CHAPTER 2

How the Collection Manager Works

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

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

The Data Collection Process

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

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

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

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

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

Raw Data Records

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

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

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

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

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

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

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

  The SCA BB application Transaction RDRs are in this category.

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

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

Information About the Collection Manager Software Package

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

- RDR Server, page 2-2
- Categorizer, page 2-3
- Priority Queues and Persistent Buffers, page 2-3

RDR Server

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

Categorizer

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

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

Priority Queues and Persistent Buffers

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

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

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

Information About Adapters

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

- JDBC Adapter, page 2-3
- CSV Adapter, page 2-4
- TA Adapter, page 2-4
- RAG Adapter, page 2-5
- HTTPC Adapter, page 2-6

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

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

JDBC Adapter

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

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

The JDBC Adapter is preconfigured to support the following databases:

- Sybase ASE 12.5 and 15.0
- Oracle 9.2 and 10

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

```
recyclebin=off
```

- MySQL 4.1 and 5

CSV Adapter

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

TA Adapter

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

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

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

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

TA Adapter Cycles

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

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

The activities in each cycle differ slightly, as follows:

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

TA Adapter Memory Requirements

It is necessary to dedicate a sufficient amount of memory to the TA Adapter. Configure the value in the **cm.conf** configuration file in the following location:

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

The recommended amount of memory to be dedicated to the TA Adapter can be calculated using the following formula:

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

Where:

- NUM_SUBSCRIBERS is the number of new subscribers that will be introduced in a day (on all SCEs sending reports to this CM).

  This is usually a high number; especially when working in anonymous subscriber mode.

  The following command provides an estimate of the number of subscribers that are known to the CM:

  ```
  ~/setup/mbean.py --getattr=Subscribers DCAdapters
  ```

- AVG_SUBS_ID_LENGTH is the average character length of a subscriber.

  In most cases, this is approximately 20.

- NUM_SERVICES is the number of subscriber usage counters and is configured in the **taadapter.conf** configuration file.
- NUM_TOP_ENTRIES is configured in the **taadapter.conf** configuration file under the *num_top_entries* value.

Note The configured memory should *not* be over 4 GB because the TA Adapter process is a 32-bit process and cannot address memory over 4 GB.

RAG Adapter

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

- RAG Adapter AggregationBuckets, page 2-5
- Flushing a Bucket, page 2-6

RAG Adapter AggregationBuckets

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

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

Example:

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

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

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

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

Flushing a Bucket

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

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

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

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

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

HTTPC Adapter

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

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

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

Databases

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

- Using the Bundled Database, page 2-7
- Using an External Database, page 2-7

Using the Bundled Database

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

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

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

Using an External Database

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