Cisco Service Control Management Suite
Collection Manager User Guide

Release 3.5.0
February 15, 2011

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

Text Part Number: OL-7208-09
# CONTENTS

**About this Guide** vii  
Document Revision History vii  
Organization xi  
Related Publications xi  
Conventions xi  
Obtaining Documentation and Submitting a Service Request xii

## CHAPTER 1

**Cisco Service Control Overview** 1-1  
 Cisco Service Control Solution 1-1  
 Service Control for Broadband Service Providers 1-1  
 Cisco Service Control Capabilities 1-2  
 SCE Platform Description 1-3  
 Management and Collection 1-4  
 Network Management 1-5  
 Subscriber Management 1-5  
 Service Configuration Management 1-6  
 Data Collection 1-6

## CHAPTER 2

**Collection Manager Overview** 2-1  
 Data Collection Process 2-1  
 Raw Data Records 2-1  
 Collection Manager Software Package 2-2  
 Raw Data Record Server 2-3  
 Categorizer 2-3  
 Priority Queues and Persistent Buffers 2-3  
 Adapter Description 2-3  
 JDBC Adapter 2-3  
 Comma Separated Value Adapter 2-4  
 Topper/Aggregator Adapter 2-4  
 TA Adapter Cycles 2-4  
 TA Adapter Memory Requirements 2-5  
 Real-Time Aggregating Adapter 2-5  
 RAG Adapter Aggregation Buckets 2-6  
 Flushing a Bucket 2-6
Using Databases 2-6
   Using the Bundled Database 2-7
   Using an External Database 2-7

CHAPTER 3 Installing the Collection Manager 3-1
   System Requirements 3-1
      Checking System Prerequisites 3-2
      Solaris Requirements 3-2
         Hardware 3-2
         Software and Environment 3-3
         Setting the Locale and Time Zone 3-4
      Red Hat Linux Requirements 3-5
         Hardware 3-5
         Software and Environment 3-5
         Setting the Locale and Time Zone 3-6
   Distribution Content 3-6
   Default Configuration Settings 3-6
   How to Install the Collection Manager 3-6
      Ports Used by the Collection Manager Software 3-7
      Installing the Sybase Database 3-7
         Actions Performed by installsyb.sh 3-8
         Prerequisites 3-8
      Installing Collection Manager Software 3-9
         Information About the install-cm.sh Script 3-9
   How to Uninstall the Sybase Database and Collection Manager Software 3-11
      Uninstalling Sybase 3-11
      Uninstalling the Collection Manager Software 3-12
   Upgrading the Collection Manager to Version 3.5.x 3-12

CHAPTER 4 Managing the Collection Manager 4-1
   Using Utility Scripts 4-1
   Collection Manager Support Information 4-2
   Configuring the Collection Manager 4-2
      Activating Servers 4-3
      Controlling Adapters 4-3
      Enabling Adapters 4-4
      Dropping an SCE Connection 4-4
      Configuring Databases 4-4
      Configuring the Categorizer 4-5
Database Configuration Files 6-2
Context Objects 6-3
tables Object 6-3
dbinfo Object 6-4
tools Object 6-4
Application Configuration 6-5
Working Sample 6-5
Macro Definitions 6-6
dbinfo Configuration 6-6
SQL Definitions 6-6
Code for drop table 6-7
Code for create table 6-7
Code for create indexes 6-7
Code for insert 6-7
Code for metadata query 6-7
Testing and Debugging 6-7
Parsing a String 6-8
Obtaining Full Debug Information 6-8
Using the JDBC Framework in Scripts 6-8
Viewing and Setting the SCE Time Zone Offset 6-9
Scalability Hints for Oracle 6-10
Using Custom tablespaces 6-10
Using Table Partitioning 6-11

**APPENDIX A**

Code Samples A-1
Application Configuration A-1
tables.xml File A-1
tables.dtd File A-3
Adapter Configuration A-4
ragadapter.conf File A-4
ragadapter.xml File A-5

**APPENDIX B**

Configuration Files Descriptions B-1
General Configuration Files B-1
CM Configuration B-1
[adapter] Section B-2
[adapter_mem] Section B-2
[adapter_start_timeout] Section B-3
[buffer] Section B-3
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>[queue] Section</td>
<td>B-3</td>
</tr>
<tr>
<td>[categorizer] Section</td>
<td>B-4</td>
</tr>
<tr>
<td>[server] Section</td>
<td>B-4</td>
</tr>
<tr>
<td>[collector] Section</td>
<td>B-4</td>
</tr>
<tr>
<td>[csv_adapter] Section</td>
<td>B-4</td>
</tr>
<tr>
<td>Queue Configuration</td>
<td>B-5</td>
</tr>
<tr>
<td>Adapter Configuration Files</td>
<td>B-5</td>
</tr>
<tr>
<td>Topper/Aggregator Adapter</td>
<td>B-5</td>
</tr>
<tr>
<td>[config] Section</td>
<td>B-6</td>
</tr>
<tr>
<td>[rdr] Section</td>
<td>B-6</td>
</tr>
<tr>
<td>[csv] Section</td>
<td>B-7</td>
</tr>
<tr>
<td>[state save] Section</td>
<td>B-7</td>
</tr>
<tr>
<td>[db] Section</td>
<td>B-8</td>
</tr>
<tr>
<td>[app] Section</td>
<td>B-8</td>
</tr>
<tr>
<td>CSV Adapter</td>
<td>B-9</td>
</tr>
<tr>
<td>[csvadapter] Section</td>
<td>B-9</td>
</tr>
<tr>
<td>JDBC Adapter</td>
<td>B-9</td>
</tr>
<tr>
<td>[bench] Section</td>
<td>B-9</td>
</tr>
<tr>
<td>[db] Section</td>
<td>B-10</td>
</tr>
<tr>
<td>[app] Section</td>
<td>B-10</td>
</tr>
<tr>
<td>RAG Adapter</td>
<td>B-11</td>
</tr>
<tr>
<td>[config] Section</td>
<td>B-11</td>
</tr>
<tr>
<td>[housekeeper] Section</td>
<td>B-11</td>
</tr>
<tr>
<td>[db] Section</td>
<td>B-11</td>
</tr>
<tr>
<td>[app] Section</td>
<td>B-12</td>
</tr>
</tbody>
</table>
About this Guide

This guide describes the installation and configuration of the Cisco Service Control Management Suite (SCMS) Collection Manager (CM). 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 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. This guide assumes a basic familiarity with the concept of the Cisco Service Control solution, the Service Control Engine (SCE) platforms, and related components.

Document Revision History

The Document Revision History below records changes to this document.
<table>
<thead>
<tr>
<th>Revision</th>
<th>Cisco Service Control Release and Date</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>OL-7208-09</td>
<td>3.5.0 February 15, 2011</td>
<td>• Added a note on case-sensitivity with table names in Windows based MySql server and Collection Manager.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the Installing the Sybase Database, page 3-7 to include a step to specify the size of the Sybase database during the installation process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the [buffer] Section, page B-3.</td>
</tr>
<tr>
<td>OL-7208-08</td>
<td>3.5.0 January 25, 2011</td>
<td>• Added bundled Sybase maximum size information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for Red Hat Linux 5 and removed support for Red Hat Linux 3.0. See Checking System Prerequisites, page 3-2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The bundled Sybase database for Red Hat is upgraded to version 15.0.2 to support Red Hat Linux 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for MySQL 5.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for Oracle 11g</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The prunetable.sh and droptable.sh scripts are now supported on all supported databases. See Common Database Management Tasks, page 5-1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The sceconf.py script has been renamed to sceconf.sh</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Obsolete configuration parameters have been removed from the configuration files. See Configuration Files Descriptions, page B-1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added dbconf.sh script to configure external databases. See Configuring Databases, page 4-4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Bundled Sybase general performance improvements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added p3stats command line utility. See Viewing Database Statistics, page 4-11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed the HTTPC Adapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• update_statistics.sh script is supported for the bundled Sybase database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• version and dbversion options for CM. See Viewing Version Information, page 4-11.</td>
</tr>
<tr>
<td>OL-7208-07</td>
<td>3.1.6 May, 2008</td>
<td>• Added ranges and default values to the configuration files descriptions. See Configuration Files Descriptions, page B-1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarification of how to implement periodic delete. See Managing the Periodic Deletion of Old Records, page 5-2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minor changes throughout the guide.</td>
</tr>
<tr>
<td>Revision</td>
<td>Cisco Service Control Release and Date</td>
<td>Change Summary</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OL-7208-06</td>
<td>3.1.5 November, 2007</td>
<td>• Added section describing the TA Adapter memory requirements. See TA Adapter Memory Requirements, page 2-5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added section describing the support information scripts. See Collection Manager Support Information, page 4-2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added module describing the configuration parameters in the configuration files. See Configuration Files Descriptions, page B-1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed the 'Controlling the Database' section and all references to the dbconf.sh script.</td>
</tr>
<tr>
<td>OL-7208-05</td>
<td>3.1.0 May, 2007</td>
<td>• Added new script to manage virtual links semantics. See Managing Virtual Links, page 4-8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for Solaris 10 and removed support for Solaris 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for MySQL 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for Oracle 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The bundled Sybase database for Solaris is upgraded to version 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed --expert mode from the Sybase install script. The regular install mode allows the user to define the database capacity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The periodic delete mechanism and dbtables.sh script are now supported for external databases. See Common Database Management Tasks, page 5-1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The upgrade procedure to release 3.1 can preserve the previous version configuration. See install-cm.sh Options, page 3-9 and Upgrading the Collection Manager to Version 3.5.x, page 3-12.</td>
</tr>
<tr>
<td>OL-7208-04</td>
<td>3.0.5 November, 2006</td>
<td>Added the following sections to the document:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Managing Users, page 4-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enabling Adapters, page 4-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Configuring the Categorizer, page 4-5</td>
</tr>
<tr>
<td>OL-7208-03</td>
<td>3.0.3 May, 2006</td>
<td>Added the following new features:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The check_prerequisites.sh script. See Checking System Prerequisites, page 3-2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The monitor.sh script. See Monitoring System Health, page 4-5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed the following, deprecated feature:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The --legacy flag of the installsyb.sh script</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added the following section to the document:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Configuring Escape of Nonprintable Characters, page 5-10</td>
</tr>
<tr>
<td>Revision</td>
<td>Cisco Service Control Release and Date</td>
<td>Change Summary</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OL-7208-02</td>
<td>3.0 December, 2005</td>
<td>Added the following new feature:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HTTPC Adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed the following, deprecated feature:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Database Adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added the following section to the document:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Code Samples, page A-1</td>
</tr>
<tr>
<td>OL-7208-01</td>
<td>2.5.5 February, 2005</td>
<td>First version of this document.</td>
</tr>
</tbody>
</table>
Organization

This guide contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cisco Service Control Overview, page 1-1</td>
<td>Provides a functional overview of the Cisco Service Control solution</td>
</tr>
<tr>
<td>2</td>
<td>Collection Manager Overview, page 2-1</td>
<td>Describes the functionality of the Collection Manager components</td>
</tr>
<tr>
<td>3</td>
<td>Installing the Collection Manager, page 3-1</td>
<td>Describes the procedures for installing the Collection Manager and its database, and explains how to run the Collection Manager</td>
</tr>
<tr>
<td>4</td>
<td>Managing the Collection Manager, page 4-1</td>
<td>Explains how to use utility scripts to view and update Collection Manager parameters and other information</td>
</tr>
<tr>
<td>5</td>
<td>Managing Databases and the Comma Separated Value Repository, page 5-1</td>
<td>Explains how to use utility scripts to manage the Collection Manager database and the CSV repository</td>
</tr>
<tr>
<td>6</td>
<td>Configuring Databases, page 6-1</td>
<td>Explains how to configure the Collection Manager to work with your database</td>
</tr>
<tr>
<td>A</td>
<td>Code Samples, page A-1</td>
<td>Provides sample lists of code for configuration files</td>
</tr>
<tr>
<td>B</td>
<td>Configuration Files Descriptions, page B-1</td>
<td>Describes the available parameters in all of the configuration files</td>
</tr>
</tbody>
</table>

Related Publications

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*

Conventions

This document uses the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bold</strong> font</td>
<td>Commands and keywords and user-entered text appear in <strong>bold</strong> font.</td>
</tr>
<tr>
<td><em>italic</em> font</td>
<td>Document titles, new or emphasized terms, and arguments for which you supply values are in <em>italic</em> font.</td>
</tr>
</tbody>
</table>
Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What’s New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:


Subscribe to the What’s New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

Note

Means reader take note.

Tip

Means the following information will help you solve a problem.

Caution

Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

Timesaver

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

Warning

Means reader be warned. In this situation, you might perform an action that could result in bodily injury.
Cisco Service Control Overview

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

- Cisco Service Control Solution, page 1-1
- Cisco Service Control Capabilities, page 1-2
- SCE Platform Description, page 1-3
- Management and Collection, page 1-4

Cisco Service Control Solution

The Cisco service control solution is delivered through a combination of hardware and specific software solutions that address various service control challenges. Service providers can use the SCE platform to support classification, analysis, and control of Internet and IP traffic.

Service control enables service providers to:

- Capitalize on existing infrastructure.
- Analyze, charge for, and control IP network traffic at multigigabit wire line speeds.
- Identify and target high-margin content-based services and 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.

Cisco service control solutions allow the service provider to capture profits from IP services through detailed monitoring, precise, real-time control, and awareness of services as they are delivered.

Service Control for Broadband Service Providers

Service providers of any access technology (DSL, cable, mobile, and so on) targeting residential and business consumers must find new ways to get maximum leverage from their existing infrastructure, while differentiating their offerings with enhanced IP services.
The Cisco service control application for broadband adds a layer of service intelligence and control to existing networks that can:

- Report and analyze network traffic at subscriber and aggregate level for capacity planning
- Provide customer-intuitive tiered application services and guarantee application service level agreements (SLAs)
- Implement different service levels for different types of customers, content, or applications
- Identify network abusers who are violating the acceptable use policy (AUP)
- Identify and manage peer-to-peer traffic, NNTP (news) traffic, and spam abusers
- Enforce the AUP
- Integrate Service Control solutions easily with existing network elements and business support systems (BSS) and operational support systems (OSS)

Cisco Service Control Capabilities

The core of the Cisco service control solution is the 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 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 through 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 bandwidth (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 adapt to new services and applications in the 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 integration process.
SCE Platform Description

The SCE family of programmable network devices performs application-layer stateful-flow inspection of IP traffic, and controls the traffic based on configurable rules. The SCE platform is a network device that uses ASIC components and reduced instruction set computer (RISC) processors to exceed beyond packet counting and expand into the contents of network traffic. Providing programmable, stateful inspection of bidirectional traffic flows, and mapping these flows with user ownership, SCE platforms provide real-time classification of network use. The classification provides the basis of the SCE platform advanced traffic-control and bandwidth-shaping functionality. Where most bandwidth shaper functionality ends, the SCE platform provides further control and shaping options, including:

- Layer 7 stateful wire-speed packet inspection and classification
- Robust support for more than 600 protocols and applications, including:
  - General—HTTP, HTTPS, FTP, Telnet, Network News Transfer Protocol (NNTP), Simple Mail Transfer Protocol (SMTP), Post Office Protocol 3 (POP3), Internet Message Access Protocol (IMAP), Wireless Application Protocol (WAP), and others
  - Peer-to-Peer (P2P) file sharing—FastTrack-KazaA, Gnutella, BitTorrent, Winny, Hotline, eDonkey, DirectConnect, Piolet, and others
  - P2P VoIP—Skype, Skinny, DingoTel, and others
  - Streaming and Multimedia—Real Time Streaming Protocol (RTSP), Session Initiation Protocol (SIP), HTTP streaming, Real Time Protocol (RTP) and Real Time Control Protocol (RTCP), and others
- Programmable system core for flexible reporting and bandwidth control
- Transparent network and BSS and OSS integration into existing networks
- Subscriber awareness that relates traffic and usage to specific customers

Figure 1-1 illustrates a common deployment of an SCE platform in a network.
Figure 1-1  SCE Platform in the Network

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).
Network Management

The Cisco service control solution provides complete network Fault, Configuration, Accounting, Performance, Security (FCAPS) Management.

Two interfaces provide 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 (through SNMP traps) and performance-monitoring functionality.

Subscriber Management

Where the Cisco service control application for broadband (SCA BB) enforces 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 OSS and 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 standards-based approach makes it easy to manage multiple devices in a large network.

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

Data Collection

Data collection occurs as follows:

1. All analysis and data processing functions of the SCE platform result in the generation of Raw Data Records (RDRs), which the SCE platform forwards using a simple TCP-based protocol (RDR-Protocol).
2. RDRs are processed by the Cisco service control management suite collection manager.
3. 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 variety of information and statistics, depending on the configuration of the system. Three main categories of RDRs include:

- 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.
Collection Manager Overview

This module provides detailed information about the functionality of 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.

- Data Collection Process, page 2-1
- Raw Data Records, page 2-1
- Collection Manager Software Package, page 2-2
- Adapter Description, page 2-3
- Using Databases, page 2-6

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).

1. 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).

2. 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.

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

You can use preinstalled utility scripts to customize 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.
Table 2-1 contains examples of RDRs produced by SCP applications:

<table>
<thead>
<tr>
<th>Report Type</th>
<th>Description</th>
<th>RDRs contain...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodic Subscriber usage report</td>
<td>SCE platforms are subscriber-aware network devices; they can report usage records per subscriber.</td>
<td>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.</td>
</tr>
<tr>
<td>Transaction level report</td>
<td>SCE platforms perform stateful tracking of each network transaction conducted on the links on which they are situated. Using this statefulness, the SCP tracks several OSI Layer 7 protocols (such as HTTP, RTSP, SIP, or Gnutella) to report on various application level attributes.</td>
<td>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-dependent Layer 7 attributes (such as user-agent, hostname 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.</td>
</tr>
<tr>
<td>SCP application activity reports</td>
<td>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.</td>
<td>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).</td>
</tr>
</tbody>
</table>

### Collection Manager Software Package

The Collection Manager software package is a group of processing and sorting modules. These include the following components:

- **Raw Data Record Server**, page 2-3
- **Categorizer**, page 2-3
- **Priority Queues and Persistent Buffers**, page 2-3
Raw Data Record Server

As each incoming raw data record (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

A 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 nonvolatile storage area that ensures that the system processes RDRs even in cases of hardware, software, or power failures.

Adapter Description

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, page 2-3
- Comma Separated Value Adapter, page 2-4
- Topper/Aggregator Adapter, page 2-4
- Real-Time Aggregating Adapter, page 2-5

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 ragadapter.conf File, page A-4.

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 Adaptive Server Enterprise (ASE) 12.5 and 15.0
Chapter 2  Collection Manager Overview

Adapter Description

- Oracle 9.2, 10.2, and 11

Note The recycle bin feature available in Oracle 10 and later versions should be disabled. You can set the initial value of the recyclebin parameter in the text initialization file init<SID>.ora, for example:
   recyclebin=off

- MySQL 4.1, 5.0, and 5.1

Comma Separated Value Adapter

The comma separated value (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.

Topper/Aggregator Adapter

The topper/aggregator (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.

Note When several CM servers use a single database, the TA adapter information may not be accurate because it is aggregated locally on each of the CMs.

- TA Adapter Cycles, page 2-4
- TA Adapter Memory Requirements, page 2-5

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 each 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 each long cycle, the adapter:
  - Adds the cycle’s aggregated Top Reports to the long 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

**TA Adapter Memory Requirements**

To work correctly, you must dedicate a sufficient amount of memory to the TA adapter. Configure the value in the `cm.conf` configuration file in the following location:

```
[adapter_mem]
com.cisco.scmscm.adapters.topper.TAAdapter=<Memory for TA Adapter>
```

To calculate the recommended amount of memory to dedicate to the TA adapter, use the following formula:

Memory (Bytes) = 2.5 * NUM_SUBSCRIBERS * (AVG_SUBS_ID_LENGTH + 64*NUM_SERVICES + 12*NUM_TOP_ENTRIES)

Where:

- NUM_SUBSCRIBERS is the number of new subscribers that will be introduced in one day (on all SCEs sending reports to this CM).
  - This is usually a high number; especially when working in anonymous subscriber mode.
  - To display an estimate of the number of subscribers that are known to the CM, use the following command:

```
~/setup/mbean.py --getattr=Subscribers DCAAdapters com.cisco.scmscm.adapters.topper.TAAdapter
```

- AVG_SUBS_ID_LENGTH is the average character length of a subscriber.
  - In most cases, this is approximately 20.

- NUM_SERVICES is the number of subscriber usage counters and is configured in the `taadapter.conf` configuration file.

- NUM_TOP_ENTRIES is configured in the `taadapter.conf` configuration file under the `num_top_entries` value.

**Note**

For Linux, the configured memory should not be over 2 GB.
For Solaris JRE 32-bit, the configured memory should not be over 3.5 GB.
For Solaris JRE 64-bit, you can set higher values for the configured memory. To configure the TA or RAG Adapters to run with the JRE 64-bit see [adapter_mem] Section, page B-2.

**Real-Time Aggregating Adapter**

The real-time aggregating (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, page 2-6
- Flushing a Bucket, page 2-6
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 ragadapter.xml File, page A-5.

Flushing a Bucket

When a bucket is flushed, it is written as one 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 reaches a configured amount
- The volume in an accumulated field in the bucket exceeds a configured amount
- The adapter, or the entire CM, goes down
- An RDR arrives at the bucket with some new value (relative to the bucket contents) in some field

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 entire CM, goes down

Using 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, page 2-7
- Using an External Database, page 2-7
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.

1. The JDBC adapter sends converted RDRs to the database to be stored in these tables.
2. 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
- Configure a database pack to supply the CM with the parameters of the database (such as its IP address and port).
- Supply a JDBC driver for the database, to be used by the adapter when connecting to it.

For details about configuring the CM to work with an external database, see Managing Databases and the Comma Separated Value Repository, page 5-1.
CHAPTER 3

Installing the Collection Manager

This module describes the procedures for installing the collection manager (CM) and its database, and explains how to run the collection manager.

For typical installation and configuration, use the Cisco Service Control Management Suite Collection Manager Quick Start Guide.

- System Requirements, page 3-1
- How to Install the Collection Manager, page 3-6
- How to Uninstall the Sybase Database and Collection Manager Software, page 3-11
- Upgrading the Collection Manager to Version 3.5.x, page 3-12

System Requirements

The CM and its database are software components that run on a server platform. They can be installed on any of the following configurations:

- Sun SPARC machine (64-bit) running 64-bit versions of Solaris 9 or Solaris 10. (See Solaris Requirements, page 3-2)
- Intel machine (32-bit or 64-bit) running 32-bit versions of Red Hat Enterprise Linux 4.0 or Red Hat Enterprise Linux 5.0. (See Red Hat Linux Requirements, page 3-5)

All configurations use a 32-bit Java virtual machine (JVM).

The CM must run on its own machine. You cannot run it on the same machine as the Subscriber Manager and/or other applications.

- Checking System Prerequisites, page 3-2
- Solaris Requirements, page 3-2
- Red Hat Linux Requirements, page 3-5
- Distribution Content, page 3-6
- Default Configuration Settings, page 3-6
Checking System Prerequisites

The CM distribution contains a script, `check_prerequisites.sh`, located in the `install_scripts` directory. The script helps to determine if 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 4 or 5)
- Additional required and optional packages
- Python installed and executable in path
- Free space for CM and Sybase homes
- Names for all network interface cards (NICs)
- Sybase kernel parameters
- Locale and time zone formats

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--sybhome=SYBHOME</td>
<td>Intended home directory for Sybase installation</td>
</tr>
<tr>
<td>--datadir=DATADIR</td>
<td>Intended data directory for Sybase data files (for the Datadir installation method)</td>
</tr>
<tr>
<td>--cmhome=CMHOME</td>
<td>Intended home directory for CM installation</td>
</tr>
</tbody>
</table>

Solaris Requirements

Collection Manager Release 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, page 3-2
- Software and Environment, page 3-3
- Setting the Locale and Time Zone, page 3-4

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 Version 5.9 64-bit build 04/01 or later (currently only Solaris Version 5.9 and 5.10 are supported).
  - Solaris 9—Patch level 9 is recommended
  - Solaris 10—Patch level 10 is recommended
- Solaris Core Installation.
- Install the following additional packages:

<table>
<thead>
<tr>
<th>system</th>
<th>SUNWbash</th>
<th>GNU Bourne-Again shell (bash)</th>
</tr>
</thead>
<tbody>
<tr>
<td>system</td>
<td>SUNWgzip</td>
<td>The GNU Zip (gzip) compression utility</td>
</tr>
<tr>
<td>system</td>
<td>SUNWzip</td>
<td>The Info-Zip (zip) compression utility</td>
</tr>
<tr>
<td>system</td>
<td>SUNWlibC</td>
<td>Sun Workshop Compilers Bundled libC</td>
</tr>
<tr>
<td>system</td>
<td>SUNWlibCx</td>
<td>Sun WorkShop Bundled 64-bit libC</td>
</tr>
</tbody>
</table>

- If you are installing the CM in bundled mode with the Sybase database, you must install the following package:

  | system | SUNWipc | Interprocess Communication |

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

<table>
<thead>
<tr>
<th>system</th>
<th>SUNWadmap</th>
<th>System administration applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>system</td>
<td>SUNWadmc</td>
<td>System administration core libraries</td>
</tr>
</tbody>
</table>

- To use the Python scripts, a Python interpreter Version 2.2.1 or later must be present on the system. You can install the following interpreter package:

<table>
<thead>
<tr>
<th>application</th>
<th>SMCpython (Solaris 9)</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>SMCpython (Solaris 10)</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>application</th>
<th>SMClibgcc</th>
<th>libgcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>SMCncurs</td>
<td>ncurses</td>
</tr>
</tbody>
</table>

- You can download these packages from http://sunfreeware.com/
System Requirements

The root (/) partition must have at least 104 MB of free space to install these packages.

- Apply the latest recommended patches from Sun:
  - 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, install the current Solaris patches recommended by Sybase.

- 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 one partition to hold the desired size of the Sybase data and logs (the sizes are configurable at install time).

- (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) Before installation, verify that all IP addresses that are configured for the machine NICs have hostnames 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) Use the set_shmmax.sh script (located under install-scripts/) to configure the kernel memory.

- Additionally, at startup you must load the IPC module by putting the following lines in the file /etc/system:

  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, U.S. English locale must be used.

  To set the locale, put the following line in the /etc/TIMEZONE configuration file (changes to this file require 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 Version 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, page 3-5
• Software and Environment, page 3-5
• Setting the Locale and Time Zone, page 3-6

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 4.0.
  – kernel -2.6.9-5
  – glibc-2.3.4-2
  – compat-libstdc++-33-3.2.3-47.3
• Red Hat Linux 5.0
  – kernel-2.6.18-8.el5
  – glibc-2.5-12
  – compat-libstdc++-33-3.2-61
• Red Hat Enterprise "Base" Installation.
• (For installations with bundled Sybase) Install the following additional package:
  – compat-libstdc++
• This package is available on the Red Hat installation CD.
How to Install the Collection Manager

- Apply latest recommended patches from Red Hat.
- (For installations with bundled Sybase) Install current patches recommended by Sybase.
- Reserve 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.
- (For installations with bundled Sybase where the legacy (pre-Version 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) Before installation, verify that all IP addresses that are configured for the machine NICs have hostnames 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) Use the set_shmmmax.sh script (located under install-scripts/) to configure the kernel memory.
- 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, U.S. 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 to enable 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, page 3-7
- Installing the Sybase Database, page 3-7
- Installing Collection Manager Software, page 3-9
Ports Used by the Collection Manager Software

Table 3-2 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.

Table 3-2 Ports that the CM Listens on Constantly

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

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.

Installing the Sybase Database

If you do not want to install Sybase (for example, when working in unbundled mode), go to Installing Collection Manager Software, page 3-9.

**Note** Installing the Sybase database can take up to three hours.

**Note** The maximum database size supported by the bundled Sybase database is 50GB. For database support larger than 50GB, use an external database.
During installation, if you want to reverse the Sybase installation actions (for example, if an installation is interrupted because of a power failure), do the following:

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

---

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

**Step 2**  
Run the script `installsyb.sh`. Enter the script 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.
  - Use a location 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.
- During the Sybase installation process, you need to enter the size of the Sybase database. Following message is displayed to enter the database size:
Please enter SIZE in 2K blocks of file to be used for the "data[maximum is 102657160; minimum is 5242798]" device.
NOTE - the actual size required will include a 1.05 overhead on the amount you specify.
SIZE in 2K blocks:

Installing Collection Manager Software

**Note**
During the installation, if you want to reverse the Sybase installation action (for example, if an installation is interrupted because of a power failure), do the following:

1. Log in as the root user.
2. End 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

Use the `install-cm.sh` script 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
Installing the Collection Manager

How to Install the Collection Manager

- 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

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, page 3-9.
        For additional information about the script, see Actions Performed by install-cm.sh, page 3-9.
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 password that you choose.
Step 4  Increase the amount of memory allocated to the topper/aggregator adapter
        For information about calculating the topper/aggregator adapter memory, see TA Adapter Memory Requirements, page 2-5.
        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 managed 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 Comma Separated Value Repository, page 5-1 for details of how to do this.

Step 6  For each adapter that your application uses, 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/3.5.0.
• 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/3.5.0.

**Step 7**

Install and activate the periodic delete procedures for the database tables.

(For information about configuring the behavior 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 to 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**

Run this script every time that the time zone of the SCE is updated; for example, when updating the time zone when moving to daylight savings time. The SCA Reporter will not use the correct time intervals unless the offset stored in the CM is consistent with the SCE’s time zone.

**Step 9**

Start the CM by running the following command:

~scmscm/cm/bin/cm start

---

**How to Uninstall the Sybase Database and Collection Manager Software**

• Uninstalling Sybase, page 3-11

• Uninstalling the Collection Manager Software, page 3-12

**Uninstalling Sybase**

To uninstall the Sybase database:
Uninstalling the Collection Manager Software

To uninstall the CM software:

Step 1 Log in as the root user.
Step 2 Uninstall the CM software
Run the following commands to uninstall the CM software:
   pkill -u scmscm
   userdel -r scmscm
   rm /etc/rc*.d/[SK]*scmscm

Upgrading the Collection Manager to Version 3.5.x

To upgrade the CM to Version 3.5.x:

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 existing scmscm user is used.
   After the upgrade, when the CM comes up for the first time, the database tables that are new in Version 3.5.x are created automatically.

Note The upgrade to Version 3.5.x can only be performed from Version 3.x.x.
Managing the Collection Manager

This module explains how to use utility scripts to view and update collection manager (CM) parameters. Any machine connected to the CM through, 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 Comma Separated Value Repository, page 5-1.

- Using Utility Scripts, page 4-1
- Collection Manager Support Information, page 4-2
- Configuring the Collection Manager, page 4-2
- Configuring the Categorizer, page 4-5
- Monitoring System Health, page 4-5
- Managing Users, page 4-7
- Managing Virtual Links, page 4-8
- How to Monitor the Collection Manager, page 4-9

Using Utility Scripts

General instructions for using utility scripts include:

- To invoke any script, log in as the scmscm user, except where otherwise noted. An attempt to run these scripts as the root user results 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

A slight variation exists 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.

```bash
> -scmscm/scripts/dbperiodic.py --help
Usage:
   -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.

Collection Manager Support Information

If you are experiencing difficulties with the CM installation, it may be necessary to provide the Cisco Technical Assistance Center (TAC) with information about the current system setup. The CM can create the required support information or files for TAC by running one of the following scripts:

- `~scmscm/unsupported/getinfo/get_cm_info.sh`—This script creates the CM support information.
- `~scmscm/unsupported/getinfo/get_support_files.sh`—This script creates a support file.

The output is a zip file with the investigation supporting files. The following example shows the output that appears when running this script:

```
bash-2.05$ ~scmscm/unsupported/getinfo/get_support_files.sh
Gathering support files into Sun_14_Oct_2007_06-05-03_PM_JST.zip, please wait............done.
The generated support file contains the output of ~scmscm/unsupported/getinfo/get_cm_info.sh
```

Configuring the Collection Manager

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

Use the following scripts to configure the CM:

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

For information about scripts for managing the database and the CSV repository, see Managing Databases and the Comma Separated Value 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 Adapters, page 4-4.
- `queue.conf`—Configuration of the adapter queues, including which RDR tags will be associated with a specific adapter. See Configuring the Categorizer, page 4-5.
Activating 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**

To view the current startup status of each component, run the script with no parameters.

To restart the CM, run the following script as the scmscm user:

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

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 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 4-1 on-boot.py Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--cm=on/off</td>
<td>Activate/do not activate the CM at startup.</td>
</tr>
<tr>
<td>--sybase=on</td>
<td>Activate/do not activate the Sybase server at startup.</td>
</tr>
</tbody>
</table>

The following example shows how to shut down an adapter named `adapter` as the scmscm user:

```
>~scmscm/scripts/adapterconf.py --op=stop --adapter=adapter
```
To bring up an adapter, as the scmscm user, run the following script:

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

## Enabling Adapters

You can define an adapter 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 shows how to define the RAG adapter to turn on when the CM starts.

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

The following example shows how to define 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.

## Dropping an SCE Connection

To drop a connection to a particular SCE, use the `sceconf.sh` script:

```
~scmscm/scripts/sceconf.sh --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, page 4-10)

### Table 4-3  `sceconf.sh` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter=IP address</td>
<td>Drop the connection at the specified IP address.</td>
</tr>
<tr>
<td>--help</td>
<td>Display these options.</td>
</tr>
</tbody>
</table>

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

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

## Configuring Databases

To configure a database, use the `dbconf.sh` script:

```
~scmscm/scripts/dbconf.sh
```

The script prompts you to choose the database type and the corresponding database configuration parameters.

The following example shows how to use the dbconf.sh script to configure the CM to work with an external Oracle database:

```
$ ~scmscm/scripts/dbconf.sh
Enter the DB type:
1 - Oracle
2 - MySQL
3 - Sybase (external not bundled)
Enter your choice: 1
Enter Oracle server host (current is localhost): 10.56.216.80
```
Enter Oracle server listening port (current is 1521) :
Enter Oracle server instance id (current is apricot) :
Enter CM schema user name (current is pqb_admin) :
Enter CM schema user password (current is pqb_admin) :
Do you want to test the DB connection? (yes/no): yes
PASS: db is up
DB connection succeeded.

Note
By default, the Collection Manager handles database related transactions in uppercase table names. However, in Windows based MySql server, table names are stored in lowercase and name comparisons are not case-sensitive; and, it automatically converts all table names to lowercase. Therefore, if you are using Windows based database server, you need to modify the lower_case_table_names server configuration entry.

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.

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

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]
The following example shows how to install monitor.sh so that it runs one time every 30 minutes.

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

The following example shows how to uninstall monitor.sh.

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

### Periodic Checker Script

The periodic checker script, `monitor.sh`, calls a series of subscripts 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 4-4  `setup-monitor.sh` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a</code></td>
<td>Make/remove an entry for <code>monitor.sh</code> in the cron.</td>
</tr>
<tr>
<td><code>-I</code></td>
<td>Run <code>monitor.sh</code> every 30 minutes, 1 hour, 12 hours, or 24 hours.</td>
</tr>
</tbody>
</table>

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

```
$ ./monitor.sh -d -a
```

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

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

```
$ ./monitor.sh -d -a
test: 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
```

#### Table 4-5  `monitor.sh` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a</code></td>
<td>Run all tests.</td>
</tr>
<tr>
<td>TEST NAME</td>
<td>The names of one or more tests. A test name is the test filename, without the leading digits and trailing .sh.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Output results in verbose mode. (Log successful tests.)</td>
</tr>
<tr>
<td><code>-d</code></td>
<td>Print results to window. (By default, results are sent to syslog.)</td>
</tr>
</tbody>
</table>

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`
Tests

The following tests can be run using `monitor.sh`:
- `db_up`—Checks that the CM database is running.
- `cm_up`—Checks that the CM application is running.
- `free_db`—Checks that the bundled Sybase database has at least 10 percent free space.
- `free_log`—Check that the bundled Sybase 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 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 script priority. For example, the test `free_db` is 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 `p3rpc` operations and options.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--set-user --username</code>=username --password=password</td>
<td>Adds and updates the username and password.</td>
</tr>
<tr>
<td><code>--validate-password --username</code>=username --password=password</td>
<td>Validates the username and password.</td>
</tr>
<tr>
<td><code>--delete-user --username</code>=username</td>
<td>Delete a user configuration.</td>
</tr>
<tr>
<td><code>--show-users</code></td>
<td>Displays all configured users.</td>
</tr>
</tbody>
</table>

Examples

The following example shows how to add and update the username and password:

```
bash-2.05$ p3rpc --set-user --username=lulu --password=lili
Command terminated successfully
bash-2.05$
```

The following example shows how to validate username and password. In this example, the user is successfully validated.

```
bash-2.05$ p3rpc --validate-password --username=lala --password=lala
Local machine: user lala was authenticated successfully : (auth level root)
Command terminated successfully
bash-2.05$
```

The following example shows how to validate username and password. In this example, the user validation fails.

```
bash-2.05$ p3rpc --validate-password --username=lulu --password=lili
```
bash-2.05$ p3rpc --validate-password --username=lala --password=lulu
Error - Failed to authenticate user lala
bash-2.05$

The following example shows how to delete a user configuration.
bash-2.05$ p3rpc --delete-user --username=lulu
Command terminated successfully
bash-2.05$

The following example shows how to display all of the configured users. In this example, only one configured user exists: clu.
bash-2.05$ p3rpc --show-users
clu
Command terminated successfully
bash-2.05$

Managing Virtual Links

A script is included in the CM distribution to allow you to manage virtual link names and indexes 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>
<thead>
<tr>
<th>Table 4-7 update_vlinks.sh Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--sce=SCE IP --file=file</code></td>
</tr>
<tr>
<td><code>--sce=SCE IP --show</code></td>
</tr>
<tr>
<td><code>--help</code></td>
</tr>
</tbody>
</table>

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)
- No duplicate virtual links names or empty names exist 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"
Chapter 4      Managing the Collection Manager

How to Monitor the Collection Manager

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
- Viewing database insertion rate statistics per table
- Viewing version information

The following scripts are used to monitor the CM:

- `~scmscm/scripts/dbfree.sh`
- `~scmscm/scripts/rdr-rate.py`
- `~scmscm/scripts/sceconf.sh`
- `~scmscm/setup/alive.sh`
- `~scmscm/cm/bin/p3stats`
- `~scmscm/cm/bin/cm version`
- `~scmscm/cm/bin/cm dbversion`

The following scripts are used to configure the CM (see Configuring the Collection Manager, page 4-2), but can also be invoked to display the relevant configuration:

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

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.

| 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.

\section*{Step 1}
As the scmscm user run the \texttt{rdr-rate.py} script

\section*{Checking the SCE Connection}
To display information about the SCE connections, use the \texttt{sceconf.sh} script:

\texttt{-scmscm/scripts/sceconf.sh --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 \textit{Dropping an SCE Connection}, page 4-4.

\section*{Step 1}
As the scmscm user, run the \texttt{sceconf.sh} script

\texttt{-scmscm/scripts/sceconf.sh --op=list}

\section*{Example:}
The following example shows SCE connection output:

\begin{table}[h]
\begin{tabular}{lll}
\hline
IP       & 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          \\
\hline
\end{tabular}
\end{table}

\section*{Verifying Server Operation}
To verify that the server is functioning correctly, use the \texttt{alive.sh} script:

\texttt{-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.

\section*{Step 1}
As the scmscm user, run the \texttt{alive.sh} script
How to Monitor the Collection Manager

**It takes time for the components to initialize after a startup. After a restart, wait 5 minutes before running this script.**

### Viewing Database Statistics

To view the statistics of the CM database, use the p3stats command line utility (CLU).

```bash
-scmscm/cm/bin/p3stats [options]
```

Table 4-8 lists the p3stats options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--show-stats</td>
<td>Displays statistics information in the format &lt;DB RDR tag&gt; &lt;RDR rate&gt; &lt;peak RDR rate&gt;</td>
</tr>
<tr>
<td>--reset-stats</td>
<td>Resets the statistics counters.</td>
</tr>
</tbody>
</table>

**Note**
The CM must be up for this CLU to operate.

The following example shows how to display the statistics of the CM database:

```bash
$ -scmscm/cm/bin/p3stats --show-stats
LUR { rate=0, peak=0 }
MALUR { rate=0, peak=0 }
TR { rate=0, peak=0 }
VLUR { rate=0, peak=0 }
NUR { rate=0, peak=0 }
SUR { rate=0, peak=0 }
MEDIA { rate=0, peak=0 }
PUR { rate=0, peak=0 }
Command terminated successfully
$
```

### Viewing Version Information

The Collection Manager includes two scripts to display the current version of the Collection Manager and of the database:

- `cm version`
- `cm dbversion`

The following example shows how to view the current Collection Manager version:

```bash
$ -scmscm/cm/bin/cm version
CM CD Version 3.5.0 Build 336
```

The following example shows how to view the current database version:

```bash
$ -scmscm/cm/bin/cm dbversion
```
Sybase ase150/
Managing Databases and the Comma Separated Value Repository

This module explains how to use utility scripts to manage the Collection Manager database and the comma separated value (CSV) repository. Many 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.

Note

By default, the Collection Manager handles database related transactions in uppercase table names. However, in Windows based MySql server, table names are stored in lowercase and name comparisons are not case-sensitive; and, it automatically converts all table names to lowercase. Therefore, if you are using Windows based database server, you need to modify the `lower_case_table_names` server configuration entry.

- Common Database Management Tasks, page 5-1
- Managing the Bundled Database, page 5-6
- Managing the CSV Repository, page 5-8

Common Database Management Tasks

Database management tasks that are applicable to all the supported databases are:

- Generating a list of the database tables
- Defining and applying the schedule for the periodic deletion of old records
- Deleting a table
- Manually deleting old records from a table

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. Use the following scripts to configure and maintain the database:

- `~scmscm/scripts/dbtables.sh`
Listing Database Tables

To display a list of 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*. Table 5-1 lists the `dbtables.sh` script options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-l</code></td>
<td>Lists the existing table names only (without statistics).</td>
</tr>
<tr>
<td><code>-a</code></td>
<td>Include the nonreport tables in the listing.</td>
</tr>
<tr>
<td><code>-f</code></td>
<td>Enable fast line counting, use the client rather than JDBC.</td>
</tr>
<tr>
<td><code>-t { sec_num }</code></td>
<td>The maximal waiting time, in seconds, for the response. The default is no timeout.</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>Prints this help message and exits.</td>
</tr>
</tbody>
</table>

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

```
>~scmscm/scripts/dbtables.sh
Executing 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_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

To manage the periodic deletion of old records, perform the following steps:

1. Install the periodic delete procedures if they were not installed during the CM installation:
Log in as the scmscm user, start the CM, wait 1 to 2 minutes for the database tables to be created, and then run the script:

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

2. Edit the periodic delete configuration file. See Configuring Periodic Delete, page 5-3.

3. To apply the new configuration, use the `dbperiodic.py` utility script. See Applying the Periodic Delete Configuration File, page 5-5

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.

All periodic delete activity is recorded in the system log file. For Solaris, the system log file is located in the `/var/adm/messages` directory; for Linux, the system log file is located in the `/var/log/messages` directory. The following is an example of the system log file at the time when the periodic delete action occurs.

```
Feb 17 05:00:07 app-redhat79 logger: /opt/scmscm/db_maint/periodic_del.sh for hourly all - running single del for RPT_LUR to 14 days
Feb 17 05:00:09 app-redhat79 logger: Starting periodic delete for RPT_LUR keeping 14 days
Feb 17 05:00:09 app-redhat79 logger: Deleted rows: 0 from the table RPT_LUR
```

**Note**

Periodic delete when using a MySQL database is supported only on Version 5 or later.

- Configuring Periodic Delete, page 5-3
- Applying the Periodic Delete Configuration File, page 5-5

### 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 the 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 appear in Table 5-2.
Common Database Management Tasks

Table 5-2  Parameters in the Periodic Delete Configuration File

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

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 lists the `dbperiodic.py` options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--load</td>
<td>Load the periodic delete configuration from /export/home/scmscm/db_maint/dbperiodic.conf</td>
</tr>
<tr>
<td>--loadfile=path to periodic delete configuration file</td>
<td>Load the periodic delete configuration file from the specified directory</td>
</tr>
<tr>
<td>--dump</td>
<td>Print the periodic delete configuration</td>
</tr>
<tr>
<td>--h</td>
<td>Display these options</td>
</tr>
</tbody>
</table>

The following example shows how to print the current periodic delete configuration:

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

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

The following example shows how to load the periodic delete configuration file from `~scmscm/db_maint/dbperiodic.conf`:

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

The following example shows how to load the periodic delete configuration file from a specified location:

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

Deleting a Table

To delete one table or all current tables from the database, use the `droptable.sh` script:

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

Table 5-4 lists the `droptable.sh` options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>Drop table_name from the database</td>
</tr>
<tr>
<td>ALL/TABLES</td>
<td>Drop all tables from the database</td>
</tr>
</tbody>
</table>
Managing the Bundled Database

Managing the bundled database includes:

- Backing up and restoring a database
- Updating Sybase with a changed IP address
Managing Databases and the Comma Separated Value Repository

Chapter 5      Managing Databases and the Comma Separated Value Repository

Managing the Bundled Database

• Updating the Sybase database statistics

Each 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/sybback.sh
• ~scmscm/scripts/sybrestore.sh
• ~scmscm/db_maint/update_statistics.sh

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 lists the sybback.sh options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d path_to_backup_directory</td>
<td>Write backup text files to the specified directory</td>
</tr>
<tr>
<td>-h</td>
<td>Display these options</td>
</tr>
</tbody>
</table>

Table 5-6 sybback.sh 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 lists the sybrestore.sh options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d path_to_restore_directory</td>
<td>Restore the database using the text files in the specified directory</td>
</tr>
<tr>
<td>-h</td>
<td>Display these options</td>
</tr>
</tbody>
</table>

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.
Chapter 5      Managing Databases and the Comma Separated Value Repository

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. To prevent disk overflow, monitor the size of the CSV repository.

Note

If the backup parameter is set to true, failure to delete CSV files may result in disk overflow (No CSV files are ever 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, page 5-8
- Configuring the CSV File Repository, page 5-9
- Configuring the Comma Escape, page 5-10
- Configuring Escape of Nonprintable Characters, page 5-10

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 when 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.)

Table 5-8 lists the csvconf.sh options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--list</td>
<td>Display the CSV repository contents (the number of RDRs currently stored in the repository).</td>
</tr>
<tr>
<td>--clear</td>
<td>Delete all files from the CSV repository. (This option deletes all CSV files, but not the directories in which they are contained.)</td>
</tr>
<tr>
<td>--maxlines=N</td>
<td>Set the maximum number of RDRs per CSV file to N (an integer between 1 and 20,000).</td>
</tr>
<tr>
<td>--maxfiles=M</td>
<td>Set the maximum number of CSV files in each subdirectory to M (an integer between 10 and 10,000.)</td>
</tr>
<tr>
<td>--backups={true</td>
<td>false}</td>
</tr>
<tr>
<td>--recordsource={true</td>
<td>false}</td>
</tr>
</tbody>
</table>

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 shows how to delete all files from the CSV repository:

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

The following example shows how to disable backing up old CSV files in the repository:
Managing the CSV 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 indicates 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

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 (\).

You enable this option in the `csvadapter.conf` file in the CSVAdapter directory. Changes in this file require a CM restart to take effect.
Configuring Databases

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, page 6-1
- Generating SQL Code Using the Velocity Template Language, page 6-1
- Database Configuration Files, page 6-2
- Working Sample, page 6-5
- Testing and Debugging, page 6-7
- Using the JDBC Framework in Scripts, page 6-8
- Scalability Hints for Oracle, page 6-10

Quick Start Guide

To use an external database with the CM, it is necessary to change basic connection parameters such as the IP address and port on which the database is deployed. To configure these parameters, use the dbconf.sh script. See Configuring Databases, page 4-4.

Generating SQL Code Using 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.

For more information regarding VTL (which is part of the Apache Jakarta Project) go to http://jakarta.apache.org/velocity/vtl-reference-guide.html.

Table 6-1 describes VTL constructs:
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 Table 6-2:

### Table 6-2 Summary of VTL Constructs

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

## 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 Table 6-2:
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

**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

### Table 6-2  Mandatory VM Definitions

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

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, page 6-3
- Application Configuration, page 6-5
- 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, page 6-5.

**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)

**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 Table 6-3:

### Table 6-3 Methods of the `tools` Object

<table>
<thead>
<tr>
<th>Method Name and Arguments</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getTableByName (allTables, name)</code></td>
<td>Locates the database table object whose logical name corresponds to name.</td>
</tr>
<tr>
<td><code>getTableByDbTabName (allTables, name)</code></td>
<td>Locates the database table object whose physical name corresponds to <code>name</code>.</td>
</tr>
<tr>
<td><code>assignParams (sql, list_of_args)</code></td>
<td>Replaces question mark characters in the sql string with consecutive elements from the <code>list_of_args</code> 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.</td>
</tr>
<tr>
<td><code>collapseWhitespace()</code></td>
<td>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.)</td>
</tr>
</tbody>
</table>
For a sample that demonstrates how to use these tools, see Using the JDBC Framework in Scripts, page 6-8.

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 tables Object, page 6-3.

Working Sample

The main.vm file can contain references to other VM files to support modularization (see Database Configuration Files, page 6-2). The names of these other files are arbitrary, except for the VM_global_library.vm file whose name is predetermined. Place macros that need to be defined in this file to ensure that they are loaded at the right time. For details about this special file, see the Velocity User Guide.

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

```velocity
#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, page 6-6
- dbinfo Configuration, page 6-6
- SQL Definitions, page 6-6
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.

```velocity
#macro (optcomma)#if ($velocityCount >1),#end#end
#macro (sqltype $field)
    #set ($maxStringLength = 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 <= $maxStringLength)$field.size
    #else $maxStringLength #end)
    #elseif ($field.type == "TEXT") long
    #elseif ($field.type == "TIMESTAMP") date
#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.

```velocity
#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, page 6-7
- Code for create table, page 6-7
- Code for create indexes, page 6-7
- Code for insert, page 6-7
- Code for metadata query, page 6-7
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. A 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 using the CM main script ~scmscm/cm/bin/cm.
The general syntax for 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, page 6-8
- Obtaining Full Debug Information, page 6-8

### 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. Examples appear below (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 '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 of 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 using the -executeUpdate flag. To perform a query and view the results, call the adapter using the -executeQuery flag.
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`; because the table may be assigned another name, it is referred to here by its logical name `TZ`. See the listing in 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 is defined in the templates.

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

```bash
#!/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

```
Scalability Hints for Oracle

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

- Using Custom tablespaces, page 6-10
- Using Table Partitioning, page 6-11

Using Custom tablespaces

Suppose you create several tablespaces and wish to distribute the CM tables among them. Specify the tablespace to be used for each table in the file tables.xml. For one table, the definition looks like this (note the code in bold):

```xml
<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>
    <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:

```bash
create table $table.dbtabname (  
  #foreach ($field in $table.fields)  
    $field,  
  #end  
)  
```

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

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

```vm
#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, page 6-10). Then augment the `create_table.vm` code as follows (note the code in bold):

```vm
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
```

Because 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, page 6-8) to issue the appropriate `alter table drop partition` and `alter table add partition` SQL commands.
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, page A-1
- Adapter Configuration, page A-4

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.

- tables.xml File, page A-1
- tables.dtd File, page A-3

tables.xml File

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

```xml
<?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"/>
      </fields>
    </rdr>
  </dbtables>
</dbtabconf>
```
<field id="4" name="PACKAGE_ID" type="INT32"/>
<field id="5" name="SERVICE_ID" type="INT32">
  <options>
    <option property="nonnull" 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>
For each table (either an RDR table, an aggregation table, or an extra table), the fields, indexes, and so on 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.

### tables.dtd File

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

```xml
<?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 dbtable (options?, fields, indexes?)>
<!ATTLIST dbtable
  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
>```
createable (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.

- ragadapter.conf File, page A-4
- ragadapter.xml File, page A-5

ragadapter.conf File

To perform RAG adapter general maintenance, use 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
Appendix A      Code Samples

Adapter Configuration

ragadapter.xml File

The sample ragadapter.xml file listed below defines two aggregations for the incoming NUR RDRs (intag="4042321920").

1. Aggregates RDRs with the RDR tag 71070 (outtag="71070") and persists them into CSV files (sinkid="csv1").
   The CSV file definition is located in the <sinks> section under the <csvsink id="csv1"> tag. This defines the output file name and directory. This also defines the rate at which new files are opened; either every five minutes, or when the current file reaches 1000 lines. Each field in the file is enclosed by quotes (").
   The buckets aggregate the data by two NUR fields: 0 and 2. These fields correspond to the SUBSCRIBER_ID and the SUBS_USG_CNT_ID. The buckets' identifiers are similar to the fields appearing in the SQL "GROUP BY" clause.
   There are two closures defined for these two fields. For field 0, if the SUBSCRIBER_ID is RonK, OmerT, or GuyM it is reported as GuyM. If the SUBSCRIBER_ID is NimrodR, YossiO, or LironL it is reported as OdedE. For field 2, if the SUBS_USG_CNT_ID is 5, 6, or 7 it is reported as 15. This functionality is similar to the Oracle SQL "DECODE" function.
   Fields 8, 9, and 10, which correspond to up, down, and sessions, are accumulated inside the buckets.
   Four monitors are then defined:
     – Two <maxmonitor> monitors are defined as checkpoints for the fields 8 (up) and 9 (down).
       When the accumulated value for either of these fields reaches 10,000, that bucket is flushed to the CSV file.
     – The third <timeoutmonitor> monitor is a checkpoint to flush all buckets to the file every 60 seconds.
     – The last <changemonitor> monitor triggers a warning to both the RAG adapter log and to the user log when an RDR arrives that contains a new value for DURATION (field 6).
2. Aggregates RDRs with the RDR tag 71071 (outtag="71071") and persists them into a database table (sinkid="dbsink1").
   The database definition is located in the <sinks> section under the <dbsink id="dbsink1"> tag.
   The table definition is located in the tables configuration file: tables.xml. The definition is of the following format:
   <ragtable ragsinkid="dbsink1" dbtablename="RPT_AGG_NUR" name="AGGNUR" ragouttag="71071"/>
   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 | metafield)+>
<!ELEMENT closures (closures)*>
<!ELEMENT closures (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
  field CDATA #REQUIRED
  active (true | false) #REQUIRED>

<!ELEMENT maxmonitor EMPTY>
<!ATTLIST maxmonitor
  field CDATA #REQUIRED
  maxvalue CDATA #REQUIRED
  active (true | false) #REQUIRED>

<!ELEMENT timeoutmonitor EMPTY>
<!ATTLIST timeoutmonitor
  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, per SCE" intag="4042321920" outtag="71071" sinkid="dbsink1">
      <bucketident>
        <field index="0" type="string"/>
<metafield source="record-source" type="int"/>
</bucketident>
<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>
</aggregations>
</aggregation>
</sinks>
<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>
Configuration Files Descriptions

This module describes each of the configuration files used by the CM along with the configuration parameters in each file.

Each configuration file is divided into sections. Each section is identified by a heading; for example, [config].

Any parameter that is not required in your configuration can be commented out by placing a # at the start of the line.

After changing any of the configuration file, the CM must be restarted to effect a change in the CM behavior.

Note

The configuration files supplied with the CM software installation contain tested valid settings. A user is not expected to change the settings. If it is necessary to change the configuration settings, care should be taken when making the changes.

General Configuration Files

This section describes the cm.conf and queue.conf configuration files.

- CM Configuration, page B-1
- Queue Configuration, page B-5

CM Configuration

This section describes the CM configuration file: cm.conf.

- [adapter] Section, page B-2
- [adapter_mem] Section, page B-2
- [adapter_start_timeout] Section, page B-3
- [buffer] Section, page B-3
- [queue] Section, page B-3
- [categorizer] Section, page B-4
- [server] Section, page B-4
- [collector] Section, page B-4
[csv_adapter] Section, page B-4

[adapter] Section

The [adapter] section contains the following parameters:

- **automatic_start**
  Defines whether starting or resetting the CM will start the adapters automatically.
  Possible values for this parameter are **true** and **false**. The default value is **true**.

- **port**
  Defines the adapter manager server port number. Possible values for this parameter are 1024-65535.
  The default value is **33001**.

- **timeout**
  Defines the timeout value in seconds for client adapters. Setting this parameter to 0 will disable the timeout.

- **connections**
  Defines the maximum number of client connections per adapter.
  Possible values for the parameter are 2-100. The default value is **20**.

- **directory**
  Defines the location to save buffered RDR data. You can use the ~/ prefix for this parameter. The default value is ~cm/adapters.

- **adapter.<number>**
  Defines the full class name of the adapter, where <number> is a unique id number. The adapter definitions must match the adapter_id parameters of the queues defined in the queue configuration file. See Queue Configuration, page B-5.

- **parestart_start**
  Defines the time in 24-hour format to start the periodic restart of the adapters. An example value for this parameter is **23:30**. The default value is **03:45**.

- **parestart_interval**
  Defines the interval in minutes between periodic adapter restarts. Possible values for this parameter are 0-2147483647 (0 to 2^{31} - 1), where 0 disables this functionality. The default value for this parameter is **10080** minutes (18 hours).

[adapter_mem] Section

The [adapter_mem] section contains the following parameter:

- **<Adapter class name>**
  Defines the maximum heap size allocated to an adapter. The following is an example of setting this parameter:

  ```
  com.cisco.scmscm.adapters.topper.TAAdapter=-Xmx512M
  ```

  **Note** To run with Solaris JRE 64-bit, you must set the -d64 flag.

  ```
  com.cisco.scmscm.adapters.topper.TAAdapter=-d64 -Xmx4096M
  ```
You must include a definition of the memory allocated for an adapter if the necessary heap size is different from JVM’s default maximum heap size.

---

**Note**
The maximum value of this parameter is 2 GB for an installation on a Linux platform and 3.5 GB for an installation on a Solaris platform. When running on Solaris JRE 64-bit you can set higher values for this parameter.

---

**[adapter_start_timeout] Section**

The **[adapter_start_timeout]** section contains the following parameter:

- `<Adapter class name>`

  Defines the number of seconds the CM should wait for a specific adapter to start. The default value is 200 seconds. The following is an example of setting this parameter:

  ```
  com.cisco.scmscm.adapters.topper.TAAdapter=600
  ```

---

**[buffer] Section**

The **[buffer]** section contains the following parameters:

- **size**
  
  Defines the minimum number of accumulated bytes in the queue before a buffer dequeue.
  
  Possible values for this parameter are 1-262144. The default value is 128000.

- **time**
  
  Defines the minimum number of milliseconds between buffer dequeues.
  
  Possible values for this parameter are 0-10000. The default value is 1050.

- **file_limit**
  
  Defines the maximum number of files to store in each adapter's queue.
  
  Possible values for this parameter are 1-49999. The recommended value is 10000.

---

**[queue] Section**

The **[queue]** section contains the following parameters:

- **upper_hysteresis**
  
  Defines the upper hysteresis level as a fraction of the total queue size.
  
  Possible values for this parameter range from 0-1. The default value is 0.11.

- **lower_hysteresis**
  
  Defines the lower hysteresis level as a fraction of the total queue size.
  
  Possible values for this parameter range from 0-1. The default value is 0.12.

- **config**
  
  Defines the queue configuration file. The file name defined for this parameter is the name of the queue configuration file described in *Queue Configuration, page B-5*. The default value is **queue.conf**.
Appendix B  Configuration Files Descriptions

General Configuration Files

[categorizer] Section

The [categorizer] section contains the following parameters:

- use_table
  Defines whether to categorize RDRs by using the table or by decoding the tag bits.
  Possible values for this parameter are true and false. The default value is true.
- rate_period_msec
  Defines the number of milliseconds between updates of incoming RDR rate calculation.
  Possible values for this parameter are 5000-30000. The default value is 5000.

[server] Section

The [server] section contains the following parameters:

- port
  Defines the RDR server port number.
  Possible values for this parameter are 1024-65535. The default value is 33000.
- connections
  Defines the maximum number of client connections.
  Possible values for this parameter are 1-50. The default value is 50.
- timeout
  Defines the client timeout in seconds. The default value is 600.
- allow_multi
  Defines whether to allow concurrent connections from the same IP.
  Possible values for this parameter are true and false. The default value is false.

[collector] Section

The [collector] section contains the following parameters:

- notification_level
  Defines the length of time in milliseconds between successive UM notifications.
  Possible values for this parameter are 0-60000. The default value is 10.
- log_directory
  Defines the directory to store log files. You can use the ~/ prefix for this parameter. The default value is ~/cm/logs.

[csv_adapter] Section

The [csv_adapter] section contains the following parameter:

- csvdir
  Defines the parent directory for CSV files. The default value for this parameter is ~/scmscm/cm/adapters/CSVAdapter/csvfiles.
Queue Configuration

This section describes the queue configuration file. The name of the queue configuration file is defined in the [queue] Section of the cm.conf configuration file.

Each section of the queue configuration file defines a queue. The section names, and hence the queue names, are all user defined. Each queue section has the following parameters:

- **adapter_id**
  Defines a unique adapter id number. The id number must match the adapters defined in the adapter.<number> parameter in the [adapter] section of the cm.conf configuration file. See [adapter] Section, page B-2.

- **priority**
  Defines the priority level of the queue. Possible values for this parameter are 1-3. This parameter is deprecated and should not be changed.

- **warning_size**
  Defines the number of RDRs that can be present in the queue before a warning is sent.

- **maximum_size**
  Defines the maximum size of the queue.

- **tags**
  Defines a list of RDR tag numbers that will be sent to this queue by the categorizer. The same tag can be sent to multiple queues.

The following is an example of a queue defined in the queue configuration file:

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

Adapter Configuration Files

This section describes the configuration files for all of the possible adapters.

- Topper/Aggregator Adapter, page B-5
- CSV Adapter, page B-9
- JDBC Adapter, page B-9
- RAG Adapter, page B-11

Topper/Aggregator Adapter

This section describes the TA adapter configuration file: taadapter.conf.

- [config] Section, page B-6
- [rdr] Section, page B-6
• [csv] Section, page B-7
• [state save] Section, page B-7
• [db] Section, page B-8
• [app] Section, page B-8

[config] Section

The [config] section contains the following parameters:

• num_top_entries
  Defines how many top entries are reported. The default value is 50.

• num_services
  Defines the maximum number of recognized services (including service 0). The default value is 32.

• all_services_index
  Provides an index that marks the "all subscribers" entry in the database/CSV files. The default value is -1. This parameter should not be changed.

• agg_lengths
  Defines the length in minutes of each aggregation cycle. Each cycle must be divisible by the preceding one, or it will be rounded on startup.

• time_base
  Defines the phase (time skew) of the aggregation cycles. The default value is 00:00.

[rdr] Section

The [rdr] section contains the following parameters:

• divide_1024
  Defines whether to divide all consumptions by 1024. Possible values for this parameter are true and false. The default value is false.

• table_name
  Specifies the logical name of the NUR table as it appears in tables.xml. The default value is NUR.

• field_subs
  Defines the name(s) of the subscriber-id field as it appears in tables.xml. The default value is SUBSCRIBER_ID.
• field_svc
  Defines the name(s) of the service counter id field as it appears in tables.xml. The default value is SERVICE_ID, SUBS_USG_CNT_ID.

• field_up
  Defines the name(s) of the upstream volume field as it may appear in tables.xml. The default value is UPSTREAM_VOLUME.

• field_down
  Defines the name(s) of the downstream volume field as it may appear in tables.xml. The default value is DOWNSTREAM_VOLUME.

• field_sess
  Defines the name(s) of the sessions field as it may appear in tables.xml. The default value is SESSIONS.

• field_sec
  Defines the name(s) of the seconds field as it may appear in tables.xml. The default value is SECONDS.

**[csv] Section**

The [csv] section contains the following parameters:

• active
  Defines whether to activate CSV file creation.
  Possible values for this parameter are true and false. The default value is true.

• file_name_format
  Defines the format for CSV file names. The default value is yyyy-mm-dd hh-mm-ss'.csv'.

• dir_name
  Defines the storage location for CSV files. The default value is ~/cm/adapters/TAAdapter/csvfiles.

• write_headers
  Defines whether to write a descriptive header at the top of each csv file.
  Possible values for this parameter are true and false. The default value is false.

• metric_separator
  Defines the character to separate the metrics in the CSV file. The default value is ‘,’.

• service_separator
  Defines the character to separate the services in the CSV file. The default value is ‘,’.

**[state save] Section**

The [state save] section contains the following parameters:

• active
  Defines whether to activate state saving/loading.
  Possible values for this parameter are true and false. The default value is true.

• file_name
Defines the name of the state file to be used for saving/loading. The default value is `taadapter.sav`.

**[db] Section**

The **[db]** section contains the following parameters:

- **name_of_total_subs**
  Defines the designation of "all subscribers" in the database.

- **db_template_dir**
  Defines the location of the main database configuration template relative to the CM config directory. The configuration file contains default values for each of the three possible databases; however, only one database can be configured at any one time. The possible values are:
  - Oracle: `dbpacks/oracle/9204e`
  - Sybase: `dbpacks/sybase/ase12.5.1`
  - MySQL: `dbpacks/mysql/4.0.20`

  The default value is `dbpacks/sybase/ase12.5.1`.

- **db_template_file**
  Defines the name of the main database configuration template. The default value is `main.vm`. This parameter should not be changed.

- **operations_timeout**
  Defines the maximum time in seconds allowed for the completion of database operations such as connecting, insertion, etc. The default value is **60**.

- **blocking_connects**
  Defines whether to block connection attempts to the database.

  Possible values for this parameter are **true** and **false**. The default value is **false**.

- **connection_timeout**
  Defines the number of seconds to allow connection attempts to the database. The default value is **900**.

  This parameter is not relevant if **blocking_connects** is set to **true**.

- **connection_wait_period**
  Defines the number of seconds to wait between connection attempts. The default value is **10**.

**[app] Section**

The **[app]** section contains the following parameters:

- **app_conf_file**
  Defines the name of the application configuration XML file. The default value is `dbtables.xml`.

- **app_dtd_file**
  Defines the name of the application configuration DTD file. The default value is `dbtables.dtd`.
• app_conf_dir
  Defines the location of the application configuration XML and DTD files relative to CM configuration directory. The default value is `apps/scasbh/3.5.0`.

**CSV Adapter**

This section describes the CSV adapter configuration file: `csvadapter.conf`.

**[csvadapter] Section**

The `[csvadapter]` section contains the following parameters:

- **maxCSVFiles**
  Defines the maximum number entries in a CSV file. When reaching this number the counting will wrap back to 1.
  Possible values for this parameter are 1-9999. The default value is **9999**.

- **rdrLimit**
  Defines the maximum number of RDRs that can be written to a single file.
  Possible values for this parameter are 1-1000. The default value is **1000**.

- **backupPreviousFiles**
  Defines whether to back up existing CSV files before overwriting them.
  Possible values for this parameter are `true` and `false`. The default value is `false`.

- **includeRecordSource**
  Defines whether to include the record source (SCE IP) in each RDR line.
  Possible values for this parameter are `true` and `false`. The default value is `false`.

- **escapeNonPrintables**
  Defines whether to escape non-printable characters.
  Possible values for this parameter are `true` and `false`. The default value is `false`.

**Note**

Setting `escapeNonPrintables` to `true` may negatively impact adapter performance.

**JDBC Adapter**

This section describes the JDBC adapter configuration file: `jdbcadapter.conf`.

- **[bench] Section, page B-9**

- **[db] Section, page B-10**

- **[app] Section, page B-10**

**[bench] Section**

The `[bench]` section contains the following parameter:
• rate_period_msec
  Defines the time period between insert rate updates in milliseconds. The default value is 5000.
  Set this parameter to 0 to disable the insertions.

[db] Section

The [db] section contains the following parameters:

• db_template_dir
  Defines the location of the main database configuration template relative to the CM config directory.
  The configuration file contains default values for each of the three possible databases; however, only
  one database can be configured at any one time. The possible values are:
  – Oracle: dbpacks/oracle/9204e
  – Sybase: dbpacks/sybase/ase12.5.1
  – MySQL: dbpacks/mysql/4.0.20
  The default value is dbpacks/sybase/ase12.5.1.

• db_template_file
  Defines the name of the main database configuration template. The default value is main.vm. This
  parameter should not be changed.

• batch_size
  Defines how many inserts can be made in a single batch. The default value is 10.

• transaction_size
  Defines how many batches can be made in a single transaction. The default value is 15.

• commit_interval
  Defines the maximum time in seconds between database commits. The default value is 6.

• operations_timeout
  Defines the maximum time in seconds allowed for the completion of database operations such as
  connecting, insertion, etc. The default value is 60.

• blocking_connects
  Defines whether to block connection attempts to the DB.
  Possible values for this parameter are true and false. The default value is true.

[app] Section

The [app] section contains the following parameters:

• app_conf_file
  Defines the name of the application configuration XML file. The default value is dbtables.xml.

• app_dtd_file
  Defines the name of the application configuration DTD file. The default value is dbtables.dtd.

• app_conf_dir
  Defines the location of the application configuration XML and DTD files relative to CM
  configuration directory. The default value is apps/scasbb/3.5.0.
RAG Adapter

This section describes the RAG adapter configuration file: `ragadapter.conf`.

- **[config] Section**, page B-11
- **[housekeeper] Section**, page B-11
- **[db] Section**, page B-11
- **[app] Section**, page B-12

[config] Section

The **[config]** section contains the following parameter:

- `xml_filename`
  Defines the location of the application configuration XML and DTD files that define the real-time aggregation to be done by the RAG adapter. The default value is `~/cm/config/ragadapter.xml`.

[housekeeper] Section

The **[housekeeper]** section contains the following parameter:

- `interval_sec`
  Defines the interval for the scheduled task to perform aggregation/sink operations. The default value is 10.

[db] Section

The **[db]** section contains the following parameters:

- `db_template_dir`
  Defines the location of the main database configuration template relative to the CM config directory. The configuration file contains default values for each of the three possible databases; however, only one database can be configured at any one time. The possible values are:
  - Oracle: `dbpacks/oracle/9204e`
  - Sybase: `dbpacks/sybase/ase12.5.1`
  - MySQL: `dbpacks/mysql/4.0.20`
  The default value is `dbpacks/sybase/ase12.5.1`.

- `db_template_file`
  Defines the name of the main database configuration template. The default value is `main.vm`. This parameter should not be changed.

- `batch_size`
  Defines how many inserts can be made in a single batch. The default value is 10.

- `transaction_size`
  Defines how many batches can be made in a single transaction. The default value is 15.

- `commit_interval`
  Defines the maximum time between database commits. The default value is 6.
• **operations_timeout**
  Defines the maximum time allowed for the completion of database operations such as connecting, insertion, etc. The default value is 60.

• **blocking_connects**
  Defines whether to block connection attempts to the DB.
  Possible values for this parameter are **true** and **false**. The default value is **true**.

### [app] Section

The [app] section contains the following parameters:

• **app_conf_file**
  Defines the name of the application configuration XML file. The default value is **dbtables.xml**.

• **app_dtd_file**
  Defines the name of the application configuration DTD file. The default value is **dbtables.dtd**.

• **app_conf_dir**
  Defines the location of the application configuration XML and DTD files relative to CM configuration directory. The default value is **apps/scasbb/3.5.0**.