



Collection Manager User Guide

Ver. 2.5.5

OL-138638-01

Corporate 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: 408 526-4100

Customer Order Number: DOC-138638=
Text Part Number: OL-138638-01



THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The following information is for FCC compliance of Class A devices: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

The following information is for FCC compliance of Class B devices: The equipment described in this manual generates and may radiate radio-frequency energy. If it is not installed in accordance with Cisco's installation instructions, it may cause interference with radio and television reception. This equipment has been tested and found to comply with the limits for a Class B digital device in accordance with the specifications in part 15 of the FCC rules. These specifications are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

Modifying the equipment without Cisco's written authorization may result in the equipment no longer complying with FCC requirements for Class A or Class B digital devices. In that event, your right to use the equipment may be limited by FCC regulations, and you may be required to correct any interference to radio or television communications at your own expense.

You can determine whether your equipment is causing interference by turning it off. If the interference stops, it was probably caused by the Cisco equipment or one of its peripheral devices. If the equipment causes interference to radio or television reception, try to correct the interference by using one or more of the following measures:

- Turn the television or radio antenna until the interference stops.
- Move the equipment to one side or the other of the television or radio.
- Move the equipment farther away from the television or radio.
- Plug the equipment into an outlet that is on a different circuit from the television or radio. (That is, make certain the equipment and the television or radio are on circuits controlled by different circuit breakers or fuses.)

Modifications to this product not authorized by Cisco Systems, Inc. could void the FCC approval and negate your authority to operate the product.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCSP, the Cisco Square Bridge logo, Follow Me Browsing, and StackWise are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, and iQuick Study are service marks of Cisco Systems, Inc.; and Access Registrar, Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, IQ Expertise, the IQ logo, IQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, Packet, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StrataView Plus, SwitchProbe, TeleRouter, The Fastest Way to Increase Your Internet Quotient, TransPath, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0501R)

Printed in the USA on recycled paper containing 10% postconsumer waste.

Collection Manager User Guide ver. 2.5.5

Copyright © 2002-2005 Cisco Systems, Inc.
All rights reserved.



Preface v

- Audience v
- Purpose v
- Document Content vi
- Document Conventions vi
- Related Publications vi
- Obtaining Technical Assistance vii
 - Cisco TAC Website vii
 - Opening a TAC Case vii
 - TAC Case Priority Definitions vii

Overview 1-1

- The Cisco Service Control Concept 1-1
 - Service Control for Wireless Service Providers 1-2
 - Service Control for DSL Providers and ISPs 1-2
 - Service Control for Cable MSOs 1-2
- The SCE Platform 1-3
- Service Control Capabilities 1-4
- Management and Collection 1-4
 - Network Management 1-5
 - Subscriber Management 1-6
 - Service Configuration Management 1-6
 - Collection 1-6
- Cisco Service Control Specific Solutions 1-6
 - Service Control Application Suite for Broadband 1-7
 - Service Control Application Suite for Mobile 1-7

How the Collection Manager Works 2-1

- The Data Collection Process 2-1

- Raw Data Records and Reports 2-2
- The Collection Manager Software Package 2-3
 - RDR Server 2-3
 - Categorizer 2-3
 - Priority Queues and Persistent Buffers 2-4
- Adapters 2-4
 - Database Adapter 2-4
 - JDBC Adapter 2-4
 - Comma-Separated Value Adapter 2-4
 - Topper/Aggregator Adapter 2-5
 - Real-Time Aggregating Adapter 2-7
- The Bundled Database 2-12
- Using an External Database 2-13

Installing the CM and Getting Started 3-1

- System Requirements 3-1
 - Solaris Requirements 3-1
 - Redhat Linux Requirements 3-4
- CD Content 3-5
- Installation 3-5
 - Phase 1: Installing Sybase 3-6
 - expert Option: Usage Guidelines 3-7
 - Phase 2: Installing Service Control Software 3-8
 - Changes Made by installsyb.sh 3-10
 - Changes Made by install-dc 3-11
 - Changes Made by select-app.sh 3-11
 - Ports Used by the Collection Manager Software 3-11
 - Uninstalling the Sybase Database and the Service Control Software 3-12
- Getting Started 3-13
- Default Configuration Settings 3-13

Managing the Collection Manager 4-1

- Using Utility Scripts 4-1
- Configuring the Collection Manager 4-2
 - Activating the Servers 4-2

- Controlling the Adapters 4-3
- Controlling the Database 4-5
- Dropping an SCE Connection 4-5
- Monitoring the Collection Manager 4-5
 - Checking the Database Capacity 4-6
 - Checking the RDR Rate 4-6
 - Checking the SCE Connection 4-7
- Verifying that the Server is Operational 4-7

Managing the Database and CSV Repository 5-1

- Managing the Database 5-1
 - Listing the Database Tables 5-2
- Managing the CSV Repository 5-8
 - CSV Repository File Structure 5-8

Database Configuration 6-1

- Quick Start Guide 6-2
- The Template Language 6-4
- Configuration Files 6-5
 - Context Objects 6-5
 - Application Configuration 6-7
- A Working Example 6-11
 - Macro Definitions 6-11
 - dbinfo Configuration 6-12
 - SQL Definitions 6-12
- Testing and Debugging 6-13
 - Parsing a String 6-14
 - Obtaining Full Debug Information 6-14
- Using the JDBC Framework in Scripts 6-14
 - Example-Viewing and Setting the SCE Timezone Offset 6-15
- Scalability-Hints for Oracle 6-16
 - Using Custom tablespaces 6-16
 - Using Table Partitioning 6-17

[Glossary of Terms 1](#)

[Index 1](#)



Preface

This guide contains instructions for installing and running the Collection Manager. It also includes relevant information about the commercial database (currently, a Sybase™ Database) that is installed on the same machine when using the *bundled* option.

This guide assumes a basic familiarity with the concept of the Service Control Solution concept, the SCE Platforms, and related components.

Audience

This guide is intended for the networking or computer technician who is responsible for the on-site installation and configuration of the Collection Manager. It is also intended for the operator who will be responsible for the daily operations of the Collection Manager, facilitating Service Provider enhancements in a subscriber-oriented environment.

Purpose

The *Collection Manager User Guide* documents the software that processes the Raw Data Records (RDRs) sent from the SCE Platforms, and the database that can be used to store the RDRs. Alternatively, RDRs can be sent to third party systems.

Document Content

This manual contains the following chapters:

Chapter 1: *Overview* provides a functional overview of the Service Control solution.

Chapter 2: *How the Collection Manager Works* provides detailed information about the functionality of the Collection Manager components.

Chapter 3: *Installing the CM and Getting Started* describes the procedures for installing the Server Platform (that hosts the Collection Manager modules) on-site and details how to start running the Collection Manager.

Chapter 4: *Managing the Collection Manager* discusses the methods of viewing and updating Collection Manager parameters and information, in particular via utility scripts.





Chapter 5: *Managing the Database* discusses how to manage the Collection Manager database, as well as the CSV repository, using utility scripts.

Chapter 6: *Database Configuration* explains how to configure the Collection Manager to work with your database.

Glossary: Brief description of Collection Manager terms.

Document Conventions

The following typographic conventions are used in this guide:

Typeface or Symbol	Meaning
<i>Italics</i>	References, new terms, field names, and placeholders.
Bold	Names of menus, options, and command buttons.
Courier	System output shown on the computer screen in the Telnet session.
Courier Bold	CLI code typed in by the user in examples.
<i>Courier Italic</i>	Required parameters for CLI code.
[<i>italic in brackets</i>]	Optional parameters for CLI code.
	Note.
	Notes contain important information.
	Warning.
	Warning means danger of bodily injury or of damage to equipment.

Related Publications

The *Collection Manager User Guide* should be used in conjunction with SCE Platform user guides (*SCE 1000 User Guide*, *SCE 2000 User Guide*) and the other Management Suite user guides (*Service Control Application Suite for Broadband User Guide*, *Service Control Application Suite for Broadband Installation Guide*, and *Service Control Application Suite for Broadband API Programmer's Guide*).

Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, the Cisco Technical Assistance Center (TAC) provides 24-hour, award-winning technical support services, online and over the phone. Cisco.com features the Cisco TAC website as an online starting point for technical assistance.

Cisco TAC Website

The Cisco TAC website (<http://www.cisco.com/tac> (<http://www.cisco.com/tac>)) provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The Cisco TAC website is available 24 hours a day, 365 days a year.

Accessing all the tools on the Cisco TAC website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a login ID or password, register at this URL:

<http://tools.cisco.com/RPF/register/register.do> (<http://tools.cisco.com/RPF/register/register.do>)

Opening a TAC Case

The online TAC Case Open Tool (<http://www.cisco.com/tac/caseopen> (<http://www.cisco.com/tac/caseopen>)) is the fastest way to open P3 and P4 cases. (Your network is minimally impaired or you require product information). After you describe your situation, the TAC Case Open Tool automatically recommends resources for an immediate solution.

If your issue is not resolved using these recommendations, your case will be assigned to a Cisco TAC engineer.

For P1 or P2 cases (your production network is down or severely degraded) or if you do not have Internet access, contact Cisco TAC by telephone. Cisco TAC engineers are assigned immediately to P1 and P2 cases to help keep your business operations running smoothly.

To open a case by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)

EMEA: +32 2 704 55 55

USA: 1 800 553-2447

For a complete listing of Cisco TAC contacts, go to this URL:

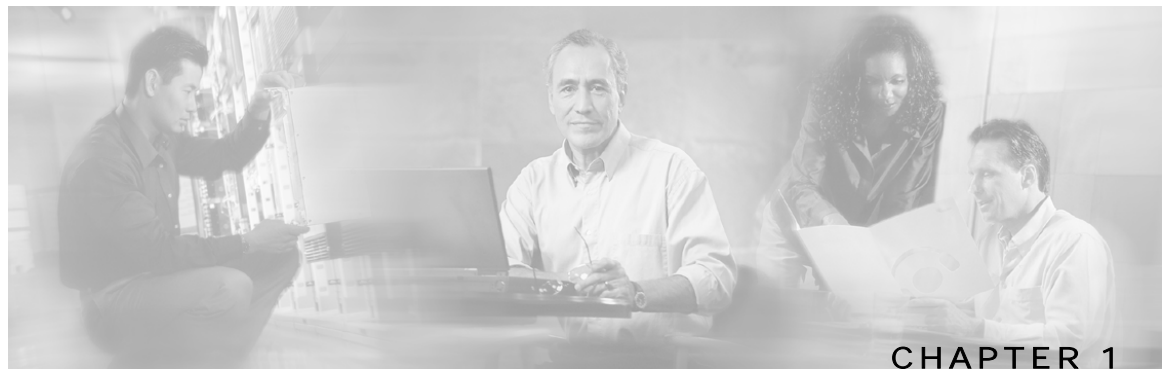
<http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml>
(<http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml>)

TAC Case Priority Definitions

To ensure that all cases are reported in a standard format, Cisco has established case priority definitions.

- **Priority 1 (P1)**—Your network is “down” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

- **Priority 2 (P2)**—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.
- **Priority 3 (P3)**—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.
- **Priority 4 (P4)**—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.



Overview

This chapter provides a general overview of the Cisco Service Control solution. It introduces the Cisco Service Control concept and the Service Control capabilities. It also briefly describes the hardware capabilities of the SCE Platform, as well as the Cisco specific applications that together compose the total Cisco Service Control solution.

This chapter contains the following sections:

- [The Cisco Service Control Concept](#) 1-1
- [The SCE Platform](#) 1-3
- [Service Control Capabilities](#) 1-4
- [Management and Collection](#) 1-4
- [Cisco Service Control Specific Solutions](#) 1-6

The Cisco Service Control Concept

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

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

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

Service Control for Wireless Service Providers

Wireless Service Providers are successfully rolling out 2.5G and 3G-based data services to their subscribers.

These services are expected to significantly increase much needed Average Revenue Per User (ARPU) for sustained business models and rapid rollout of new services. These data services require new ways of service offering and new ways of billing these services to the subscribers. The Cisco Service Control solutions enable:

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

Service Control for DSL Providers and ISPs

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

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

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

Service Control for Cable MSOs

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

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

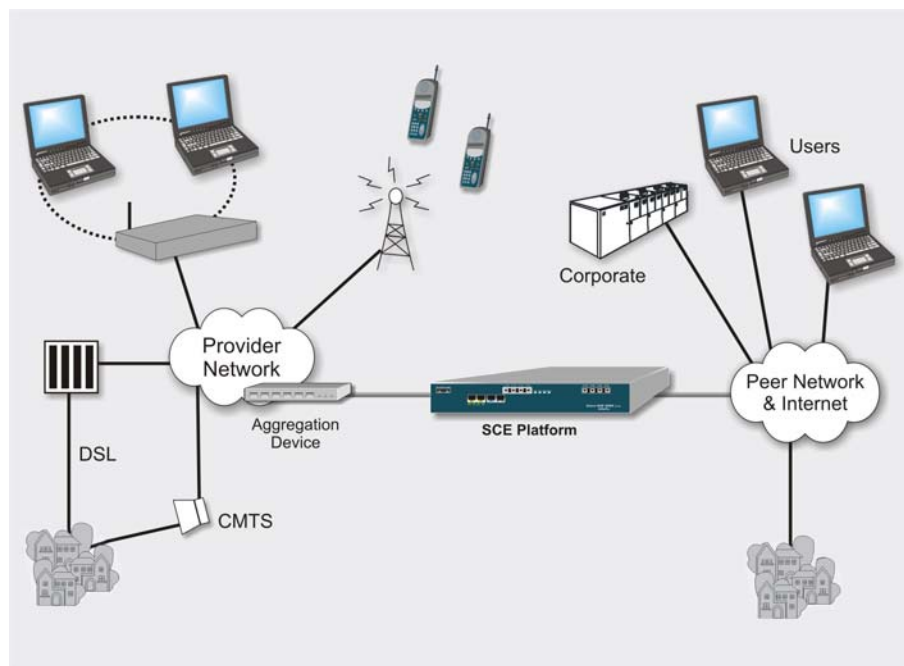
The SCE Platform

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

- Layer 7-3 stateful wire-speed packet inspection and classification
- Robust support for over 600 protocol/applications including:
 - General: HTTP, HTTPS, FTP, TELNET, NNTP, SMTP, POP3, IMAP, WAP, and others
 - P2P: FastTrack-KazaA, Gnutella, WinMX, Winny, Hotline, eDonkey, DirectConnect, Piolet, and others
- Streaming & Multimedia: RTSP, SIP, HTTP-STREAMING, RTP/RTCP, and others
- Programmable system core for flexible reporting and bandwidth control
- Transparent network and BSS/OSS integration into existing networks
- Subscriber awareness for relating traffic and usage to specific customers

The following diagram demonstrates a deployment of an SCE Platform in the network.

Figure 1-1: SCE Platform in the Network



Service Control Capabilities

At the core of the Cisco Service Control Platform stands the purpose-built network hardware device: the Service Control Engine (SCE). Implementing a complete Service Control solution requires that the Service Control Engine provide certain functionalities and capabilities. The following are the core capabilities of the Cisco Service Control Engine, which support a wide range of applications for delivering Service Control solutions:

- **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 for maintaining the state of each subscriber transmitting traffic through the platform, and enforcing the appropriate policy on this subscriber traffic.

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

For an application protocol that is 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.

- **Stateful, real time traffic control:** The ability to perform advanced control functions, including granular BW metering and shaping, quota management and redirection, utilizing stateful real-time traffic transaction processing. This requires highly adaptive protocol and application level intelligence.
- **Programmability:** The ability to quickly add new protocols and easily adapt to new services and applications in the ever-changing service provider environment. Programmability is achieved using the SML language.

Programmability means that new services can 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 3rd party systems at the Service Provider, such as provisioning systems, subscriber repositories, billing systems, and OSS systems. The Service Control Engine provides a set of open and well-documented APIs that allows a quick and robust integration process.
- **Scalable High-Performance Service Engines:** The ability to execute all operations described above at wire speed.

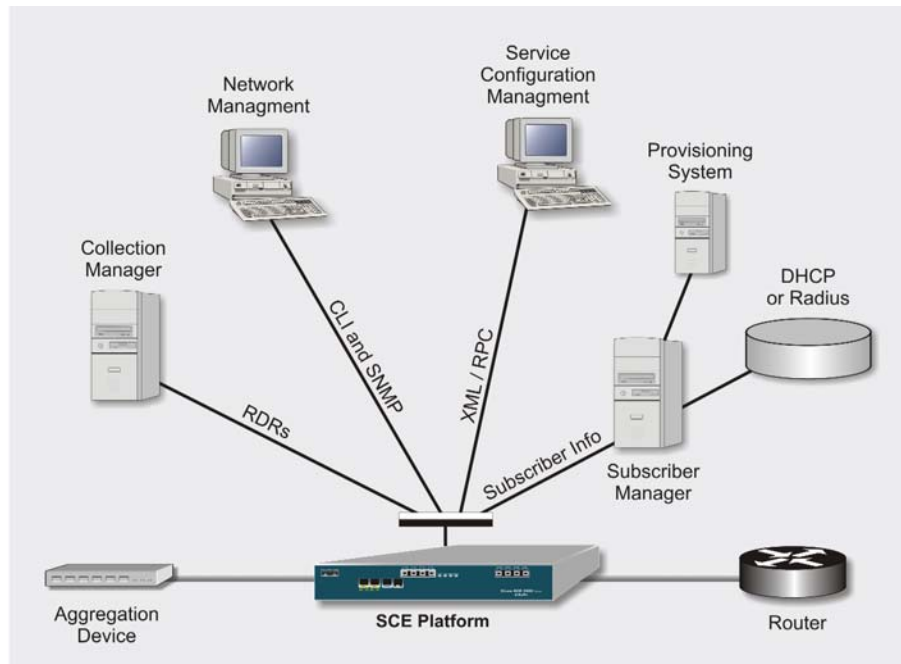
Management and Collection

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

- Network management
- Subscriber management
- Service Control Management

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

Figure 1-2: Service Control Management Infrastructure



Network Management

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

Two interfaces are provided for network management:

- **CLI** (Command Line Interface). The CLI is accessible through the Console port or through a Telnet connection.

CLI is used for configuration and security functions.

- **SNMP** (Simple Network Management Protocol).

SNMP provides fault management via SNMP traps, as well as performance monitoring functionality.

Subscriber Management

The smartSUB Manager (SM) is a middleware software component used for bridging between the OSS and the SCE Platform(s). Subscriber information is stored in the SM database and can then be distributed between multiple devices according to actual subscriber placement.

The SM provides subscriber awareness, mapping network IDs to subscriber IDs. It obtains subscriber information using dedicated integration modules, which integrate with AAA devices like Radius or DHCP servers.

Subscriber information may be introduced into the SCE platform in one of two ways:

- Push Mode: The SM pushes subscriber information to the SCE Platform automatically upon logon of a subscriber.
- Pull Mode: On-demand, in response to a query from the SCE Platform to the SM.

Service Configuration Management

Service configuration management is the ability to configure the general service definitions of a Service Control application. Service Configuration is performed by creating an XML file and then applying it onto the SCE Platform using the Service Configuration utilities and management commands. This XML based approach is simple to use and easy to automate.

Collection

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

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

- Transaction RDRs: Records generated for each transaction, where a transaction is a single event detected in network traffic. The identification of a transaction will depend on the particular application/protocol.
- Subscriber 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.

Cisco Service Control Specific Solutions

Cisco provides two specific solutions that run on top of the SCE Platform. Each solution addresses a different IP network control challenge that service providers face.

The Cisco specific solutions are:

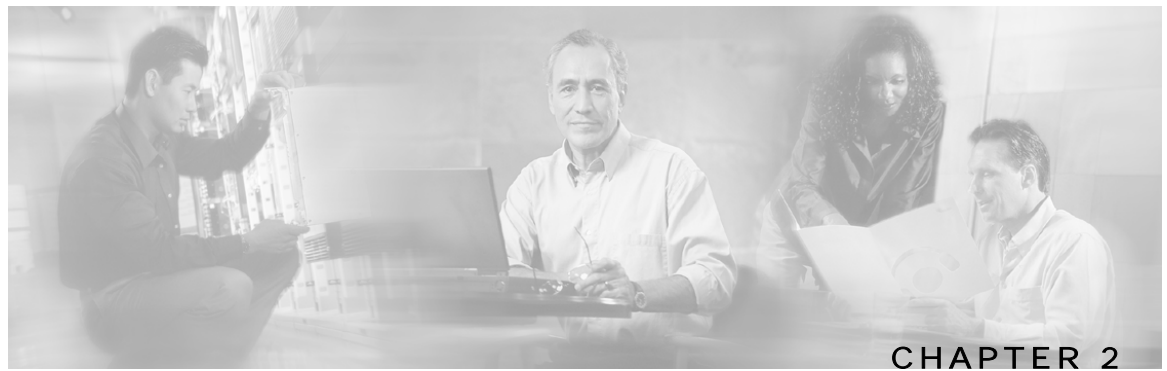
- *Service Control Application Suite for Broadband*
- *Service Control Application Suite for Mobile*

Service Control Application Suite for Broadband

The *Service Control Application Suite for Broadband* allows service providers to detect complex and evasive network application protocols (such as P2P), and to control them as per their business and service delivery requirements. It also enables the creation of differentiated tiered services that the service provider uses to boost revenues and provide competitive services to end customers. *Service Control Application Suite for Broadband*'s programmable application detection and subscriber awareness makes tiered service possible from one central point in the network. The *Service Control Application Suite for Broadband* requires no network changes or upgrades, and is compatible with all existing IP network switches, routers, and infrastructure.

Service Control Application Suite for Mobile

In this solution the SCE Platform is instrumental as a real-time post- and pre-paid network billing and traffic control device. It implements post-paid and pre-paid billing plans that relate subscriber access and network bandwidth consumption. The *Service Control Application Suite for Mobile* solution tracks detailed user specific traffic/application metrics and applies service and quota controls depending on their pre-paid balances.



How the Collection Manager Works

This chapter describes how the Collection Manager works. It describes the Raw Data Records (RDRs) that the Collection Manager produces and provides an overview of the components of the Collection Manager software package.

This chapter contains the following sections:

- [The Data Collection Process](#) 2-1
- [Raw Data Records and Reports](#) 2-2
- [The Collection Manager Software Package](#) 2-3
- [Adapters](#) 2-4
- [The Bundled Database](#) 2-12
- [Using an External Database](#) 2-13

The Data Collection Process

SCE Platforms create Raw Data Records (RDRs). The specifications of the RDRs are defined by the application running on the SCE Platform, such as **Service Control Application Suite for Broadband**.

The Collection Manager (CM) collects the Raw Data Records (RDRs) that are produced by the SCE Platforms, and processes them using one or more of its adapters:

- Database Adapter
- JDBC Adapter
- Comma-Separated Value Adapter
- Topper/Aggregator Adapter

Each adapter performs a specific function on RDRs (stores it in a CSV formatted file on a local machine, sends it to a RDBMS application or performs custom operations). For more information, see *Adapters* (on page [2-4](#)).

Raw Data Records and Reports

Raw Data Records (RDRs) are usage reports produced by the SCE Platforms. The list of RDRs, their fields, and their semantics, depend on the specific SCP (Service Control Protocol) application. The following are some examples of possible RDRs produced by SCP applications:

- **Periodic Subscriber usage report:** Since the SCE Platform is a subscriber-aware network device, it can report usage records per subscriber.

These RDRs would 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), etc. These types of usage reports are necessary for usage-based billing services, or for network analysis and capacity planning.

The *Service Control Application Suite for Broadband* application generates RDRs of this type, called Subscriber Usage RDRs.

- **Transaction level report:** The SCE Platforms perform stateful tracking of each network transaction conducted on the links on which they are situated. Using this statefulness, the SCP tracks a number of OSI layer-7 protocols (such as HTTP, RTSP, SIP, GNUTELLA) to report on various application level attributes.

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

The *Service Control Application Suite for Broadband* application generates RDRs of this type, called Service Access Reports.

- **SCP application activity reports:** The SCP application can program the SCE Platform to perform various actions on network traffic. This includes such actions as blocking transactions, shaping traffic to certain rates and limits, performing application level redirections etc. In different applications, when such an operation is performed, the SCP application may produce a report.

The *Service Control Application Suite for Broadband* application Breaching and Blocking RDRs are of this category. Breaching RDRs are generated when the system changes its active enforcement on a subscriber (due to a usage exceeding a certain quota). Blocking RDRs are generated when the SCE Platform blocks a network transaction (as indicated in its rule-base).

Each RDR type has a unique ID or RDR-tag that can be used by the collection system to understand how to interpret it and its list of fields. RDRs are streamed from the SCE Platform using a simple and reliable protocol called the *RDR-Protocol*. Integration with Service Control solution for the collection of data records involves implementing RDR-Protocol support in the collection system (a simple and easy development process).

The various types of RDRs are recognized and sorted, based on pre-set categories and according to type and priority, by the CM software modules. You can use the utility scripts for determining many of the parameters that influence the behavior of the CM.

The CM collects the RDRs generated by the SCE Platforms via the RDR Server and pipes the RDRs through the various components. The Categorizer (see *Categorizer* (on page 2-3)) classifies the RDRs according to their RDR tag and then places them in Priority Queues. The RDRs are then stored in local Persistent Buffers to assure that they are processed by the system, even in cases of hardware, software or power failures. From the persistent buffers, they are channeled to the appropriate Adapter module for further processing: to be stored in text files (CSV format), to be stored in a local or remote database, or to be sent to a third party system.

The CM components are described in detail later in this chapter.

The Collection Manager Software Package

The Collection Manager Software Package consists of a group of processing and sorting modules, as shown in the figure below. These include the following components:

- **RDR Server:** Accepts the RDRs forwarded by the SCE Platform(s).
- **Categorizer:** Separates the RDRs into categories and maps them to the appropriate Adapter.
- **Priority Queue:** Queues the RDRs by priority level before storing them in a Persistent Buffer.
- **Persistent Buffer:** A per-adapter non-volatile storage area that holds the RDRs until they are processed by the Adapter(s).

The data is continually piped through the Collection Manager modules and stored in the Persistent Buffer.

RDR Server

The RDR Server processes the incoming RDRs as they arrive from the connected SCE Platforms. The RDR Server adds to each RDR the timestamp for when it arrived and the ID of the source SCE Platform. The RDRs are then sent to the Categorizer.

Categorizer

The Categorizer identifies the type of RDR based on their tag numbers. It decides the destination Adapter(s) for each RDR and through which Priority Queue the RDR should be sent. An RDR can be mapped to more than one Adapter. Based on your requirements, this flow will be configured by a qualified technician, through the configuration file.

Priority Queues and Persistent Buffers

Each Adapter has one or more specific Priority Queues and a specific Persistent Buffer assigned to it. The RDRs are stored in the Adapter's Persistent Buffer until the Adapter processes them. Note that RDRs that are already stored in the Persistent Buffer will not be lost if there is an electricity outage or a network failure.

Adapters

Adapters are software modules that translate the RDRs to match the target system's requirements and disseminate them upon request. At this time four Adapters are shipped with the system:

- Database (DB) Adapter
- JDBC Adapter
- Comma-Separated Value (CSV) Adapter
- Topper/Aggregator (TA) Adapter

Database Adapter

The Database (DB) Adapter accepts records, processes them, and stores them in the internal database.

This adapter is designed for compatibility with the Sybase Database when the latter is bundled with the Service Control Collection Manager.

JDBC Adapter

The Database Adapter accepts records, processes them, and stores them in a database.

This adapter is designed for compatibility with any database server that is JDBC compliant (Java Database Connectivity), and transforms the records accordingly. The JDBC Adapter can be configured to utilize a database operating on a remote machine.

The JDBC Adapter can be used as an alternative to the DB Adapter even when using the bundled Sybase database.

Comma-Separated Value Adapter

The Comma-Separated Value Adapter writes RDRs to files on the disk in comma-separated value format. These records can also be retrieved by a service provider's OSS or by a third party billing system, via standard mechanisms such as FTP, in order to generate enhanced accounting and network traffic analysis records.

Specific parameters for this adapter, such as the length of an RDR CSV file or whether to keep backups of old CSV files, are managed through the *csvconf.sh* utility script or using the *csvadapter.conf* file (See *Managing the CSV Repository* (on page 5-8).)

Topper/Aggregator Adapter

The Topper/Aggregator adapter collects usage information for Subscriber Usage RDRs as they arrive from the SCE. It stores this information in memory, aggregates it, and periodically outputs statistics to the database and/or CSV files.

The information that the Topper/Aggregator adapter outputs to the DB is formatted as Top Reports, which can later be presented by the Reporter. The information that the Topper/Aggregator adapter outputs to CSV files is the aggregated daily statistics of all subscribers, not just the top consumers.

The Topper/Aggregator adapter uses the following metrics:

- For the DB Top Reports: Up Volume, Down Volume, Combined Volume, Sessions, Seconds
- For the CSV files: Up Volume, Down Volume, Sessions, Seconds

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

The adapter can be configured to utilize any JDBC-compliant database, either locally or remotely, similarly to the JDBCAdapter

Topper/Aggregator Cycles

The adapter works in two cycles, a short cycle and a long cycle. Cycles are fixed intervals at which time the adapter outputs its aggregated information to the DB and CSV adapters. The interval for the short cycle is one hour, while for the long cycle it is 24 hours (every day at midnight). The activities in each cycle differ slightly, as follows:

- Short Cycle: At every short cycle period, the adapter does the following:
 - Adds the last cycle's aggregated Top Reports to the short cycle DB table
 - Saves the current state file in case of power failure
- Long Cycle: At every long cycle period, the adapter does the following:
 - Performs the action of the short cycle
 - Creates a CSV file containing all the day's aggregated statistics

Topper/Aggregator Adapter CSV Files

As previously mentioned, at the end of every long cycle, the adapter can generate a CSV file. The name of the CSV file is the date and time of its creation. The default format of the file name is `yyyy-MM-dd_HH-mm-ss.csv` (for example, `2003-10-27_18-30-01.csv`). The location of the CSV files by default is `~/pump/adapters/TAAdapter/csvfiles`.

**Note:**

In order to ensure that local disk space does not run out, an external process must periodically delete these created files. For information on the periodic deletion of database records available with the bundled option, see *Configuring the Periodic Deletion of Old Records* (on page 5-5).

By default, the fields in each row of the CSV file are as follows:

```
TIMESTAMP, TAG, subsID, svcALLup, svcALLdown, svcALLsessions,
svcALLseconds, svc0up, svc0down, svc0sessions, svc0seconds, svc1up,
svc1down, svc1sessions, svc1seconds ..., ..., ..., svcNup, svcNdown,
svcNsessions, svcNseconds
```

Where `subsID` is the Subscriber ID and `svcXY` is the aggregated volume of metric Y for service X. (The N in `svcN` is the highest service number, which is the configured number of services minus 1.)

The adapter can optionally include in the beginning of each line, a field indicating the "Record Source", i.e. the encoded IP address of the SCE engine that sent the line's RDR. To turn this option on, edit the file `csvadapter.conf`, change the value of `includeRecordSource` to true, and restart the CM.

Note that the sum of upstream and downstream volume is not stored in the CSV file, since it is easily obtained by adding metrics 0 and 1.

The adapter can be configured to insert a descriptive comment at the beginning of every CSV file. The comment can contain a timestamp when it was created and an explanation of its format. This feature is disabled by default.

Handling of Database Tables

As mentioned above, the Topper/Aggregator Adapter periodically outputs a list of the top subscribers in different metrics (for example, the top 50 volume or session consumers in the last hour).

This information is stored in the following DB tables

- RPT_TOPS_PERIOD0 for the hourly reports
- RPT_TOPS_PERIOD1 for the daily reports

The tables are created in the same location in the DB as all the DB Adapter tables, and may be manipulated by the standard CM table operations and scripts, such as the Periodic Delete mechanism (see *Configuring the Periodic Deletion of Old Records* (on page 5-5)), the `dbtables.sh` script, and the `droptable.sh` script.

The structure of the tables depends on the SCE Platform application being used and is in the application documentation.

The adapter ensures that in each Top Report in the DB, the actual subscriber/consumption pairs are ordered from the highest consumption to lowest. This is followed by the "(total)" statistic, which is the sum of all subscribers for this metric.

In case of an empty report, typically when there was no traffic reported for the designated service/metric pair during the aggregation period, the DB will still be updated, but the only row in the report will be the (total) row with a consumption of 0. The reason the DB is updated is to avoid the perception in the Reporter that the report is not there due to a malfunction.

The reported metrics are as follows:

- Metric 0 = Up Volume
- Metric 1 = Down Volume
- Metric 2 = Combined Volume

- Metric 3 = Sessions
- Metric 4 = Seconds

Real-Time Aggregating Adapter

The RAG Adapter processes RDRs of one or more particular tags and accumulates the data from specific field positions into buckets. The accumulation is additive on pre-designated fields. An aggregation bucket is indexed by combination of values from other fields in the RDR, where 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 can be defined that will be performed if 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.

Flushing a Bucket

The trigger for flushing a bucket (a “checkpoint”) will be the earliest occurrence of any of the following:

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

When a bucket is flushed, it is written as a CSV line into a file in the adapters CSV repository. A CSV file will be named according to its initial creation date, similarly to the way the TA Adapter names its CSV files. The trigger to close a CSV file will be the earliest occurrence of one of the following:

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

Real-Time Aggregating Adapter CSV Files

Each line output to the CSV file may have some synthesized fields added to it, such as timestamps of first and last RDRs that contributed to this bucket and the total number of RDRs in this bucket. Similarly, other fields may be removed altogether.

The CSV repository will be flat (all CSV files in one directory). Alternatively, the adapter can be configured to use a subdirectory structure where each subdirectory is named as the tag that is written in it, similar to the operation of the CSVAdapter.

Fields in the output line that were not used for aggregation will have values corresponding to the values in the first RDR that contributed to the bucket. However, the timestamp field prepended to the bucket in the CSV file will have a value corresponding to the timestamp of the last RDR in the bucket.

Real-Time Aggregating Adapter Configuration Example

Following is an example of the configuration possibilities of the RAG Adapter.

```
<?xml version="1.0"?>
<!DOCTYPE ragadapterconf [
  <!ELEMENT ragadapterconf (fileversion, config)>
  <!ELEMENT fileversion (#PCDATA)>
  <!ELEMENT config (aggregations, sinks)>
  <!ELEMENT aggregations (aggregation+)>
  <!ELEMENT aggregation (bucketident, closures, accumulators, monitors)>
  <!ATTLIST aggregation
    id CDATA #REQUIRED
    intag CDATA #REQUIRED
    outtag CDATA #REQUIRED
    sinkid CDATA #REQUIRED
  >
  <!ELEMENT bucketident (field+)>
  <!ELEMENT closures (closure*)>
  <!ELEMENT closure (closurespec+)>
  <!ATTLIST closure
    field CDATA #REQUIRED
  >
  <!ELEMENT closurespec (equivvalue+)>
  <!ATTLIST closurespec
    type (string | int | long | double) #REQUIRED
    primaryvalue CDATA #REQUIRED
  >
  <!ELEMENT equivvalue EMPTY>
  <!ATTLIST equivvalue
    val CDATA #REQUIRED
  >
  <!ELEMENT accumulators (field+)>
  <!ELEMENT monitors (changemonitor | maxmonitor | timeoutmonitor)*>
  <!ELEMENT changemonitor EMPTY>
  <!ATTLIST changemonitor
    action (warn | checkpoint) #REQUIRED
    field CDATA #REQUIRED
    active (true | false) #REQUIRED
  >
  <!ELEMENT maxmonitor EMPTY>
  <!ATTLIST maxmonitor
    action (warn | checkpoint) #REQUIRED
    field CDATA #REQUIRED
    maxvalue CDATA #REQUIRED
    active (true | false) #REQUIRED
  >
  <!ELEMENT timeoutmonitor EMPTY>
  <!ATTLIST timeoutmonitor
    action (warn | checkpoint) #REQUIRED
    maxsec CDATA #REQUIRED
    active (true | false) #REQUIRED
  >
  <!ELEMENT field EMPTY>
  <!ATTLIST field
    index CDATA #REQUIRED
    type (string | int | long | double) #REQUIRED
  >
  <!ELEMENT sinks (csvsink | dbsink | generalsink)+>
  <!ELEMENT csvsink EMPTY>
  <!ATTLIST csvsink
    id CDATA #REQUIRED
    classname CDATA #REQUIRED
```

```

        filenameformat CDATA #REQUIRED
        dirname CDATA #REQUIRED
        maxagesec CDATA #REQUIRED
        maxlines CDATA #REQUIRED
        usequotes (true | false) #REQUIRED
        active (true | false) #REQUIRED
    >
    <!ELEMENT dbsink EMPTY>
    <!ATTLIST dbsink
        id CDATA #REQUIRED
        classname CDATA #REQUIRED
        active (true | false) #REQUIRED
    >
    <!ELEMENT generalsink EMPTY>
    <!ATTLIST generalsink
        id CDATA #REQUIRED
        classname CDATA #REQUIRED
        active (true | false) #REQUIRED
    >
]>
<ragadapterconf>
    <fileversion>
        $File: //depot/App/DC/V2.5.5/pump/config/ragadapter.xml $
$Revision: #2 $
        $Author: ronv $
        $DateTime: 2004/12/13 11:35:28 $
    </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">
                        <closurestypename type="string"
primaryvalue="GuyM">
                            <equivvalue val="RonK"/>
                            <equivvalue val="OmerT"/>
                            <equivvalue val="GuyM"/>
                        </closurestypename>
                    </closure>
                    <closure field="0">
                        <closurestypename type="string"
primaryvalue="OdedE">
                            <equivvalue val="NimrodR"/>
                            <equivvalue val="YossiO"/>
                            <equivvalue val="LironL"/>
                        </closurestypename>
                    </closure>
                    <closure field="2">
                        <closurestypename type="int"
primaryvalue="15">
                            <equivvalue val="5"/>
                            <equivvalue val="6"/>
                            <equivvalue val="7"/>
                        </closurestypename>
                    </closure>
                </closures>
            </aggregation>
        </aggregations>
        <accumulators>
            <!-- up=8, down=9, sessions=10 -->

```

```

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

```

RAG Adapter general maintenance is done in the file `~pcube/pump/config/ragadapter.conf`, which is similar in structure to the JDBC Adapter configuration file. Following is an example:

```
#
# RAGAdapter main configuration file
#

[config]
xml_filename = ~/pump/config/ragadapter.xml

[housekeeper]
interval_sec = 10

[db]
operations_timeout = 60
batch_size = 10
transaction_size = 15
commit_interval = 6
blocking_connects = true
db_template_file = main.vm
db_template_dir = dbpacks/sybase/ase12.5.1

[app]
app_conf_file = dbtables.xml
app_dtd_file = dbtables.dtd
app_conf_dir = apps/engage/2.5
```

The Bundled Database

The CM platform can utilize a database for the purpose of storing the transaction records supplied by the system's SCE Platform(s).

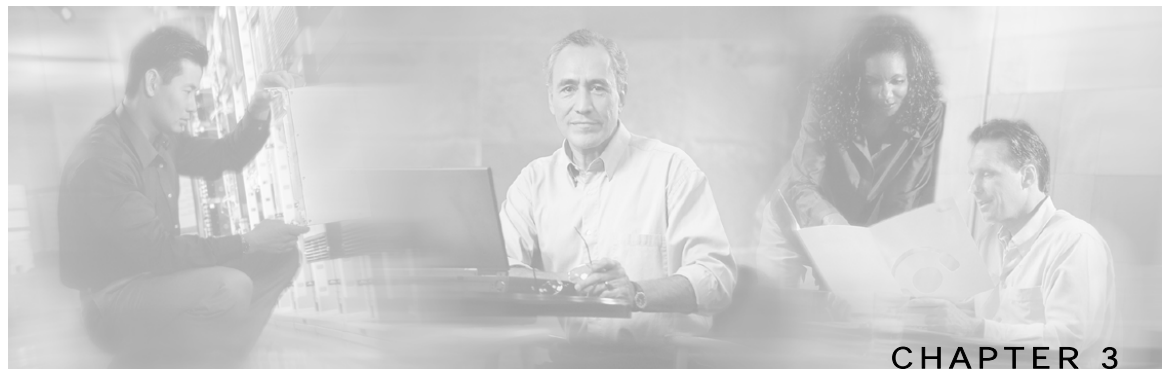
In *bundled* mode, the CM uses the Sybase Adaptive Server Enterprise for implementing the commercial database. The Sybase data management platform supports transaction-intensive enterprise applications. It enables you to store and retrieve information on-line and can warehouse information as needed.

The bundled Sybase Database, located on the Server HW together with the other CM components, uses a simple schema consisting of a group of small, simple tables. The CM DB Adapter (or JDBC Adapter) sends the converted RDRs to the Sybase Database to be stored within these tables. The records can then be accessed using standard database query and reporting tools. As an example, Cisco provides a template-based Reporting tool for the purpose of generating reports on subscriber usage or network resource and traffic analysis. (see the *Service Control Application Suite for Broadband User Guide* for more information on the Service Control Reporting tool.)

To carry out database maintenance, use the operating system commands and scripts. 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 *Configuring the Periodic Deletion of Old Records* (on page 5-5).

Using an External Database

As noted above, when using the JDBC Adapter and TA Adapter, any JDBC-compliant database (for example, Oracle™ or MySQL) may be used with the CM. In this case, the database can be local or remote. The user should configure the JDBC and or TA adapter to use this database, and also configure a so-called database pack to configure the CM for the database's parameters such as IP address, port and so forth. The user should also supply a JDBC driver for this database, to be used by the adapters when connecting to it. See *Database Configuration* (on page [6-1](#)) for more details on configuring the CM to work with an external database.



Installing the CM and Getting Started

This chapter describes how to install the Collection Manager and optionally, its bundled database.

This chapter contains the following sections:

- [System Requirements](#) 3-1
- [CD Content](#) 3-5
- [Installation](#) 3-5
- [Getting Started](#) 3-13
- [Default Configuration Settings](#) 3-13

System Requirements

The Collection Manager and its database are software components that run on a Server Platform. They can be installed on either of the following configurations that conform to the requirements listed below:

- SUN Sparc machine running Solaris 8
- IA32 machine running Redhat Enterprise Linux 3.0

Solaris Requirements

Hardware

- Minimum 500Mhz CPU
- Minimum 1GB RAM
- Hard disk:
 - For unbundled installations: One hard disk, at least 18 GB
 - For bundled installations: a second hard disk is recommended (also at least 18 GB), which may be used to keep Sybase data.
- 100baseT network interface
- Recommended: CDROM drive

Software and Environment

- Solaris 5.8 64-bit build 04/01 and above (currently only Solaris 5.8 is supported).
- Solaris Core Installation
- The following additional packages should be installed:

system	SUNWbash	GNU Bourne-Again shell (bash)
system	SUNWgzip	The GNU Zip (gzip) compression utility
system	SUNWzip	The Info-Zip (zip) compression utility
system	SUNWlibC	Sun Workshop Compilers Bundled libC
system	SUNWlibCx	Sun WorkShop Bundled 64-bit libC
- If using the CM in bundled mode with Sybase, the following package should also be installed:

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

Optionally the following package may be installed (for sysadmin applications such as sys-unconfig):

system	SUNWadmap	System administration applications
system	SUNWadmc	System administration core libraries

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

application	SMCpythn	python
-------------	----------	--------

Note that as of this writing, this python package requires the installation of two additional packages as follows:

application	SMClibgcc	libgcc
application	SMCncurs	application ncurses

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

The root (/) partition must have enough free space to install these packages. As of this writing, the amount of free space needed is 104 MB.

- Latest recommended patches from SUN should be applied
 - See <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/xos-8&nav=pub-patches>
 - For Java, see <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/J2SE>
- If using Sybase, current Solaris patches recommended by Sybase should be installed

See <http://my.sybase.com/detail?id=1016173>

- At least 8 Gb free on the partition where the CM is to be installed. (Used for CSV storage and persistent buffers)
- At least 3 Gb free on some partition for installations with bundled Sybase, for the sybase home directory. When using the "legacy" install mode for Sybase, an additional 1Gb will be required on this partition for the "tempdb" device.

- For installations with bundled Sybase - free space on some partition(s) to hold the desired size of Sybase data and log (these sizes are configurable in install time)
- [*OPTIONAL, and only for installation with bundled Sybase*] Install the sudo package (from e.g. <http://sunfreeware.com>) and configure the following line in the sudoers file:

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

where XXX is the intended home directory for pcube

If you choose not to install sudo: in the rare event of Sybase crash, the CM will not be able to revive it by itself.

- For installations with a bundled DB where the Service Control Reporter is to be used, some ftp server should be listening on port 21 so the Reporter can authenticate against it.
- If using Sybase, you must make sure before installation, that all IP addresses that are configured for the machine NICs, have host names associated with them in /etc/hosts or some other active naming service. (this is a limitation of Sybase ASE).
- If using Sybase, the kernel should be configured with at least:
 - 512000000 bytes in shmmax
 - 32 in shmseg

Additionally, the IPC module should be loaded at boot time. This is achieved by putting the following lines in the file /etc/system:

```
forceload: sys/semsys
forceload: sys/shmsys
```

- If using database periodic delete (in CM in bundled mode with Sybase), the pcube user should be able to schedule and run cron jobs.

Setting the Locale and Time Zone

- US English locale must be used:

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

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

```
LANG=en_US
```

Also, Solaris needs to have this locale installed. You can verify that it is installed by checking that the directory /usr/lib/locale/en_US exists. If it does not, install the locale files from the Solaris CDs.

- POSIX format for time zone is not recommended:

Setting the OS time zone as offset from GMT in POSIX format is not recommended, and may lead to problems. It is best to set the time zone in the /etc/TIMEZONE configuration file by (supported) country name, for example:

```
TZ=Japan
```

You can 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`.

In case GMT offset must be used, please use the "zoneinfo" format by prepending a `:Etc/` prefix, for example:

`TZ=:Etc/GMT+5`

Redhat Linux Requirements

Hardware

- Minimum 800Mhz CPU.
- Minimum 1Gb RAM per CPU
- Hard drive:
 - For unbundled installations: One hard disk, at least 18 Gb.
 - For bundled installations: a second hard disk is recommended (also at least 18 GB), which may be used to keep Sybase data.
- 100baseT network interface.
- Recommended: CDROM drive.

Software and Environment

- Redhat Enterprise Linux 3.0 and up (currently only Redhat 3.0 is supported).
- Redhat Enterprise "Base" Installation.
- If using Sybase, the following additional package should be installed:

`compat-libstdc++`

this package is available on the Redhat installation CD.

- Latest recommended patches from Redhat should be applied
- Current patches recommended by Sybase should be installed
- At least 8 Gb free on the partition where the CM is to be installed.
(Used for CSV storage and persistent buffers)
- At least 1 Gb free on some partition for installations with bundled Sybase, for the sybase home directory.
- [OPTIONAL, and only for installation with DB] Install the sudo package and configure the following line in the sudoers file:

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

where XXX is the intended home directory for pcube.

If you choose not to install sudo, in the rare event of Sybase crash, the CM will not be able to revive it by itself.

- For installations with a bundled DB where the Service Control Reporter is to be used, an ftp server should be listening on port 21 so the Reporter can authenticate against it.

- If using Sybase, you must make sure before installation, that all IP addresses that are configured for the machine NICs, have host names associated with them in */etc/hosts* or some other active naming service. (this is a limitation of Sybase ASE).
- If using Sybase, the kernel should be configured with at least:
 - 512000000 bytes in *shmmax*
- If using database periodic delete (in CM in bundled mode with Sybase), the *pcube* user should be able to schedule and run cron jobs.

Setting the Locale and Timezone

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

CD Content

The Collection Manager installation CD contains installation scripts for installing the Collection Manager and the Sybase database. In addition, a number of useful utilities are included. These utilities are located in the **unsupported** directory. You can use them on the machine where you are installing the Collection Manager to increase its security.

These utilities include:

- **The SSH package:** This utility can be used to enable an SSH daemon on the target host, so that secure shell connections can be made to it.
- **The IPFilter package:** This is a simple firewall that you can install to secure the host platform. When using this firewall, make sure you do not block access to the machine in a manner that will disrupt operation of the Collection Manager software.
- **Scripts for support file gathering**
- **Scripts for periodic Sybase maintenance**



Important Note:

The included SSH and IPFilter packages are not, by themselves, sufficient to ensure the security of your machine. Most significantly, it is crucial to always use the most recent versions of these packages available on the web, and keep them up to date. These packages are provided on the Collection Manager CD for demonstration purposes only.

Installation

In the Bundled DB case, the Collection Manager stores its data using a local Sybase database. Version information is as follows:

- Solaris: Adaptive Server Enterprise/12.5.1/EBF 11420/P/Sun_svr4/OS 5.8/ase1251/1823/64-bit/FBO/Wed Sep 17 09:05:24 2003
- Redhat Enterprise Linux: Adaptive Server Enterprise/12.5.1/EBF 11429/P/Linux Intel/Enterprise Linux/ase1251/1823/32-bit/OPT/Tue Sep 16 23:43:54 2003

This section describes how to install the CM and the Sybase database.
To prepare to install the Collection Manager and the Sybase database:

-
- Step 1** Log on as the **root** user.
- Step 2** Mount the CD-ROM on any mount point (usually, **/cdrom**) or make its contents available on your local network.
-

Phase 1: Installing Sybase

If you do not want to install Sybase (for example, when working in *unbundled mode*), go to *Phase 2: Installing Service Control Software* (on page 3-8).



Note:

If at any point during the installation you want to revert the Sybase installation actions (for example, in the rare case that an installation is interrupted due to a power failure) you can do so as follows:

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

To install Sybase:

-
- Step 1** Change directory to *'sybase'* under the CDROM root.
- Step 2** Execute the script *'./installsyb.sh'*
-

The script usage is as follows:

Usage:

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

- SYBHOME is the new home directory of the Sybase user (should have 1,000,000 Kbytes free)
- Data location options:
 - Specify DATADIR as a directory in which all Sybase data will be stored in the files.
This location should be in a partition where at least 15 GB is free.

OR

- *einteractively* for file/device location and size (no error checking will be performed - use with care!)

OR

- Specify `--legacy` to create old-style partitions/devices on a 2nd disk. (this option is available on Solaris only)

If you specify a *DATADIR*, all Sybase data will be stored as normal files in that directory, with default sizes of 10 GB for data and 3 GB for log. The directory will be chown'ed to the Sybase user during installation. While this option is convenient, the `--expert t` option provides better DB performance.

--expert Option: Usage Guidelines

The `--expert` option allows you to specify files (or devices) and sizes for the data, log, and "tempdb" storage areas. However, this option must be used with care, due to the following important considerations:

- No error-checking is performed.
- If you use devices, you must verify that each device is a raw (character-special) device, and has enough space to hold the "sybase device" assigned to it. (As mentioned above, the system does NOT verify this.)



Note:

Note that the size you specify for the "sybase device" is expressed in 2048-byte blocks.

- Sybase will overwrite and destroy any existing data in the files/devices you specify!



Important Note for Linux Users:

Enabling raw devices in Redhat Linux and preparing them for use with Sybase, involves a few steps and is described in the following URL:http://sybooks.sybase.com/onlinebooks/group-as/asp1251e/installlnx/@Generic__BookTextView/8022#X

EXAMPLE:

The following is a sample from a sessions where `--expert` was used:

```
Please enter a file or character-device for each of the following.
NOTE - no error checking on existence/size is performed.  If anything
goes wrong, the install will fail with an error message somewhere along the
way.
```

```
file/device for data: /dev/rdsk/clt1d0s6
size (2K blocks) for data: 10485760
file/device for log: /dev/rdsk/clt1d0s4
size (2K blocks) for log: 4194304
file/device for tempdb: /opt/sybdata/tempdb.dat
size (2K blocks) for tempdb: 4194304
Do you want to input the values again? (yes/no): no
```

Phase 2: Installing Service Control Software



Note:

If at any point during the installation you want to revert the Service Control software installation actions (for example, in the rare case that an installation is interrupted due to a power failure) you can do so as follows:

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

To install the Service Control software:

Step 1 Change directories to **install-scripts** under the CD-ROM root.

Step 2 Run the `./install-dc.sh` script.

The usage message for this script is:

```
Usage: install-dc.sh [-h] (-d PQBDIR | -o)

Options: -d PQBDIR   select directory for ~pcube
                    (must not exist and must be on 8 GB free partition)
        -o           keep an old installation of the software
                    (can't be used with -d)
        -h           print this help and exit

Description of the options:

-d PQBDIR
  Used to designate the directory of the newly created
  pcube 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 the existing pcube user home
  (can't be used with -d)
```

Step 3 After the script completes, set a password for the **pcube** user by running the command `passwd pcube`. Make sure to record the password that you choose.

Step 4 If you are going to run an application that utilizes the Topper/Aggregator adapter (such as the Service Control Application Suite 2.0.1 and above), you may need to increase the amount of memory allocated to this adapter. This depends on the number of subscribers to be handled by the CM. To increase the memory allocation:

- a) Open the file `~pcube/pump/config/pump.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, use the value "**-Xmx1024M**".
- d) Save and close the file.

If you are going to run an application that utilizes the Realtime Aggregating adapter (RAGAdapter), 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 RAGAdapter configuration. To change the setting:

- e) Open the file `~pcube/pump/config/pump.conf`
- f) Locate the setting containing "**RAGAdapter**" in the "[*adapter_mem*]" section.
- g) Change the default value (512 MB) to a larger value. For example, to allocate 1024 MB of memory, use the value "~~Xmx~~**1024M**".
- h) Save and close the file.

Step 5 Log on as the **pcube** user.

Step 6 If you plan to use a database, you must select either the DB Adapter or JDBCAdapter for connecting to the DB.

Edit the file `~/pump/config/pump.conf` and uncomment in the **adapters** section one of these adapters.

Step 7 If you need to work with a database, one of the DB-related adapters of the CM should be enabled, as follows:

- To use a bundled Sybase database, enable either the DBAdapter or JDBCAdapter.
- To use another database such as Oracle, enable the JDBCAdapter.

Do this by editing the file `~pcube/pump/config/pump.conf`, and uncommenting the appropriate "**adapter.1**" line, depending on the adapter you want to enable.

Note that to use an external DB, you must also configure a '**dbpack**' to enable the CM to connect to it.

See *Database Configuration* (on page 6-1) for more information.

Step 8 Select the desired application. The procedure for this depends on the adapter selected:

- **JDBCAdapter with a bundled Sybse DB:** Optionally, log in as '**pcube**', start the DC, wait 1-2 minutes for the DB tables to be created, and then run the script `~/db_maint/create_periodic_del_procs.sh` to install the periodic delete procedures for the DB tables.



Note:

You may be asked whether to install periodic delete procedures for the DB tables. Choosing not to install the periodic delete procedures, may cause your second disk to overflow if reports are sent to the database, and no other mechanism is used to delete them periodically.

- **DBAdapter:** Log in as '**pcube**', and run the script `~/pump/bin/select-app.sh` to select the desired application (**Service Control Application Suite for Broadband, Service Control Application Suite for Mobile**).



Note:

You may be asked whether to install periodic delete procedures for the DB tables. Choosing not to install the periodic delete procedures, may cause your second disk to overflow if reports are sent to the database, and no other mechanism is used to delete them periodically.

- **JDBCAdapter:** Edit the file `~pcube/pump/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/engage/2.5**".
- **TAAdapter:** Edit the file `~pcube/pump/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/engage/2.5**".

Step 9 Set the SCE device time zone by running the command:

```
~pcube/pump/bin/select-se-tz.sh --offset=<offset-in-minutes from GMT>
```

where the offset parameter is the offset of the SCE device(s) from GMT in minutes.

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

```
~pcube/pump/bin/select-se-tz.sh --offset=120
```

If the SCE is located in GMT-10 use:

```
~pcube/pump/bin/select-se-tz.sh --offset=-600
```



Important Note for the JDBC Adapter:

If using the JDBC Adapter, replace `select-se-tz.sh` with `jselect-se-tz.sh`



Note:

This step allows the CM and reporting software to create reports and charts according to the service engines' time zone.



Note:

The database must be running to run this script.

Changes Made by `installsyb.sh`

The `installsyb.sh` script performs the following steps: (This script is used in *Phase 1: Installing Sybase* (on page 3-6).)

- Verifies the **shmemp** setting for Sybase in `/etc/system`. If the setting is not there, it inserts it and reboots (after prompting the user).
- Adds a user **sybase** and group **sybase**.
- Populates the home of user **sybase** with Sybase 12.0.0.5 distribution.
- Optionally formats the second disk and creates partitions there to hold Sybase tables.
- Builds a Sybase server including Sybase users and passwords.
- Starts Sybase.

- Runs SQL scripts to create the Service Control database structure. This involves restarting Sybase several times, and is a lengthy process.

Changes Made by `install-dc`

The `install-dc.sh` script performs the following steps: (This script is used in *Phase 2: Installing Service Control Software* (on page 3-8).)

If needed, creates a `pcube` user and `pcube` group.

- Optionally creates the home for this user (see *Phase 2: Installing Service Control Software* (on page 3-8)).
- Populates the home of `pcube` with CM files and scripts.
- Installs the following extra components:
 - JRE 1.4.2_03 in `~pcube/lib/java`
 - Creates boot script symbolic links for `sybase` and `pcube` in `/etc/init.d` and `/etc/rc*.d`

Changes Made by `select-app.sh`

The `select-app.sh` script makes the following change: (This script is used in step 6 of *Phase 2: Installing Service Control Software* (on page 3-8).)

- Optionally creates periodic delete procedure for Sybase report tables, this involves installing some stored procedures in Sybase and creating a cron job for user `pcube` to run them (see *Phase 2: Installing Service Control Software* (on page 3-8)).

Ports Used by the Collection Manager Software

The following table describes the TCP/UDP ports on which the CM software, and associated components (such as the Sybase database), listens. This list may be helpful to the network administrator in understanding the behavior of the software, and its adherence to the security policy.

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

The ports listed below will appear in conjunction with the operation that spawns them. It is important to allow access on these port numbers, otherwise certain operations may fail.

Table 3-1 Ports that CM listens on constantly

Port Number	Description
TCP-33000	Used by the SCE devices to send RDRs for data-collection
TCP-21	Used by the Service Control Reporter to authenticate against the <code>pcube</code> user on the CM machine.
TCP-33001	Internal Collection Manager. Note: Access required only from the local machine, and can be blocked for external access.

TCP-9092	HTTP technician interface
TCP-4100	(<i>bundled mode only</i>) Sybase database connectivity through ODBC/JDBC. Required for access to the data-base (Service Control reporter)
TCP-1099 - 1120	RMI. Used as management interface between the data-collector and the Service Control management server
TCP-22000	FTP server of CM agent. Note: FTP transactions may listen on other ports (22001 – 22100) for data transfer, as negotiated by the protocol
TCP-7787	Management userlog internal logging. Note: Access required only from the local machine, and can be blocked for external access.
TCP-14375	Used by the SCAS BB Console to send symbol definitions ("valyes.ini") to the CM.

Uninstalling the Sybase Database and the Service Control Software

To uninstall Sybase:

-
- Step 1** Log in as the root user.
- Step 2** Run the following commands:
- ```
pkill -u sybase

userdel -r sybase

rm /etc/rc*.d/[SK]*sybase
```
- Step 3** (Optional) Edit `/etc/system` and remove the Sybase `shmem` setting.
- 

To uninstall the Service Control software:

- 
- Step 1** Log in directly as the `root` user.
- Step 2** Run the following commands:
- ```
pkill -u pcube

userdel -r pcube

rm /etc/rc*.d/[SK]*pcube
```
-

Getting Started

The Server HW is configured to start the CM components automatically on machine boot, and stop them on machine shutdown. To manage, monitor, and configure the Collection Manager, use the various utility scripts installed with the Collection Manager.

The following chapters explain how to use utility scripts:

- Chapter 5, *Managing the Collection Manager* (on page 4-1) contains information about the use of scripts to configure and monitor certain aspects of the Collection Manager.
- Chapter 6, *Managing the Database and CSV Repository* (on page 5-1) contains information about the use of scripts to manage both the commercial database and the CSV repository that is part of the CM.

Default Configuration Settings

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

**Note:**

It is possible to send a particular RDR (by tag value) to more than one Adapter.



Managing the Collection Manager

This chapter describes how to manage the Collection Manager (CM) using utility scripts. The Collection Manager is monitored and managed using utility scripts, from any machine connected to the CM, for example via Telnet or SSH. The utility scripts are located in the installation directory of the CM.

**Note:**

For information on managing the database and the CSV repository, see *Managing the Database and CSV Repository* (on page 5-1). For information on the periodic deletion of database records, see *Configuring the Periodic Deletion of Old Records* (on page 5-5).

This chapter contains the following sections:

- [Using Utility Scripts](#) 4-1
- [Configuring the Collection Manager](#) 4-2
- [Monitoring the Collection Manager](#) 4-5
- [Verifying that the Server is Operational](#) 4-7

Using Utility Scripts

The following are general instructions for using the utility scripts.

- To invoke any script other than the exceptions listed below, always log in as the **pcube** user. An attempt to run these scripts as the **root** user will result in an error.
- To display a description of the script, with an explanation of all flags and parameters, invoke the script with the help flag.

Note that there is a slight variation in the help flag. Some scripts use "-h", and others "--help". Consult the specific script definition.

Example:

The following example shows how to display a description of the `~pcube/scripts/dbperiodic.py` script.

```

> ~pcube/scripts/dbperiodic.py --help

Usage:

/export/home/dc/scripts/dbperiodic.py --load
    load configuration from
    /export/home/dc/db_maint/dbperiodic.conf

/export/home/dc/scripts/dbperiodic.py --loadfile=FILE
    load configuration from FILE

/export/home/dc/scripts/dbperiodic.py --dump
    print the current configuration in INI format to standard
output

/export/home/dc/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, see www.python.org.

Configuring the Collection Manager

Use utility scripts to:

- Specify which servers will be activated at boot time
- Start or stop the database
- Start or stop an adapter
- Drop an SCE connection
- Synchronize the Sybase interfaces with the actual IP address

The following scripts are relevant to configuring the Collection Manager:

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

**Note:**

For information on managing the database and the CSV repository, see *Managing the Database and CSV Repository* (on page 5-1). For information on the periodic deletion of database records, see *Configuring the Periodic Deletion of Old Records* (on page 5-5).

Activating the Servers

```
~pcube/setup/on-boot.py
```

This script defines which of the servers, the Collection Manager (also called pump) and Sybase, are activated at boot time. This takes effect starting on next boot.

Options

`--pump=on/off` Activate/Do not activate the Collection Manager at boot.
`--sybase=on/off` Activate/deactivate the bundled Sybase at boot

To define which servers are activated at boot:

Step 1 As the **pcube** user, type:

```
~pcube/setup/on-boot.py --pump=flag --sybase=flag
```

and press **Enter**.

EXAMPLE:

The following example shows how to set the pump and Sybase to be activated (by default) on boot.

```
> ~pcube/setup/on-boot.py --pump=on --sybase=on
```

Controlling the Adapters

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

This script can be used to shutdown or bring up any configured adapter, or to list the currently running Collection Manager adapters.

Options

`action = start` Bring up the adapter specified in the **adapter** parameter.
`action = stop` Shut down the adapter specified in the **adapter** parameter.
`action = list` List the currently running Collection Manager adapters.
`Adapter = adapter name` Identifies adapter. Use only with **start** and **stop** actions.
`--help` Display these options.

To shut down an adapter:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

To bring up an adapter:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

EXAMPLE:

The following example shows how to bring up an adapter

```
> ~pcube/scripts/adapterconf.py --op=start --adapter=csvadapter
```

Options

<code>--op=start</code>	Start the Collection Manager database.
<code>--op=stop</code>	Shut down the Collection Manager database.
<code>--op=status</code>	Display the current operational status of the database.

To stop the Collection Manager database:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

To start the Collection Manager database:

Step 1 As the **pcube** user, type:

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

and press **Enter**.



Note:

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

Controlling the Database

This script is only useful for the *bundled* DB case.

```
~pcube/scripts/dbconf.sh --op=operation
```

This script can be used to shutdown or start the Collection Manager database, or to show the operational status of the database.

Dropping an SCE Connection

```
~pcube/scripts/seconf.py --op=drop --ip=IP address
```

This script can be used to drop a connection to a particular SCE. In order to use this script, the HTTP adaptor of the CM management agent must be operating.

This script is also used to display information about the SCE connection. (See *Checking the SCE Connection* (on page 4-7)).

Options

IP address Drop the connection at the specified IP address.

--help Display these options.

To drop an SCE connection:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/seconf.py --op=drop --ip=IP address
```

and press **Enter**.

Monitoring the Collection Manager

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

- Percentage of free space in the database
- SCE Platform connection data
- Rate of reports entering the Collection Manager

The following scripts are relevant to monitoring the Collection Manager:

- `~pcube/setup/alive.sh`
- `~pcube/scripts/dbfree.sh`
- `~pcube/scripts/rdr-rate.py`
- `~pcube/scripts/seconf.py`

The following scripts are used to configure the Collection Manager, but also can be invoked to display the relevant configuration:

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

Checking the Database Capacity

This script is only useful for the *bundled* DB case.

```
~pcube/scripts/dbfree.sh
```

This script displays the percentage of free space in the database report tables and the associated transaction log.

To monitor the database capacity:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/dbfree.sh
```

and press **Enter**.

Checking the RDR Rate

```
~pcube/scripts/rdr-rate.py
```

This script displays the momentary total rate of reports entering the Collection Manager. The output is a single floating point number representing the total rate per second of incoming RDRs (from all sources) that have entered the Collection Manager in the past 5 seconds.

In order for this script to operate, the HTTP adaptor of the CM management agent must be up.

To monitor the RDR rate:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/rdr-rate.py
```

and press **Enter**.

Checking the SCE Connection

```
~pcube/scripts/seconf.py --op=list
```

This script can be used to display information about the SCE connection. In order for this script to operate, the HTTP adaptor of the CM management agent must be up.

This script is also be used to drop a connection from a particular SCE. (See *Dropping an SCE Connection* (on page 4-5)).

To display information about the SCE connection:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/seconf.py --op=list
```

and press **Enter**.

EXAMPLE:

The following example shows SCE connection output.

```
> ~pcube/scripts/seconf.py --op=list
```

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

Verifying that the Server is Operational

```
~pcube/setup/alive.sh
```

You can use this script to verify that the Server is functioning correctly. It verifies that the following components are operational:

- Collection Manager
- Database (in *bundled* DB case)
- Report tables (in *bundled* DB case)

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

To verify that the Server is operational:

Step 1 As the **pcube** user, type:

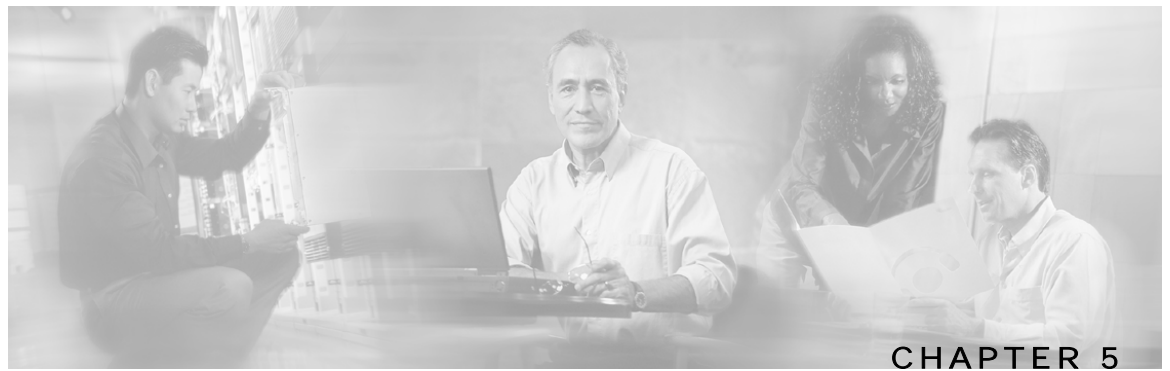
```
~pcube/setup/alive.sh
```

and press **Enter**.

Verifying that the Server is Operational

**Note:**

Since it takes time for the components to initialize after a boot, it is recommended to wait around five minutes after a reboot before running this script.



Managing the Database and CSV Repository

This chapter describes how to manage the database using utility scripts.

**Note:**

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

This chapter contains the following sections:

- [Managing the Database](#) 5-1
- [Managing the CSV Repository](#) 5-8

Managing the Database

**Important Note:**

This entire section pertains only to an installation using a *bundled* DB.

Managing a database includes:

- Generating a list of the database tables
- Deleting a table
- Manually deleting old records from a table
- Backing up and restoring a database
- Configuring the periodic deletion of old records

Every record stored in the database is given a timestamp indicating the record's time of entry to the Collection Manager. This timestamp is used when performing various maintenance operations on the database tables.

The following scripts are relevant to configuring the Collection Manager:

- `~pcube/scripts/dbtables.sh`
- `~pcube/scripts/droptable.sh`
- `~pcube/scripts/prunetable.sh`
- `~pcube/scripts/sybback.sh`
- `~pcube/scripts/sybrestore.sh`

- `~pcube/scripts/dbperiodic.py`

Listing the Database Tables

`~pcube/scripts/dbtables.py`

This script displays a list of all the tables in the database, and if applicable, the number of lines and the minimum/maximum timestamp in the table.

Actual content of the tables, that is existing records, is accessed via the reporter application. For more information, see the *Service Control Application Suite for Broadband User Guide*.

To display a list of all the tables in the database:

Step 1 As the `pcube` user, type:

```
~pcube/scripts/dbtables.sh
```

and press **Enter**.

EXAMPLE:

The following example shows a sample output of `dbtables.sh`:

```
>~pcube/scripts/dbtables.sh

MAX_TIME          TABLE | NUM_LINES |          MIN_TIME |
-----|-----|
2003  3:28PM      RPT_LUR |    53971 | Jul 24 2003  2:52PM | Jul 30
2003 10:47AM      RPT_NUR |    33350 | Jul 29 2003  2:08PM | Jul 30
2003 11:13AM      RPT_PUR |    42167 | Jul 22 2003  5:45PM | Jul 23
2003  3:37PM      RPT_SUR |    38390 | Jul 24 2003  3:08PM | Jul 30
2003  3:37PM      RPT_TR |    29436 | Jul 22 2003  5:27PM | Jul 30
```

Deleting a Table

`~pcube/scripts/droptable.sh`

The following script deletes a single table or all current tables from the database.

Options

<code>table_name</code>	Drop <code>table_name</code> from the database.
<code>ALLTABLES</code>	Drop all tables from the database
<code>-f</code>	Drop 'by force' (no questions asked/errors reported).

-h Display these options.

To drop a table from the database with no request for confirmation:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/droptable.sh -f table_name
```

and press **Enter**.

EXAMPLE:

The following example shows how to drop a table named SubscriberTable.

```
> ~pcube/scripts/droptable.sh -f SubscriberTable
```

To drop all tables from the database:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

Deleting Old Records

```
~pcube/scripts/prunetable.sh num_days table_name
```

This script selectively removes records from a database table based on the record's timestamp. The argument specifies the maximum age (in days) that will not be deleted.

Options

-f Drop 'by force' (no questions asked/errors reported).

-h Display these options.

To delete old records from a database:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/prunetable.sh num_days table_name
```

and press **Enter**.

EXAMPLE:

The following example shows how to delete all records that are more than 7 days old from a table named SubscriberTable.

Since the `-f` flag is not specified, there may be requests for confirmation and/or errors reported.

```
> ~pcube/scripts/prunetable.sh 7 SubscriberTable
```

Creating a Backup of the Database

```
~pcube/scripts/sybback.sh
```

This script creates text file backups of all the tables in the database. It converts all tables to ASCII representation and puts them in a backup directory.

Options

`-d` *path to backup directory* Write backup text files to *backup directory*.
`-h` Display these options.

To backup the database to a specified directory:

Step 1 As the `pcube` user, type:

```
~pcube/scripts/sybback.sh path_to_backup_directory
```

and press **Enter**.

Restoring a Database from the Backup

```
~pcube/scripts/sybrestore.sh
```

This script restores a database from the backup file that was created by the `sybback` script. The backup is taken from the directory specified by the command line argument.

Options

`-d` *path to restore directory* Restore the database using the text files in *restore directory*.
`-h` Display these options.

To restore a database from a specified directory:

Step 1 As the `pcube` user, type:

```
~pcube/scripts/sybrestore.sh -d path_to_backup_directory
```

and press **Enter**.

**Important Note:**

The scripts *sybback* and *sybrestore* are designed for use in situations such as transferring a small amount of data between machines. These scripts do not constitute a viable backup/restore mechanism for Sybase. In cases where such a mechanism is required, please consult the Sybase “Backup Server” product documentation.

Configuring the Periodic Deletion of Old Records

Configuring the periodic deletion of old records is a two-step process:

- Edit the periodic deletion configuration file.
- Use the **dbperiodic.py** utility script to apply the new configuration. This script parses the configuration file, verifies its validity, and updates the pcube users crontab to reflect the changes.

Periodic deletion of a table does not begin when a previous periodic is still running. This avoids excessive load on the database, which impedes 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 sequence they are listed in the periodic deletion configuration file.

For ease of configuration, you have the option to schedule periodic deletion for all tables consecutively on the same schedule.

**Note:**

All periodic deletion activity is recorded in the system log file (`/var/adm/messages`).

Viewing the Currently Active Periodic Deletion Configuration File

To print the current periodic deletion configuration:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

**Note:**

This script prints the current periodic deletion configuration. However, if the current periodic deletion configuration file was not yet loaded, the actual deletion configuration may vary from the contents of this file.

Editing the Periodic Deletion Configuration File

Edit the periodic deletion configuration file (**dbperiodic.conf**). This file is structured similar to an INI file, where each section describes a scheduled data reduction operation for a specific set of tables, to be performed according to a specified schedule. The name of each section of the file is not used when parsing the file; you can use whatever names you wish.

Each section begins with the section name in brackets, and should contain the fields shown in the following table. (Not all fields are required in each section of the configuration file.) Separate the fields and their values by an equal sign (=). Examples are provided later in this chapter.

Table 5-1 Periodic Deletion Fields

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



Note:

Values for all fields 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**.

Example 1

In this 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 = *
```

Example 2

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 DB 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 realtime]
active = true
tablenames = RPT_SUR,RPT_LUR,RPT_PUR
daystokeep = 3
minute = 0
hour = *
```

Applying the Periodic Deletion Configuration File

`~pcube/scripts/dbperiodic.py`

Use this script to apply a new periodic deletion configuration file or to view an existing file.

Options

<code>--load</code>	Load the periodic deletion configuration from <code>/export/home/dc/db_maint/dbperiodic.conf</code>
<code>--loadfile=path to periodic deletion configuration file</code>	Load the periodic deletion configuration from the file specified.
<code>--dump</code>	Print the periodic deletion configuration.
<code>-h</code>	Display these options.

To load the periodic deletion configuration file from `/export/home/dc/db_maint/dbperiodic.conf`:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

To load the periodic deletion configuration file from a specified location:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/dbperiodic.py --loadfile=path to periodic deletion
configuration file
```

and press **Enter**.

Managing the CSV Repository

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



Note:

Failure to delete CSV files may result in disk overflow if the backup parameter is set to true (no CSV files will ever be deleted).

It is the responsibility of the third party to manage the CSV files and delete them as necessary.

In order for this script to operate, the HTTP adaptor of the CM management agent must be operational. If it is down, an error message will be printed.

CSV Repository File Structure

CSV files are stored in several subdirectories. Each subdirectory is given the number of an RDR tag. (RDR tags describe the type of RDR.) RDRs are stored in the subdirectory with their tag number, 4042321936, 404232193, etc. For more information on RDR tags, see the *Service Control Application Suite for Broadband User 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 `pump.conf` file located in the `pump/config` directory.

Configuring the CSV File Repository

```
~pcube/scripts/csvconf.sh
```

This script is used for:

- Listing the number of RDRs currently stored in the repository.
- Configuring the maximum number of CSV files and the maximum permissible number of reports (lines) in each file.
- Controlling whether a backup will be made whenever an old CSV file is about to be overwritten.
- Controlling whether each line in a CSV file will contain an indication of the IP of the SCE that sent this RDR (this option is off by default)

**Note:**

Instead of using this script, the user can edit the file `~pcube/pump/config/csvadapter.conf`. Changes in this file need a CM restart to take effect.

**Note:**

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

**Note:**

Setting these parameters does not change CSV files that already exist; it only has an effect on files that are created subsequently.

Options

<code>--list</code>	Display the CSV repository contents (number of RDRs currently stored in the repository).
<code>--clear</code>	Delete all files from CSV repository. This deletes all CSV files, but not the directories in which they are contained.
<code>--maxlines=N</code>	Set max number of RDRs per CSV file to N. (Integer between 1 and 20,000)
<code>--maxfiles=M</code>	Set max number of CSV files in each subdirectory to M. (Integer between 10 and 10,000.)
<code>--backups=true/false</code>	Enable or disable backing-up of old CSV files
<code>--recordsource=true/false</code>	Enable or disable inclusion of record source in CSV files

To set the maximum number of CSV files per subdirectory:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/csvconf.sh --maxfiles=M
```

and press **Enter**.

EXAMPLE:

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

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

To set the maximum number of RDRs per CSV file:

Step 1 As the **pcube** user, type:

```
~pcube/scripts/csvconf.sh --maxlines=N
```

and press **Enter**.

EXAMPLE:

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

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

To delete all files from the CSV repository:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

To disable backing up of old CSV files in the repository:

Step 1 As the **pcube** user, type:

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

and press **Enter**.

Configuring the Comma Escape

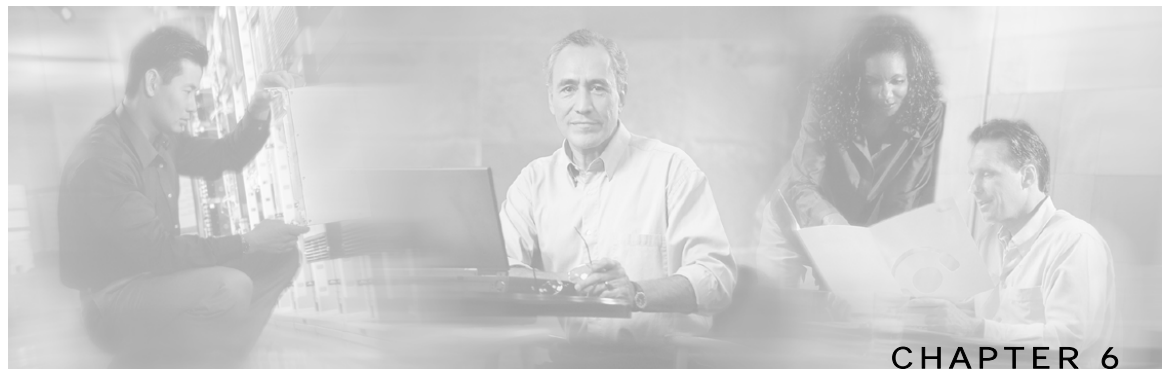
When a comma is contained within a field in a CSV file, an escape method is used to indicate that the comma does not mark the end of the field.

Three escape methods are supported:

- **Single quotation marks** – Single quotation marks surround any field that contains one or more commas. When using this method, 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**. When using this method, 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 (****). When using this method, 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 Microsoft® 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.cfg* file in the *CSVAdapter* directory. The value must be: **backslash**, **quote**, or **url**.



Database Configuration

Starting with CM 2.1, all database access is done using JDBC. This is true for the adapter inserting records to the database, scripts accessing or managing the database, etc. This chapter outlines how to configure the CM to work with your database, and how to utilize the DB infrastructure of the CM to extend its functionality.

This chapter contains the following sections:

- A quickstart guide to configuring the basic communication parameters for the CM adapters. If there is no need to extend the functionality of the CM, and the database has no special configuration, then the instructions in this chapter will suffice for configuring your CM to work successfully with a JDBC database.
- A brief overview of VTL, the template language used for the database-related configuration files of the CM, and links to further resources for the language.
- A description of the configuration files involved in configuring the CM to work with a particular database.

This includes the application-related configuration files, as these contain information that influences the parsing of the database configuration directives.

- A concrete example of all configuration files in the simplest case of working with an Oracle database.
- A description of the mechanisms in the DB framework that may help in debugging or testing your templates.
- An example of how to use this framework in command-line scripts.
- Hints on using the framework to support scalability-related configuration in Oracle.

This chapter contains the following sections:

- [Quick Start Guide](#) 6-2
- [The Template Language](#) 6-4
- [Configuration Files](#) 6-5
- [A Working Example](#) 6-11
- [Testing and Debugging](#) 6-13
- [Using the JDBC Framework in Scripts](#) 6-14
- [Scalability-Hints for Oracle](#) 6-16

Quick Start Guide

If you will be using an Oracle database with the standard configuration supplied with the CM, this Quick Start Guide explains how to change the basic connection parameters such as the IP address and port where Oracle is deployed. No other configuration changes are necessary for the CM to work with your Oracle database.

Step 1 Stop the CM if it is running.

Step 2 Configure the CM to use the JDBCAdapter:

Edit the file `~pcube/pump/config/pump.conf`, and search for the string “**adapter.1**”. You will find two lines, one for each adapter. By default, they are both commented out.

You should make sure that:

- DBAdapter line is commented out
- JDBCAdapter line is uncommented

as illustrated below:

```
#adapter.1 = com.pcube.pump.adapters.database.DBAdapter
adapter.1 = com.pcube.pump.adapters.jdbc.JDBCAdapter
```

Step 3 Configure the JDBCAdapter to use Oracle:

Edit the file `~pcube/pump/config/jdbcadapter.conf`, and search for the string “**db_template_dir**”. You will find two lines, one for Sybase and one for Oracle. By default, the *Oracle* line is commented out.

You should:

- uncomment the *Oracle* line
- comment out the *Sybase* line

as illustrated below:

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

Step 4 Configure the TAAdapter to use Oracle:

If you use the TA Adapter, you should repeat Step 3 for this adapter:

Edit the file `~pcube/pump/config/taadapter.conf`, and search for the string “**db_template_dir**”. You will find two lines, one for Sybase and one for Oracle. By default, the *Oracle* line is commented out.

You should:

- uncomment the *Oracle* line
- comment out the *Sybase* line

as illustrated below:

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

Step 5 Configure your database connection parameters:

Edit the file `~pcube/pump/config/dbpacks/oracle/9204e/dbinfo.vm`, and change the following lines to reflect your setup:

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



Note:

The `dbinfo.sh` file is not a shell script; the '#' symbols are part of the declarations and not a comment sign.

The relevant parameters are:

- The host name or IP address of the machine where Oracle is installed.
- The port number on which the Oracle server is listening.
- The user name and password for authentication against Oracle.
- An existing Oracle SID to be used by the CM.

Step 6 (Re)start the CM.

The Template Language

The JDBC configuration files use the Velocity template language from the Apache Jakarta Project (version 1.3.1). For full details about this simple language, see the site <http://jakarta.apache.org/velocity/>.

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

A full reference to the Velocity Template Language (VTL) can be found on the web at <http://jakarta.apache.org/velocity/vtl-reference-guide.html>.

The following table provides a concise description of VTL constructs:

Table 6-1 Summary of VTL constructs

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

Configuration Files

When initializing the Database access framework, the first file searched for is *main.vm*, which contains definitions or pointers to all the required database SQL definitions. The location used for searching to 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 selection of a *dbpack* is done in the adapter level. For instance, the following snippet from the *jdbcadapter.conf* file configures it to work with an Oracle *dbpack*:

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

The directory location is interpreted relative to the main CM configuration directory (usually *~pcube/pump/config*).

The *main.vm* file will generally point to other files to make the configuration more modular; however this is not strictly necessary. The files can contain arbitrary definitions, which can later be used, for example in scripts. Some definitions are mandatory because they are used by the CM JDBCAdapter for its operation. These definitions are listed in the following table (see examples below):

Table 6-2 Mandatory VM Definitions

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

The important thing to note about VTL template parsing in the CM, is that some objects representing the CM configuration are available in the parsing context to be used in the templates. These objects are described in the next section.

Context Objects

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

- The *tables* object: describes all application-related DB configuration, such as the structure of RDRs that should be stored in the database, the structure of the database tables where they are stored, and the structure of any other database tables that might be used by the CM.
- The *dbinfo* object: describes all configuration that is specific to the database, such as the parameters to be used when opening a DB connection, and the SID or schema to be used.
- The *tools* object: a holder for several utility methods that can be used in the templates.

The tables object

This object is an array in which each element represents one of the database tables used by the CM. For each table, the element may contain the following information (not all items are relevant to all tables):

- logical name
- physical name
- RDR tag associated with this table
- list of fields/columns in this table, with the following attributes for each:
 - field ID
 - field name
 - field native type
 - free-form field options
- list of indexes for this table, with the following attributes for each:
 - index name
 - names of columns indexed
 - free-form index options

The contents of the *tables* object can be inspected or manipulated when loading the templates. The *tables* object is initialized using the application-specific XML configuration file (see *Application Configuration* (on page 6-7)).

The dbinfo object

This object holds database-specific configuration options. It holds the following information:

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

The tools object

This object is a container for a few methods that might be useful when developing templates or manipulating the context data structures.

Any of the methods can be invoked as *\$tools.method(arg1, ..., argN)* where *method* is the name of the method.

Currently the included methods are:

- *getTableByName (allTables, name)* – locates the DB table object whose logical name corresponds to *name*.
- *getTableByDbTableName (allTables, name)* – locates the DB table object whose physical name corresponds to *name*.

- *assignParams (sql, args)* – replaces question mark characters in the *sql* string with consecutive elements from the *args* parameter, represented as a String. This is handy in templates that create SQL insert statements using the JDBC Prepared Statement string as a base.
- *collapseWhitespace (s)* – converts all instances of one or more consecutive white space characters to one space, and trims beginning and ending whitespace. This may be useful for databases that like their SQL with a minimum of newlines etc. (Sybase and Oracle do not require this).

For examples demonstrating how to use these tools, see the section *Using the JDBC Framework in Scripts* (on page 6-14).

Application Configuration

As mentioned above, all application-related configuration is done in one file that includes the following items:

- Name and version of the application
- Names and properties of each database table, and specifically the structure of application RDRs that will be stored in database tables
- For each database table, the following items:
 - 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.

The tables.xml file

Following is a listing of a portion of the *Service Control Application Suite for Broadband tables.xml* file:

```
<?xml version="1.0" encoding="ISO8859_1"?>
<!DOCTYPE dbtabconf PUBLIC "-//P-Cube//Engage DB RDR Configuration
2.1.0//EN" "databases.dtd">
<dbtabconf>
  <fileversion>
    ...
  </fileversion>
  <application name="Engage" version="2.1"/>
  <databases>
    <rdr name="SUR" dbtabname="RPT_SUR" tag="4042321922"
createtable="true">
      <fields>
        <field id="1" name="TIME_STAMP" type="TIMESTAMP">
          <options>
            <option property="source" value="timestamp"/>
          </options>
        </field>
        <field id="2" name="RECORD_SOURCE" type="INT32">
          <options>
            <option property="source" value="recordsource"/>
          </options>
        </field>
        <field id="3" name="SUBSCRIBER_ID" type="STRING" size="64"/>
        <field id="4" name="PACKAGE_ID" type="INT32"/>
        <field id="5" name="SERVICE_ID" type="INT32">
          <options>
            <option property="notnull" value="true"/>
          </options>
        </field>
        <field id="6" name="MONITORED_OBJECT_ID" type="INT32"/>
        <field id="7" name="BREACH_STATE" type="INT32"/>
        <field id="8" name="REASON" type="INT32"/>
        <field id="9" name="CONFIGURED_DURATION" type="INT32"/>
        <field id="10" name="DURATION" type="INT32"/>
        <field id="11" name="END_TIME" type="INT32"/>
        <field id="12" name="UPSTREAM_VOLUME" type="UINT32"/>
        <field id="13" name="DOWNSTREAM_VOLUME" type="UINT32"/>
        <field id="14" name="SESSIONS" type="UINT32"/>
      </fields>
      <indexes>
        <index name="RPT_SUR_I1" columns="END_TIME">
          <options>
            <option property="clustered" value="true"/>
          </options>
        </index>
      </indexes>
    </rdr>
    <rdr name="LUR" dbtabname="RPT_LUR" tag="4042321925"
createtable="true">
      <fields>
        <field id="1" name="TIME_STAMP" type="TIMESTAMP">
          <options>
            <option property="source" value="timestamp"/>
          </options>
        </field>
        <field id="2" name="RECORD_SOURCE" type="INT32">
          <options>
```



```

        <option property="source" value="recordsource"/>
    </options>
</field>
<field id="3" name="LINK_ID" type="INT32"/>
<field id="4" name="GENERATOR_ID" type="INT32"/>
<field id="5" name="SERVICE_ID" type="INT32"/>
<field id="6" name="CONFIGURED_DURATION" type="INT32"/>
<field id="7" name="DURATION" type="INT32"/>
<field id="8" name="END_TIME" type="INT32"/>
<field id="9" name="UPSTREAM_VOLUME" type="UINT32"/>
<field id="10" name="DOWNSTREAM_VOLUME" type="UINT32"/>
<field id="11" name="SESSIONS" type="UINT32"/>
</fields>
<indexes>
    <index name="RPT_LUR_I1" columns="END_TIME">
        <options>
            <option property="clustered" value="true"/>
            <option property="allowduprow" value="true"/>
        </options>
    </index>
</indexes>
</rdr>
<aggtable name="TOP_HOURLY" dbtabname="RPT_TOPS_PERIOD0"
aggperiod="0">
    <fields>
        <field id="1" name="RECORD_SOURCE" type="INT32"/>
        <field id="2" name="METRIC_ID" type="INT8"/>
        <field id="3" name="SERVICE_ID" type="INT8"/>
        <field id="4" name="TIME_STAMP" type="TIMESTAMP"/>
        <field id="5" name="AGG_PERIOD" type="INT8"/>
        <field id="6" name="SUBSCRIBER_ID" type="STRING" size="64"/>
        <field id="7" name="CONSUMPTION" type="UINT32"/>
    </fields>
    <indexes>
        <index name="RPT_TOPS_PERIOD0_I1" columns="TIME_STAMP">
            <options>
                <option property="clustered" value="true"/>
                <option property="allowduprow" value="true"/>
            </options>
        </index>
    </indexes>
</aggtable>
<table name="TZ" dbtabname="JCONF_SE_TZ_OFFSET">
    <fields>
        <field id="1" name="TIME_STAMP" type="TIMESTAMP"/>
        <field id="2" name="OFFSET_MIN" type="INT16"/>
    </fields>
</table>
</dbtables>
</dbtabconf>

```

For each table (either RDR table, aggregation table, or an extra table) the fields, indexes, etc are listed. Note that a table, index or fields can have arbitrary free text *options* that can be accessed in the templates.

The XML file is verified in runtime against a simple DTD, reproduced below.

The tables.dtd file

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

```
<?xml version="1.0" encoding="ISO8859_1"?>

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

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

A Working Example

As explained above, the *main.vm* file can contain references to other *vm* files to support modularization. The names of these files are arbitrary. There is, however, one file whose name is predetermined – *VM_global_library.vm*. Any macros that need to be defined should be put in this file, to make sure they are loaded at the right time. See the *Velocity User Guide* for more details on this special file.

Here are the contents of the *main.vm* file for an Oracle setup:

```
#parse ('dbinfo.vm')

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

In this example, the mandatory DB and SQL definitions (see Table 7-2 - Mandatory VM Definitions) were moved to separate files, to be loaded and parsed using the *#parse* directive.

The next 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

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

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

dbinfo Configuration

In this example, note that the only required fields are the URL and connection options (for authentication).

The insertion of blank lines to separate the code into to distinct fields for host, port, *sid* is for readability and to ease later configuration changes.

```
#set ($dbinfo.driver = "oracle.jdbc.OracleDriver")
#set ($dbinfo.driverjarfile = "ojdbc14.jar")

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

#set ($dbinfo.url =
"jdbc:oracle:thin:@$dbinfo.options.host:$dbinfo.options.port:$dbinfo.options
.sid")

#set ($dbinfo.connOptions.user = $dbinfo.options.user)
#set ($dbinfo.connOptions.password = $dbinfo.options.password)
## the vendor-specific piece of SQL that will return the current
## date and time:
#set ($dbinfo.options.getdate = "sysdate")
```

SQL Definitions

Code for "drop table"

This code sample drops the table using normal SQL syntax

```
drop table $table.dbtaname
```

Code for "create table"

This code sample creates the 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*, use table partitioning, etc.

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

Code for "create indexes"

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

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

Code for "insert"

This code creates the JDBC “prepared statement” idiom 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

This code simply defines a query that will be used to get the table metadata (column names and types). Any query that returns an empty result set will do.

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

Testing and Debugging

While developing a set of templates for your database, it is useful to be able to see the results of parsing directly. To enable, this, the JDBCAdapter supports direct invocation via the CM main script `~pcube/pump/bin/pump`.

The general syntax for such an invocation is:

```
~/pump/bin/pump invoke com.pcube.pump.adapters.jdbc.JDBCAdapter arguments
```

Where *arguments* are one of the directives described below. It is possible to use this mechanism whether the CM is running at the time or not.

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

Parsing a String

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

```
$ ~/pump/bin/pump invoke com.pcube.pump.adapters.jdbc.JDBCAdapter -parse
'xxx'

xxx

$ ~/pump/bin/pump invoke com.pcube.pump.adapters.jdbc.JDBCAdapter -parse
'$dbinfo.url'

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

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

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

Obtaining Full Debug Information

To see a dump of all the contents of the *tables* and *dbinfo* structures as created by the templates, use the *-debug* flag. This will print out a very detailed view of all the fields, properties, and options of those structures to standard output.

Using the JDBC Framework in Scripts

It is possible to send arbitrary SQL commands to the database for execution and view the resulting data. This may be handy for periodic database maintenance, monitoring the database tables contents, managing extra database tables or any other purpose.

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

Example-Viewing and Setting the SCE Timezone Offset

As an example for an update operation, suppose we want to programmatically change the value in the database table holding the SCE timezone offset setting. The name of this table is usually *JCONF_SE_TZ_OFFSET*, but it may be configured to be another name, so we refer to the table by its logical name *TZ* (see the listing of *tables.xml* above).

To avoid the need to first check that the table exists and then update it, we drop the table (ignoring the error status if it didn't exist), then re-create it and insert the proper values. Since the table contains a timestamp column, we need to get the current date in the database. This operation is specific to each database vendor, so we call the preconfigured *getdate* operation that has been defined in the templates.

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

```
#!/bin/bash

this=$0
tableName=TZ

usage () {
    cat <<EOF
Usage:
    $this --status          - show currently configured TZ offset
    $this --offset=N       - set the offset to N minutes (-1440 <= N <= 1440)
    $this --help           - print this message
EOF
}

query () {
    ~/pump/bin/pump invoke com.pcube.pump.adapters.jdbc.JDBCAdapter -
executeQuery "$*"
}

update () {
    ~/pump/bin/pump invoke com.pcube.pump.adapters.jdbc.JDBCAdapter -
executeUpdate "$*"
}

get_tz () {
    query 'select * from $tools.getTableByName($tables, "TZ").dbtabname'
}

set_tz () {
    update '$tools.getTableByName($tables, "TZ").sql.dropTable'
    update '$tools.getTableByName($tables, "TZ").sql.createTable'
    update '$tools.assignParams($tools.getTableByName($tables,
"TZ").sql.insert, [$dbinfo.options.getdate, '$1'])'
}

case $1 in
    --status)
        get_tz
        ;;
    --help)
        usage
        exit 0
        ;;
    --offset=*)
        n=$(echo $1 | egrep 'offset=[-]?[0-9]+$' | sed 's/.*=//')

```

```

        if [ "$n" ]; then
            if [ "$n" -ge -1440 -a "$n" -le 1440 ]; then
                set_tz $n &>/dev/null
                ok=1
            fi
        fi
        if [ ! "$ok" ]; then
            usage
            exit 2
        fi
        get_tz
        ;;
    *)
        usage
        exit 3
        ;;
esac

```

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

Scalability-Hints for Oracle

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

Using Custom tablespaces

Suppose you have created several tablespaces and wish to distribute the CM tables among them. An easy way to achieve this, is to specify the tablespace to be used in the *tables.xml*. For one table, the definition will look like this (note the **bold** parts):

```

<rdrr name="LUR" dbtabname="RPT_LUR" tag="4042321925"
createtable="true">
  <options>
    <option property="tablespace" value="tspace1"/>
  </options>
  <fields>
    <field id="1" name="TIME_STAMP" type="TIMESTAMP">
    <!-- (other field declarations) -->
    <field id="10" name="DOWNSTREAM_VOLUME" type="UINT32"/>
    <field id="11" name="SESSIONS" type="UINT32"/>
  </fields>
  <indexes>
    <index name="RPT_LUR_I1" columns="END_TIME">
      <options>
        <option property="clustered" value="true"/>
        <option property="allowduprow" value="true"/>
        <option property="tablespace" value="tspace2"/>
      </options>
    </index>
  </indexes>
</rdrr>

```


We added the required tablespaces (*tspacel* and *tspac2*) for the index and table itself. 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, we modify *create_table.vm* as follows:

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

And to create the index in its own tablespace, we similarly modify **create_indexes.vm** as follows:

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

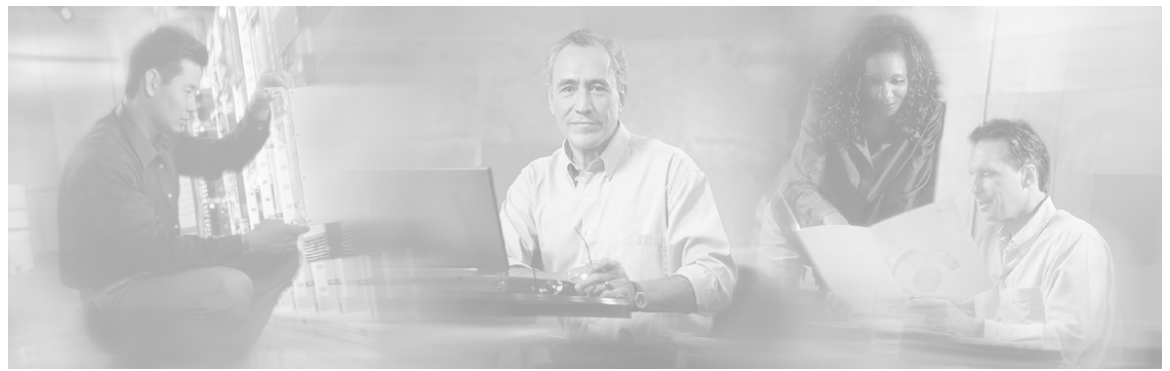
Using Table Partitioning

Suppose you want to implement rolling partitioning for a particular table on a weekly basis. You can control this option by creating a *partitioned* option for the table in the *tables.xml* file similar to the example described above. Then you can augment the *create_tables.vm* code as follows:

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

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

You will then want to create a *cron* job to roll the partitions (delete an old partition and create a new one) on a weekly basis. This *cron* job would run a script that would call the command-line interface of the JDBC adapter (as explained in *Using the JDBC Framework in Scripts* (on page 6-14)) to issue the appropriate *alter table drop partition* and *alter table add partition* SQL commands.



Glossary of Terms

A

Anonymous subscriber mode

A mode in which entities defined as an IP address(es) or VLAN(s) are treated as subscribers. The correlation to actual subscriber IDs is not performed by the system, but can be done externally by the collection system. Anonymous subscriber mode does not require an SM.

C

Collection Manager (CM)

A software application that is responsible for collecting RDRs from the SCE Platforms, processing them, and preparing them for reports.

Command Line Interface (CLI)

One of the management interfaces to the SCE Platform. It is accessed through a Telnet session or directly via the console port on the front panel of the SCE Platform

P

PQI (Service Control Application Installation) File

An application package file that is installed on the SCE Platform or associated software modules.

R

RDR (Raw Data Record)

A data record produced by the SCE Platform that reports on events in the traffic. RDRs produced by the SCE Platform are sent to the Collection Manager and then stored in the Collection Manager database or forwarded to third-party systems. The RDR typically contains quota (see Quota) request or reports service usage.

S

SCE Platform

The Service Control purpose-built hardware service component. This hardware device is capable of performing smart analysis of the packets at wire speed. It monitors the traffic on the line, producing raw data to be provided to the loaded application, which processes the data for functions such as reporting, policy management, subscription management, and implementation of tiered service subscriber aware traffic policies.

The SCE Platform comes in three models: SCE 1000 2xGBE, SCE 2000 4xGBE and SCE 2000 4/8xFE. There may be one or more SCE Platform on the provider network.

Service Control Application

An SML program that determines how the SCE Platform operates.

SLI (SML Loadable Image) File

An SLI file is a software package (part of an Service Control Application solution) that contains the SML application that is loaded onto a SCE Platform. The SML application determines the behavior of the SCE Platform. Different SCE Platforms can have different SML applications, even when they are within the same POP. (Operators do not need to access the SLI file.)

Subscriber

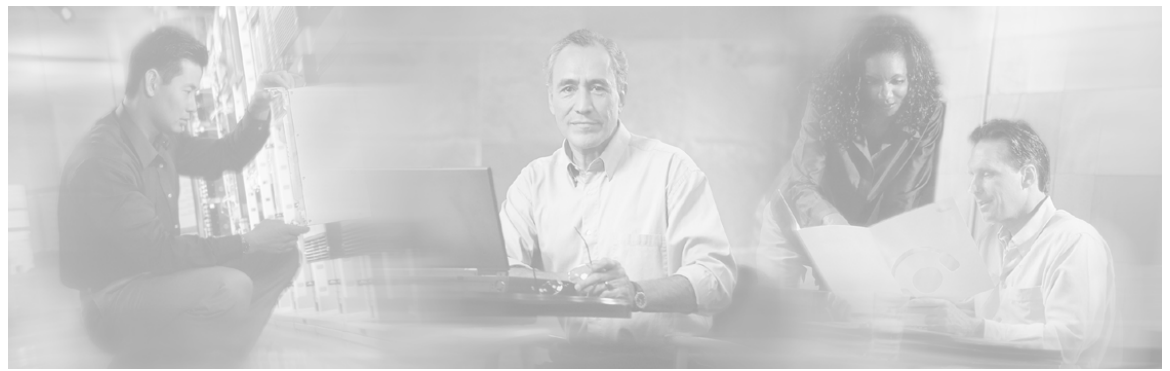
A Service Provider's client. This can refer to either an individual (single IP address) or a company (range of IP addresses).

Subscriber aware mode

A mode in which actual subscribers are defined in the system, thus requiring no external correlation to actual subscriber IDs.

Subscriber-less mode

A mode of the solution that requires no integration, so that the SM component is not required. This mode is not influenced by the number of subscribers or inbound IP addresses, therefore the total amount of subscribers utilizing the monitored link is unlimited from the perspective of the SCE Platform. It is the choice for sites where control and level analysis functions are required only at a global device resolution.



Index

A

- A Working Example • 6-11
- Activating the servers • 4-2
- Activating the Servers • 4-2
- ActivePython • 3-11
- adapterconf.py • 4-3
- Adapters • 2-4, 4-3
- alive.sh • 4-7
- Anonymous subscriber mode • 1
- Application Configuration • 6-7
- Applying the Periodic Deletion
 - Configuration File • 5-7
- Audience • v

B

- Backup
 - creating a backup of the database • 5-4
 - restoring a database from the backup • 5-4

C

- Categorizer • 2-3
- CD content • 3-5
- CD Content • 3-5
- Changes Made by install-dc • 3-11
- Changes Made by installsyb.sh • 3-10
- Changes Made by select-app.sh • 3-11
- Checking the database capacity • 4-6
- Checking the Database Capacity • 4-6
- Checking the RDR rate • 4-6
- Checking the RDR Rate • 4-6
- Checking the SCE Connection • 4-7
- Checking the SE connection • 4-7
- Cisco Service Control Specific Solutions • 1-6
- Cisco TAC Website • vii
- Code for • 6-12, 6-13

- Code for metadata query • 6-13
- Collection • 1-6
- Collection Manager (CM) • 1
- Comma escape for CSV files • 5-10
- Command Line Interface (CLI) • 1
- Comma-Separated Value Adapter • 2-4
- Configuration
 - Comma escape for CSV files • 5-10
 - CSV repository • 5-8
 - Data Collector • 4-1
 - Sybase Database • 5-1
- Configuration Files • 6-5
- Configuring the Collection Manager • 4-2
- Configuring the Comma Escape • 5-10
- Configuring the CSV File Repository • 5-8
- Configuring the Periodic Deletion of Old Records • 5-5
- Context Objects • 6-5
- Controlling the adapters • 4-3
- Controlling the Adapters • 4-3
- Controlling the database • 4-5
- Controlling the Database • 4-5
- Creating a backup of the database • 5-4
- Creating a Backup of the Database • 5-4
- CSV Repository File Structure • 5-8
- CSV repository, configuring • 5-8
- csvconf.sh • 5-8

D

- Data Collector

- Categorizer • 2-3
- configuring • 4-2
- default configuration settings • 3-13
- getting started • 3-1, 3-13
- how it works • 2-1
- installation • 3-1
- managing the CSV repository • 5-8
- managing the Data Collector • 4-1
- managing the database • 5-1
- Persistent Buffer • 2-3, 2-4
- Priority Queue • 2-3, 2-4
- RDR Server • 2-3
- software package • 2-3
- Sybase database • 2-12
- system requirements • 3-1
- using scripts for • 3-13
- verifying it is operational • 4-7
- Database Adapter • 2-4
- Database Configuration • 6-1
- DatabaseSee Sybase database • 2-12
- dbconf.sh • 4-5
- dbfree.sh • 4-6
- dbinfo Configuration • 6-12
- dbperiodic.conf • 5-6
- dbperiodic.py • 5-5, 5-7
- dbtables.py • 5-2
- Default configuration settings • 3-13
- Default Configuration Settings • 3-13
- Deleting a database table • 5-2
- Deleting a Table • 5-2
- Deleting old database records • 5-3
- Deleting Old Records • 5-3
- Document content • vi
- Document Content • vi
- Document Conventions • vi
- Dropping an SCE Connection • 4-5
- Dropping an SE connection • 4-5
- droptable.py • 5-2
- E**
- Editing the Periodic Deletion Configuration File • 5-6
- Escape for CSV files • 5-10
- Example 1 • 5-6
- Example 2 • 5-7
- Example-Viewing and Setting the SCE Timezone Offset • 6-15
- expert Option
- Usage Guidelines • 3-7
- F**
- Firewall • 3-5
- Flushing a Bucket • 2-7
- G**
- Getting started • 3-1, 3-13
- Getting Started • 3-13
- H**
- Handling of Database Tables • 2-6
- Hardware • 3-1, 3-4
- How the Collection Manager Works • 2-1
- How the Data Collector works • 2-1
- I**
- Installation • 3-1, 3-5
 - changes made by install-dc.sh • 3-11
 - changes made by installsyb.sh • 3-10
 - changes made by select-app.sh • 3-11
 - Data Collector • 3-5
 - Sybase database • 3-5
- install-dc.sh • 3-5, 3-11
- install-dc.sh, changes made by • 3-11
- Installing the CM and Getting Started • 3-1
- installsyb.sh • 3-5, 3-10
- installsyb.sh, changes made by • 3-10
- Introduction • 1-1
- IPFilter • 3-5
- J**
- JDBC Adapter • 2-4
- JRE 1.3.1_04 • 3-11
- L**
- Listing the database tables • 5-2
- Listing the Database Tables • 5-2
- M**
- Macro Definitions • 6-11
- Management and Collection • 1-4
- Managing the Collection Manager • 4-1
- Managing the CSV repository • 5-8
- Managing the CSV Repository • 5-8
- Managing the Data Collector • 4-1, 4-5
- Managing the Database • 5-1
- Managing the Database and CSV Repository • 5-1
- Managing the Sybase database • 5-1

Monitoring the Collection Manager • 4-5

N

Network Management • 1-5

O

Obtaining Full Debug Information • 6-14

Obtaining Technical Assistance • vii

on-boot.py • 4-2

Opening a TAC Case • vii

Options • 4-3, 4-4, 4-5, 5-2, 5-3, 5-4, 5-7, 5-9

Overview • 1-1

P

Parsing a String • 6-14

Periodic deletion of old records • 5-5, 5-7

Persistent Buffer • 2-3, 2-4

Phase 1

 Installing Sybase • 3-6

Phase 2

 Installing Service Control Software • 3-8

Ports Used by the Collection Manager

 Software • 3-11

Ports, used by Data-Collector software • 3-11

pqb_admin • 3-5

PQI (Service Control Application Installation) File • 1

Preface • v

Priority Queue • 2-3, 2-4

Priority Queues and Persistent Buffers • 2-4

prunetable.py • 5-3

Purpose • v

Purpose of this guide • v

Python/Sybase bridge • 3-11

Q

Quick Start Guide • 6-2

R

Raw Data Records (RDRs) • 2-2

 adapters • 2-4

 RDR Server • 2-3

 types sent by SE Platform • 2-2

Raw Data Records and Reports • 2-2

RDR (Raw Data Record) • 1

RDR Server • 2-3

rdr-rate.py • 4-6

RDRSee Raw Data Records (RDRs) • 2-2

Real-Time Aggregating Adapter • 2-7

Real-Time Aggregating Adapter

 Configuration Example • 2-9

Real-Time Aggregating Adapter CSV Files • 2-8

Redhat Linux Requirements • 3-4

Related publications • vi

Related Publications • vi

Report tables, verifying they are operational • 4-7

Reporter application • 5-2

Reports • 2-2

Restoring a database from the Backup • 5-4

Restoring a Database from the Backup • 5-4

S

Scalability-Hints for Oracle • 6-16

SCE Connection

 checking • 4-7

 dropping a connection • 4-5

SCE Platform • 1

Scripts

 adapterconf.py • 4-3

 alive.sh • 4-7

 csvconf.sh • 5-8

 dbconf.sh • 4-5

 dbfree.sh • 4-6

 dbperiodic.py • 5-5, 5-7

 dbtables.py • 5-2

 droptable.py • 5-2

 help • 4-1

 install-dc.sh • 3-5, 3-11

 installsyb.sh • 3-5, 3-10

 on-boot.py • 4-2

 prunetable.py • 5-3

 rdr-rate.py • 4-6

 seconf.py • 4-5, 4-7

 select-app.sh • 3-5, 3-11

 sybback.sh • 5-4

 sybrestore.sh • 5-4

 usage instructions • 4-1

 using to manage the CSV repository • 5-8

 using to manage the Data Collector • 4-1, 4-2, 4-5

 using to manage the database • 5-1

seconf.py • 4-5, 4-7

select-app.sh • 3-5, 3-11

select-app.sh, changes made by • 3-11

Servers, activating • 4-2

Service Configuration Management • 1-6

- Service Control Application • 1
- Service Control Application Suite for Broadband • 1-7
- Service Control Application Suite for Mobile • 1-7
- Service Control Capabilities • 1-4
- Service Control for Cable MSOs • 1-2
- Service Control for DSL Providers and ISPs • 1-2
- Service Control for Wireless Service Providers • 1-2
- Setting the Locale and Time Zone • 3-3
- Setting the Locale and Timezone • 3-5
- SLI (SML Loadable Image) File • 2
- Software and Environment • 3-4
- Software and Environment • 3-2
- Solaris Requirements • 3-1
- SQL Definitions • 6-12
- SSH • 3-5, 4-1
- Subscriber • 2
- Subscriber aware mode • 2
- Subscriber Management • 1-6
- Subscriber-less mode • 2
- Sybase database • 3-5
 - checking capacity • 4-6
 - checking the RDR rate • 4-6
 - controlling • 4-5
 - creating a backup of • 5-4
 - Database Adapter • 2-4
 - dbperiodic.conf • 5-6
 - deleting old database records • 5-3
 - deleting tables from • 5-2
 - generating list of tables • 5-2
 - installation • 3-1
 - managing • 5-1
 - periodic deletion of records • 5-5, 5-7
 - restoring from the backup • 5-4
 - server • 3-10
 - system requirements • 3-1
 - using scripts for managing • 3-13
 - verifying it is operational • 4-7
- sybback.sh • 5-4
- sybrestore.sh • 5-4
- System
 - verifying that it is operational • 4-7
- System requirements • 3-1
- System Requirements • 3-1

T

- TAC Case Priority Definitions • vii

- Telnet • 4-1
- Testing and Debugging • 6-13
- The Bundled Database • 2-12
- The Cisco Service Control Concept • 1-1
- The Collection Manager Software Package • 2-3
- The Data Collection Process • 2-1
- The dbinfo object • 6-6
- The SCE Platform • 1-3
- The tables object • 6-6
- The tables.dtd file • 6-10
- The tables.xml file • 6-8
- The Template Language • 6-4
- The tools object • 6-6
- Topper/Aggregator Adapter • 2-5
- Topper/Aggregator Adapter CSV Files • 2-5
- Topper/Aggregator Cycles • 2-5

U

- Uninstalling the Sybase Database and the Service Control Software • 3-12
- Using an External Database • 2-13
- Using Custom tablespaces • 6-16
- Using Table Partitioning • 6-17
- Using the JDBC Framework in Scripts • 6-14
- Using Utility Scripts • 4-1

V

- Verifying that the Server is Operational • 4-7
- Viewing the Currently Active Periodic Deletion Configuration File • 5-5