# Cisco Service Control Management Suite Collection Manager User Guide

Release 3.1

May 2007

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Customer Order Number:
Text Part Number: OL-7208-05

# CONTENTS

# About This Guide

**Revised: May 30, 2007, OL-7208-05**

This preface describes who should read the *Cisco Service Control Management Suite Collection Manager User Guide*, how it is organized, its document conventions, and how to obtain documentation and technical assistance. This guide assumes a basic familiarity with the concept of the Cisco Service Control solution, the Service Control Engine (SCE) platforms, and related components.

This introduction provides information about the following topics:

- Document Revision History
- Audience
- Document Organization
- Document Conventions
- Related Documentation
- Obtaining Documentation, Obtaining Support, and Security Guidelines

## Document Revision History

| Cisco Service Control Release | Part Number | Publication Date |
|---|---|---|
| Release 3.1.0 | OL-7208-05 | May, 2007 |

### Description of Changes

- Added new script to manage virtual links semantics. See Managing Virtual Links, page 7.
- Added support for Solaris 10 and removed support for Solaris 8
- Added support for MySQL 5
- Added support for Oracle 10
- The bundled Sybase database for Solaris is upgraded to version 15
- Removed --expert mode from the Sybase install script. The regular install mode allows the user to define the DB capacity.
- The periodic delete mechanism and **dbtables.sh** script are now supported for external databases. See Common Database Management Tasks, page 1.

- The upgrade procedure to release 3.1 can preserve the previous version configuration. See install-cm.sh Options, page 9 and Upgrading the CM to Version 3.1, page 13.

| Cisco Service Control Release | Part Number | Publication Date |
|---|---|---|
| Release 3.0.5 | OL-7208-04 | November, 2007 |

**Description of Changes**

Added the following sections to the document:

- Managing Users, page 6
- Enabling the Adapters, page 3
- Configuring the Categorizer, page 4

| Cisco Service Control Release | Part Number | Publication Date |
|---|---|---|
| Release 3.0.3 | OL-7208-03 | May, 2006 |

**Description of Changes**

Added the following new features:

- The **check_prerequisites.sh** script. See Checking System Prerequisites, page 1.
- The **monitor.sh** script. See Information About Monitoring System Health, page 5.

Removed the following, deprecated feature:

- The **--legacy** flag of the **installsyb.sh** script

Added the following section to the document:

- Configuring Escape of Nonprintable Characters, page 9

| Cisco Service Control Release | Part Number | Publication Date |
|---|---|---|
| Release 3.0 | OL-7208-02 | December, 2005 |

**Description of Changes**

Added the following new feature:

- HTTPC Adapter, page 5

Removed the following, deprecated feature:

- Database Adapter

Added the following section to the document:

- Code Samples, page 1

| Cisco Service Control Release | Part Number | Publication Date |
|---|---|---|
| Release 2.5.5 | OL-7208-01 | February, 2005 |

**Description of Changes**

- First version of this document.

## Audience

This guide is intended for the networking or computer technician responsible for the onsite installation and configuration of the Cisco Service Control Management Suite (SCMS) Collection Manager (CM). It is also intended for the operator responsible for the daily operations of the CM, allowing the Service Provider operator to make enhancements in a subscriber-oriented environment.

## Document Organization

This guide contains the following chapters:

| Chapter | Title | Description |
|---|---|---|
| 1 | General Overview, page 1 | Provides a functional overview of the Cisco Service Control solution |
| 2 | How the Collection Manager Works, page 1 | Provides detailed information about the functionality of the Collection Manager components |
| 3 | Installing the Collection Manager, page 1 | Describes the procedures for installing the Collection Manager and its database, and explains how to run the Collection Manager |
| 4 | Managing the Collection Manager, page 1 | Explains how to use utility scripts to view and update Collection Manager parameters and other information |
| 5 | Managing Databases and the CSV Repository, page 1 | xplains how to use utility scripts to manage the Collection Manager database and the CSV repository |
| 6 | Database Configuration, page 1 | Explains how to configure the Collection Manager to work with your database |
| A | Code Samples, page 1 | Provides sample listings of code for configuration files |

## Document Conventions

This guide uses the following conventions:

- **Bold** is used for commands, keywords, and buttons.
- *Italics* are used for command input for which you supply values.
- `Screen` font is used for examples of information that are displayed on the screen.
- **`Bold screen`** font is used for examples of information that you enter.
- Vertical bars ( | ) indicate separate alternative, mutually exclusive elements.
- Square brackets ( [ ] ) indicate optional elements.
- Braces ( {} ) indicate a required choice.
- Braces within square brackets ( [{}] ) indicate a required choice within an optional element.

**Note** Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the guide.

**Timesaver** Means the *described action saves time*. You can save time by performing the action described in the paragraph.

**Caution** Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

**Warning** **Means *danger*. You are in a situation that could cause bodily injury. Before you work on any equipment, you must be aware of the hazards involved with electrical circuitry and familiar with standard practices for preventing accidents. To see translated versions of warnings, refer to the *Regulatory Compliance and Safety Information* document that accompanied the device.**

## Related Documentation

Use this *Cisco SCMS Collection Manager User Guide* in conjunction with the following Cisco documentation:

- *Cisco Service Control Application for Broadband User Guide*
- *Cisco Service Control Application for Broadband Reference Guide*
- *Cisco SCA BB Service Configuration API Programmer Guide*
- *Cisco Service Control Application Reporter User Guide*

## Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

CHAPTER 1

# General Overview

This chapter provides a general overview of the Cisco Service Control solution. It introduces the Cisco Service Control concept and the Service Control capabilities.

It also briefly describes the hardware capabilities of the Service Control Engine (SCE) platform and the Cisco specific applications that together compose the total Cisco Service Control solution.

- Information About the Cisco Service Control Concept
- Cisco Service Control Capabilities
- The SCE Platform
- Information About Management and Collection
- The Cisco Service Control Application for Broadband

## Information About the Cisco Service Control Concept

- The Cisco Service Control Solution

### The Cisco Service Control Solution

The Cisco Service Control solution is delivered through a combination of purpose-built hardware and specific software solutions that address various service control challenges faced by service providers. The SCE platform is designed to support classification, analysis, and control of Internet/IP traffic.

Service Control enables service providers to create profitable new revenue streams while capitalizing on their existing infrastructure. With the power of Service Control, service providers have the ability to analyze, charge for, and control IP network traffic at multigigabit wire line speeds. The Cisco Service Control solution also gives service providers the tools they need to identify and target high-margin content-based services and to enable their delivery.

As the downturn in the telecommunications industry has shown, IP service providers' business models need to be reworked to make them profitable. Having spent billions of dollars to build ever larger data links, providers have incurred massive debts and faced rising costs. At the same time, access and bandwidth have become commodities where prices continually fall and profits disappear. Service providers have realized that they must offer value-added services to derive more revenue from the traffic and services running on their networks. However, capturing real profits from IP services requires more than simply running those services over data links; it requires detailed monitoring and precise, real-time control and awareness of services as they are delivered. Cisco provides Service Control solutions that allow the service provider to bridge this gap.

- Service Control for Wireless Service Providers
- Service Control for DSL Providers and ISPs
- Service Control for Cable MSOs

## Service Control for Wireless Service Providers

Wireless service providers are successfully rolling out 2.5G and 3G-based data services to their subscribers. These services are expected to significantly increase much needed average revenue per user (ARPU) for sustained business models and rapid rollout of new services.

These data services require new ways of offering services and new ways of billing these services to subscribers. The Cisco Service Control solutions enable:

- Support for multiple billing models
- Elimination of revenue leakage via real-time service control
- Flexible pricing plans—Postpaid, prepaid, MRC, pay-per-use
- Content-based billing for various applications
- Subscription-based and tiered application services

## Service Control for DSL Providers and ISPs

DSL providers and ISPs targeting residential and business broadband customers must find new ways to get maximum leverage from their existing infrastructures, while differentiating their offerings with enhanced IP services.

Cisco products add a new layer of service intelligence and control to existing networks They:

- Provide granular visibility into network usage
- Automatically enforce application SLAs or acceptable use policies
- Implement different service levels for different types of customers, content, or applications
- Deploy from network edge to network core for end-to-end service control
- Integrate Cisco solutions easily with existing network elements and BSS/OSS systems

## Service Control for Cable MSOs

Cable MSOs have successfully deployed high-speed cable modem services to millions of homes. Now they must move beyond providing commodity broadband access by introducing differentiated services and by implementing the service control necessary to fully manage service delivery through their broadband infrastructure. Cisco Service Control solutions enable:

- Reporting and analyzing network traffic at subscriber and aggregate level for capacity planning
- Identification of network abusers who are violating the Acceptable Use Policy (AUP)
- Identification and management of peer-to-peer traffic, NNTP (news) traffic, and spam abusers Enforcement of the AUP
- Limiting or preventing the use of servers in the subscriber residence and the use of multiple (unpaid) computers
- Customer-intuitive tiered application services and guarantee application SLAs
- Full integration with standard or legacy OSS for subscriber management and billing

## Cisco Service Control Capabilities

The core of the Cisco Service Control solution is the purpose-built network hardware device: the Service Control Engine (SCE). The core capabilities of the SCE platform, which support a wide range of applications for delivering Service Control solutions, include:

- Subscriber and application awareness—Application-level drilling into IP traffic for real-time understanding and controlling of usage and content at the granularity of a specific subscriber.
  - Subscriber awareness—The ability to map between IP flows and a specific subscriber in order to maintain the state of each subscriber transmitting traffic through the SCE platform and to enforce the appropriate policy on this subscriber's traffic.

  Subscriber awareness is achieved either through dedicated integrations with subscriber management repositories, such as a DHCP or a Radius server, or via sniffing of Radius or DHCP traffic.
  - Application awareness—The ability to understand and analyze traffic up to the application protocol layer (Layer 7).

  For application protocols implemented using bundled flows (such as FTP, which is implemented using Control and Data flows), the SCE platform understands the bundling connection between the flows and treats them accordingly.
- Application-layer, stateful, real-time traffic control—The ability to perform advanced control functions, including granular BW metering and shaping, quota management, and redirection, using application-layer stateful real-time traffic transaction processing. This requires highly adaptive protocol and application-level intelligence.
- Programmability—The ability to quickly add new protocols and easily adapt to new services and applications in the ever-changing service provider environment. Programmability is achieved using the Cisco Service Modeling Language (SML).

  Programmability allows new services to be deployed quickly and provides an easy upgrade path for network, application, or service growth.
- Robust and flexible back-office integration—The ability to integrate with existing third-party systems at the service provider, including provisioning systems, subscriber repositories, billing systems, and OSS systems. The SCE provides a set of open and well-documented APIs that allows a quick and robust integration process.
- Scalable high-performance service engines—The ability to perform all these operations at wire speed.

## The SCE Platform

The SCE family of programmable network devices is capable of performing application-layer stateful-flow inspection of IP traffic, and controlling that traffic based on configurable rules. The SCE platform is a purpose-built network device that uses ASIC components and RISC processors to go beyond packet counting and delve deeper into the contents of network traffic. Providing programmable, stateful inspection of bidirectional traffic flows and mapping these flows with user ownership, the SCE platforms provide real-time classification of network usage. This information provides the basis of the SCE platform advanced traffic-control and bandwidth-shaping functionality. Where most bandwidth shaper functionality ends, the SCE platform provides more control and shaping options, including:

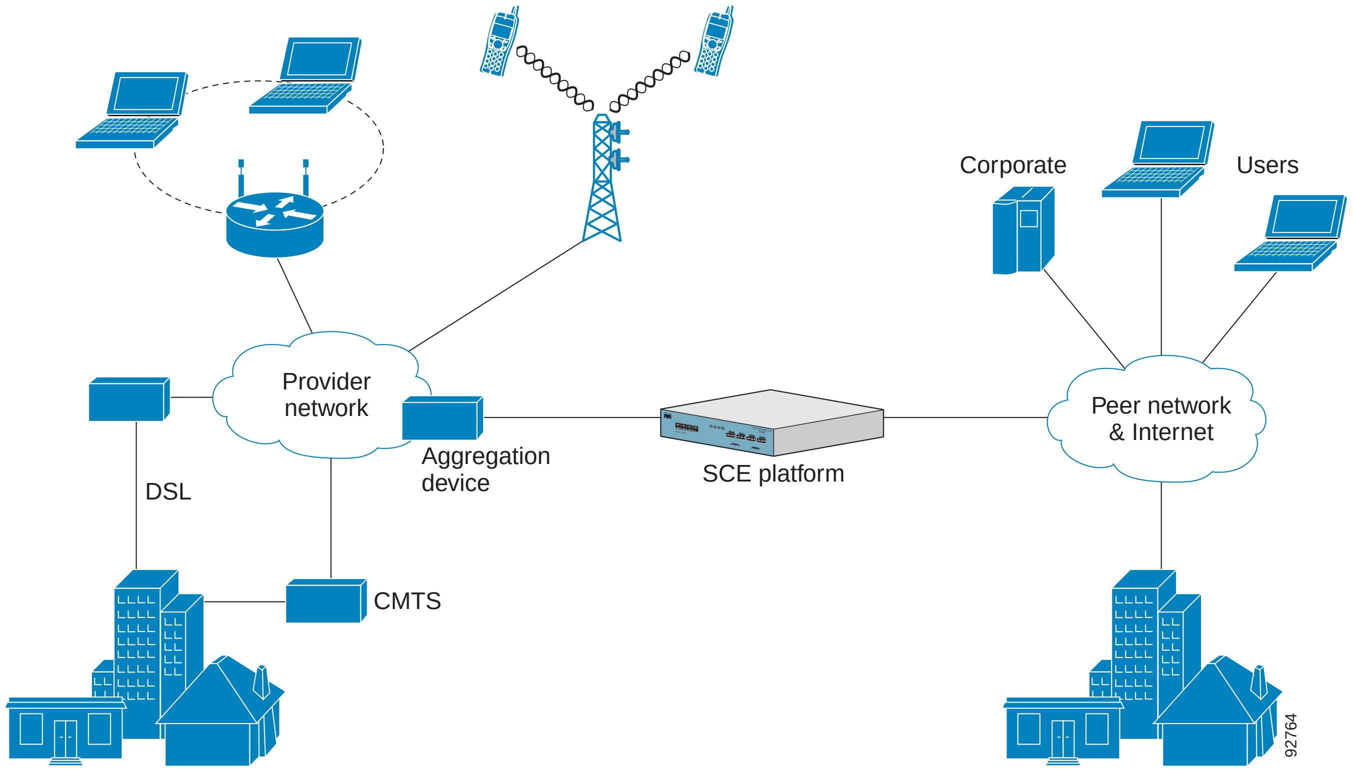
- Layer 7 stateful wire-speed packet inspection and classification
- Robust support for over 600 protocols and applications, including:

- General—HTTP, HTTPS, FTP, TELNET, NNTP, SMTP, POP3, IMAP, WAP, and others
- P2P file sharing—FastTrack-KazaA, Gnutella, BitTorrent, Winny, Hotline, eDonkey, DirectConnect, Piolet, and others
- P2P VoIP—Skype, Skinny, DingoTel, and others
- Streaming and Multimedia—RTSP, SIP, HTTP streaming, RTP/RTCP, and others

- Programmable system core for flexible reporting and bandwidth control
- Transparent network and BSS/OSS integration into existing networks
- Subscriber awareness that relates traffic and usage to specific customers

The following diagram illustrates a common deployment of an SCE platform in a network.

***Figure 1-1 SCE Platform in the Network***



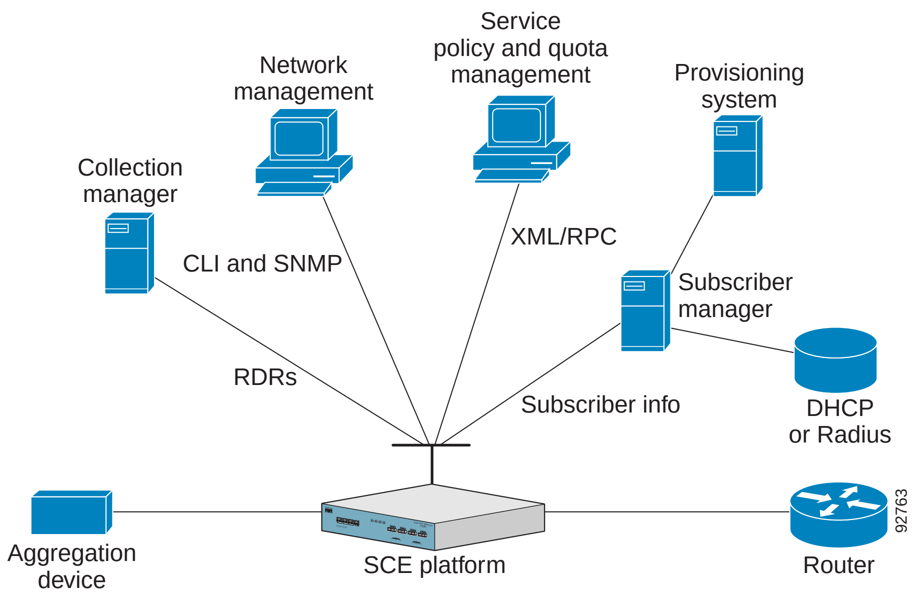
## Information About Management and Collection

The Cisco Service Control solution includes a complete management infrastructure that provides the following management components to manage all aspects of the solution:

- Network management
- Subscriber management
- Service Control management

These management interfaces are designed to comply with common management standards and to integrate easily with existing OSS infrastructure.

***Figure 1-2 Service Control Management Infrastructure***



## Network Management

Cisco provides complete network FCAPS (Fault, Configuration, Accounting, Performance, Security) Management.

Two interfaces are provided for network management:

- Command-line interface (CLI)—Accessible through the Console port or through a Telnet connection, the CLI is used for configuration and security functions.
- SNMP—Provides fault management (via SNMP traps) and performance monitoring functionality.

## Subscriber Management

Where the Cisco Service Control Application for Broadband (SCA BB) enforces different policies on different subscribers and tracks usage on an individual subscriber basis, the Cisco Service Control Management Suite (SCMS) Subscriber Manager (SM) may be used as middleware software for bridging between the OSS and the SCE platforms. Subscriber information is stored in the SM database and can be distributed between multiple platforms according to actual subscriber placement.

The SM provides subscriber awareness by mapping network IDs to subscriber IDs. It can obtain subscriber information using dedicated integration modules that integrate with AAA devices, such as Radius or DHCP servers.

Subscriber information may be obtained in one of two ways:

- Push Mode—The SM pushes subscriber information to the SCE platform automatically upon logon of a subscriber.
- Pull Mode—The SM sends subscriber information to the SCE platform in response to a query from the SCE platform.

### Service Configuration Management

Service configuration management is the ability to configure the general service definitions of a service control application. A service configuration file containing settings for traffic classification, accounting and reporting, and control is created and applied to an SCE platform. The SCA BB application provides tools to automate the distribution of these configuration files to SCE platforms. This simple, standards-based approach makes it easy to manage multiple devices in a large network.

Service Control provides an easy-to-use GUI to edit and create these files and a complete set of APIs to automate their creation.

### Data Collection

All analysis and data processing functions of the SCE platform result in the generation of Raw Data Records (RDRs). These RDRs are processed by the Cisco Service Control Management Suite Collection Manager. The Collection Manager software is an implementation of a collection system that receives RDRs from one or more SCE platforms. It collects these records and processes them in one of its adapters. Each adapter performs a specific action on the RDR.

RDRs contain a wide variety of information and statistics, depending on the configuration of the system. There are three main categories of RDRs:

- Transaction RDRs—Records generated for each transaction, where a transaction is a single event detected in network traffic. The identification of a transaction depends on the particular application and protocol.
- Subscriber Usage RDRs—Records generated per subscriber, describing the traffic generated by that subscriber for a defined interval.
- Link RDRs—Records generated per link, describing the traffic carried on the link for a defined interval.

## The Cisco Service Control Application for Broadband

Cisco provides a specific solution that runs on top of the SCE platform and addresses the IP network control challenges that service providers face. This solution is the Cisco Service Control Application for Broadband (SCA BB).

SCA BB allows service providers to detect complex and evasive network application protocols (such as P2P), and to control them according to their business and service delivery requirements. It also enables the creation of differentiated tiered services that the service provider uses to boost revenues and provide competitive services to end customers. SCA BB’s programmable application detection and subscriber awareness makes tiered service possible from a central point in the network. SCA BB requires no network changes or upgrades, and it is compatible with all existing IP network switches, routers, and infrastructure.

CHAPTER 2

# How the Collection Manager Works

This module provides detailed information about the functionality of the Collection Manager components

This module describes how the Cisco Service Control Management Suite (SCMS) Collection Manager (CM) works. It describes the Raw Data Records (RDRs) that the Service Control Engine (SCE) platforms produce and send to the Collection Manager, and provides an overview of the components of the CM software package. It also gives an overview of the database used to store the RDRs.

- The Data Collection Process
- Raw Data Records
- Information About the Collection Manager Software Package
- Information About Adapters
- Databases

## The Data Collection Process

Cisco SCE platforms create RDRs whose specifications are defined by the application running on the SCE platform, such as the Cisco Service Control Application for Broadband (SCA BB).

RDRs are streamed from the SCE platform using the simple, reliable RDR-Protocol . Integrating the collection of data records with the Service Control solution involves implementing RDR-Protocol support in the collection system (a straightforward development process).

After the CM receives the RDRs from the SCE platforms, CM software modules recognize and sort the various types of RDR, based on preset categories and according to type and priority, and queue them in persistent buffers.

One or more of the CM adapters then processes each RDR. Each adapter performs a specific function on RDRs (stores it in a CSV formatted file on a local machine, sends it to an RDBMS application, or performs custom operations).

You can use preinstalled utility scripts to determine many of the parameters that influence the behavior of the CM.

## Raw Data Records

Raw Data Records (RDRs) are reports produced by SCE platforms. The list of RDRs, their fields, and their semantics depend on the specific Service Control Protocol (SCP) application. Each RDR type has a unique ID known as an RDR tag.

The following are some examples of RDRs produced by SCP applications:

- Periodic Subscriber usage report—SCE platforms are subscriber-aware network devices; they can report usage records per subscriber.

  These RDRs typically contain a subscriber identifier (such as the OSS subscriber ID), the traffic type (such as HTTP, Streaming, or Peer-to-Peer traffic), and usage counters (such as total upstream and downstream volume). These types of usage reports are necessary for usage-based billing services, and for network analysis and capacity planning.

  The SCA BB application Subscriber Usage RDRs are in this category.

- Transaction level report—SCE platforms perform stateful tracking of each network transaction conducted on the links on which they are situated. Using this statefulness, the SCP tracks a number of OSI Layer 7 protocols (such as HTTP, RTSP, SIP, or Gnutella) to report on various application level attributes.

  These RDRs typically contain transaction-level parameters ranging from basic Layer 3-4 attributes (such as source IP, destination IP, and port number) to protocol-dependant Layer 7 attributes (such as user-agent, host-name for HTTP, or e-mail address of an SMTP mail sender), and also generic parameters (such as time of day and transaction duration). These RDRs are important for content-based billing schemes and for detailed usage statistics.

  The SCA BB application Transaction RDRs are in this category.

- SCP application activity reports—The SCP application can program the SCE platform to perform various actions on network traffic. These actions include blocking transactions, shaping traffic to certain rates and limits, and performing application-level redirections. When such an operation is performed, the SCP application may produce an RDR.

  The SCA BB application Breaching RDRs and Blocking RDRs are in this category. Breaching RDRs are generated when the system changes its active enforcement on a subscriber (because usage exceeded a certain quota). Blocking RDRs are generated when an SCE platform blocks a network transaction (according to rules contained in the current service configuration).

## Information About the Collection Manager Software Package

The Collection Manager Software Package consists of a group of processing and sorting modules. These include the following components:

- RDR Server
- Categorizer
- Priority Queues and Persistent Buffers

### RDR Server

As each incoming RDR arrives from an SCE platform, the RDR Server adds an arrival timestamp and the ID of the source SCE platform to it, and then sends the RDR to the Categorizer.

### Categorizer

The Categorizer classifies each RDR according to its RDR tag. It decides the destination adapters for the RDR and through which Priority Queue it should be sent.

An RDR can be mapped to more than one adapter. A qualified technician defines the flow in a configuration file based on user requirements.

### Priority Queues and Persistent Buffers

Each adapter has one or more Priority Queues; a persistent buffer is assigned to each Priority Queue.

A Priority Queue queues each RDR according to its priority level and stores it in a persistent buffer until the adapter processes it.

A persistent buffer is a non-volatile storage area that ensures that the system processes RDRs even in cases of hardware, software, or power failures.

## Information About Adapters

Adapters are software modules that transform RDRs to match the target system's requirements, and distribute the RDRs upon request. At this time, the following adapters are shipped with the system:

- JDBC Adapter
- CSV Adapter
- Topper/Aggregator (TA) Adapter
- Real-Time Aggregating (RAG) Adapter
- HTTPC Adapter

Some of the adapters send data to the database or write it to CSV files. The structures of the database tables, and the location and structures of these CSV files are described in the *Cisco Service Control Application for Broadband Reference Guide* .

Each adapter has its own configuration file; all the configuration files are similar in structure. For a sample RAG Adapter configuration file, see The ragadapter.conf File, page A-4.

- JDBC Adapter
- CSV Adapter
- TA Adapter
- RAG Adapter
- HTTPC Adapter

### JDBC Adapter

The JDBC Adapter receives RDRs, processes them, and stores the records in a database.

This adapter is designed to be compatible with any database server that is JDBC-compliant, and transforms the records accordingly. The JDBC Adapter can be configured to use a database operating on a remote machine.

The JDBC Adapter is preconfigured to support the following databases:

- Sybase ASE 12.5 and 15.0
- Oracle 9.2 and 10
- MySQL 4.1 and 5

## CSV Adapter

The CSV Adapter receives RDRs, processes them, and writes the records to files on the disk in comma-separated value format. Using standard mechanisms such as FTP, a service provider's OSS or a third-party billing system can retrieve these records to generate enhanced accounting and network traffic analysis records.

## TA Adapter

The TA Adapter receives Subscriber Usage RDRs, aggregates the data they contain, and outputs 'Top Reports' to the database and aggregated daily statistics of all subscribers (not just the top consumers) to CSV files. Top Reports are lists of the top subscribers for different metrics (for example, the top 50 volume or session consumers in the last hour).

This adapter maintains a persistent saved state (saved to disk) to minimize any data loss in case of failure.

The TA Adapter, which uses the JDBC Adapter infrastructure, can be configured to use any JDBC-compliant database, either locally or remotely.

### TA Adapter Cycles

The TA Adapter works in two cycles: short and long. Cycles are fixed intervals at the end of which the adapter can output its aggregated information to the database and to a CSV file. The default interval for the short cycle is 1 hour; for the long cycle it is 24 hours (every day at midnight). The intervals (defined in minutes) and their start and end times are configurable.

**Note** The long-cycle interval must be a multiple of the short-cycle interval.

The activities in each cycle differ slightly, as follows:

- Short Cycle—At the end of every short cycle, the adapter:
  - Adds the cycle's aggregated Top Reports to the short cycle database table
  - Saves the current state file in case of power failure
- Long Cycle—At the end of every long cycle, the adapter:
  - Adds the cycle's aggregated Top Reports to the short cycle database table
  - Saves the current state file in case of power failure
  - Creates a CSV file containing the aggregated statistics for the long-cycle period

## RAG Adapter

The RAG Adapter processes RDRs of one or more types and aggregates the data from predesignated field positions into buckets. The contents of the buckets are written to CSV files.

- RAG Adapter Aggregation Buckets
- Flushing a Bucket

### RAG Adapter Aggregation Buckets

A RAG Adapter aggregation bucket is indexed by combining values from fields in the RDR. The indexing relation can be one-to-one or many-to-one.

The values in the bucket-identifying fields are processed using closures (equivalence classes), which are configured per type of RDR.

**Example:**

```
Bucket-identifying field = field number 3
Closures: 4 = 4,5,6; 10 = 8,10,11
Value in field 3 = 4, 5, or 6; field reported as 4
Value in field 3 = 8, 10, or 11; field reported as 10
```

The adapter can be configured to monitor the values in certain fields for change relative to the values in the first RDR that entered the bucket. For each monitored field, an action is performed when a value change is detected. The supported actions are:

- Checkpoint the bucket without aggregating this RDR into it, and start a new bucket with this RDR
- Issue a warning to the user log

Buckets, closures, triggers, and trigger actions are defined in an XML file. For a sample XML file, see The ragadapter.xml File, page A-5.

### Flushing a Bucket

When a bucket is flushed, it is written as a single line to a CSV file.

The trigger for flushing a bucket (a checkpoint ) is the earliest occurrence of any of the following:

- The time elapsed since the creation of the bucket has reached a configured amount
- The volume in an accumulated field in the bucket has exceeded a configured amount
- The adapter, or the whole CM, is going down
- An RDR having some new value (relative to the bucket contents) in some field arrived at the bucket

The trigger to close a CSV file is the earliest occurrence of one of the following:

- The time elapsed since creation of the file has reached a set amount
- The number of lines in the file has reached a set amount
- The adapter, or the whole CM, is going down

## HTTPC Adapter

The HTTPC Adapter receives RDRs, processes them, and sends them to a Policy Server over HTTP.

The HTTPC Adapter can be configured to set the various HTTP requests according to the various Policy Server modes and the required action for a specific flow.

The HTTPC Adapter receives only two types of RDR: one to signal to the Policy Server that a flow has started, the other to signal that the flow has ended.

## Databases

The CM can use either a bundled database or an external database to store RDRs supplied by the system's SCE platforms.

- Using the Bundled Database
- Using an External Database

### Using the Bundled Database

In bundled mode, the CM uses the Sybase Adaptive Server Enterprise database, which supports transaction-intensive enterprise applications, allows you to store and retrieve information online, and can warehouse information as needed.

The Sybase database is located on the same server as the other CM components. It uses a simple schema consisting of a group of small, simple tables. The JDBC Adapter sends converted RDRs to the database to be stored in these tables. Records can then be accessed using standard database query and reporting tools. (Cisco provides a template-based reporting tool that can generate reports on subscriber usage, network resource analysis, and traffic analysis; for information about the Service Control reporting tool, see the *Cisco Service Control Application Reporter User Guide* .)

Database maintenance is performed using operating system commands and scripts. The CM supports automatic purging of old records from the bundled database. By default, the report tables are automatically purged of every record that is more than two weeks old. The records are polled once every hour. Database maintenance can be configured using the **dbperiodic.py** utility script. For more information, see Managing the Periodic Deletion of Old Records, page 5-2.

### Using an External Database

Any JDBC-compliant database (for example, Oracle™ or MySQL) may be used with the CM in conjunction with the JDBC Adapter. In this case, the database can be local or remote. You should configure the JDBC Adapter to use this database, and also configure a database pack to supply the CM with the parameters of the database (such as its IP address and port). You should also supply a JDBC driver for the database, to be used by the adapter when connecting to it. For more details about configuring the CM to work with an external database, see Managing Databases and the CSV Repository, page 5-1.

CHAPTER 3

# Installing the Collection Manager

This module describes the procedures for installing the Collection Manager and its database, and explains how to run the Collection Manager.

- System Requirements
- How to Install the Collection Manager
- How to Uninstall the Sybase Database and the Service Control Software
- Upgrading the CM to Version 3.1

## System Requirements

The CM and its database are software components that run on a Server Platform. They can be installed on either of the following configurations:

- Sun SPARC machine running Solaris 9 or Solaris 10. (See Solaris Requirements )
- IA32 machine running Red Hat Enterprise Linux 3.0 or Red Hat Enterprise Linux 4.0. (See Red Hat Linux Requirements )
- Checking System Prerequisites
- Solaris Requirements
- Red Hat Linux Requirements
- Distribution Content
- Default Configuration Settings

## Checking System Prerequisites

The CM distribution contains a script, **check_prerequisites.sh** , located in the **install_scripts** directory, that helps to determine whether a system meets the requirements for installing a CM or the bundled Sybase database.

The script checks overall readiness of the system for a CM or Sybase installation. The main prerequisites checked are:

- CPU speed
- Amount of RAM
- Operating System version (Solaris 9 or 10, Red Hat Enterprise Linux 3.0 or 4.0)

- Additional required and optional packages
- Python installed and executable in path
- Free space for CM and Sybase homes
- Names for all NICs
- Sybase kernel parameters
- Locale and time zone formats

**check_prerequisites.sh**[ **--sybhome**=*SYBHOME*] [ **--cmhome**=*CMHOME*] [ **--datadir**=*DATADIR*]

*Table 3-1 check_prerequisites.sh Script Options*

| | |
|---|---|
| **--sybhome**=*SYBHOME* | Intended home directory for Sybase installation |
| **--datadir**=*DATADIR* | Intended data directory for Sybase data files (for the Datadir installation method) |
| **--cmhome**=*CMHOME* | Intended home directory for CM installation |

## Solaris Requirements

Collection Manager 3.1.0 or later can be installed on any Sun SPARC Machine running Solaris that conforms to the requirements listed in the following sections.

- Hardware
- Software and Environment
- Setting the Locale and Time Zone

### Hardware

- Minimum 500 MHz CPU
- Minimum 1 GB RAM per CPU
- Hard disk:
  - One hard disk, at least 18 GB
  - (Recommended for bundled installations) A second hard disk (at least 18 GB), to store Sybase data
- 100BASE-T network interface

### Software and Environment

- Solaris 5.9 64-bit build 04/01 or later (currently only Solaris 5.9 and 5.10 are supported).
- Solaris Core Installation.
- The following additional packages should be installed:

| | | |
|---|---|---|
| system | SUNWbash | GNU Bourne-Again shell (bash) |
| system | SUNWgzip | The GNU Zip (gzip) compression utility |

| | | |
|---|---|---|
| system | SUNWzip | The Info-Zip (zip) compression utility |
| system | SUNWlibC | Sun Workshop Compilers Bundled libC |
| system | SUNWlibCx | Sun WorkShop Bundled 64-bit libC |

- If you are installing the CM in bundled mode with the Sybase database, the following package should also be installed:

| | | |
|---|---|---|
| system | SUNWipc | Interprocess Communication |

- (Optional) The following packages may be installed (for sysadmin applications such as sys-unconfig):

| | | |
|---|---|---|
| system | SUNWadmap | System administration applications |
| system | SUNWadmc | System administration core libraries |

- To use the Python scripts, a Python interpreter version 2.2.1 or later must be present on the system. One way to get such an interpreter is to install the following package:

| | | |
|---|---|---|
| application | SMCpythn (Solaris 9)<br>SMCpython (Solaris 10) | Python |

- The Python package requires the installation of two additional packages:

| | | |
|---|---|---|
| application | SMClibgcc | libgcc |
| application | SMCncurs | ncurses |

- These packages can be downloaded from http://sunfreeware.com/

  The root (/) partition must have at least 104 MB of free space to install these packages.
- The latest recommended patches from Sun should be applied:
  - For Solaris 9, go to http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/xos-9&nav=pub-patches
  - For Solaris 10, go to http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/xos-10&nav=pub-patches
  - For Java, go to http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/J2SE
- If you are using Sybase, current Solaris patches recommended by Sybase should be installed:
  - Go to http://my.sybase.com/detail?id=1016173
- At least 8 GB free on the partition where the CM is to be installed. (This is used for CSV storage and persistent buffers.)
- (For installations with bundled Sybase) At least 3 GB free on one partition for the Sybase home directory.

- (For installations with bundled Sybase) Free space on a single partition to hold the desired size of the Sybase data and logs (these sizes are configurable at install time).
- (Optional, and only for installations with bundled Sybase) Install the sudo package (from, for example, http://sunfreeware.com), and configure the following line in the sudoers file:

  ```
  scmscm ALL= NOPASSWD: XXX/scripts/dbconf.sh
  ```

  where XXX is the intended home directory for scmscm.

  If you choose not to install sudo: in the rare event of a Sybase crash, the CM will not be able to revive the database by itself.
- (For installations with bundled Sybase where the legacy (pre-3.0) Cisco Service Control Application Suite (SCAS) Reporter is to be used) An FTP server should be listening on port 21 so that the SCA Reporter can authenticate against it.
- (For installations with bundled Sybase) Verify before installation that all IP addresses that are configured for the machine NICs have host names associated with them in **/etc/hosts** or in another active naming service. (This is a limitation of Sybase Adaptive Server Enterprise.)
- (For installations with bundled Sybase) The kernel should be configured with at least:
  - 512000000 bytes in shmmax
- Additionally, the IPC module should be loaded at startup. This is achieved by putting the following lines in the file **/etc/system** :

  ```
  forceload: sys/semsys
  forceload: sys/shmsys
  ```
- If you are using database periodic delete, the scmscm user should be able to schedule and run cron jobs.

## Setting the Locale and Time Zone

- For correct CM and Sybase operation, US English locale must be used.

  The easiest way to set the locale is by putting the following line in the **/etc/TIMEZONE** configuration file (changes in this file need a restart to take effect):

  ```
  LANG=en_US
  ```

  Solaris also needs to have this locale installed. Verify that the locale is installed by checking that the directory **/usr/lib/locale/en_US** exists. If the directory does not exist, install the locale files from the Solaris CDs.
- Setting the OS time zone as an offset from GMT in POSIX format is not recommended, and may lead to problems. Best is to set the time zone in the **/etc/TIMEZONE** configuration file by (supported) country name, as in the following example.

  ```
  TZ=Japan
  ```

  Verify that the country name is supported as a time zone setting by checking that it is listed in the directory **/usr/share/lib/zoneinfo**.

  If GMT offset must be used, use the zoneinfo format by prepending an **:Etc/** prefix, as in the following example:

  ```
  TZ=:Etc/GMT+5
  ```

## Red Hat Linux Requirements

Collection Manager 3.1.0 or later can be installed on any i386 running Red Hat Linux that conforms to the requirements listed in the following sections.

- Hardware
- Software and Environment
- Setting the Locale and Time Zone

### Hardware

- Minimum 800 MHz CPU
- Minimum 1 GB RAM per CPU
- Hard disk:
  - One hard disk, at least 18 GB
  - (Recommended for bundled installations) A second hard disk (at least 18 GB), to store Sybase data
- 100BASE-T network interface

### Software and Environment

- Red Hat Linux 3.0 or 4.0.
- Red Hat Enterprise "Base" Installation.
- (For installations with bundled Sybase) The following additional package should be installed:

  ```
  compat-libstdc++
  ```
- This package is available on the Red Hat installation CD.
- Latest recommended patches from Red Hat should be applied.
- (For installations with bundled Sybase) Current patches recommended by Sybase should be installed.
- At least 8 GB free on the partition where the CM is to be installed. (This is used for CSV storage and persistent buffers.)
- (For installations with bundled Sybase) At least 1 GB free on some partition for the Sybase home directory.Optional, and only for installation with a bundled database) Install the sudo package and configure the following line in the sudoers file:

  scmscm ALL= NOPASSWD: XXX/scripts/dbconf.sh

  where XXX is the intended home directory for scmscm.

  If you choose not to install sudo, in the rare event of a Sybase crash, the CM will not be able to revive the database by itself.
- (For installations with bundled Sybase where the legacy (pre-3.0) Cisco Service Control Application Suite (SCAS) Reporter is to be used) An FTP server should be listening on port 21 so that the SCA Reporter can authenticate against it.
- (For installations with bundled Sybase) Verify before installation that all IP addresses that are configured for the machine NICs have host names associated with them in **/etc/hosts** or in another active naming service. (This is a limitation of Sybase Adaptive Server Enterprise.)

- (For installations with bundled Sybase) The kernel should be configured with at least:
  - 512000000 bytes in shmmax
- If you are using database periodic delete, the scmscm user should be able to schedule and run cron jobs.

### Setting the Locale and Time Zone

- For correct CM and Sybase operation, US English locale ( **en_US** ) must be used.

## Distribution Content

The Collection Manager installation kit contains installation scripts for installing the CM and the Sybase database.

It also contains:

- Scripts to support file gathering
- Scripts for periodic Sybase maintenance

## Default Configuration Settings

Settings for the CM are configured during installation. These settings include which adapters should be enabled and their locations, Priority Queue parameters, the target adapters for each type of RDR (by RDR tag value), and various logging policies. Only qualified personnel should change these settings.

## How to Install the Collection Manager

This section describes how to install CM version 3.1.0 or later and the Sybase database on a computer running Solaris or Red Hat Linux.

- Ports Used by the Collection Manager Software
- Installing the Sybase Database
- Installing the Collection Manager Software

## Ports Used by the Collection Manager Software

The following table describes the TCP/UDP ports on which the CM software and associated components (such as the Sybase database) listen. This table may help the network administrator understand the behavior of the software and its adherence to the security policy.

The ports listed are those on which the device listens constantly. You should allow access on these port numbers; otherwise, certain operations may fail.

Some operations (such as file transfer) cause a device to *temporarily* open ports other than those listed; however, these ports close automatically when the operation ends.

***Table 3-2 Ports that the CM Listens on Constantly***

| Port Number | Description |
|---|---|
| 33000 | Used by the SCE devices to send RDRs for data collection. |
| 21 | Used by the legacy (pre-3.0) SCAS Reporter to authenticate against the CM user on the CM machine. |
| 33001 | Internal Collection Manager. **Note** Access is required only from the local machine; external access can be blocked. |
| 9092 | HTTP technician interface. |
| 4100 | (For installations with bundled Sybase) Sybase database connectivity through ODBC/JDBC. Required for access to the database. |
| 1099—1120 | RMI. Used as the management interface between the data collector and the Service Control management server. |
| 22000 | FTP server of the CM. **Note** FTP transactions may listen on other ports (22001 to 22100) for data transfer, as negotiated by the protocol. |
| 7787 | Internal logging of the management user log. **Note** Access is required only from the local machine; external access can be blocked. |
| 14375 | Used by the Cisco Service Control Application Suite for Broadband (SCAS BB) Console to send symbol definitions ( **values.ini** ) to the CM. |

## Installing the Sybase Database

If you do not want to install Sybase (for example, when working in unbundled mode), Installing the Collection Manager Software.

**Note** If at any point during the installation you want to reverse the Sybase installation actions (for example, in the rare case that an installation is interrupted because of a power failure), do the following:

1. Log on as the root user.
2. Kill any Sybase processes by typing **pkill -u sybase**.
3. Remove the Sybase user and home directory by typing **userdel -r sybase**.
4. Restart the Sybase installation process from the beginning.

## Actions Performed by installsyb.sh

The **installsyb.sh** script installs the Sybase database. The script performs the following actions:

- Verifies the **shmem** setting for Sybase in **/etc/system**. If the setting is not there, the script inserts it and reboots (after prompting the user).
- Adds a user sybase and group sybase.
- Runs the Sybase installer for your platform.
- Builds a Sybase server including Sybase users and passwords.
- Starts Sybase.
- Runs SQL scripts to create the Collection Manager database structure. This is a lengthy process that involves restarting Sybase several times.

## Prerequisites

Log on as the root user and make the distribution kit contents available on your system or local network.

### SUMMARY STEPS

1. Change directory to **sybase** in the distribution kit root.
2. Run the script **installsyb.sh**

### DETAILED STEPS

**Step 1** Change directory to **sybase** in the distribution kit root.

**Step 2** Run the script **installsyb.sh**

The script usage is as follows:

```
installsyb.sh --sybhome=SYBHOME{ --datadir=DATADIR}
```

- **SYBHOME** is the home directory of the Sybase user (and should have 1 GB free)
- Select one of the following data location options:
  - Specify **--datadir=DATADIR** , where **DATADIR** is a directory in which all Sybase data is to be stored.

  This location should be in a partition where at least 15 GB is free.
- If you specify a **DATADIR** , all Sybase data is stored as normal files in that directory, with default sizes of 10 GB for data, 3 GB for logs, and 3 GB for Sybase temporary storage. The ownership of the directory is changed to the Sybase user during installation.

## Installing the Collection Manager Software

**Note** If at any point during the installation you want to reverse the Sybase installation actions (for example, in the rare case that an installation is interrupted because of a power failure), do the following:

1. Log on as the root user.
2. Kill any Sybase processes by typing **pkill -u sybase**.
3. Remove the Sybase user and home directory by typing **userdel -r sybase**.
4. Restart the Sybase installation process from the beginning.

### Information About the install-cm.sh Script

The **install-cm.sh** script is used to install the Collection Manager Server.

#### install-cm.sh Options

The usage message for the **install-cm.sh** script is:

```
Usage: install-cm.sh [-h] (-d CMDIR | -o)
Options: -d CMDIR   select directory for ~scmscm
(must not exist and must be on 8 GB free partition)
-o   upgrade the existing installation
while preserving the current configuration
(can't be used with -d)
-h   print this help and exit
```

Description of the options:

```
-d CMDIR
Used to designate the directory of the newly created
scmscm user's home. Should be the name of a
non-existing directory, whose parent resides on a
partition where at least 8 GB is free.
As an alternate to this option, you can specify -o :
-o
Use this option when you wish to upgrade the existing
installation while preserving the current configuration.
(can't be used with  -d)
```

#### Actions Performed by install-cm.sh

The **install-cm.sh** script performs the following actions:

- If needed, creates an scmscm user and an scmscm group
- Optionally, creates the home for this user
- Populates the home of scmscm with CM files and scripts
- Installs the following extra component:
  - private JRE in **~scmscm/cm/lib**
- Creates boot script symbolic links for the sybase and scmscm users in /etc/init.d and /etc/rcX.d

#### SUMMARY STEPS

1. Change directory to **install-scripts** under the distribution kit root
2. Run the **install-cm.sh** script

3. After the script completes, set a password for the scmscm user
4. Increase the amount of memory allocated to the Topper/Aggregator Adapter
5. Open the file **~scmscm/cm/config/cm.conf**.
6. Locate the setting containing **TAAdapter** in the **[adapter_mem]** section.
7. Change the default value (512 MB) to a larger value.
8. Save and close the file.
9. Increase the amount of memory allocated to the Real-Time Aggregating Adapter
10. Open the file **~scmscm/cm/config/cm.conf**.
11. Locate the setting containing **RAGAdapter** in the **[adapter_mem]** section.
12. Change the default value (512 MB) to a larger value.
13. Save and close the file.
14. For each adapter that your application will use, configure the adapter to point to the application
15. Install and activate the periodic delete procedures for the database tables.
16. Install the periodic delete procedures
17. Activate the automatic invocation of the periodic delete procedures
18. Set the Service Control Engine (SCE) device time zone
19. Start the CM

## DETAILED STEPS

**Step 1** Change directory to **install-scripts** under the distribution kit root

**Step 2** Run the **install-cm.sh** script

For more information about the **install-cm.sh** script options, see install-cm.sh Options.

For additional information about the script, see Actions Performed by install-cm.sh.

**Step 3** After the script completes, set a password for the scmscm user

Run the following command to set the password for the scmscm user:

```
passwd scmscm
```

Be sure to record the passwork that you choose.

**Step 4** Increase the amount of memory allocated to the Topper/Aggregator Adapter

If you are going to run an application that uses the Topper/Aggregator (TA) Adapter, you may need to increase the amount of memory allocated to this adapter. This depends on the number of subscribers to be handled by the CM. To increase the memory allocation:

a. Open the file **~scmscm/cm/config/cm.conf**.
b. Locate the setting containing **TAAdapter** in the **[adapter_mem]** section.
c. Change the default value (512 MB) to a larger value.

   For example, to allocate 1024 MB of memory, set the value to **-Xmx1024M**.
d. Save and close the file.

**Step 5** Increase the amount of memory allocated to the Real-Time Aggregating Adapter

If you are going to run an application that uses the Real-Time Aggregating (RAG) Adapter, you may need to increase the amount of memory allocated to this adapter. This depends on the number of subscribers to be handled by the CM and on your RAG Adapter configuration. To change the setting:

**a.** Open the file **~scmscm/cm/config/cm.conf**.

**b.** Locate the setting containing **RAGAdapter** in the **[adapter_mem]** section.

**c.** Change the default value (512 MB) to a larger value.

For example, to allocate 1024 MB of memory, set the value to **-Xmx1024M**.

**d.** Save and close the file.

**Note** To use an external database, you must also configure a dbpack to enable the CM to connect to the database. See Managing Databases and the CSV Repository, page 5-1 for details of how to do this.

**Step 6** For each adapter that your application will use, configure the adapter to point to the application

- JDBC Adapter— Edit the file **~scmscm/cm/config/jdbcadapter.conf** , and, in the **[app]** section, change the value of **app_conf_dir** to point to your desired application.

  By default, it is set to **apps/scasbb/3.1.0**.

  TA Adapter— Edit the file **~scmscm/cm/config/taadapter.conf** , and, in the **[app]** section, change the value of **app_conf_dir** to point to your desired application. By default, it is set to **apps/scasbb/3.1.0**.

**Step 7** Install and activate the periodic delete procedures for the database tables.

(For more information about configuring the behaviour of periodic delete, see Managing the Periodic Deletion of Old Records, page 5-2.)

**Note** If reports are sent to the database and you do not install and activate the periodic delete procedures, the second disk may overflow.

**a.** Install the periodic delete procedures

Log on as the scmscm user, start the CM, wait 1-2 minutes for the database tables to be created, and then run the script:

```
~scmscm/db_maint/create_periodic_del_procs.sh.
```

**b.** Activate the automatic invocation of the periodic delete procedures

Run the following command:

```
~scmscm/scripts/dbperiodic.py --load
```

**Step 8** Set the Service Control Engine (SCE) device time zone

Use the following command to set the time zone:

```
~scmscm/cm/bin/jselect-sce-tz.sh --offset=offset-in-minutes from GMT
```

For example, if the SCE device is located in GMT+2, use:

```
~scmscm/cm/bin/jselect-sce-tz.sh --offset=120
```

If the SCE is located in GMT-10, use:

```
~scmscm/cm/bin/jselect-sce-tz.sh --offset=-600
```

**Note** This script should be run on every occasion that the time zone of the SCE is updated; for example, when updating the time zone when moving to daylight savings time.

**Step 9** Start the CM

Start the CM by running the following command:

```
~scmscm/cm/bin/cm start
```

## How to Uninstall the Sybase Database and the Service Control Software

- Uninstalling Sybase
- Uninstalling the Service Control Software

### Uninstalling Sybase

#### SUMMARY STEPS

1. Log in as the root user
2. Uninstall Sybase
3. Edit **/etc/system** and remove the Sybase **shmem** setting

#### DETAILED STEPS

**Step 1** Log in as the root user

**Step 2** Uninstall Sybase

Run the following commands to uninstall Sybase:

```
pkill -u sybase userdel -r sybase rm /etc/rc*.d/[SK]*sybase
```

**Step 3** Edit **/etc/system** and remove the Sybase **shmem** setting

### Uninstalling the Service Control Software

#### SUMMARY STEPS

1. Log in as the root user.
2. Uninstall the Service Control software

#### DETAILED STEPS

**Step 1** Log in as the root user.

**Step 2** Uninstall the Service Control software

Run the following commands to uninstall the Service Control software:

```
pkill -u scmscm userdel -r scmscm rm /etc/rc*.d/[SK]*scmscm
```

## Upgrading the CM to Version 3.1

### SUMMARY STEPS

1. Stop the CM
2. Install the new CM using the **install-cm.sh** script.

### DETAILED STEPS

**Step 1** Stop the CM

**Step 2** Install the new CM using the **install-cm.sh** script.

When upgrading, use the **-o**option to preserve the existing configuration.

The current scmscm user is used.

The database tables that are new in 3.1 will be created automatically when the CM comes up for the first time after the upgrade.

**Note** The upgrade to version 3.1 can be done only from version 3.x.

CHAPTER 4

# Managing the Collection Manager

This module explains how to use utility scripts to view and update Collection Manager parameters.

Any machine connected to the CM via, for example, Telnet or SSH can use utility scripts to monitor and manage the CM. The utility scripts are located in the installation directory of the CM.

For information on managing the database and the CSV repository, see Managing Databases and the CSV Repository, page 5-1.

- Using Utility Scripts
- Configuring the CM
- Configuring the Categorizer
- Information About Monitoring System Health
- Managing Users
- Managing Virtual Links
- How to Monitor the CM

## Using Utility Scripts

The following are general instructions for using the utility scripts:

- To invoke any script, log in as the scmscm user, except where otherwise noted. An attempt to run these scripts as the root user will result in an error.
- To display a description of the script, with an explanation of all flags and parameters, invoke the script with the help flag.

**Note** There is a slight variation in the help flag. Scripts for managing the CM use **--help**; scripts for managing the database use **-h**. Consult the specific script definition.

The following example shows how to display a description of the **dbperiodic.py** script.

```
>~scmscm/scripts/dbperiodic.py --helpUsage:
~scmscm/scripts/dbperiodic.py --load
load configuration from
/export/home/scmscm/db_maint/dbperiodic.conf
~scmscm/scripts/dbperiodic.py --loadfile=FILE
load configuration from FILE
~scmscm/scripts/dbperiodic.py --dump
```

```
print the current configuration in INI format to standard output
~scmscm/scripts/dbperiodic.py --help
print this help message
```

**Note** Some of the scripts used to control and monitor the data-collector software use the Python scripting language. For more information about Python, go to http://www.python.org.

## Configuring the CM

Use utility scripts to:

- Specify which servers are to be activated at startup
- Start or stop the database
- Start or stop an adapter
- Drop a Service Control Engine (SCE) connection

The following scripts are used to configure the CM:

- `~scmscm/setup/on-boot.py`
- `~scmscm/scripts/adapterconf.py`
- `~scmscm/scripts/dbconf.sh`
- `~scmscm/scripts/sceconf.py`

For information about scripts for managing the database and the CSV repository, see Managing Databases and the CSV Repository, page 5-1.

The following files are also used to configure the CM:

- **cm.conf** —General configuration of the CM, including which adapters will be turned on when the CM starts. See Enabling the Adapters.
- **queue.conf** —Configuration of the adapter queues, including which RDR tags will be associated with a specific adapter. See Configuring the Categorizer.

## Activating the Servers

To set which servers (CM or Sybase) are activated at startup, use the **on-boot.py** script:

**`~scmscm/setup/on-boot.py--cm=`***`flag`***`--sybase=`***`flag`*

Changes take effect the next time the system restarts.

**Note** Run the script with no parameters to see the current startup status of each component.

Run the following script as the scmscm to restart the Collection Manager:

**`~scmscm/cm/bin/cm restart`**

***Table 4-1 on-boot.py Options***

| | |
|---|---|
| **`--cm`**`={ `*`on`*` | `*`off`*`}` | Activate/do not activate the CM at startup. |
| **`--sybase`**`={ on | off }` | Activate/do not activate the Sybase server at startup. |

The following example shows how to set the CM and Sybase servers to be activated at startup. (This is the default setting of the script.)

```
>~scmscm/setup/on-boot.py --cm=on --sybase=on
```

## Controlling the Adapters

To shut down or bring up a configured adapter, or to list the currently running CM adapters, use the **adapterconf.py** script:

```
~scmscm/scripts/adapterconf.py--op=action[--adapter=adapter name]
```

*Table 4-2 adapterconf.py Options*

| | |
|---|---|
| **--op**=*start* | Bring up the adapter specified in the **adapter** parameter. |
| **--op**=*stop* | Shut down the adapter specified in the **adapter** parameter. |
| **--op**=*list* | List the currently running CM adapters. |
| **adapter**=*adapter name* | Identify the adapter to be operated on. Use only with **start** and **stop** actions. |
| **--help** | Display these options. |

To shut down an adapter, as the scmscm user, run the following script:

```
~scmscm/scripts/adapterconf.py--op=stop--adapter=adapter name
```

To bring up an adapter, as the scmscm user, run the following script:

```
~scmscm/scripts/adapterconf.py--op=start--adapter=adapter name
```

## Enabling the Adapters

An adapter can be defined to turn on when the CM starts by removing the remark character at the start of the appropriate line in the **cm.conf** file.

The following example defines the RAG adapter to turn on when the CM starts.

```
adapter.4 = com.cisco.scmscm.adapters.rag.RAGAdapter
```

The following example defines the CSV adapter to remain off when the CM starts.

```
#adapter.2 = com.cisco.scmscm.adapters.CSVAdapter
```

**Note** The value of the **adapter.<number>**must match the **adapter_id** parameter value defined in the queue.conf file for the corresponding adapter.

## Controlling the Database

To shut down or start the CM database, or to show the operational status of the database, use the **dbconf.sh** script:

```
~scmscm/scripts/dbconf.sh--op=action
```

The script can only be used with a bundled database.

**Note** This script only operates when the sudo package is installed. If you did not install sudo, you must log in as the root user and run the **/etc/init.d/sybase** script to start or stop Sybase.

*Table 4-3 dbconf.sh Options*

| | |
|---|---|
| `--op=start` | Start the CM database. |
| `--op=stop` | Shut down the CM database. |
| `--op=status` | Display the current operational status of the database. |

To shut down the CM database, as the scmscm user, run the following command:

```
~scmscm/scripts/dbconf.sh--op=stop
```

To start the CM database, as the scmscm user, run the following command:

```
~scmscm/scripts/dbconf.sh--op=start
```

## Dropping an SCE Connection

To drop a connection to a particular SCE, use the **sceconf.py** script:

```
~scmscm/scripts/sceconf.py--op=drop--ip=IP address
```

This script can be used only if the HTTP Adaptor of the CM is running.

This script is also used to display information about the SCE connection. (See Checking the SCE Connection )

*Table 4-4 sceconf.py Options*

| | |
|---|---|
| `Adapter=IP address` | Drop the connection at the specified IP address. |
| `--help` | Display these options. |

To drop an SCE connection, as the scmscm user, run the following command:

```
~scmscm/scripts/sceconf.py--op=drop--ip=IP address
```

## Configuring the Categorizer

The Categorizer classifies each RDR according to its RDR tag. An RDR can be routed to a specific adapter by adding its RDR tag to the tags parameter (a comma-separated list of RDR tags) of the adapter. This configuration is contained in the **queue.conf** file.

The following example configures the RDR tags **4042321920** and **4042321922** to be sent to the Topper/Aggregator Adapter.

```
# Topper/Aggregator Adapter
[topper-hi]
adapter_id=3
priority=3
warning_size=40000
maximum_size=50000
tags=4042321920,4042321922
```

**Note** The value of the **adapter_id** parameter must match the **adapter.<number>**defined in the **cm.conf** file for the corresponding adapter.

## Information About Monitoring System Health

The CM contains a small, expandable framework that monitors the system and issues alerts for predefined, potentially problematic conditions.

The following scripts are used to monitor the CM:

- `~scmscm/setup/monitor/setup-monitor.sh`
- `~scmscm/setup/monitor/monitor.sh`
- Installing the Periodic Checker
- Information About the Periodic Checker Script

## Installing the Periodic Checker

To make (or remove) an entry for **monitor.sh** , the periodic checker script, in the cron (periodic scheduler) subsystem, use the **setup-monitor.sh** script:

```
~scmscm/setup/monitor/setup-monitor.sh-a flag[-i flag]
```

***Table 4-5 setup-monitor.sh Options***

| | |
|---|---|
| `-a{ install| uninstall}` | Make/remove an entry for **monitor.sh** in the cron. |
| `-i{ 30m| 1h| 12h| 24h}` | Run **monitor.sh** every 30 minutes, 1 hour, 12 hours, or 24 hours. |

The following example shows how to install **monitor.sh** so that it will run once every 30 minutes.

```
$ ./setup-monitor.sh -a install -i 30m
```

The following example shows how to uninstall monitor.sh.

```
$ ./setup-monitor.sh -a uninstall
```

## Information About the Periodic Checker Script

- The Periodic Checker Script
- Tests

### The Periodic Checker Script

The periodic checker script, **monitor.sh** , calls a series of sub-scripts that monitor different aspects of a running system:

```
~scmscm/setup/monitor/monitor.sh{ -a| TEST NAME} [ -v] [ -d]
```

The script is not intended to be run from the command line, although you can do so. Test results are sent to the syslog subsystem and are logged in the file **/var/log/messages**.

***Table 4-6 monitor.sh Options***

| | |
|---|---|
| **-a** | Run all tests. |
| *TEST NAME* | The names of one or more tests. A test name is the test file name, without the leading digits and trailing .sh. |
| **-v** | Output results in verbose mode. (Log successful tests.) |
| **-d** | Print results to screen. (By default, results are sent to syslog.) |

Any test that is run returns a result in the following format:

```
STATUS: Message
```

- STATUS—PASS or FAIL
- Message—A short informative status message

For example, **FAIL: db "apricot" has only 1523 free blocks**

The following example shows how to run all available tests and print system output to the screen.

```
$ ./monitor.sh -d -aTest: 01free_db.sh. Status: PASS. Message: db apricot has 1532 free
blocks
Test: 02cm_is_up.sh. Status: FAIL. Message: cm process is not running
```

The following example shows how to run a single test to check that the installed database has sufficient free space.

```
$ ./monitor.sh -d free_dbTest: 01free_db.sh. Status: PASS. Message: db apricot has 1532
free blocks
```

## Tests

The following tests can be run using **monitor.sh** :

- **db_up** —Checks that the bundled Sybase database is running.
- **cm_up** —Checks that the CM application is running.
- **free_db** —Checks that the database has at least 10 percent free space.
- **free_log** —Check that the database transaction log has at least 70 percent free space.
- **cm_persistent_buffers** —Checks that each CM adapter´s persistent buffer contains less than 500 files.

The scripts for all of these tests are located in the **~/setup/monitor/tests** directory.

When calling a test called **test_name** , the script expects to find a file called **NNtest_name.sh** , where NN is a number that denotes the script´s overall priority. For example, the test **free_db** will be mapped to the file **01free_db.sh**.

## Managing Users

The CM uses the **p3rpc** utility to manage users for authenticated RPC calls.

The command format is: **p3rpc OPERATION [OPTIONS]**

The following table lists the **p3rpc**operations and options.

***Table 4-7 p3rpc Operations***

| Operation | Description |
|---|---|
| **--set-user--username**=*username***--password**=*password* | Adds and updates the username and password. |
| **--validate-password--username**=*username***--password**=*password* | Validates the username and password. |
| **--delete-user--username**=*username* | Delete a user configuration. |
| **--show-users** | Displays all configured users. |

## Managing Virtual Links

A script is included in the CM distribution to allow you to manage virtual link names and indices that are configured for a specific SCE.

To show or set virtual links, use the **update_vlinks.sh** script:

```
~scmscm/cm/bin/update_vlinks.sh--sce=SCE IP address[ --file=file| --show]
```

***Table 4-8 update_vlinks.sh Options***

| | |
|---|---|
| **--sce**=*SCE IP***--file**=*file* | Update the VLINK_INI table with the data in the supplied csv formatted file for the specified SCE. |
| **--sce**=*SCE IP***--show** | Query the VLINK_INI table for entries for the specified SCE. |
| **--help** | Display these options. |

To set the virtual link details, as the scmscm user, run the following command:

```
~scmscm/cm/bin/update_vlinks.sh--sce=SCE IP address--file=file
```

The CSV file format is: link id (positive integer), link direction (0=upstream, 1=downstream), name (string).

The following validation steps are performed on the file:

- The file exists
- There are no duplicate virtual links ids for each direction
- The virtual links id is a positive value from 0 to 1024.
- The direction is either 0 (upstream) or 1 (downstream)
- There are no duplicate virtual links names or empty names for each direction
- Virtual links names can contain up to 256 characters. All printable characters with an ASCII code between 32 and 126 (inclusive) can be used; except for 34 ("), 39 ('), and 96 (`).

After the file is successfully validated, the script performs the following actions:

1. All entries containing the SCE IP address in their SCE_IP field are deleted from the VLINK_INI table
2. Two entries will be added to the VLINK_INI table in the following format:
   – Timestamp, sce ip, 0, 0, "Default Virtual Link Up"

– Timestamp, sce ip, 0, 1, "Default Virtual Link Down"

3. The CSV file is parsed and each line in the CSV file will be entered as a line entry in the VLINK_INI table.

To show the virtual link details, as the scmscm user, run the following command:

**~scmscm/cm/bin/update_vlinks.sh--sce**=*SCE IP address***--show**

## How to Monitor the CM

You can use scripts to monitor system statistics that are relevant to the CM, such as:

- Percentage of free space in the database
- Rate of RDRs entering the CM
- SCE platform connection data

The following scripts are used to monitor the CM:

- `~scmscm/scripts/dbfree.sh`
- `~scmscm/scripts/rdr-rate.py`
- `~scmscm/scripts/sceconf.py`
- `~scmscm/setup/alive.sh`

The following scripts are used to configure the CM (see Configuring the CM ), but can also be invoked to display the relevant configuration:

- `~scmscm/setup/on-boot.py`
- `~scmscm/scripts/adapterconf.py`
- `~scmscm/scripts/dbconf.sh`

## Checking the Database Capacity

To display the percentage of free space in the database report tables and the associated transaction log, use the **dbfree.sh** script:

**~scmscm/scripts/dbfree.sh**
The script can be used only with a bundled database.

### DETAILED STEPS

**Step 1** As the scmscm user, run the **dbfree.sh**script

## Checking the RDR Rate

To display the momentary total rate of reports entering the CM, use the **rdr-rate.py** script

**~scmscm/scripts/rdr-rate.py**
The output is a single floating-point number representing the total rate per second of incoming RDRs (from all sources) that have entered the CM in the past 5 seconds.

This script can be used only if the HTTP Adaptor of the CM is running.

### DETAILED STEPS

**Step 1** As the scmscm user run the **rdr-rate.py**script

## Checking the SCE Connection

To display information about the SCE connections, use the **sceconf.py** script:

```
~scmscm/scripts/sceconf.py --op=list
```

This script can be used only if the HTTP Adaptor of the CM is running.

The script is also used to drop a connection from a particular SCE. See Dropping an SCE Connection.

### DETAILED STEPS

**Step 1** As the scmscm user, run the **sceconf.py**script

```
~scmscm/scripts/sceconf.py --op=list
```

### Example:

The following example shows SCE connection output:

```
>~scmscm/scripts/sceconf.py --op=listIP                     Rate                    Peak
-------                 -------                 -------
10.1.6.93               0.71798986              0.718
10.1.9.36               0.14420895              0.1442139
10.1.9.35               0.0                     0.027929332
10.1.12.11              0.0                     0.0
```

## Verifying that the Server is Operational

To verify that the Server is functioning correctly, use the **alive.sh** script:

```
~scmscm/setup/alive.sh
```

The script verifies that the following components are operational:

- Collection Manager
- Database (in the bundled database case)
- Report tables (in the bundled database case)

If any component is down, the script issues an error message.

### DETAILED STEPS

**Step 1** As the scmscm user, run the **alive.sh**script

**Note** It takes time for the components to initialize after a startup; after a restart, wait five minutes before running this script.

CHAPTER 5

# Managing Databases and the CSV Repository

This module explains how to use utility scripts to manage the Collection Manager database and the CSV repository.

Many of the database management tasks are applicable only to the bundled Sybase database.

**Note** For general instruction on using utility scripts, see Using Utility Scripts, page 4-1.

- Common Database Management Tasks
- Managing the Bundled Database
- Managing the CSV Repository

## Common Database Management Tasks

The database management tasks that are applicable to all of the supported databases are:

- Generating a list of the database tables
- Defining and applying the schedule for the periodic deletion of old records

Every record stored in the database is given a timestamp indicating the time that the Cisco Service Control Management Suite (SCMS) Collection Manager (CM) received the Raw Data Record (RDR). This timestamp is used when various maintenance operations are performed on the database tables.

The following scripts are used to configure and maintain the database:

- `~scmscm/scripts/dbtables.sh`
- `~scmscm/scripts/dbperiodic.py`
- `~scmscm/db_maint/create_periodic_del_procs.sh`

### Listing the Database Tables

To display a list all of the tables in the database, use the **dbtables.sh** script:

```
~scmscm/scripts/dbtables.sh
```

Where applicable, the number of lines in the table and the earliest and latest timestamps are displayed.

Actual content of the tables can be displayed using the Cisco Service Control Application (SCA) Reporter. For more information, see the *Cisco Service Control Application Reporter User Guide* .

The following table lists the **dbtables.sh** script options.

***Table 5-1 dbtables.sh Options***

| Option | Description |
|---|---|
| **-l** | Lists the existing table names only (without statistics) |
| **-a** | Include the non-report tables in the listing |
| **-f** | Enable fast line counting, use the client rather than JDBC<br>**Note** Applicable only for the bundled Sybase database. |
| **-t { sec_num }** | The maximal waiting time, in seconds, for the response. The default is no timeout. |
| **-h** | Prints this help message and exits. |

The following is a sample output from the **dbtables.sh** script:

```
>~scmscm/scripts/dbtables.shExecuting query ...
name| num_lines|                 min_time|                 max_time|
----------------+--------+------------------------+------------------------+
RPT_SUR|  131000|  2006-10-30 16:46:42.24| 2007-02-15 12:00:32.216|
RPT_LUR|  170000|  2007-04-10 15:25:45.31|  2007-04-11 07:06:05.45|
RPT_VLUR|    4694| 2007-04-11 13:12:39.683| 2007-04-11 13:18:07.396|
RPT_PUR|  116000| 2007-04-09 04:45:55.566| 2007-04-11 07:44:09.186|
RPT_TR|   57766| 2007-04-11 13:12:39.683| 2007-04-11 13:18:07.396|
RPT_MALUR|  109000| 2007-04-09 04:46:35.903|  2007-04-09 13:32:18.42|
RPT_MEDIA|  120000| 2007-04-05 17:14:24.443| 2007-04-11 13:16:29.436|
RPT_TOPS_PERIOD0|  194250|  2007-03-18 20:00:00.01|  2007-04-23 06:00:00.16|
RPT_TOPS_PERIOD1|   46940|  2007-03-19 00:00:00.05|   2007-04-23 00:00:00.1|
```

## Managing the Periodic Deletion of Old Records

In order to manage the periodic deletion of old records, it is necessary to perform the following general steps:

- Install the periodic delete procedures if they were not installed during the CM installation:

  Log on as the scmscm user, start the CM, wait 1-2 minutes for the database tables to be created, and then run the script:

  **~scmscm/db_maint/create_periodic_del_procs.sh.**
- Edit the periodic delete configuration file.
- Use the **dbperiodic.py** utility script to apply the new configuration.

Periodic deletion of a table does not begin while a previous periodic deletion is still running. This prevents excessive load on the database, which would degrade insertion performance in the adapters.

When two or more tables are scheduled to be reduced at the same time, the tables are processed in the order in which they are listed in the periodic delete configuration file.

For ease of configuration, you can schedule periodic deletion for all tables consecutively on one schedule.

**Note** All periodic delete activity is recorded in the system log file ( **/var/adm/messages** ).

**Note** Periodic delete when using a MySQL database is supported only on version 5.

- Configuring Periodic Delete
- Applying the Periodic Delete Configuration File

## Configuring Periodic Delete

The periodic delete configuration file ( **dbperiodic.conf** ) is, by default, located at **~scmscm/db_maint/**. The file has a structure similar to an INI file, where each section describes a particular data reduction operation for a specific set of tables, to be performed according to a specified schedule.

**Note** The name of each section of the file is not used when the file is parsed; use whatever names you wish.

Each section begins with the section name in brackets, and should contain the parameters shown in the following table. (Not all parameters are required in each section of the configuration file.) Separate the parameters and their values by an equal sign (=). Examples of periodic delete configuration files are given following the table.

***Table 5-2 Parameters in the Periodic Delete Configuration File***

| Parameter Name | Explanation | Values | Default | Example |
|---|---|---|---|---|
| active | Whether or not to use this section of the configuration file | true/false | true | false |
| tablenames | Names of the tables to which this section applies | Names of tables separated by commas, or * for all tables | * (all) | RPT_SUR,RPT_LUR |
| daystokeep | Number of days to keep records | Positive integers | 14 | 30 |
| minute<br>hour<br>day<br>month | When to perform the deletion in this section of the configuration file | 0 … 59, *<br>0 … 23, *<br>1 … 31, *<br>1 … 12, * | 0 *<br>(all) *<br>(all) *<br>(all) | 0<br>0,4,8,12,16,20<br>1<br>1,3,5,7,9,11 |

**Note** Values for all parameters except **active** and **daystokeep** can be either a single value, a list of values separated by commas, a range of values (two values separated by a dash), or an asterisk (*) which signifies all possible values. A range is not possible for **tablenames**.

In the following example, all fields are set to their default values.

```
# This dbperiodic.conf file emulates the legacy style for periodic
# deletion. All tables are processed every hour on the hour, and
# records are kept for 14 days.
[hourly all]
active = true
tablenames = *
daystokeep = 14
minute = 0
hour = *
```

In this example, all tables are reduced at 4:30 A.M., leaving 10 days of data in each table. In addition, the real-time tables are reduced every hour, leaving three days of data in each table.

```
# This dbperiodic.conf file reduces all tables once a day and
# real-time tables once an hour.
[daily all]
active = true
tablenames = *
daystokeep = 10
minute = 30
hour = 4
[hourly real-time]
active = true
tablenames = RPT_SUR,RPT_LUR,RPT_PUR
daystokeep = 3
minute = 0
hour = *
```

## Applying the Periodic Delete Configuration File

To load and apply a new periodic delete configuration file or to view the current file, use the **dbperiodic.py** script:

**~scmscm/scripts/dbperiodic.py**[**--dump**] [**--load**| **--loadfile**=*path_to_dbperiodic.conf*]

When the script is used to load a new configuration file, it parses the file, verifies its validity, and updates the scmscm user's crontab to reflect the changes.

*Table 5-3 dbperiodic.py Options*

| Option | Description |
|---|---|
| **--load** | Load the periodic delete configuration from **/export/home/scmscm/db_maint/ dbperiodic.conf** |
| **--loadfile**=*path to periodic delete configuration file* | Load the periodic delete configuration file from the specified directory |
| **--dump** | Print the periodic delete configuration |
| **--h** | Display these options |

The following example prints the current periodic delete configuration:

**~scmscm/scripts/dbperiodic.py --dump**

**Note** This script prints the *loaded* periodic delete configuration. If the current periodic delete configuration file was not yet loaded, the actual configuration may vary from the script´s output.

The following example loads the periodic delete configuration file from **~scmscm/db_maint/dbperiodic.conf** :

```
~scmscm/scripts/dbperiodic.py --load
```

The following example loads the periodic delete configuration file from a specified location:

```
~scmscm/scripts/dbperiodic.py --loadfile=path_to_periodic_delete_configuration_file
```

## Managing the Bundled Database

Managing the bundled database includes:

- Deleting a table
- Manually deleting old records from a table
- Backing up and restoring a database
- Updating Sybase with a changed IP address

Every record stored in the database is given a timestamp indicating the time that the Cisco Service Control Management Suite (SCMS) Collection Manager (CM) received the Raw Data Record (RDR). This timestamp is used when various maintenance operations are performed on the database tables.

The following scripts are used to maintain the bundled Sybase database only:

- `~scmscm/scripts/droptable.sh`
- `~scmscm/scripts/prunetable.sh`
- `~scmscm/scripts/sybback.sh`
- `~scmscm/scripts/sybrestore.sh`

### Deleting a Table

To delete a single table or all current tables from the database, use the **droptable.sh** script:

```
~scmscm/scripts/droptable.sh[-f] tableParameter
```

***Table 5-4 droptable.sh Options***

| Option | Description |
|---|---|
| *table_name* | Drop table_name from the database |
| ALLTABLES | Drop all tables from the database |
| **-f** | Drop by force (no questions asked or errors reported) |
| **-h** | Display these options |

The following example drops a table named RPT_SUR from the database with no request for confirmation:

```
~scmscm/scripts/droptable.sh –f RPT_SUR
```

The following example drops all tables from the database:

```
~scmscm/scripts/droptable.sh ALLTABLES
```

## Deleting Old Records

To remove records from a database table based on the timestamps of the records, use the **prunetable.sh** script:

```
~scmscm/scripts/prunetable.sh[-f] num_days table_name
```

*Table 5-5 prunetable.sh Options*

| Option | Description |
|---|---|
| *num_days* | The maximum age (in days) of records that will *not* be deleted. |
| *table_name* | The table whose records are to be deleted. |
| **-f** | Drop by force (no questions asked or errors reported). |
| **-h** | Display these options. |

The following example shows how to delete all records that are more than seven days old from a table named RPT_SUR.

Since the **–f**flag is not specified, there may be requests for confirmation and errors will be reported.

```
>~scmscm/scripts/prunetable.sh 7 RPT_SUR
```

## Backing Up the Database

To create text file backups of all the tables in the database, use the **sybback.sh** script:

```
~scmscm/scripts/sybback.sh -d path_to_backup_directory
```

The script converts all tables to ASCII files and copies the files to a backup directory.

*Table 5-6 sybback.sh Options*

| Option | Description |
|---|---|
| **-d** *path_to_backup_directory* | Write backup text files to the specified directory |
| **-h** | Display these options |

## Restoring a Database

To restore a database from the backup file that was created by the **sybback.sh** script, use the **sybrestore.sh** script:

```
~scmscm/scripts/sybrestore.sh -d path_to_restore_directory
```

*Table 5-7 sybrestore.sh Options*

| Option | Description |
|---|---|
| **-d** *path_to_restore_directory* | Restore the database using the text files in the specified directory |
| **-h** | Display these options |

**Note** The scripts **sybback.sh** and **sybrestore.sh** are not a viable backup mechanism for Sybase. They are designed for backing up and restoring small amounts of data; for example, transferring small tables between machines.

**Note** If you require a viable backup mechanism, please consult the *Sybase Backup Server* product documentation.

## Updating Sybase with a Changed IP Address

It is necessary to update the Sybase server when you change its IP addreass. As the root user run the following command:

```
~scmscm/setup/syb_interfaces.sh
```

## Managing the CSV Repository

You can use a utility script to manage the repository of CSV files output by the CM. These files are written to the disk by the Comma-Separated Value (CSV) Adapter for use by a service provider's operational support system (OSS) or by a third-party billing system. The size of the CSV repository should be monitored to prevent disk overflow.

**Note** If the backup parameter is set to **true**, failure to delete CSV files may result in disk overflow (No CSV files will ever be deleted.)

**Note** The third-party application is responsible for managing the CSV files and deleting them as necessary.

To successfully invoke this script, the HTTP Adaptor of the CM must be running. If the adapter is down, an error message is printed.

- CSV Repository File Structure
- Configuring the CSV File Repository
- Configuring the Comma Escape
- Configuring Escape of Nonprintable Characters

### CSV Repository File Structure

CSV files are stored in several subdirectories. Each subdirectory is given the number of a Raw Data Record (RDR) tag. (RDR tags define the type of the RDR.) Each RDR is stored in the subdirectory whose name matches its RDR tag number. For more information on RDR tags, see the *Cisco Service Control Application for Broadband Reference Guide* .

The CSV files are (automatically) sequentially numbered, with separate numbering in each directory. You can change the location of the parent directory by editing the **cm.conf** file located in the **cm/config** directory.

## Configuring the CSV File Repository

Use the **csvconf.sh** script, **~scmscm/scripts/csvconf.sh** , to:

- List the number of RDRs currently stored in the repository.
- Configure the maximum number of CSV files and the maximum permissible number of reports (lines) in each file.
- Control whether a backup is made whenever an old CSV file is about to be overwritten.
- Control whether each line in a CSV file contains an indication of the IP of the Service Control Engine (SCE) that sent this RDR. (By default, this option is off.)

**Note** Instead of using this script, you can edit the file **~scmscm/cm/config/csvadapter.conf**. Changes in this file require a CM restart to take effect.

**Note** The same configuration is applied to all subdirectories in the CSV Repository.

**Note** Setting these parameters does not change existing CSV files; it affects only files that are created subsequently.

***Table 5-8 csvconf.sh Options***

| Option | Description |
|---|---|
| `--list` | Display the CSV repository contents (the number of RDRs currently stored in the repository). |
| `--clear` | Delete all files from the CSV repository. (This option deletes all CSV files, but not the directories in which they are contained.) |
| **--maxlines**=*N* | Set the maximum number of RDRs per CSV file to *N*(an integer between 1 and 20,000). |
| **--maxfiles**=*M* | Set the maximum number of CSV files in each subdirectory to *M*(an integer between 10 and 10,000.) |
| **--backups**={**true**\|**false**} | Enable or disable backup of old CSV files. |
| **--recordsource**={**true**\|**false**} | Enable or disable the inclusion of the record source in CSV files. |

The following example shows how to set the maximum number of CSV files per subdirectory to 1000.

```
>~scmscm/scripts/csvconf.sh --maxfiles=1000
```

The following example shows how to set the maximum number records per CSV files to 10,000.

```
>~scmscm/scripts/csvconf.sh --maxlines=10000
```

The following example deletes all files from the CSV repository:

```
~scmscm/scripts/csvconf.sh --clear
```

The following example disables backing up of old CSV files in the repository:

```
~scmscm/scripts/csvconf.sh --backups=false
```

## Configuring the Comma Escape

When a comma is contained within a field in a CSV file, an escape sequence is used to indicate that the comma does not mark the end of the field.

Three escape methods are supported:

- Single quotation marks—Single quotation marks surround any field that contains one or more commas. There is no special treatment of single quotation marks already present in existing RDRs.
- URL—Each comma contained within a field is replaced by %2C. There is no special treatment of such sequences already present in existing RDRs.
- Backslash—Each comma contained within a field is preceded by a backslash (\). There is no special treatment of backslashes already present in existing RDRs.

The first two escape methods are compatible with Microsoft® Excel. The Backslash method is not compatible with Excel, but is retained for backward compatibility.

By default, single quotation marks are used. You can change the escape method by modifying the value of the **escapeMethod** attribute. This attribute is located in the **csvadapter.conf** file in the **CSVAdapter** directory. The value must be one of: **backslash** , **quote** , or **url**.

## Configuring Escape of Nonprintable Characters

Optionally, the CSV Adapter can escape nonprintable characters. Enabling this option incurs a performance penalty for the adapter; by default, the option is disabled.

When the option is enabled, each non-printable character, such as CR and LF, contained within a field is preceded by a backslash (\).

This option can be enabled in the **csvadapter.conf** file in the **CSVAdapter** directory. Changes in this file require a CM restart to take effect.

CHAPTER 6

# Database Configuration

This module describes how to configure the Cisco Service Control Management Suite (SCMS) Collection Manager (CM) to work with your database, and how to use the database infrastructure of the CM to extend its functionality.

## Quick Start Guide for Oracle Users

To use an Oracle database with the CM, you will have to change basic connection parameters such as the IP address and port where Oracle is deployed; no other configuration changes are necessary. The following steps describe the necessary changes:

1. If the CM is running, stop the CM.
2. Configure the JDBC Adapter to use Oracle:
   a. Open the file **~scmscm/cm/config/jdbcadapter.conf** in a text editor.
   b. Search for the string **db_template_dir**.

      There are two lines containing this string, one for Sybase and one for Oracle. By default, the *Oracle* line is commented out.
   c. Uncomment the Oracle line.
   d. Comment out the Sybase line.
   e. Save your changes.

This is illustrated in the following code fragment (after the changes have been made):

```
#db_template_dir = dbpacks/sybase
db_template_dir = dbpacks/oracle/9204e/
```

3. Configure the Topper/Aggregator (TA) Adapter to use Oracle:
   a. Open the file **~scmscm/cm/config/taadapter.conf** in a text editor.

b. Search for the string **db_template_dir**.

There are two lines containing this string, one for Sybase and one for Oracle. By default, the *Oracle* line is commented out.

c. Uncomment the Oracle line.

d. Comment out the Sybase line.

e. Save your changes.

This is illustrated in the following code fragment (after the changes have been made):

```
#db_template_dir = dbpacks/sybase
db_template_dir = dbpacks/oracle/9204e/
```

4. Configure your database connection parameters:

a. Open the file **~scmscm/cm/config/dbpacks/oracle/9204e/dbinfo.vm** in a text editor.

b. Change the following lines to reflect your setup:

```
#set ($dbinfo.options.host = "localhost")
#set ($dbinfo.options.port = "1521")
#set ($dbinfo.options.user = "pqb_admin")
#set ($dbinfo.options.password = "pqb_admin")
#set ($dbinfo.options.sid = "apricot")
```

**Note** The **dbinfo.vm** file is not a shell script; each pound sign (#) is part of the declaration and not a comment sign.

The relevant parameters are:

– The host name or IP address of the machine where Oracle is installed

– The port number on which the Oracle server is listening

– The user name and password for authentication against Oracle

– An existing Oracle SID to be used by the CM

c. Save your changes.

5. Start the CM.

## The Velocity Template Language

The JDBC Adapter framework uses macros written in the Velocity Template Language (VTL) to generate all SQL code that is passed to the database server. The following sections describe the configuration file used to control the generation process.

A full reference to VTL, which is part of the Apache Jakarta Project, can be found on the Web at http://jakarta.apache.org/velocity/vtl-reference-guide.html.

The following table briefly describes VTL constructs:

*Table 6-1 Summary of VTL Constructs*

| Directive | Syntax Example | Purpose |
|---|---|---|
| `#foreach` | `#foreach ($item in $collection)`<br>`item is $item`<br>`#end` | Iterates over a collection, array, or map. |
| `#if`<br>`...`<br>`#else`<br>`...`<br>`#elseif` | `#if ($order.total == 0)`<br>`No charge`<br>`#end` | Conditional statement. |
| `#parse` | `#parse("header.vm")` | Loads, parses, and includes the specified template in the generated output. |
| `#macro` | `#macro(currency $amount)`<br>`${formatter.currency($amount)}`<br>`#end` | Defines a new directive and any required parameters. The result is interpreted when used later in the template. |
| `#include` | `#include("disclaimer.txt")` | Includes the specified file, as is, in the generated output. |
| `#set` | `#set ($customer = ${order.customer})` | Assigns a value to a context object. If the context object does not exist, it is added; otherwise, it is overwritten. |
| `#stop` | `#if ($debug) #stop #end` | Stops template processing. |

## Database Configuration Files

When you initialize the Database access framework, the first file the Database access framework searches for is **main.vm** , which contains definitions or pointers to all the required database SQL definitions. The location used to search for this file depends on the dbpack used in the CM. A dbpack is a collection of configuration files pertaining to a specific database installation. The adapter (in accordance with its configuration file) selects the dbpack. The following code fragment from the **jdbcadapter.conf** file configures it to work with an Oracle dbpack:

```
db_template_dir = dbpacks/oracle/9204e/
db_template_file = main.vm
```

**Note** The directory location is interpreted relative to the main CM configuration directory (usually **~scmscm/cm/config** ).

To make the configuration more modular, the **main.vm** file generally points to other files; however this is not strictly necessary. The files can contain arbitrary definitions that can later be used, for example, in scripts. Some definitions are mandatory because the JDBC Adapter uses them for its operation. These definitions are listed in the following table:

*Table 6-2 Mandatory VM Definitions*

| Object Name | Mandatory Definition |
|---|---|
| $table.sql.dropTable<br>$table.sql.createTable<br>$table.sql.createIndexes<br>$table.sql.insert<br>$table.sql.metaDataQuery | For each table, these settings control how SQL is generated for the indicated operation |
| $dbinfo.driverjarfile<br>$dbinfo.driver | Location and class name for JDBC driver |
| $dbinfo.cmdSeparator | Pattern used to separate multiple SQL statements |
| $dbinfo.url<br>$dbinfo.connOptions | URL for connecting to the database, and various connection properties |

Some objects representing the CM configuration in the VTL parsing context are available to be used in the templates. These objects are described in the following sections.

- Context Objects
- Application Configuration

## Context Objects

Before the VM templates are loaded and parsed by any CM components (for instance, a TA or JDBC Adapter, or a script), the parsing context is initialized with the following Java objects:

- The tables Object
- The dbinfo Object
- The tools Object

### The tables Object

The **tables** object describes application-related database configuration, such as the structure of RDRs that should be stored in the database, the structure of the database tables and where they are stored, and the structure of any other database tables that the CM might use. The object is an array in which each row represents one of the database tables used by the CM. For each table, the row may contain the following information (not all items are relevant to all tables):

- Logical name
- Physical name
- RDR tag associated with this table
- List of fields/columns in this table, with the following attributes for each:
  - Field ID
  - Field name
  - Field native type

- Free-form field options
- List of indexes for this table, with the following attributes for each:
  - Index name
  - Names of columns indexed
  - Free-form index options

The contents of the **tables** object can be inspected or manipulated when loading the templates. The **tables** object is initialized using the application-specific XML configuration file. See Application Configuration.

## The dbinfo Object

The **dbinfo** object describes configuration that is specific to the database, such as the parameters and the SID or schema to be used when opening a database connection. The object holds database-specific configuration options. It contains the following information:

- The JDBC class name to be used as a driver for this database
- The name of the JAR file containing the driver
- The location of the database expressed as a JDBC URL
- Free-form JDBC connection options, such as authentication data (user and password)

## The tools Object

The **tools** object is a container for several utility methods that you might find useful when developing templates or manipulating the context data structures.

You invoke the object's methods by using **$tools.method(arg1, ..., argN)** , where **method** is the name of the method.

The included methods are listed in the following table:

***Table 6-3 A Summary of Methods of the tools Object***

| Method Name and Arguments | Functionality |
|---|---|
| `getTableByName (allTables, name)` | Locates the database table object whose logical name corresponds to name. |
| `getTableByDbTabName (allTables, name)` | Locates the database table object whose physical name corresponds to **name**. |

*Table 6-3 A Summary of Methods of the tools Object*

| Method Name and Arguments | Functionality |
|---|---|
| `assignParams (sql, list_of_args)` | Replaces question mark characters in the sql string with consecutive elements from the **list_of_args** parameter, represented as a String. This method is useful if working with templates that create SQL insert statements using the JDBC PreparedStatement string as a base. |
| `collapseWhitespace()` | Converts all instances of more than one consecutive white-space characters to one space, and trims beginning and ending white space. This method may be useful for databases that require SQL with a minimum of newline and other white-space characters. (Sybase and Oracle do not require this.) |

For a sample that demonstrates how to use these tools, see Using the JDBC Framework in Scripts.

## Application Configuration

All application-related configuration is done in one file ( **tables.xml** ) that includes the following items:

- Name and version of the application
- Name and properties of each database table, and specifically the structure of application RDRs that are to be stored in database tables
- For each database table:
  - Names and native types of the table/RDR fields
  - Names and properties of the table indexes

This information is used primarily to populate the **tables** object in the template parsing context. See The tables Object.

## A Working Sample

The **main.vm** file can contain references to other VM files to support modularization (see Database Configuration Files ). The names of these other files are arbitrary, except for the **VM_global_library.vm** file whose name is predetermined. Any macros that need to be defined should be put in this file, to ensure that they are loaded at the right time. For more details about this special file, see the *Velocity User Guide* .

The following sample illustrates the contents of main.vm for an Oracle setup:

```
#parse ('dbinfo.vm')
#foreach ($table in $tables)
#set ($table.sql.dropTable = "#parse ('drop_table.vm')")
#set ($table.sql.createTable = "#parse ('create_table.vm')")
#set ($table.sql.createIndexes = "#parse ('create_indexes.vm')")
#set ($table.sql.insert = "#parse ('insert.vm')")
#set ($table.sql.metaDataQuery = "#parse ('metadata.vm')")
#end
```

In this sample, the mandatory database and SQL definitions (see Table 6-2) are moved to separate files, to be loaded and parsed using the **#parse** directive.

The following sections list possible contents for the various files in the Oracle dbpack. Some of the definitions use macros that are defined in the **VM_global_library.vm** file. This file should contain all macro definitions used by any template.

- Macro Definitions
- dbinfo Configuration
- SQL Definitions

## Macro Definitions

The following sample illustrates definitions for the mapping between native types and SQL types, and utility macros such as the **optcomma** macro, which inserts a comma between successive elements of lists.

```
#macro (optcomma)#if ($velocityCount >1),#end#end
#macro (sqltype $field)
#set ($maxStringLen = 2000)
#if     ($field.type == "INT8") integer
#elseif ($field.type == "INT16") integer
#elseif ($field.type == "INT32") integer
#elseif ($field.type == "UINT8") integer
#elseif ($field.type == "UINT16") integer
#elseif ($field.type == "UINT32") integer
#elseif ($field.type == "REAL") real
#elseif ($field.type == "BOOLEAN") char(1)
#elseif ($field.type == "STRING") varchar2(#if($field.size <= $maxStringLen)$field.size
#else $maxStringLen #end)
#elseif ($field.type == "TEXT") long
#elseif ($field.type == "TIMESTAMP") date
#end
#end
```

## dbinfo Configuration

In the following code sample, note that the only required fields are the URL and connection options (for authentication).

Blank lines in the code separate the code into distinct fields for readability and to ease later configuration changes.

```
#set ($dbinfo.driver = "oracle.jdbc.OracleDriver")
#set ($dbinfo.driverjarfile = "ojdbc14.jar")
#set ($dbinfo.options.host = "localhost")
#set ($dbinfo.options.port = "1521")
#set ($dbinfo.options.user = "pqb_admin")
#set ($dbinfo.options.password = "pqb_admin")
#set ($dbinfo.options.sid = "apricot")
#set ($dbinfo.url =
"jdbc:oracle:thin:@$dbinfo.options.host:$dbinfo.options.port:$dbinfo.options.sid")
#set ($dbinfo.connOptions.user = $dbinfo.options.user)
#set ($dbinfo.connOptions.password = $dbinfo.options.password)
## the vendor-specific piece of SQL that will return the current
## date and time:
#set ($dbinfo.options.getdate = "sysdate")
```

## SQL Definitions

### Code for "drop table"

The following code sample drops a table using normal SQL syntax

```
drop table $table.dbtabname
```

### Code for "create table"

The following code sample creates a table using normal SQL syntax. Any customized database configuration that requires special directives for table creation can be implemented using this definition. For example, you can modify it to create the table in some unique tablespace or to use table partitioning.

```
create table $table.dbtabname (
#foreach ($field in $table.fields)
#optcomma()$field.name #sqltype($field)
#if ("$!field.options.notnull" == "true")
not null
#end
#end)
```

### Code for "create indexes"

The following code creates the indexes using normal SQL syntax. Any customized database configuration requiring special directives for index creation can be implemented using this definition. For example, you can modify it to create the indexes in some unique tablespace.

```
#foreach ($index in $table.indexes)
create index $index.name on $table.dbtabname ($index.columns)
#end
```

### Code for "insert"

The following code creates the JDBC PreparedStatement corresponding to the table structure.

```
insert into ${table.dbtabname} (
#foreach ($field in $table.fields)
#optcomma()${field.name}
#end)
values (
#foreach ($field in $table.fields)
#optcomma()?
#end)
```

### Code for metadata query

The following code defines a simple query that is used to get the table metadata (column names and types). Any query that returns an empty result set can be used.

```
select * from ${table.dbtabname} where 1=0
```

## Testing and Debugging

While you develop a set of templates for your database, it is useful to be able to see the results of parsing directly. To enable, this, the JDBC Adapter supports direct invocation via the CM main script **~scmscm/cm/bin/cm**.

The general syntax for such an invocation is:

```
~/cm/bin/cm invoke com.cisco.scmscm.adapters.jdbc.JDBCAdapter argument
```

where **argument** is one of the flags described in the following sections. You can use this mechanism whether or not the CM is running.

Additionally, the query and update execution methods described in the following section can be used to test the template results against a live database.

- Parsing a String
- Obtaining Full Debug Information

### Parsing a String

Any string can be parsed as a VTL template with the complete context in place. The result of the parsing is displayed on the standard output. To parse a string, call the adapter with the -parse flag. Here are a few examples (the responses are shown in **bold** ):

```
$ ~/cm/bin/cm invoke com.cisco.scmscm.adapters.jdbc.JDBCAdapter -parse 'xxx'

xxx

$ ~/cm/bin/cm invoke com.cisco.scmscm.adapters.jdbc.JDBCAdapter -parse '$dbinfo.url'

jdbc:oracle:thin:@localhost:1521:apricot

$ ~/cm/bin/cm invoke com.cisco.scmscm.adapters.jdbc.JDBCAdapter -parse
'$tools.getTableByName($tables, "LUR").sql.createTable'

create table RPT_LUR (
 TIME_STAMP   date
   ,RECORD_SOURCE   integer
   ,LINK_ID   integer
   ,GENERATOR_ID   integer
   ,SERVICE_ID   integer
   ,CONFIGURED_DURATION   integer
   ,DURATION   integer
   ,END_TIME   integer
   ,UPSTREAM_VOLUME   integer
   ,DOWNSTREAM_VOLUME   integer
   ,SESSIONS   integer
  )
```

## Obtaining Full Debug Information

To see a dump of all the contents of the **tables** and **dbinfo** structures as created by the templates, use the **-debug** flag. When this flag is used, a very detailed view of all the fields, properties, and options of these structures is printed to standard output.

## Using the JDBC Framework in Scripts

You can send arbitrary SQL commands to the database for execution and view the resulting data. This may be useful for periodic database maintenance, monitoring the contents of database tables, managing extra database tables, or any other purpose.

To perform an **update** operation, call the adapter with the **-executeUpdate** flag. To perform a query and view the results, call the adapter with the **-executeQuery** flag.

- Sample - Viewing and Setting the SCE Time Zone Offset

## Sample - Viewing and Setting the SCE Time Zone Offset

The following sample of an update operation demonstrates how to programmatically change the value in the database table holding the Service Control Engine (SCE) time zone offset setting. The name of this table is usually **JCONF_SE_TZ_OFFSET** ; since the table may be assigned another name, it is referred to here by its logical name TZ. See the listing in The tables.xml File, page A-1.

To avoid the need to first check that the table exists and then update it, the table is dropped (ignoring the error status if it does not exist) and then recreated, and the proper values are inserted. Since the table contains a timestamp column, you must get the current date from the database. This operation is specific to each database vendor; therefore this example uses the preconfigured **getdate** operation that has been defined in the templates.

Note the use of the tools **assignParams** and **getTableByName** to generate the SQL.

```
#! /bin/bash
this=$0
tableName=TZ
usage () {
cat <<EOF
Usage:
$this --status      - show currently configured TZ offset
$this --offset=N    - set the offset to N minutes (-1440 <= N <= 1440)
$this --help        - print this message
EOF
}
query () {
~/cm/bin/cm invoke com.cisco.scmscm.adapters.jdbc.JDBCAdapter -executeQuery "$*"
}
update () {
~/cm/bin/cm invoke com.cisco.scmscm.adapters.jdbc.JDBCAdapter -executeUpdate "$*"
}
get_tz () {
query 'select * from $tools.getTableByName($tables, "TZ").dbtabname'
}
set_tz () {
update '$tools.getTableByName($tables, "TZ").sql.dropTable'
update '$tools.getTableByName($tables, "TZ").sql.createTable'
update '$tools.assignParams($tools.getTableByName($tables, "TZ").sql.insert,
[$dbinfo.options.getdate, '$1'])'
}
case $1 in
--status)
get_tz
;;
--help)
usage
exit 0
;;
--offset=*)
n=$(echo $1 | egrep 'offset=[-]?[0-9]+$' | sed 's/.*=//')
if [ "$n" ]; then
if [ "$n" -ge -1440 -a "$n" -le 1440 ]; then
set_tz $n &>/dev/null
ok=1
fi
fi
if [ ! "$ok" ]; then
usage
exit 2
fi
get_tz
;;
*)
usage
exit 3
;;
esac
```

When a result set is returned by an executed query, it is displayed to standard output in tabular form with appropriate column headers.

## Scalability - Hints for Oracle

The following two sections demonstrate ways to make your database handling more scalable for the CM. These are specific to Oracle, and are provided merely as hints to illustrate the possibilities.

- Using Custom tablespaces
- Using Table Partitioning

### Using Custom tablespaces

Suppose you have created several tablespaces and wish to distribute the CM tables among them. An easy way to do this is to specifythe tablespace to be used for each table in the file **tables.xml**. For one table, the definition looks like this (note especially the code in **bold** ):

```
<rdr name="LUR" dbtabname="RPT_LUR" tag="4042321925" createtable="true">
<options >
<option property="tablespace" value="tspace1" />
</options >
<fields>
<field id="1" name="TIME_STAMP" type="TIMESTAMP">
<!-- (other field declarations) -->
<field id="10" name="DOWNSTREAM_VOLUME" type="UINT32"/>
<field id="11" name="SESSIONS" type="UINT32"/>
</fields>
<indexes>
<index name="RPT_LUR_I1" columns="END_TIME">
<options>
```

```
<option property="clustered" value="true"/>
<option property="allowduprow" value="true"/>
<option property="tablespace" value="tspace2" />
</options>
</index>
</indexes>
</rdr>
```

This sample adds the required tablespaces ( **tspace1** and **tspace2** ) for the index and for the table. There is no preconfigured meaning to the option **tablespace** in the CM; any new option name could have been used. Its meaning is derived from its subsequent use in the templates.

To create the table in the correct tablespace, modify **create_table.vm** as follows:

```
create table $table.dbtabname (
#foreach ($field in $table.fields)
#optcomma()$field.name #sqltype($field)
#if ("$!field.options.notnull" == "true")
not null
#end
#end)#if ("$!table.options.tablespace" != "") TABLESPACE $table.options.tablespace #end
```

To create the index in its own tablespace, modify **create_indexes.vm** as follows:

```
#foreach ($index in $table.indexes)
create index $index.name on $table.dbtabname ($index.columns)
#if ("$!index .options.tablespace" != "") TABLESPACE $index.options.tablespace #end #end
```

## Using Table Partitioning

To implement rolling partitioning for a particular table on a weekly basis, you can create a partitioned option for the table in the **tables.xml** file in a similar manner to the example in the previous section ( Using Custom tablespaces ). Then augment the **create_table.vm** code as follows (note especially the code in **bold** ):

```
create table $table.dbtabname (
#foreach ($field in $table.fields)
#optcomma()$field.name #sqltype($field)
#if ("$!field.options.notnull" == "true")
not null
#end
#end)#if ("$!table.options.partitioned" != "") partition by range (timestamp) (partition
week_1 values less than (to_date ('01-JAN-2005 00:00:00','DD-MON-YYYY HH24:MI:SS')),
partition week_2 values less than (to_date ('08-JAN-2005 00:00:00','DD-MON-YYYY
HH24:MI:SS')) partition week_3 values less than (to_date ('15-JAN-2005
00:00:00','DD-MON-YYYY HH24:MI:SS')) partition week_4 values less than (to_date
('22-JAN-2005 00:00:00','DD-MON-YYYY HH24:MI:SS')) ); #end
```

Since Oracle does not accept nonconstant expression for the time boundaries, the values must be hardwired for the time the tables are created.

Create a cron job to roll the partitions (delete an old partition and create a new one) on a weekly basis. This cron job runs a script that calls the command-line interface of the JDBC Adapter (as explained in Using the JDBC Framework in Scripts ) to issue the appropriate **alter table drop partition** and **alter table add partition** SQL commands.

APPENDIX A

# Code Samples

This appendix contains samples of files used to configure the Cisco Service Control Management Suite (SCMS) Collection Manager (CM) and the adapters that process the data that the CM receives.

- Application Configuration
- Adapter Configuration

## Application Configuration

The following sections list part of the XML file ( **tables.xml** ) used to configure the database tables, and the DTD file used to verify the structure of the XML file.

- The tables.xml File
- The tables.dtd File

### The tables.xml File

The following is a listing of a portion of the Cisco Service Control Application for Broadband **tables.xml** file:

```
<?xml version="1.0" encoding="ISO8859_1"?>
<!DOCTYPE dbtabconf PUBLIC "-//P-Cube//Engage DB RDR Configuration 2.1.0//EN"
"dbtables.dtd">
<dbtabconf>
<fileversion>
...
</fileversion>
<application name="Engage" version="2.1"/>
<dbtables>
<rdr name="SUR" dbtabname="RPT_SUR" tag="4042321922" createtable="true">
<fields>
<field id="1" name="TIME_STAMP" type="TIMESTAMP">
<options>
<option property="source" value="timestamp"/>
</options>
</field>
<field id="2" name="RECORD_SOURCE" type="INT32">
<options>
<option property="source" value="recordsource"/>
</options>
</field>
<field id="3" name="SUBSCRIBER_ID" type="STRING" size="64"/>
```

```
<field id="4" name="PACKAGE_ID" type="INT32"/>
<field id="5" name="SERVICE_ID" type="INT32">
<options>
<option property="notnull" value="true"/>
</options>
</field>
<field id="6" name="MONITORED_OBJECT_ID" type="INT32"/>
<field id="7" name="BREACH_STATE" type="INT32"/>
<field id="8" name="REASON" type="INT32"/>
<field id="9" name="CONFIGURED_DURATION" type="INT32"/>
<field id="10" name="DURATION" type="INT32"/>
<field id="11" name="END_TIME" type="INT32"/>
<field id="12" name="UPSTREAM_VOLUME" type="UINT32"/>
<field id="13" name="DOWNSTREAM_VOLUME" type="UINT32"/>
<field id="14" name="SESSIONS" type="UINT32"/>
</fields>
<indexes>
<index name="RPT_SUR_I1" columns="END_TIME">
<options>
<option property="clustered" value="true"/>
</options>
</index>
</indexes>
</rdr>
<rdr name="LUR" dbtabname="RPT_LUR" tag="4042321925" createtable="true">
<fields>
<field id="1" name="TIME_STAMP" type="TIMESTAMP">
<options>
<option property="source" value="timestamp"/>
</options>
</field>
<field id="2" name="RECORD_SOURCE" type="INT32">
<options>
<option property="source" value="recordsource"/>
</options>
</field>
<field id="3" name="LINK_ID" type="INT32"/>
<field id="4" name="GENERATOR_ID" type="INT32"/>
<field id="5" name="SERVICE_ID" type="INT32"/>
<field id="6" name="CONFIGURED_DURATION" type="INT32"/>
<field id="7" name="DURATION" type="INT32"/>
<field id="8" name="END_TIME" type="INT32"/>
<field id="9" name="UPSTREAM_VOLUME" type="UINT32"/>
<field id="10" name="DOWNSTREAM_VOLUME" type="UINT32"/>
<field id="11" name="SESSIONS" type="UINT32"/>
</fields>
<indexes>
<index name="RPT_LUR_I1" columns="END_TIME">
<options>
<option property="clustered" value="true"/>
<option property="allowduprow" value="true"/>
</options>
</index>
</indexes>
</rdr>
<aggtable name="TOP_HOURLY" dbtabname="RPT_TOPS_PERIOD0" aggperiod="0">
<fields>
<field id="1" name="RECORD_SOURCE" type="INT32"/>
<field id="2" name="METRIC_ID" type="INT8"/>
<field id="3" name="SERVICE_ID" type="INT8"/>
<field id="4" name="TIME_STAMP" type="TIMESTAMP"/>
<field id="5" name="AGG_PERIOD" type="INT8"/>
<field id="6" name="SUBSCRIBER_ID" type="STRING" size="64"/>
<field id="7" name="CONSUMPTION" type="UINT32"/>
```

```
</fields>
<indexes>
<index name="RPT_TOPS_PERIOD0_I1" columns="TIME_STAMP">
<options>
<option property="clustered" value="true"/>
<option property="allowduprow" value="true"/>
</options>
</index>
</indexes>
</aggtable>
<table name="TZ" dbtabname="JCONF_SE_TZ_OFFSET">
<fields>
<field id="1" name="TIME_STAMP" type="TIMESTAMP"/>
<field id="2" name="OFFSET_MIN" type="INT16"/>
</fields>
</table>
</dbtables>
</dbtabconf>
```

For each table (either an RDR table, an aggregation table, or an extra table), the fields, indexes, and so forth are listed.

**Note** A table, an index, or fields can have arbitrary free text options that can be accessed in the templates.

The XML file is verified at runtime against a simple DTD, reproduced in the following section.

## The tables.dtd File

The following is a listing of the DTD file used to verify the **tables.xml** definition file:

```
<?xml version="1.0" encoding="ISO8859_1"?>
<!ELEMENT dbtabconf (fileversion, application, db?, dbtables)>
<!ELEMENT fileversion (#PCDATA)>
<!ELEMENT application EMPTY>
<!ATTLIST application
name CDATA #REQUIRED
version CDATA #REQUIRED
>
<!ELEMENT db (options)>
<!ELEMENT dbtables (rdr*, aggtable*, table*)>
<!ELEMENT table (options?, fields, indexes?)>
<!ATTLIST table
name CDATA #REQUIRED
dbtabname CDATA #REQUIRED
createtable (true | false) "true"
inserttodb (true | false) "false"
>
<!ELEMENT aggtable (options?, fields, indexes?)>
<!ATTLIST aggtable
name CDATA #REQUIRED
dbtabname CDATA #REQUIRED
aggperiod CDATA #REQUIRED
createtable (true | false) "true"
>
<!ELEMENT rdr (options?, fields, indexes?)>
<!ATTLIST rdr
name CDATA #REQUIRED
dbtabname CDATA #REQUIRED
tag CDATA #REQUIRED
createtable (true | false) "true"
```

```
inserttodb (true | false) "true"
>
<!ELEMENT fields (field+)>
<!ELEMENT field (options?)>
<!-- the id attribute below is presumably a numeric index, but it is for future
use, we currently don't look at it, as the order is imposed in the XML -->
<!ATTLIST field
id CDATA #REQUIRED
name CDATA #REQUIRED
type CDATA #REQUIRED
size CDATA #IMPLIED
>
<!ELEMENT indexes (index+)>
<!ELEMENT index (options?)>
<!ATTLIST index
name CDATA #REQUIRED
columns CDATA #REQUIRED
create (true | false) "true"
>
<!ELEMENT options (option+)>
<!ELEMENT option EMPTY>
<!ATTLIST option
property CDATA #REQUIRED
value CDATA #REQUIRED
>
```

The location and name of the DTD and XML files can be set separately for each adapter in the adapter's configuration file.

## Adapter Configuration

The following sections list the configuration file ( **ragadapter.conf** ) and the associated XML file ( **ragadapter.xml** ) used to configure the Real-Time Aggregating (RAG) Adapter.

The configuration files of the other adapters are similar to the RAG Adapter configuration file. Only the RAG Adapter has an associated XML file.

- The ragadapter.conf File
- The ragadapter.xml File

### The ragadapter.conf File

RAG Adapter general maintenance is performed using the file **~scmscm/cm/config/ragadapter.conf**. The following is a sample of the RAG Adapter configuration file:

```
#
# RAGAdapter main configuration file
#
[config]
xml_filename = ~/cm/config/ragadapter.xml
[housekeeper]
interval_sec = 10
[db]
operations_timeout = 60
batch_size = 10
transaction_size = 15
commit_interval = 6
blocking_connects = true
db_template_file = main.vm
```

```
db_template_dir = dbpacks/sybase/ase12.5.1
[app]
app_conf_file = dbtables.xml
app_dtd_file = dbtables.dtd
app_conf_dir = apps/scasbb/3.1.0
```

## The ragadapter.xml File

The following code is a sample of the configuration possibilities of the RAG Adapter.

```
<?xml version="1.0"?>
<!DOCTYPE ragadapterconf [
<!ELEMENT ragadapterconf (fileversion, config)>
<!ELEMENT fileversion (#PCDATA)>
<!ELEMENT config (aggregations, sinks)>
<!ELEMENT aggregations (aggregation+)>
<!ELEMENT aggregation (bucketident, closures, accumulators, monitors)>
<!ATTLIST aggregation
id CDATA #REQUIRED
intag CDATA #REQUIRED
outtag CDATA #REQUIRED
sinkid CDATA #REQUIRED
>
<!ELEMENT bucketident (field+)>
<!ELEMENT closures (closure*)>
<!ELEMENT closure (closurespec+)>
<!ATTLIST closure
field CDATA #REQUIRED
>
<!ELEMENT closurespec (equivvalue+)>
<!ATTLIST closurespec
type (string | int | long | double) #REQUIRED
primaryvalue CDATA #REQUIRED
>
<!ELEMENT equivvalue EMPTY>
<!ATTLIST equivvalue
val CDATA #REQUIRED
>
<!ELEMENT accumulators (field+)>
<!ELEMENT monitors (changemonitor | maxmonitor | timeoutmonitor)*>
<!ELEMENT changemonitor EMPTY>
<!ATTLIST changemonitor
action (warn | checkpoint) #REQUIRED
field CDATA #REQUIRED
active (true | false) #REQUIRED
>
<!ELEMENT maxmonitor EMPTY>
<!ATTLIST maxmonitor
action (warn | checkpoint) #REQUIRED
field CDATA #REQUIRED
maxvalue CDATA #REQUIRED
active (true | false) #REQUIRED
>
<!ELEMENT timeoutmonitor EMPTY>
<!ATTLIST timeoutmonitor
action (warn | checkpoint) #REQUIRED
maxsec CDATA #REQUIRED
active (true | false) #REQUIRED
>
<!ELEMENT field EMPTY>
<!ATTLIST field
index CDATA #REQUIRED
type (string | int | long | double) #REQUIRED
>
<!ELEMENT sinks (csvsink | dbsink | generalsink)+>
<!ELEMENT csvsink EMPTY>
<!ATTLIST csvsink
id CDATA #REQUIRED
classname CDATA #REQUIRED
filenameformat CDATA #REQUIRED
dirname CDATA #REQUIRED
maxagesec CDATA #REQUIRED
maxlines CDATA #REQUIRED
usequotes (true | false) #REQUIRED
active (true | false) #REQUIRED
>
<!ELEMENT dbsink EMPTY>
<!ATTLIST dbsink
id CDATA #REQUIRED
classname CDATA #REQUIRED
active (true | false) #REQUIRED
>
<!ELEMENT generalsink EMPTY>
<!ATTLIST generalsink
id CDATA #REQUIRED
classname CDATA #REQUIRED
active (true | false) #REQUIRED
>
]>
<ragadapterconf>
<fileversion>
$File: ragadapter.xml $ $Revision: #3 $
$Author: ronv $
$DateTime: 2005/08/15 15:48:23 $
</fileversion>
<config>
<aggregations>
<aggregation id="NUR's by subscriber and subs usage counter"
intag="4042321920" outtag="71070" sinkid="csv1">
<bucketident>
<!-- SUBSCRIBER_ID=0, SUBS_USG_CNT_ID=2 -->
<field index="0" type="string"/>
<field index="2" type="int"/>
</bucketident>
<closures>
<closure field="0">
<closurespec type="string" primaryvalue="GuyM">
<equivvalue val="RonK"/>
<equivvalue val="OmerT"/>
<equivvalue val="GuyM"/>
</closurespec>
<closurespec type="string" primaryvalue="OdedE">
<equivvalue val="NimrodR"/>
<equivvalue val="YossiO"/>
<equivvalue val="LironL"/>
</closurespec>
</closure>
<closure field="2">
<closurespec type="int" primaryvalue="15">
<equivvalue val="5"/>
<equivvalue val="6"/>
<equivvalue val="7"/>
</closurespec>
</closure>
</closures>
<accumulators>
```

```
<!-- up=8, down=9, sessions=10 -->
<field index="8" type="long"/>
<field index="9" type="long"/>
<field index="10" type="long"/>
</accumulators>
<!-- nothing to monitor for change in NUR really.
For sake of testing, let's warn if DURATION changes. -->
<monitors>
<maxmonitor action="checkpoint" field="8" maxvalue="10000" active="true"/>
<maxmonitor action="checkpoint" field="9" maxvalue="10000" active="true"/>
<changemonitor action="warn" field="6" active="true"/>
<timeoutmonitor action="checkpoint" maxsec="60" active="true"/>
</monitors>
</aggregation>
<aggregation id="NUR's by subscriber only"
intag="4042321920" outtag="71071" sinkid="dbsink1">
<bucketident>
<field index="0" type="string"/>
</bucketident>
<closures/>
<accumulators>
<field index="8" type="long"/>
<field index="9" type="long"/>
<field index="10" type="long"/>
</accumulators>
<monitors>
<timeoutmonitor action="checkpoint" maxsec="60" active="true"/>
</monitors>
</aggregation>
</aggregations>
<sinks>
<csvsink id="csv1"
classname="com.cisco.scmscm.adapters.rag.sinks.CSVSink"
filenameformat="yyyy-MM-dd_HH-mm-ss-SSS'.csv'"
dirname="~/cm/adapters/RAGAdapter/csvfiles"
maxagesec="300" maxlines="1000" usequotes="true" active="true"/>
<dbsink id="dbsink1"
classname="com.cisco.scmscm.adapters.rag.sinks.JDBCSink" active="false"/>
</sinks>
</config>
</ragadapterconf>
```