warning

- Accept the default minimum (logmin) or maximum (logmax) severity by omitting the minimum or maximum severity. For example:

  info-—Defines messages of minimum severity level info and maximum severity level logmax

  -warning—Defines messages of minimum severity level logmin and maximum severity level warning

- Specify no severity to log all event messages.

- Table 14 on page 87 shows some examples of filters.

Table 14: Examples of Filters for Event Messages

Syntax	Event Messages Saved
/	All event messages
/info-	Event messages of level info and higher from all categories
vta/debug	Debug events from the SRC VTA category only
!vta,/debug	All debug events except those from the SRC VTA category
!VtaMsg/info-,vtaMsg,vta	All messages from the SRC VTA category, except those from the VtaMsg category with level lower than info

4. Specify the maximum file size. This option disables or enables and sets the maximum size of the log file and the rollover file.

   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set maximum-file-size *maximum-file-size*

   This is specified in number of kilobytes in the range 0–4294967295. The default is 1000000.

   Do not set the maximum file size to a value greater than the available disk space.

5. Specify the rollover filename.

   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set rollover filename *rollover filename*

   Specify the path and filename of the rollover log file. When the log file reaches the maximum size, the software closes the log file and renames it with the name you specify for the rollover file. If a previous rollover file exists, the software overwrites it. The software then reopens the log file and continues to save event messages in it.

   For example:

   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set rollover filename *vta_debug.alt*

Logging SRC VTA Messages to a System Logging Server

Use this procedure to configure the SRC VTA to save messages, such as event messages, SRC VTA startup messages, errors, and so on, on a system logging server.

Use the following statement to configure the SRC VTA to save messages on a system logging server:

```
shared vta group name logger name syslog {
  filter filter;
```

```
    host host;
}
```

1. From configuration mode, access the statement that configures logging to the system log facility. For example, to configure a logger called vta1-logger for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 logger vta1-logger syslog
   ```

2. (Optional) Specify a filter that determines the type of messages that this log file contains.

   ```
   [edit shared vta group vta1 logger vta1-logger syslog]
   user@host# set filter filter
   ```

   The filter is specified in an expression. The software filters events by evaluating each subexpression from left to right. When the software finds a match, it logs or ignores the message accordingly. You can specify an unlimited number of subexpressions. The order in which you specify the subexpressions affects the result. Expressions have the format:

   singlematch [,singlematch]

   Where

   singlematch—[!] (<category> | ([<category>]/[<severity>] | [<minimumSeverity>]-[<maximumSeverity>]))

   - !—Do not log matching events.
   - <category>—SRC component that generated the event message. To log only events in a specific category, you can define the category, which is a text string that matches the name of a category. The text string is not case sensitive. For the names of categories, view a log file for a default filter. Juniper Networks Technical Assistance Center (JTAC) can also provide category names.
   - [<severity>] | [<minimumSeverity>]-[<maximumSeverity>]—Name or number in the range 1–127. A higher number indicates a higher severity level. Table 15 on page 88 shows common severity levels that you can specify by name.

Table 15: Named Severity Levels

Name	Severity Level
logmin	1
debug	10
info	20
notice	30
warning	40

Table 15: Named Severity Levels *(continued)*

Name	Severity Level
error	50
crit	60
alert	70
emerg	80
panic	90
logmax	127

For the SRC VTA, you can perform *event trail* logging, which is helpful when debugging the SRC VTA configuration. Event trail logging logs messages that show you exactly how the SRC VTA is handling each event as it passes through the SRC VTA. You can follow a particular event as it passes through the SRC VTA and see exactly how each event handler and action handles that event.

To set up event trail logging, set the filter option to **filter EventTrail/9-**. Enabling event trail logging generates messages such as the following:

```
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA received event:
    Event type = SAEPluginEvent
    Event name = USERSTART
    Attributes:
      PA_ACCT_INTERIM_TIME = 0
      PA_DOMAIN = vtatest.com
      PA_EVENT_TIME = 1321471198
      PA_EVENT_TIME_MILLISECOND = 1321471198995
      PA_INTERFACE_NAME = ip10.227.1.96
      PA_INTERFACE_SPEED = 0
      PA_IN_OCTETS = 0
      PA_IN_PACKETS = 0
      PA_LAC_IP = 0
      PA_LOGIN_NAME = vtatest1@vtatest.com
      PA_NAS_INET_ADDRESS = 0.0.0.0
      PA_NAS_IP = 0.0.0.0
      PA_NAS_PORT = 0
      PA_OUT_OCTETS = 0
      PA_OUT_PACKETS = 0
      PA_PRIMARY_USER_NAME = vtatest1@vtatest.com
      PA_ROUTER_NAME = sim
      PA_ROUTER_TYPE = junos
      PA_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_SESSION_TIME = 0
      PA_SESSION_TIMEOUT = -1
      PA_SSP_HOST = kimberley
      PA_TERMINATE_CAUSE = 0
      PA_TUNNEL_ID = 0
```

```
      PA_TUNNEL_SESSION_ID = 0
      PA_USER_DN =
uniqueId=vtatest1,ou=group01,retailerName=vtatest,o=Users,o=UMC
      PA_USER_INET_ADDRESS = 10.227.1.96
      PA_USER_IP_ADDRESS = 10.227.1.96
      PA_USER_SESSION_HANDLE = SAE:KcVGVRM6OAwL0AAU@sim
      PA_USER_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_USER_TYPE = AUTHINTF
      PA_VPN_ID =
      currentTime = 1321471199007
      processingUuid = afe51007-7e35-465b-a515-3be12735afd6
      subscriberId = vtatest1@vtatest.com
    SAE information:
      SAE references =
[IOR:000000000000003549444C3A736D67742E6A756E697065722E6E657
42F7361652F536572766963654163746976617469 6F6E456E67696E653A312E3000000000000000100
0000000000006C0001020000000000D31302E3232372E362E31323600002261000000146B696D6265726
C65792F736165504F412F53414500000002000000000000000800000004A414300000000010000002 0
0000000000100010000000105010001000101090000000205010001000101 00]
      NIC proxy namespace = /nicProxies/np1
      VTA subscriber ID solution = [Subscriber ID = PA_LOGIN_NAME, NIC key
= PA_LOGIN_NAME, SAE key = LOGIN_NAME]
      SAE subscriber ID = LOGIN_NAME
    SAE subscriber references: 0
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA ignoring event.
                .
                .
                .
                .
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  Event handler NoQuota: Received event.
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event handler NoQuota: Ignoring event cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
14:19:59.040 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  VTA completed processing event.
```

This is only an excerpt of the log and does not show all messages. Notice that each message starts with a unique ID generated when the SRC VTA starts processing an event—for example, “Event trail afe51007-7e35-465b-a515-3be12735afd6.”

Enabling debug log messages has a negative effect on system performance. Do not enable debug log messages unless JTAC instructs you to do so.

You can define a severity level as follows:

- Specify an explicit severity. For example:

  warning—Defines only warning messages

- Specify a minimum severity and a maximum severity. For example:

  info-warning—Defines messages of minimum severity level of info and a maximum severity level of warning

- Accept the default minimum (logmin) or maximum (logmax) severity by omitting the minimum or maximum severity. For example:

  info-—Defines messages of minimum severity level info and maximum severity level logmax

  -warning—Defines messages of minimum severity level logmin and maximum severity level warning

- Specify no severity to log all event messages.

- Example—Table 16 on page 91 shows some examples of filters.

Table 16: Examples of Filters for Event Messages

Syntax	Event Messages Saved
/	All event messages
/info-	Event messages of level info and higher from all categories
vta/debug	Debug events from the SRC VTA category only
!vta,/debug	All debug events except those from the SRC VTA category
!VtaMsg/info-,vtaMsg,vta	All messages from the SRC VTA category, except those from the VtaMsg category with level lower than info

3. Specify the host information for the system log server.

   [edit shared vta group vta1 logger vta1-logger syslog]
   user@host# set host *host*

   Specify an IP address or name of a host that collects event messages with a standard system logging daemon.

Related Documentation

- Logging SRC VTA Messages to a System Logging Server on page 87
- Overview of the SRC VTA on page 9
- How the SRC VTA Works on page 12
- Logging SRC VTA Messages to a Text File on page 83

Enabling, Disabling, and Restarting the SRC VTA (SRC CLI)

After you create and commit an SRC VTA shared configuration, a new top-level SRC component called vta-*name* automatically appears in the SRC CLI. This component is visible when you execute the following SRC CLI operational commands:

- **show component**
- **enable component vta-*name***

- **disable component vta-*name***
- **restart component vta-*name***

Where the **vta-*name*** is the name you specified for the group name under the **edit shared vta group *name*** configuration statement. You can configure multiple SRC VTA instances (groups), each with a unique name.

After you delete an SRC VTA group and commit the configuration, the SRC component called **vta-*name*** automatically disappears from the SRC CLI.

When you issue the **show component** command, a component called VTA is always visible. However, this component cannot be enabled or disabled or restarted. It is only displayed so that you can view the version number of the SRC VTA. Only SRC VTA group components with names like vta-*name* can be enabled and disabled and restarted.

To enable an SRC VTA:

- From operational mode, enable the SRC VTA.

  user@host> **enable component vta-*name***

To disable an SRC VTA:

- From operational mode, enable the SRC VTA.

  user@host> **disable component vta-*name***

To restart an SRC VTA:

- From operational mode, enable the SRC VTA.

  user@host> **restart component vta-*name***

Related Documentation

- Overview of the SRC VTA on page 9
- How the SRC VTA Works on page 12

Using One SRC VTA Account for Multiple Subscriber Sessions

The SRC VTA allows multiple subscriber sessions to share the same SRC VTA account. The SRC VTA debits usage for all the subscriber sessions from the account. When the account is empty, service sessions for all subscribers are stopped. When the account is refilled, the SRC VTA starts services for all subscriber sessions that share the account.

To use this feature, you use the subscriberId event attribute to map a group of subscribers to the SRC VTA account. You then use the accounting-id attribute as the **subscriber-id-solution** parameter under the **shared vta group *name*** configuration statement. For information about this option, see "Creating and Configuring an SRC VTA Shared Group Configuration (SRC CLI)" on page 59. You also set up the NIC to use the accounting ID to look up the SAE that manages a subscriber.

To set up the SRC software to use one SRC VTA account for multiple subscribers:

1. In the subscriber classifier script, assign a value to the accountingUserId attribute. For example, you could assign it to the userName, interfaceName, loginName, or a combination of classification criteria. The purpose of the assignment is to allow the SRC VTA to identify subscribers by many different subscriber attributes using accountingUserId as a wrapper.

   For example, the following subscriber classifier script assigns the value of the userName to the accountingUserId attribute:

   ```
   [<-retailerDn->?accountingUserId=<-userName->?sub?(uniqueID=<-userName->)]
   ```

2. Configure the SAE to publish the PA_ACCOUNTING_ID plug-in attribute in subscriber-tracking events to the NIC SAE agent plug-in.

   See Configuring Internal Plug-Ins (SRC CLI).

3. Configure the NIC to use the OnePopAcctId NIC scenario.

   See Configuring the NIC (SRC CLI).

4. Configure the SRC VTA shared group configuration in the SRC CLI as follows:

   - For the **subscriber-id-solution** option enter **accounting-id**.
   - For the **nic-proxy** option, enter the name of the NIC proxy configuration you entered for the **shared vta nic-proxy-configuration *name*** statement. Be sure to configure the NIC proxy so that it uses a NIC that maps from the accounting ID to the SAE.

     See "Creating and Configuring an SRC VTA Shared Group Configuration (SRC CLI)" on page 59.

5. (Optional) Set up an action to apply functions to all subscriber sessions that share the same SRC VTA account. For example, the following action starts services for all subscriber sessions that have the same subscriber ID.

   ```
   [edit shared vta group vta1]
   user@host# edit action xyz
   [edit shared vta group vta1 action xyz]
   user@host#y# set function sae-start-service
   [edit shared vta group pear action xyz]
   user@host## show

   function sae-start-service;
   parameter;
   ```

   Make sure that the **current-subscriber-only** option is set to false (not present). If it is set, delete it.

Related Documentation

- Overview of the SRC VTA on page 9
- Managing SRC VTA Accounts and Sessions on page 30
- Configuring Subscribers and Subscriptions to SRC VTA Services on page 44

Enabling the SOAP Interface for an SRC VTA Group (SRC CLI)

To use the SOAP interface, you need to enable it for the SRC VTA group. You can also specify whether basic HTTP authentication is required when using the SOAP interface. By default, authentication is required. If you disable authentication, the CLI editing level must be set to expert.

NOTE: If you enable authentication, you must configure each user within the shared application server and set the user account role to VTA-*group name*.

After you install the SRC VTA, you can access a Web Services Definition Language (WSDL) file for the application. The WSDL file defines the SOAP properties that you or your customers can use to develop a client. The URL for the SRC VTA WSDL file is:

http://<host-name>:8080:/<vta-group>/api/soap/services/vtaAPI?wsdl

- **<host-name>**—Name of the C Series Controller that is running the SRC VTA
- **<vta-group>**—Name of the SRC VTA group

Use the following configuration statements to enable the SOAP interface to configure user accounts at the **[edit]** hierarchy level:

```
shared vta group name api {
  enable-soap-api;
  disable-client-authentication
}
```

1. From configuration mode, access the configuration statement that enables the SOAP interface. For example, to enable the SOAP interface for the SRC VTA group called Quota:

   ```
   user@host#  edit shared vta group Quota api
   ```

2. Enable the SOAP interface.

   ```
   [edit shared vta group Quota api]
   user@host# set enable-soap-api
   ```

3. To disable client authentication when using the SOAP interface:

   ```
   [edit shared vta group Quota api]
   user@host# set disable-client-authentication
   ```

Related Documentation

- SRC VTA SOAP Interface on page 31
- Viewing SRC VTA SOAP API Statistics (SRC CLI)
- Methods for the Volume Tracking Application SOAP Interface on page 95
- Configuring User Accounts for Web Applications (SRC CLI) on page 56

Methods for the Volume Tracking Application SOAP Interface

This topic describes the methods associated with the Volume Tracking Application SOAP interface and provides information additional to that in the Web Services Definition Language (WSDL) file.

openAccount

- Open an account and optionally create a balance change for the account as part of one database transaction.
- Arguments:
  - accountData—Account data object carrying account information.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

closeAccount

- Close an account and optionally record the change to the balance change table as part of one database transaction. This method only sets the status of the account to closed. If the account is already closed, do nothing.
- Arguments:
  - subscriberID—ID of the subscriber to whom the account belongs.
  - accountName—Name of the account to be closed.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

closeAccountsOfSubscriber

- Close accounts for a subscriber and optionally create balance changes for the accounts as part of one database transaction. This method only sets the status of the accounts to closed. Do nothing for closed accounts.
- Arguments:
  - subscriberID—ID of the subscriber to whom the account belongs.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

closeAccountsByName

- Close the named accounts for all subscribers. This method only sets the status of the accounts to closed. Do nothing for closed accounts.

- Arguments:
  - accountName—Name of the account to be closed.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

changeStatus

- Change the status of a specified account and optionally create a balance change record as part of one database transaction. This method does not perform any status check. It can be used to set a closed account to active again.
- Arguments:
  - subscriberID—ID of the subscriber to whom the account belongs.
  - accountName—Name of the account.
  - status—New status of the account.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

changeBalance

- Add the given amount to the balance (decreasing it if the amount is negative) and optionally create a balance change record for the account as part of one database transaction.
- Arguments:
  - subscriberID—ID of the subscriber to whom the account belongs.
  - accountName—Name of the account.
  - status—New status of the account.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

topUpBalance

- Top up the named accounts with the given amount for all subscribers. This method optionally creates balance change records for the accounts as part of one database transaction. Do nothing for closed accounts.
- Arguments:
  - subscriberID—ID of the subscriber to be topped up.
  - accountName—Name of the account to be topped up.

- amount—Amount to be added to the balance of each account.
- date—Date that the account's lastupdatetime field, in milliseconds since January 1, 1970 UTC, is to be set.
- writeBalanceChange—If true, an account balance change record is written to the database.
- description—Description of the account balance change record.

topUpBalance

- Top up the named accounts with the given amount for all subscribers. This method optionally creates balance change records for the accounts as part of one database transaction. Do nothing for closed accounts.
- Arguments:
  - subscriberID—ID of the subscriber to be topped up.
  - accountName—Name of the account to be topped up.
  - status—Status of the account to be topped up.
  - amount—Amount to be added to the balance of each account.
  - date—Date that the account's lastupdatetime field, in milliseconds since January 1, 1970 UTC, is to be set.
  - writeBalanceChange—If true, an account balance change record is written to the database.
  - description—Description of the account balance change record.

getAccount

- Get the named account for the given subscriber.
- Arguments:
  - subscriberID—ID of the subscriber to whom the account belongs.
  - accountName—Name of the account.

getAccountsOfSubscriber

- Search for accounts with the given status for a subscriber.
- Arguments:
  - subscriberID—ID of the subscriber to whom the account belongs.
  - status—Account status. If no status is specified, it returns all accounts for the subscriber regardless of their statuses.
- Expected output—Collection of AccountData objects.

getAccountsByName

- Search for accounts by name and status.
- Arguments:
  - accountName—Name of the account.
  - status—Account status. If no status is specified, it returns all accounts for the subscriber regardless of their statuses.
- Expected output—Collection of AccountData objects.

getBalanceChanges

- Search for balance changes in a given time period for an account. If both subscriberID and accountName are null, it returns balance changes in a time period for all accounts. If both start and end times are zero, it returns all balance changes for a given account.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountName—Name of the account.
  - start—Start time of the period to query.
  - end—End time of the period to query.
- Expected output—Collection of BalanceChangeData objects.

getSessionBalanceChange

- Given a session ID and a qualifier, return the session balance change record for a named account.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountName—Name of the account.
  - sessionId—ID of the session to query.
  - qualifier—Qualifier of the session to query.
- Expected output—SessionBalanceChangeData object.

getSessionBalanceChange

- Given a session ID and a qualifier, return the session balance change value objects.
- Arguments:
  - sessionId—ID of the session to query.
  - qualifier—Qualifier of the session to query.
- Expected output—Collection of SessionBalanceChangeData objects.

getSessionBalanceChanges

- Search for session balance changes in a given time period for an account. If subscriberID and accountName are null, it returns session balance changes in a time period for all accounts. If the start and end times are zero, it returns session balance changes for a given account.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountName—Name of the account.
  - start—Start time of the period to query.
  - end—End time of the period to query.
- Expected output—Collection of SessionBalanceChangeData objects.

getSessionBalanceChanges

- Given a session ID and qualifier, get the related session balance changes on the named accounts.
- Arguments:
  - sessionId—ID of the session to query.
  - qualifier—Qualifier of the session to query.
  - subscriberID—ID of the subscriber.
  - accountNames—Names of the accounts.
- Expected output—Map from the account name to the SessionBalanceChangeData object. If accountNames is null, it returns the related session balance change records on all accounts for the subscriber.

getAllBalanceChanges

- Search for balance changes and session balance changes in a given time period for an account. This method converts the balance changes to session balance changes and merges the balance changes and session balance changes.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountName—Name of the account.
  - start—Start time of the period to query.
  - end—End time of the period to query.
- Expected output—Collection of SessionBalanceChangeData and BalanceChangeData sorted by date field.

terminateSessions

- Terminate sessions for the subscriber.

- Arguments:
  - subscriberIDs—IDs of the subscribers whose sessions will be terminated.

getSession

- Given a subscriber ID, session ID, and a qualifier, return the session value object.
- Arguments:
  - subscriberID—ID of the subscriber.
  - sessionId—ID of the session to query.
  - qualifier—Qualifier of the session to query.
- Expected output—SessionData of the session that is found.

getSessions

- Given the session ID and status, return a collection of session value objects. If no status is specified, it returns sessions with the session ID regardless of their status.
- Arguments:
  - sessionId—ID of the session to query.
  - status—Status of the session to query.

getSessions

- Get sessions with the given status starting within a given time period for a subscriber. If the start and end times are zero, it returns sessions with the given status for the subscriber. If status is null, it returns all the relevant sessions regardless of their status.
- Arguments:
  - subscriberID—ID of the subscriber.
  - status—Status of the session to query.
  - start—Start time of the period to query.
  - end—End time of the period to query.
- Expected output—Collection of SessionData objects.

deleteAccount

- Delete an account.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountName—Name of the account.

deleteAccountsOfSubscriber

- Delete accounts for a subscriber. Only closed accounts are removed. For each unclosed account, an error message is logged in the debug log.

- Arguments:
  - subscriberID—ID of the subscriber.

deleteAccountsByName

- Delete named accounts for all subscribers. Only closed accounts are deleted. For each unclosed account, an error message is logged in the debug log.
- Arguments:
  - accountName—Name of the account.

deleteBalanceChanges

- Delete balance changes in a given time period for a set of accounts. If no account names are specified, it deletes balance changes in the given time period on all subscriber accounts. If no time period is specified, it deletes all balance changes for the set of accounts. If no account names or time period are specified, it deletes all balance changes on all subscriber accounts. If both subscriberID and accountNames are not specified, it deletes the balance changes in the given time period for all subscribers and accounts.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountNames—Names of accounts.
  - start—Start time of the period to query.
  - end—End time of the period to query.

deleteSessionBalanceChanges

- Delete session balance changes in a given time period for a set of accounts. If no account names are specified, it deletes session balance changes in the given time period on all subscriber accounts. If no time period is specified, it deletes all session balance changes for the set of accounts. If no account names or time period are specified, it deletes all balance changes on all the subscriber accounts.
- Arguments:
  - subscriberID—ID of the subscriber.
  - accountNames—Names of accounts.
  - start—Start time of the period to query.
  - end—End time of the period to query.

deleteSession

- Given a session IdD and qualifier, delete the session.
- Arguments:
  - subscriberID—ID of the subscriber.

- sessionId—ID of the session to query.
- qualifier—Qualifier of the session to query.

deleteSessions

- Given a session ID, delete the related sessions. Only closed sessions are deleted. For each unclosed session, an error message is logged in the debug log.
- Arguments:
  - sessionId—ID of the session to query.

deleteSessions

- Delete sessions starting within a given time period for a subscriber. If no time period is specified, it deletes all sessions for the subscriber. If no subscriber is specified, it deletes sessions in a given time period for all subscribers. Only closed sessions are deleted.
- Arguments:
  - subscriberID—ID of the subscriber.
  - start—Start time of the period to query.
  - end—End time of the period to query.

cleanup

- Clean up accounts whose last update time is earlier than the given time and whose status is equal to the given status. If no subscriber is specified, it performs the cleanup for all subscribers. If no status is specified, it only deletes closed accounts.
- Arguments:
  - subscriberID—ID of the subscriber.
  - time—Earliest last update time of accounts to clean up.
  - status—Status of the accounts to clean up.

cleanup

- Clean up sessions, sessionBalanceChanges, and balanceChanges whose last update time is earlier than the given time. If no subscriber is specified, it performs the cleanup for all subscribers.
- Arguments:
  - subscriberID—ID of the subscriber.
  - time—Earliest last update time of accounts to clean up.

updateAccounts

- Modify accounts selected by subscriberId, accountName, and accountStatus arguments. In those selected accounts, it:

- Overwrites the account status with newStatus (if newStatus is not null).
- Overwrites the account balance with balanceAmount (if balanceAmount is not null and amountIsDelta is false).
- Changes the account balance by balanceAmount (if balanceAmount is not null and amountIsDelta is true).
- If the balanceChangeDescription is not null, also creates a balance change record containing the specified description, the current time, and the amount by which the account balance was changed.
- If terminateSessions is true, generates a Callback:TERMINATESESSIONS event for every subscriber associated with the specified accounts.

- Arguments:
  - subscriberID—ID of the subscriber.
  - accountName—Name of the account.
  - accountStatus—Status of the account.
  - newStatus—New status of the account.
  - balanceAmount—Amount to set the balance to.
  - amountIsDelta—If true, sets the balance to the value specified by balanceAmount.
  - balanceChangeDescription—Description of the account balance change record.
  - terminateSessions—If true, generates a Callback:TERMINATESESSIONS event for every subscriber associated with the specified accounts.
- Expected output—Message describing the number of accounts that matched the subscriberId, accountName, and accountStatus arguments, the number of subscribers for those accounts, and the number of accounts that were modified.

CHAPTER 7

Examples

Example: Limiting Subscriber Access Based on Account Balances

The sample data provides an example called Quota that limits a subscriber's access rate based on the balances of accounts that record the subscriber's use of network resources. Subscribers receive a quota of transfer (upload and download) volume in two ways:

- Periodic quota—Volume that is periodically added to a subscriber's account. For example, a subscriber may receive a 25-MB periodic quota each month. The periodic quota is tracked in the periodic account.
- Bought quota—Additional volume that a subscriber can purchase and use at any time. For example, a subscriber may purchase 25 MB of bought quota in January, and use the bought quota between January and March. Bought quota is tracked in a bought account.

As a subscriber consumes volume, the SRC VTA debits the accounts, using first the periodic quota and, if no periodic quota is available, the bought quota.

Subscribers managed by the SRC VTA require a subscription to quota services—services for which the SRC VTA monitors and manages usage. You must configure these subscriptions to be activated when the subscriber logs in. When a subscriber logs in to the SAE, the SAE tries to activate the quota services. However, if neither the periodic account nor the bought account has a positive balance, the SRC VTA deactivates the quota services, and the SAE applies to the subscriber either the default policy or another policy that implements a service with a lower bandwidth.

The units of the accounts depend on the formula that you define to determine the use of network resources. With this formula, the SRC VTA calculates the change to the accounts and the interim accounting interval.

The Quota configuration example provides an SRC VTA that operates as follows:

1. When a service session for the quota service starts, the SAE sends a start event to the SRC VTA.
2. The SRC VTA starts a session that has the same identifier as the service session and a qualifier of zero.

3. When the SRC VTA receives the first interim update from the SAE in the SRC VTA session, it records a balance change that details the use of resources and the event time for the SRC VTA session. When the SRC VTA receives interim updates in the SRC VTA session, it updates the use of resources and the event time in the balance change recorded previously.
   - If the periodic account contains sufficient resources to cover the balance change, the SRC VTA changes only the balance of that account.
   - If the periodic account does not contain sufficient resources for the change, the SRC VTA records one balance change for the resources available in that account and records another balance change for the difference in the bought account. In this case, the SRC VTA records subsequent balance changes to the bought account.
   - If neither account has sufficient resources, the SRC VTA deactivates the quota service.
4. If the SAE session ends, the SRC VTA session ends. When a new service session starts, Steps 1 to 3 recur. However, a service session may last for several SRC VTA sessions. In this case, the SAE and SRC VTA continue the process described in Step 3.
5. When an administrator replenishes the periodic quota, the SRC VTA ends the session, finalizes all balance changes for the session, and records a credit to the periodic account.
6. When the subscriber buys additional volume, the SRC VTA ends the session, finalizes all balance changes for the session, and records a credit to the bought account.
7. When the SRC VTA next receives an interim update event from the SAE, it starts a new session. The SRC VTA obtains the start time for the session from the SAE event, and records debits to the accounts as described in Step 3.

   The SRC VTA always ends the session when an administrator replenishes periodic quota or the subscriber buys volume. However, a service session may last for several billing periods. In this case, when the SRC VTA starts a new session, it continues to assign the SAE session identifier to the session and increments the qualifier by one. Keeping the session within a billing period allows the SRC VTA to finalize balance changes.

Related Documentation

- Loading Sample Data into a Juniper Networks Database (SRC CLI)
- Overview of the SRC VTA on page 9
- Identifying Subscribers, SAEs, and Sessions on page 29
- Managing SRC VTA Accounts and Sessions on page 30

PART 3

Administration

CHAPTER 8

Managing the SRC VTA

Deleting Balance Change History Records from the Database (SRC CLI)

Use the **request vta group *vtaName* delete balance-changes before *date*** command to delete old balance change records from the SRC VTA database. This command deletes balance change records from the SRC VTA database that have a last updated time before the specified date.

- From operational mode, execute:

  user@host> **request vta group *vtaName* delete balance-changes before *date***

  Specify the date in the format yyyy-mm-dd.

Related Documentation

- Deleting Session History Records from the Database (SRC CLI) on page 110
- Deleting Subscriber SRC VTA Accounts (SRC CLI) on page 110

Deleting Session History Records from the Database (SRC CLI)

Use the **request vta group *vtaName* delete sessions before *date*** command to delete old session history records from the SRC VTA database. This command deletes session history records from the SRC VTA database that have a last updated time before the specified date.

- From operational mode, execute:

  user@host> **request vta group *vtaName* delete sessions before *date***

  Specify the date in the format yyyy-mm-dd.

Related Documentation

- Deleting Balance Change History Records from the Database (SRC CLI) on page 109
- Deleting Subscriber SRC VTA Accounts (SRC CLI) on page 110

Deleting Subscriber SRC VTA Accounts (SRC CLI)

Use the **request vta group *vtaName* delete subscriber *subscriber-id*** command to delete a specified subscriber's SRC VTA accounts and subscriber's sessions that have a status of "Closed." This command also deletes all balance changes (from administrative actions) and session balance changes (from session usage) associated with those accounts.

- From operational mode, execute:

  user@host> **request vta group *vtaName* delete subscriber *subscriber-id***

Related Documentation

- Deleting Balance Change History Records from the Database (SRC CLI) on page 109
- Deleting Session History Records from the Database (SRC CLI) on page 110

Modifying SRC VTA Accounts and Service Sessions (SRC CLI)

Use the **request vta group *vtaName* update-accounts account-name *account-name*** command to modify subscriber SRC VTA accounts and service sessions. By using this command, you can modify either all service sessions in an account or a specific subscriber's service sessions in the account.

You can overwrite the status, description, last update time, or balance in a set of accounts. You can also change the existing account balance by a specific positive or negative amount.

- To modify all service sessions in an account, execute the following command from operational mode:

  user@host> request vta group *vtaName* update-accounts account-name *account-name*

- To modify a specific subscriber's service sessions in the account, execute the following command from operational mode:

  user@host> **request vta group** *vtaName* **update-accounts account-name** *account-name* **subscriber-id** *subscriber-id*

  You can optionally add the **account-status** option if you only want to update accounts with a particular status.

Use one or more of the options described in Table 17 on page 111 to modify the account.

Table 17: Arguments Used to Modify Accounts

Argument	Description
new-status *newAccountStatus*	Changes the status of the account or specified subscriber's service sessions.
new-balance *newAccountBalance*	Sets the account balance to the amount specified by the *newAccountBalance*. For example, an account starts with a balance of 10. You set the *newAccountBalance* to 2. The account now has a balance of 2.
balance-change *accountBalanceChangeAmount*	Adds the amount specified for the *accountBalanceChangeAmount* to the existing account balance. For example, an account starts with a balance of 10. You set the *accountBalanceChangeAmount* to 2. The account now has a balance of 12.
balance-change-description *description*	(Optional) Add a description for the balance change. If you specify this option, a new balance change record is created in the database that contains the specified description, as well as the amount by which the account balance was changed. If the account balance was not changed, this amount will be zero.
terminate-sessions	(Optional) Trigger a **callback:terminatesessions** event for each specified subscriber. The SRC VTA ignores these events unless you configure an event-handler to process them.

When you execute either of these commands, the SRC VTA takes the following steps:

- · If you specify the terminate-sessions option, the SRC VTA generates a **callback:terminatesessions** event for every subscriber that owns a selected account.
- The SRC VTA updates all service sessions in the account, or all service sessions for the specified subscriber as requested. This is purely a database operation.
- For each account that is updated, the SRC VTA creates an **account-update** event. This event contains the relevant account's old and new balances, and other information. The SRC VTA ignores these events unless you configure an event handler to process this event type.

Terminating Sessions (SRC CLI)

For better integration with billing systems, it is sometimes desirable to terminate a set of subscribers' SRC VTA sessions at the end of a billing period.

- To terminate SRC VTA sessions, from operational mode, execute:

  user@host> **request vta group** *vtaName* **terminate-sessions subscriber-id** *subscriber-id*

NOTE: To use **request vta group *vtaName* terminate-sessions subscriber-id *subscriber-id*** to terminate sessions, you need to configure an event handler to process **callback:terminatesessions** events by invoking an action that in turn invokes the **db-engine-terminate-session** function. If you do not configure the event handler to process these events in this manner, executing this command does nothing.

Viewing a Subscriber's SRC VTA Accounts (SRC CLI)

Purpose View the list of all specified SRC VTA accounts of a subscriber.

Action To view information about a subscriber's SRC VTA account:

```
user@host> show vta accounts group name subscriber-id subscriberId account-name account-name
```

The **group** and **subscriber-id** options are mandatory. The **account-name** option is optional.

Related Documentation

- Viewing Balance Change History for a Subscriber (SRC CLI) on page 112
- Viewing a Subscriber's Session History (SRC CLI) on page 113

Viewing Balance Change History for a Subscriber (SRC CLI)

Purpose View a list of the specified SRC VTA accounts and all associated balance changes and session balance changes if the changes have a timestamp between the *from* and *to* dates. Balance changes (from administrative actions) and session balance changes (from service usage) are displayed in a single list and ordered by their timestamps from the oldest to most recent.

Action To view a list of the specified SRC VTA accounts and all associated balance changes and session balance changes, from operational mode, execute:

```
user@host> show vta balance-changes group name subscriber-id subscriber-id account-name account-name from from to to
```

Specify the date in the format yyyy-mm-dd.

The system displays both balance changes from administrative actions and session balance changes from service usage. The from and to dates are optional. If you omit the to date, the system defaults to the infinite future. If you omit the from date, the system defaults to the previous day.

Related Documentation

- Viewing a Subscriber's SRC VTA Accounts (SRC CLI) on page 112
- Viewing a Subscriber's Session History (SRC CLI) on page 113

Viewing a Subscriber's Session History (SRC CLI)

Purpose View a subscriber's SRC VTA-tracked session history.

Action To view a list of the subscriber's SRC VTA-tracked session history, from operational mode, execute:

```
user@host> show vta sessions group name from from to to subscriber-id subscriber-id
```

Specify the date in the format yyyy-mm-dd.

The system displays a list of the subscriber's SRC VTA-tracked sessions that have a last update time on or between the from and to dates. The list is ordered by session start time (not last update time), from the oldest to most recent.

The from and to dates are optional. If you omit the to date, the system defaults to the infinite future. If you omit the from date, balance changes are displayed for the last six days. Balance changes are displayed from the oldest to most recent.

To display only the subscriber ID, session ID, status, up bytes, and down bytes, execute:

```
user@host> show vta sessions group name from from to to subscriber-id subscriber-id terse
```

Related Documentation

- Viewing a Subscriber's SRC VTA Accounts (SRC CLI) on page 112
- Viewing Balance Change History for a Subscriber (SRC CLI) on page 112

Viewing SRC VTA Performance Statistics (SRC CLI)

Purpose View performance statistics for a SRC VTA's queue, event handlers, and actions.

Action To view performance statistics for a SRC VTA group, from operational mode, execute:

```
user@host> show vta statistics performance group name
```

```
Uptime
Uptime (seconds) 44
Up Since         Tue Oct 18 14:53:04 EDT 2011

Event Queue
Configured max queue size                       5000
Current queue size                              0
Events received                                 0
Events rejected due to full queue               0
Events dispatched                               0
Events received in last 60s: Min time in SAE    0
Events received in last 60s: Avg time in SAE    0
Events received in last 60s: Max time in SAE    0
Events dispatched in last 60s: Min time in queue 0
Events dispatched in last 60s: Avg time in queue 0
Events dispatched in last 60s: Max time in queue 0
Events received/second in last 60s              0.0
Events dispatched/second in last 60s            0.0

Event Handler
Event handler name                              EndofBilling
```

```
Events received                                 0
Events ignored                                  0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.0

Event Handler
Event handler name                              GetQuota
Events received                                 0
Events ignored                                  0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.0

Event Handler
Event handler name                              NoQuota
Events received                                 0
Events ignored                                  0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.0

Event Handler
Event handler name                              QuotaRefilled
Events received                                 0
Events ignored                                  0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.0

Event Handler
Event handler name                              RecordUsage
Events received                                 0
Events ignored                                  0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.0

Event Handler
Event handler name                              SetInterim
Events received                                 0
Events ignored                                  0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
```

```
Successful processing in last 60s: Max time      0
Successfully processed events/second in last 60s 0.0

Action
Action name                                      CalcUsage
Events received                                  0
Events processed                                 0
Event processing failures                        0
Successful processing in last 60s: Min time      0
Successful processing in last 60s: Avg time      0
Successful processing in last 60s: Max time      0
Successfully processed events/second in last 60s 0.00

Action
Action name                                      CalculateInterim
Events received                                  0
Events processed                                 0
Event processing failures                        0
Successful processing in last 60s: Min time      0
Successful processing in last 60s: Avg time      0
Successful processing in last 60s: Max time      0
Successfully processed events/second in last 60s 0.00

Action
Action name                                      DebitAccounts
Events received                                  0
Events processed                                 0
Event processing failures                        0
Successful processing in last 60s: Min time      0
Successful processing in last 60s: Avg time      0
Successful processing in last 60s: Max time      0
Successfully processed events/second in last 60s 0.00

Action
Action name                                      GetAccountBalances
Events received                                  0
Events processed                                 0
Event processing failures                        0
Successful processing in last 60s: Min time      0
Successful processing in last 60s: Avg time      0
Successful processing in last 60s: Max time      0
Successfully processed events/second in last 60s 0.00

Action
Action name                                      SetInterim
Events received                                  0
Events processed                                 0
Event processing failures                        0
Successful processing in last 60s: Min time      0
Successful processing in last 60s: Avg time      0
Successful processing in last 60s: Max time      0
Successfully processed events/second in last 60s 0.00

Action
Action name                                      StartptspService
Events received                                  0
Events processed                                 0
Event processing failures                        0
Successful processing in last 60s: Min time      0
Successful processing in last 60s: Avg time      0
Successful processing in last 60s: Max time      0
```

```
Successfully processed events/second in last 60s 0.00

Action
Action name                                     StopptspService
Events received                                 0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.00

Action
Action name                                     TerminateSession
Events received                                 0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.00

Action
Action name                                     test
Events received                                 0
Events processed                                0
Event processing failures                       0
Successful processing in last 60s: Min time     0
Successful processing in last 60s: Avg time     0
Successful processing in last 60s: Max time     0
Successfully processed events/second in last 60s 0.00

user@host>
```

Related Documentation

- Viewing a Subscriber's SRC VTA Accounts (SRC CLI) on page 112
- Viewing Balance Change History for a Subscriber (SRC CLI) on page 112

Overview of Testing the SRC VTA Configuration

You can use SRC VTA test commands to simulate events and test a specific SRC VTA group configuration. You can simulate subscriber-tracking events, service-tracking events, and callback events.

There are two steps involved in configuring and testing the SRC VTA configuration.

- First, you need to define the SRC VTA test events. You define a test event by assigning values to a set of attributes. The attributes you can assign to the test event depends on the event type you specify. SRC VTA test event configurations are stored in the Juniper Networks database so that you can reference them when executing SRC VTA test commands from any C Series Controller.
- After you define a test event, you can reference the test event when you execute test commands.

Table 18 on page 117 lists the event types and the configuration statements you use to define SRC VTA test events.

Table 18: SRC VTA Event Types and Test Event Configuration Statements

Test Events	Configuration Statement Used to Define the Test Event	Event Type
Subscriber-and service-tracking test events	Use the following statement to configure subscriber- and service-tracking test events: **shared vta test-events** *event-name* **type** *type* **attributes {** The *event-name* variable is an arbitrary name you specify for the event. When you execute a test command, you specify this name for the *test-event* variable. See Table 19 on page 117 for setting the *attribute-name* variable for either subscriber- or service-tracking test events.	You can define the following event types for the *event-type* variable when defining subscriber-tracking events: • **user-start** • **user-interim** • **user-stop** You can define the following event types for the *event-type* variable when defining service-tracking events: • **service-start:service-name** • **service-interim:service-name** • **service-stop:service-name**
Callback test events	Use the following statement to configure callback test events: **shared vta test-events type call-back:***name* **callback-attributes {** You can define the following attribute type for callback test events: • Boolean • Long • Integer • Double • Float • String The *event-name* variable is an arbitrary name you specify for the event. When you execute a test command, you specify this name for the *test-event* variable.	The only event type you can define for callback test events is **callback**.

Table 19: Attributes for Subscriber- and Service-Tracking Test Events

Attributes Supported for Subscriber-Tracking Events	Attributes Supported for Service-Tracking Events
PA_ACCOUNTING_ID	PA_ACCOUNTING_ID
PA_AUTH_USER_ID	PA_AGGR_ACCOUNTING_ID
PA_DHCP_PACKET	PA_AGGR_AUTH_USER_ID
PA_EVENT_TIME	PA_AGGR_LOGIN_NAME

Table 19: Attributes for Subscriber- and Service-Tracking Test Events *(continued)*

PA_EVENT_TIME_MILLISECOND	PA_AGGR_SESSION_ID
PA_IF_RADIUS_CLASS	PA_AGGR_USER_DN
PA_IF_SESSION_ID	PA_AGGR_USER_INET_ADDRESS
PA_INTERFACE_ALIAS	PA_AUTH_USER_ID
PA_INTERFACE_DESCR	PA_DHCP_PACKET
PA_INTERFACE_NAME	PA_DOWNSTREAM_BANDWIDTH
PA_LOGIN_ID	PA_EVENT_TIME
PA_LOGIN_NAME	PA_EVENT_TIME_MILLISECOND
PA_NAS_INET_ADDRESS	PA_IF_RADIUS_CLASS
PA_NAS_IP	PA_IF_SESSION_ID
PA_NAS_PORT	PA_IN_OCTETS
PA_OPERATIONAL	PA_IN_PACKETS
PA_PORT_ID	PA_INTERFACE_ALIAS
PA_PRIMARY_USER_NAME	PA_INTERFACE_DESCR
PA_RADIUS_CLASS	PA_INTERFACE_NAME
PA_ROUTER_NAME	PA_LOGIN_ID
PA_SESSION_TIMEOUT	PA_LOGIN_NAME
PA_SSP_HOST	PA_NAS_INET_ADDRESS
PA_SUBSCRIPTION_NAME	PA_NAS_IP
PA_TERMINATE_CAUSE	PA_NAS_PORT
PA_USER_DN	PA_OPERATIONAL
PA_USER_INET_ADDRESS	PA_OUT_OCTETS
PA_USER_IP_ADDRESS	PA_OUT_PACKETS
PA_USER_MAC_ADDRESS	PA_PORT_ID
PA_USER_RADIUS_CLASS	PA_PRIMARY_USER_NAME

Table 19: Attributes for Subscriber- and Service-Tracking Test Events *(continued)*

PA_USER_TYPE	PA_RADIUS_CLASS
	PA_ROUTER_NAME
	PA_SERVICE_NAME
	PA_SERVICE_SCOPE
	PA_SERVICE_SESSION_NAME
	PA_SERVICE_SESSION_TAG
	PA_SESSION_ID
	PA_SESSION_TIME
	PA_SSP_HOST
	PA_SUBSCRIPTION_NAME
	PA_TERMINATE_CAUSE
	PA_UPSTREAM_BANDWIDTH
	PA_USER_DN
	PA_USER_INET_ADDRESS
	PA_USER_IP_ADDRESS
	PA_USER_MAC_ADDRESS
	PA_USER_RADIUS_CLASS
	PA_USER_TYPE

Related Documentation

- Overview of Configuring Event Handlers on page 18

Testing SRC VTA Events (SRC CLI)

You can test SRC VTA events for subscriber- and service-tracking events, as well as callback events.

- From operational mode, execute:

  user@host> **test vta events subscriber-id** ***subscriber-identifier*** **event-name** ***event-name*** **group** ***group***

For the *event-name* variable, specify the event name you configured when you defined the test event with either the **shared vta test-events *name* callback-attributes *name*** or **shared vta test-events *name* attributes *name*** statement.

Related Documentation

- Overview of Configuring Event Handlers on page 18
- Overview of Testing the SRC VTA Configuration on page 116

Deleting Event Queues for SRC VTA Groups (SRC CLI)

Use the **request vta delete event-queue group *vtaName*** to delete the event queue for the SRC VTA group. Use this command to delete empty event queues after you disable the SRC VTA using the **disable component vta-*xxx*** command. You should execute this command on each system where the SRC VTA group was previously running.

- From operational mode, execute:

  user@host> **request vta delete event-queue group *vtaName***

Related Documentation

- Configuring Event Queues for SRC VTA Groups (SRC CLI) on page 67

PART 4

Troubleshooting

CHAPTER 9

Troubleshooting Initial Configuration

Logging SRC VTA Messages to a Text File

Use this procedure to configure the SRC VTA to save messages, such as event messages, SRC VTA startup messages, errors, and so on, in a text file.

Use the following statement to configure the SRC VTA to save messages in text files:

```
shared vta group name logger name file {
  filename filename;
  filter filter;
  maximum-file-size maximum-file-size;
  rollover-filename rollover-filename;
}
```

1. From configuration mode, access the statement that configures logging to a text file. For example, to configure a logger called vta1-logger for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 logger vta1-logger file
   ```

2. Specify the path and filename of the current log file. For example:

   ```
   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set filename pathname/filename.log
   ```

   Make sure you have write access to the folder.

3. (Optional) Specify a filter that determines the type of messages that this log file contains.

   ```
   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set filter filter
   ```

   The filter is specified in an expression. The software filters events by evaluating each subexpression from left to right. When the software finds a match, it logs or ignores

the message accordingly. You can specify an unlimited number of subexpressions. The order in which you specify the subexpressions affects the result. Expressions have the format:

singlematch [,singlematch]

Where

singlematch—[!] (<category> | ([<category>]/[<severity>] | [<minimumSeverity>]-[<maximumSeverity>]))

- !—Do not log matching events.
- <category>—SRC component that generated the event message. To log only events in a specific category, you can define the category, which is a text string that matches the name of a category. The text string is not case sensitive. For the names of categories, view a log file for a default filter. Juniper Networks Technical Assistance Center (JTAC) can also provide category names.
- [<severity>] | [<minimumSeverity>]-[<maximumSeverity>]—Name or number in the range 1–127. A higher number indicates a higher severity level. Table 13 on page 84 shows common severity levels that you can specify by name.

Table 20: Named Severity Levels

Name	Severity Level
logmin	1
debug	10
info	20
notice	30
warning	40
error	50
crit	60
alert	70
emerg	80
panic	90
logmax	127

You can set up *event trail* logging for the SRC VTA, which is helpful when debugging the SRC VTA configuration. Event trail logging logs messages that show you exactly how the SRC VTA is handling each event as it passes through the SRC VTA. You

can follow a particular event as it passes through the SRC VTA and see exactly how each event handler and action handles that event.

To set up event trail logging, set the filter option to **EventTrail/9-**. Enabling event trail logging generates messages such as the following:

```
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA received event:
    Event type = SAEPluginEvent
    Event name = USERSTART
    Attributes:
      PA_ACCT_INTERIM_TIME = 0
      PA_DOMAIN = vtatest.com
      PA_EVENT_TIME = 1321471198
      PA_EVENT_TIME_MILLISECOND = 1321471198995
      PA_INTERFACE_NAME = ip10.227.1.96
      PA_INTERFACE_SPEED = 0
      PA_IN_OCTETS = 0
      PA_IN_PACKETS = 0
      PA_LAC_IP = 0
      PA_LOGIN_NAME = vtatest1@vtatest.com
      PA_NAS_INET_ADDRESS = 0.0.0.0
      PA_NAS_IP = 0.0.0.0
      PA_NAS_PORT = 0
      PA_OUT_OCTETS = 0
      PA_OUT_PACKETS = 0
      PA_PRIMARY_USER_NAME = vtatest1@vtatest.com
      PA_ROUTER_NAME = sim
      PA_ROUTER_TYPE = junos
      PA_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_SESSION_TIME = 0
      PA_SESSION_TIMEOUT = -1
      PA_SSP_HOST = kimberley
      PA_TERMINATE_CAUSE = 0
      PA_TUNNEL_ID = 0
      PA_TUNNEL_SESSION_ID = 0
      PA_USER_DN =
uniqueId=vtatest1,ou=group01,retailerName=vtatest,o=Users,o=UMC
      PA_USER_INET_ADDRESS = 10.227.1.96
      PA_USER_IP_ADDRESS = 10.227.1.96
      PA_USER_SESSION_HANDLE = SAE:KcVGVRM6OAwL0AAU@sim
      PA_USER_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_USER_TYPE = AUTHINTF
      PA_VPN_ID =
      currentTime = 1321471199007
      processingUuid = afe51007-7e35-465b-a515-3be12735afd6
      subscriberId = vtatest1@vtatest.com
    SAE information:
      SAE references =
[IOR:000000000000003549444C3A736D67742E6A756E697065722E6E657
42F7361652F536572766963654163746976174696F6E456E67696E653A312E30000000000000000100
0000000000006C000102000000000D31302E3232372E362E313236000022610000000146B696D6265726
C65792F736165504F412F534145000000020000000000000008000000004A414300000000010000002
0000000000010001000000010501000100010109000000020501000100010100]
      NIC proxy namespace = /nicProxies/np1
      VTA subscriber ID solution = [Subscriber ID = PA_LOGIN_NAME, NIC key
= PA_LOGIN_NAME, SAE key = LOGIN_NAME]
      SAE subscriber ID = LOGIN_NAME
    SAE subscriber references: 0
```

```
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA ignoring event.
              .
              .
              .
              .
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  Event handler NoQuota: Received event.
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event handler NoQuota: Ignoring event cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
14:19:59.040 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  VTA completed processing event.
```

This is only an excerpt of the log and does not show all messages. Notice that each message starts with a unique ID generated when the SRC VTA starts processing an event—for example, “Event trail afe51007-7e35-465b-a515-3be12735afd6.”

Enabling debug log messages has a negative effect on system performance. Do not enable debug log messages unless JTAC instructs you to do so.

You can define a severity level as follows:

- Specify an explicit severity. For example:

  warning—Defines only warning messages

- Specify a minimum severity and a maximum severity. For example:

  info-warning—Defines messages of minimum severity level of info and a maximum severity level of warning

- Accept the default minimum (logmin) or maximum (logmax) severity by omitting the minimum or maximum severity. For example:

  info-—Defines messages of minimum severity level info and maximum severity level logmax

  -warning—Defines messages of minimum severity level logmin and maximum severity level warning

- Specify no severity to log all event messages.

- Table 14 on page 87 shows some examples of filters.

Table 21: Examples of Filters for Event Messages

Syntax	Event Messages Saved
/	All event messages
/info-	Event messages of level info and higher from all categories
vta/debug	Debug events from the SRC VTA category only
!vta,/debug	All debug events except those from the SRC VTA category
!VtaMsg/info-,vtaMsg,vta	All messages from the SRC VTA category, except those from the VtaMsg category with level lower than info

4. Specify the maximum file size. This option disables or enables and sets the maximum size of the log file and the rollover file.

   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set maximum-file-size *maximum-file-size*

   This is specified in number of kilobytes in the range 0–4294967295. The default is 1000000.

   Do not set the maximum file size to a value greater than the available disk space.

5. Specify the rollover filename.

   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set rollover filename *rollover filename*

   Specify the path and filename of the rollover log file. When the log file reaches the maximum size, the software closes the log file and renames it with the name you specify for the rollover file. If a previous rollover file exists, the software overwrites it. The software then reopens the log file and continues to save event messages in it.

   For example:

   [edit shared vta group vta1 logger vta1-logger file]
   user@host# set rollover filename *vta_debug.alt*

Related Documentation

- Logging SRC VTA Messages to a System Logging Server on page 87

Logging SRC VTA Messages to a System Logging Server

Use this procedure to configure the SRC VTA to save messages, such as event messages, SRC VTA startup messages, errors, and so on, on a system logging server.

Use the following statement to configure the SRC VTA to save messages on a system logging server:

```
shared vta group name logger name syslog {
  filter filter;
  host host;
}
```

1. From configuration mode, access the statement that configures logging to the system log facility. For example, to configure a logger called vta1-logger for the SRC VTA group called vta1:

   ```
   [edit]
   user@host# edit shared vta group vta1 logger vta1-logger syslog
   ```

2. (Optional) Specify a filter that determines the type of messages that this log file contains.

   ```
   [edit shared vta group vta1 logger vta1-logger syslog]
   user@host# set filter filter
   ```

   The filter is specified in an expression. The software filters events by evaluating each subexpression from left to right. When the software finds a match, it logs or ignores the message accordingly. You can specify an unlimited number of subexpressions. The order in which you specify the subexpressions affects the result. Expressions have the format:

   singlematch [,singlematch]

   Where

   singlematch—[!] (<category> | ([<category>]/[<severity>] | [<minimumSeverity>]-[<maximumSeverity>]))

   - !—Do not log matching events.
   - <category>—SRC component that generated the event message. To log only events in a specific category, you can define the category, which is a text string that matches the name of a category. The text string is not case sensitive. For the names of categories, view a log file for a default filter. Juniper Networks Technical Assistance Center (JTAC) can also provide category names.
   - [<severity>] | [<minimumSeverity>]-[<maximumSeverity>]—Name or number in the range 1–127. A higher number indicates a higher severity level. Table 15 on page 88 shows common severity levels that you can specify by name.

Table 22: Named Severity Levels

Name	Severity Level
logmin	1
debug	10

Table 22: Named Severity Levels *(continued)*

Name	Severity Level
info	20
notice	30
warning	40
error	50
crit	60
alert	70
emerg	80
panic	90
logmax	127

For the SRC VTA, you can perform *event trail* logging, which is helpful when debugging the SRC VTA configuration. Event trail logging logs messages that show you exactly how the SRC VTA is handling each event as it passes through the SRC VTA. You can follow a particular event as it passes through the SRC VTA and see exactly how each event handler and action handles that event.

To set up event trail logging, set the filter option to **filter EventTrail/9-**. Enabling event trail logging generates messages such as the following:

```
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA received event:
    Event type = SAEPluginEvent
    Event name = USERSTART
    Attributes:
      PA_ACCT_INTERIM_TIME = 0
      PA_DOMAIN = vtatest.com
      PA_EVENT_TIME = 1321471198
      PA_EVENT_TIME_MILLISECOND = 1321471198995
      PA_INTERFACE_NAME = ip10.227.1.96
      PA_INTERFACE_SPEED = 0
      PA_IN_OCTETS = 0
      PA_IN_PACKETS = 0
      PA_LAC_IP = 0
      PA_LOGIN_NAME = vtatest1@vtatest.com
      PA_NAS_INET_ADDRESS = 0.0.0.0
      PA_NAS_IP = 0.0.0.0
      PA_NAS_PORT = 0
      PA_OUT_OCTETS = 0
      PA_OUT_PACKETS = 0
      PA_PRIMARY_USER_NAME = vtatest1@vtatest.com
      PA_ROUTER_NAME = sim
```

```
      PA_ROUTER_TYPE = junos
      PA_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_SESSION_TIME = 0
      PA_SESSION_TIMEOUT = -1
      PA_SSP_HOST = kimberley
      PA_TERMINATE_CAUSE = 0
      PA_TUNNEL_ID = 0
      PA_TUNNEL_SESSION_ID = 0
      PA_USER_DN =
uniqueId=vtatest1,ou=group01,retailerName=vtatest,o=Users,o=UMC
      PA_USER_INET_ADDRESS = 10.227.1.96
      PA_USER_IP_ADDRESS = 10.227.1.96
      PA_USER_SESSION_HANDLE = SAE:KcVGVRM6OAwL0AAU@sim
      PA_USER_SESSION_ID = KcVGVRM6OAwL0AAU
      PA_USER_TYPE = AUTHINTF
      PA_VPN_ID =
      currentTime = 1321471199007
      processingUuid = afe51007-7e35-465b-a515-3be12735afd6
      subscriberId = vtatest1@vtatest.com
    SAE information:
      SAE references =
[IOR:000000000000003549444C3A736D67742E6A756E697065722E6E657
42F7361652F536572766963654163746976617469 6F6E456E67696E653A312E3000000000000000100
0000000000006C000102000000000D31302E3232372E362E313236000022610000001 46B696D6265726
C65792F736165504F412F53414500000002000000000000000800000004A414300000000010000002 0
000000000010001000000001050100010001010900000002050100010 0010100]
      NIC proxy namespace = /nicProxies/np1
      VTA subscriber ID solution = [Subscriber ID = PA_LOGIN_NAME, NIC key
= PA_LOGIN_NAME, SAE key = LOGIN_NAME]
      SAE subscriber ID = LOGIN_NAME
    SAE subscriber references: 0
14:19:59.008 EST 16.11.2011 [Thread-21
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail afe51007-7e35-465b-a515-3be12735afd6
  VTA ignoring event.
                .
                .
                .
                .
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  Event handler NoQuota: Received event.
14:19:59.039 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event handler NoQuota: Ignoring event cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
14:19:59.040 EST 16.11.2011 [Thread-19
(group:HornetQ-client-global-threads-1542419015)] [EventTrail] [9]
  Event trail cea6f3b3-efc7-46c1-9e8a-b26b721f03cc
  VTA completed processing event.
```

This is only an excerpt of the log and does not show all messages. Notice that each message starts with a unique ID generated when the SRC VTA starts processing an event—for example, “Event trail afe51007-7e35-465b-a515-3be12735afd6.”

Enabling debug log messages has a negative effect on system performance. Do not enable debug log messages unless JTAC instructs you to do so.

You can define a severity level as follows:

- Specify an explicit severity. For example:

  warning—Defines only warning messages

- Specify a minimum severity and a maximum severity. For example:

  info-warning—Defines messages of minimum severity level of info and a maximum severity level of warning

- Accept the default minimum (logmin) or maximum (logmax) severity by omitting the minimum or maximum severity. For example:

  info-—Defines messages of minimum severity level info and maximum severity level logmax

  -warning—Defines messages of minimum severity level logmin and maximum severity level warning

- Specify no severity to log all event messages.

- Example—Table 16 on page 91 shows some examples of filters.

Table 23: Examples of Filters for Event Messages

Syntax	Event Messages Saved
/	All event messages
/info-	Event messages of level info and higher from all categories
vta/debug	Debug events from the SRC VTA category only
!vta,/debug	All debug events except those from the SRC VTA category
!VtaMsg/info-,vtaMsg,vta	All messages from the SRC VTA category, except those from the VtaMsg category with level lower than info

3. Specify the host information for the system log server.

   [edit shared vta group vta1 logger vta1-logger syslog]
   user@host# set host *host*

   Specify an IP address or name of a host that collects event messages with a standard system logging daemon.

Related Documentation

- Logging SRC VTA Messages to a Text File on page 83

Troubleshooting Database Deadlocks

Problem The JBoss application server logs the following error when the database reports a deadlock—a condition in which the database operation cannot continue because two processes are both waiting for the other process to be completed before they proceed.

java.sql.SQLException: General error, message from server: "Deadlock found when trying to get lock; Try restarting transaction

Solution Deadlocks can occur for a variety of reasons in normal database operation. The SRC VTA resolves deadlocks in the database, and you should ignore this message.

Related Documentation

- Installing Web Applications in the SRC Web Application Server on page 57
- Configuring a Database to Store Account and Session Data (SRC CLI) on page 42
- Configuring the Initial Balance and Status of a Subscriber Account in the External Database (SRC CLI) on page 69

PART 5

Index

Index

T