Cisco Universal Gateway Call Analyzer User Guide

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

Text Part Number: OL-3257-01

Cisco Universal Gateway Call Analyzer User Guide

CONTENTS

Preface

Cisco Universal Gateway Call Analyzer (Cisco UGCA) is a lightweight reporting application that is dedicated to detecting and troubleshooting Cisco AS5000 series universal gateways used in dial deployments. For additional discussion of the application and its architecture, see Chapter 1, "Introduction to Cisco UGCA." For gateway, client, and server system requirements, see Chapter 2, "Installing Cisco UGCA."

This preface presents the following major topics:

- Document and Application Release
- Audience
- Scope
- Document Organization
- Obtaining Documentation
- Obtaining Technical Assistance

Document and Application Release

This is the first release of this document, which covers Release 1.0 of Cisco UGCA application software. Document version history is detailed below.

Document Version	Application Release	Date	Notes
1	1.0	10/14/02	This document was first released as a beta draft.
2	1.0	12/20/02	This document was released in final form.
3	1.0(1)	03/26/03	This document was released with minor updates.

Audience

The target audience for this document is assumed to have basic knowledge in the following areas:

- Familiarity with basic UNIX commands and operations
- Familiarity with the operation of Cisco AS5000 series universal gateways
- Familiarity with syslog and Cisco IOS CallTracker
- Familiarity with dial modems and troubleshooting their operation in service provider networks

Scope

This document deals with the fundamentals of installing and maintaining the Cisco UGCA application. For help in using the application as it is running, refer to the online help that comes with the product.

Document Organization

The major sections of this document are as follows:

Section	Title	Major Topics
Preface	Preface	Brief product overview and links to resources
Chapter 1	Introduction to Cisco UGCA	Application architecture, basic tasks, report types
Chapter 2	Installing Cisco UGCA	Hardware and software requirements, essential gateway configurations, installation on the server
Chapter 3	Starting, Using, and Maintaining Cisco UGCA	Launching the application, basic housekeeping
Appendix A	Cisco UGCA Configuration File: ugca.xml	A listing of the annotated configuration file for the application
Appendix B	BNF for Markup Interpreter: Terminals and Nonterminals	Terminals and nonterminals (compiler syntax) used by the markup interpreter
Appendix C	Cisco UGCA Database Schema	Discussion and examples of how the application stores call record data.
Appendix D	Cisco UGCA Markup: markup.xml and call_record.discid	A text listing of the file markup.xml, with details of the disconnect attributes
Appendix E	Cisco UGCA Enumeration Types: enum.xml	A text listing of the file enum.xml.

Obtaining Documentation

These sections explain how to obtain documentation from Cisco Systems.

World Wide Web

You can access the most current Cisco documentation on the World Wide Web at this URL:

http://www.cisco.com

Translated documentation is available at this URL:

http://www.cisco.com/public/countries_languages.shtml

Documentation CD-ROM

Cisco documentation and additional literature are available in a Cisco Documentation CD-ROM package, which is shipped with your product. The Documentation CD-ROM is updated monthly and may be more current than printed documentation. The CD-ROM package is available as a single unit or through an annual subscription.

Ordering Documentation

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Networking Products MarketPlace:

  http://www.cisco.com/cgi-bin/order/order_root.pl

- Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:

  http://www.cisco.com/go/subscription

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, U.S.A.) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can submit comments electronically on Cisco.com. In the Cisco Documentation home page, click the **Fax** or **Email** option in the "Leave Feedback" section at the bottom of the page.

You can e-mail your comments to bug-doc@cisco.com.

You can submit your comments by mail by using the response card behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Document Resource Connection
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain online documentation, troubleshooting tips, and sample configurations from online tools by using the Cisco Technical Assistance Center (TAC) Web Site. Cisco.com registered users have complete access to the technical support resources on the Cisco TAC Web Site.

Cisco.com

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information, networking solutions, services, programs, and resources at any time, from anywhere in the world.

Cisco.com is a highly integrated Internet application and a powerful, easy-to-use tool that provides a broad range of features and services to help you with these tasks:

- Streamline business processes and improve productivity
- Resolve technical issues with online support
- Download and test software packages
- Order Cisco learning materials and merchandise
- Register for online skill assessment, training, and certification programs

If you want to obtain customized information and service, you can self-register on Cisco.com. To access Cisco.com, go to this URL:

http://www.cisco.com

Technical Assistance Center

The Cisco Technical Assistance Center (TAC) is available to all customers who need technical assistance with a Cisco product, technology, or solution. Two levels of support are available: the Cisco TAC Web Site and the Cisco TAC Escalation Center.

Cisco TAC inquiries are categorized according to the urgency of the issue:

- Priority level 4 (P4)—You need information or assistance concerning Cisco product capabilities, product installation, or basic product configuration.
- Priority level 3 (P3)—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.
- Priority level 2 (P2)—Your production network is severely degraded, affecting significant aspects of business operations. No workaround is available.
- Priority level 1 (P1)—Your production network is down, and a critical impact to business operations will occur if service is not restored quickly. No workaround is available.

The Cisco TAC resource that you choose is based on the priority of the problem and the conditions of service contracts, when applicable.

Cisco TAC Web Site

You can use the Cisco TAC Web Site to resolve P3 and P4 issues yourself, saving both cost and time. The site provides around-the-clock access to online tools, knowledge bases, and software. To access the Cisco TAC Web Site, go to this URL:

http://www.cisco.com/tac

All customers, partners, and resellers who have a valid Cisco service contract have complete access to the technical support resources on the Cisco TAC Web Site. The Cisco TAC Web Site requires a Cisco.com login ID and password. If you have a valid service contract but do not have a login ID or password, go to this URL to register:

http://www.cisco.com/register/

If you are a Cisco.com registered user, and you cannot resolve your technical issues by using the Cisco TAC Web Site, you can open a case online by using the TAC Case Open tool at this URL:

http://www.cisco.com/tac/caseopen

If you have Internet access, we recommend that you open P3 and P4 cases through the Cisco TAC Web Site.

Cisco TAC Escalation Center

The Cisco TAC Escalation Center addresses priority level 1 or priority level 2 issues. These classifications are assigned when severe network degradation significantly impacts business operations. When you contact the TAC Escalation Center with a P1 or P2 problem, a Cisco TAC engineer automatically opens a case.

To obtain a directory of toll-free Cisco TAC telephone numbers for your country, go to this URL:

http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml

Before calling, please check with your network operations center to determine the level of Cisco support services to which your company is entitled: for example, SMARTnet, SMARTnet Onsite, or Network Supported Accounts (NSA). When you call the center, please have available your service agreement number and your product serial number.

CHAPTER 1

Introduction to Cisco UGCA

This chapter introduces the Cisco UGCA and presents the following topics:

- Overview
- Supported Gateways and Required Cisco IOS Releases
- Client/Server Architecture
- Basic Cisco UGCA Tasks
- Report Fundamentals
- Cisco UGCA Report Types, Report Specifications, and the Export Function

Overview

Cisco UGCA uses call characteristic data to detect service-affecting degradations in call-connection success and call quality, and assists in pinpointing root causes. Data in the form of syslog messages is collected and stored on database servers. A commercial Web browser client allows the user to create, edit, view, run, and export reports for analysis. The application analyzes call detail records (CDRs) that contain the call characteristics.

Note Cisco UGCA is not an element management system (EMS) for Cisco AS5000 series platforms. Rather, it is an application that complements such products as Cisco Universal Gateway Manager (UGM). If you require an EMS, refer to Cisco UGM product information at the following URL:

http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/ugm/

Cisco UGCA provides call characteristic reports that classify historical calls by metrics that indicate call quality. If a large percentage of calls fail or have poor quality, the user can generate call-connection reports according to resources and characteristics to determine if the problems are associated with a particular trunk, a modem or universal port, a calling or called prefix (an NPA-NXX number in North America), a service type, or other characteristics. Reports can be run on call volume, call duration, call disconnect reason, Tx/Rx bytes, Tx/Rx rates, modulation type, modem retrains, signal-to-noise ratio, signal quality, and call connection time. A relational database holds the raw data in a documented schema that can be queried.

Supported Gateways and Required Cisco IOS Releases

Table 1-1 lists the gateways and Cisco IOS releases that Cisco UGCA supports. The application requires Cisco IOS Call Tracker (also known as Cisco Call Tracker), which is available on all Cisco AS5000 series platforms since Cisco IOS Release 12.1(3)T. Cisco UGCA supports reports on data collected in Cisco IOS Call Tracker call records and modem call records. (See Role of Cisco IOS Call Tracker, page 1-9.)

Note Refer to the release notes for this product for any known issues specific to a given release.

Table 1-1 Supported Gateways and Required Cisco IOS Releases

Gateway	Cisco IOS Release
Cisco AS5300	12.1(3)T and later
Cisco AS5350	
Cisco AS5400	
Cisco AS5800	
Cisco AS5850	

For supporting documentation, go to the following URLs:

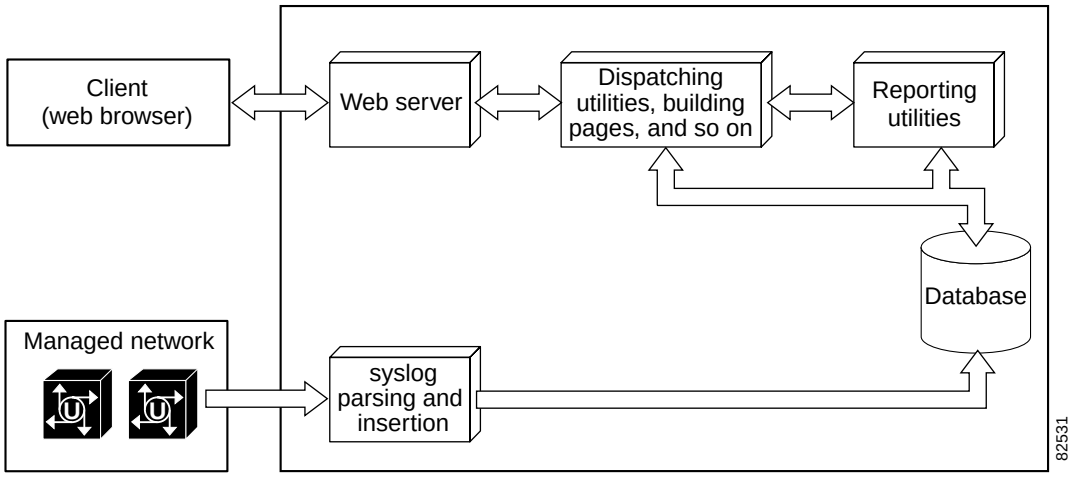
- Cisco AS5300
  http://www.cisco.com/univercd/cc/td/doc/product/access/acs_serv/5300/index.htm
- Cisco AS5350
  http://www.cisco.com/univercd/cc/td/doc/product/access/acs_serv/as5350/index.htm
- Cisco AS5400
  http://www.cisco.com/univercd/cc/td/doc/product/access/acs_serv/as5400/index.htm
- Cisco AS5800
  http://www.cisco.com/univercd/cc/td/doc/product/access/acs_serv/as5800/index.htm
- Cisco AS5850
  http://www.cisco.com/univercd/cc/td/doc/product/access/acs_serv/as5850/index.htm

Client/Server Architecture

Figure 1-1 illustrates the client-server architecture of Cisco UGCA. The client is a commercial Web browser. The Cisco IOS Call Tracker feature (see Role of Cisco IOS Call Tracker, page 1-9) is used to obtain syslog messages from the gateways in the managed network. Both call records and modem call records from the syslog messages are then parsed and loaded into the database as call data records (CDRs), by means of a utility known as CDR Loader.

Note Version 1.0(1) of Cisco UGCA supports only Microsoft Internet Explorer as the client browser and Sun Solaris as the backend operating system. See Prerequisites, page 2-1.

Figure 1-1 Cisco UGCA Architecture



Basic Cisco UGCA Tasks

Table 1-2 summarizes basic Cisco UGCA configuration, installation, maintenance, and troubleshooting tasks discussed in this user guide.

Table 1-2 Basic Cisco UGCA Tasks

Category	Task
Configuration	• Disable Sun JVM, page 2-3 • Enable Cookies and Java Script, page 2-3 • Restart the Client Computer, page 2-4 • Enable Cisco IOS Call Tracker and syslog on the Gateways, page 2-4 • Configure Timing on Gateways, page 2-5
Installation	• Determine Free Disk Space, page 2-6 • Install the Application Package, page 2-6
Starting	• Launching Cisco UGCA from a Browser, page 3-1

Table 1-2 Basic Cisco UGCA Tasks (continued)

Category	Task
Maintenance	• Purging Data, page 3-17 • Backing Up the System, page 3-18 • Restoring the System, page 3-19 • Managing crontab Entries for Backups and Purges, page 3-22 • Stopping Cisco UGCA, page 3-19 • Resetting the Cisco UGCA Database, page 3-21 • Starting Cisco UGCA, page 3-22 • Removing Cisco UGCA, page 3-22 • Loading CDR Data, page 3-23 • Maintaining Markup Data, page 3-26 • Managing the Volume of Log Files, page 3-34 • Accessing and Reading Raw Data, page 3-34 • Changing Database Account Passwords, page 3-34
Troubleshooting	• Optimizing System Performance, page 3-39

Other tasks, such as working with reports and administering user passwords and access privileges, are discussed in the online help within the application.

Cisco UGCA Report Types, Report Specifications, and the Export Function

This section lists and describes the basic report types supplied with the application, followed by useful preconfigured report specifications for call volume and modem call volume. All report types report by DS1 or modem slot/port for any specific gateway or set of gateways. The export function exports each item in a report result. By default, a variety of items are marked up for each CDR (call detail record).

Report Fundamentals

There are three major components to the report processing within Cisco UGCA:

- Report type

  A template for a report specification. Report types define the basic structure of a report, and are predefined with the shipped product.

- Report specification

  A named persistent definition of a query, which produces a report result when invoked. Every report has a name, a description, a definition of a query, and a time-coverage parameter associated with it. Report specifications can be run one or more times.

- Report result

  The result of a single invocation of a report specification. Report results are stored as XML documents, with name and contents identifying the report type and the time of invocation, along with the results. Results can be single numbers (scalar values) or tables.

Although the report types are static, the report specifications that are generated from them are not. The report specifications supplied with the product are a set of useful specifications that can cover a wide range of needs. You can copy and edit existing report specifications to meet local requirements, as well as create new ones.

This section presents the following topics:

- Basic Report Types
- CallVolume Report Specifications
- ModemCallVolume Report Specifications
- About CDR Markup
- How Modem and Nonmodem Report Types Are Differentiated
- Export File Format

Basic Report Types

Table 1-3 lists, in alphabetic order, the basic report types supported by Cisco UGCA, along with descriptions.

Note For a technical discussion of how modem and nonmodem report types are differentiated, see How Modem and Nonmodem Report Types Are Differentiated, page 1-8.

Table 1-3 Report Types Supplied by Cisco UGCA

Report Type	Description
CallDuration[1]	Average, minimum, maximum, and sum of call durations. The report uses duration buckets to group calls by user-configurable duration intervals.
CallVolume	Total number of calls and percentage, with breakdown by type of service and modulation type. Type of service includes PPP, multilink PPP, L2TP, tcpClear, and other. For details, see CallVolume Report Specifications, page 1-6.
ModemCallDuration[1]	Average, minimum, maximum, and sum of call durations. The report uses duration buckets to group calls by user-configurable duration intervals.
ModemCallVolume	Total number of calls and percentage with breakdown by type of service and modulation type. Type of service includes PPP, multilink PPP, L2TP, tcpClear, and other. For details, see ModemCallVolume Report Specifications, page 1-7.
ModemRetrains	Average, minimum, and maximum number of local and remote retrains and retrain failures.
ModemSelectAll	Raw selection of all columns from a join[2] of call_record, modem_call_rec, and markup CDR tables. Attributes to specify conditions are subsets of the above columns. No groupings are available.
ModemServiceTimes	Average, minimum, and maximum average connection, physical layer steady-state, service establishment, and user-validation times.
ModemSignalQuality	Average, minimum, and maximum of signal quality.
ModemSignalToNoiseRatio	Average, minimum, and maximum of signal-to-noise ratio.
ModemTxRxChars[1]	Average, minimum, and maximum number of characters transmitted and received.
ModemTxRxRates	Average, minimum, maximum, and final receive and transmit rates.
SelectAll	Raw selection of all columns from a join of *call_record* and markup CDR tables. Attributes to specify conditions are subsets of the above columns. No groupings are available.
ServiceTimes	Average, minimum, and maximum average connection, physical layer steady-state, service establishment, and user-validation times.
TxRxChars[1]	Average, minimum, and maximum number of characters transmitted and received.
TxRxRates	Average, maximum, and minimum initial and final receive and transmit rates.

1. CallDuration and TxRxChars appear in both call_record and modem_call_rec. The values used depend on which report type—modem or nonmodem—is selected. See How Modem and Nonmodem Report Types Are Differentiated, page 1-8.
2. Join means an inner join, which combines records from two tables whenever there are matching values in a common field.

CallVolume Report Specifications

Table 1-4 lists, in alphabetical order, the CallVolume report specifications supplied by Cisco UGCA, along with descriptions.

Note The following are preconfigured examples judged to be useful. Users are encouraged to tailor them to the local environment.

Table 1-4 CallVolume Report Specifications Supplied by Cisco UGCA

Report Specification	Description
CallVolumeByCall Category	Total number of calls grouped by call category (such as modem, V.110, CasDigital, IsdnSync, etc.) with breakdown by type of service and modulation type.
CallVolumeByNode	Total number of calls and percentage of calls grouped by node. Percentage is calculated as a percentage of each call classification for a single node.
SuccessfulCallVolume	Total number of successful calls with breakdown by type of service and modulation type. A successful call is defined as a call that has a service successfully set up.

ModemCallVolume Report Specifications

Table 1-5 lists, in alphabetic order, the basic ModemCallVolume distribution report specifications supplied by Cisco UGCA, along with descriptions. All distributions are by number and percentage of calls.

Note The following are preconfigured examples judged to be useful. Users are encouraged to tailor them to the local environment.

Table 1-5 ModemCallVolume Report Specifications Supplied by Cisco UGCA

Report Specification	Description
CallClassAndDiscIdAnd DiscReasonGrDistribution	Distribution of modem calls by call classification, IOS disconnection ID, and grouped *disc_reason*. Call classification is assigned to each call based on a set of configurable rules.
DisconnectReason Distribution	Distribution of call disconnect reason.
FailedRetrainsDistribution	Distribution of failed retrains.
FinalModulation Distribution	Distribution of final modulation.
FinalRxRateDistribution	Distribution of final receive rates.
FinalTxRateDistribution	Distribution of final transmit rates.
GrantedRetrains Distribution	Distribution of granted retrains.
InitialModulation Distribution	Distribution of initial modulation.
InitialRxRateDistribution	Distribution of initial receive rates.
InitialTxRateDistribution	Distribution of initial transmit rates.
LocalRetrainsDistribution	Distribution of local retrains.
ModemCallVolumeByNode CallingPrefixCallClass	Number and percentage of calls grouped by node, calling prefix, and call classification. Call classification is assigned to each call based on a set of configurable rules. Percentage is calculated as a percentage of each call classification for a node/calling_prefix pair.
RemoteRetrains Distribution	Distribution of remote retrains.

About CDR Markup

The use of markup facilitates reporting on data aspects that are not immediately available from the raw data. By default, Cisco UGCA provides markup for the following call-related data aspects in each CDR, listed in alphabetic order:

- Call classification
- Call record call duration bucket
- Called prefix
- Calling prefix
- Disconnect reason group
- Modem call record call duration bucket
- User group

For the criteria that are used to mark up each of these aspects, refer to Appendix D, "Cisco UGCA Markup: markup.xml and call_record.discid."

How Modem and Nonmodem Report Types Are Differentiated

Generally speaking, nonmodem report types are of interest when one is looking at the basic characteristics of all call types, whereas modem report types provide in-depth characteristics specific to modem calls.

The differences between, for example, ServiceTime and ModemServiceTime report types, are two-fold, as discussed below. (The same principle applies to CallDuration and ModemCallDuration, CallVolume and ModemCallVolume, and so on.)

The first difference concerns the scope of the data going into the query. The second report runs over an "inner join" operation with modem call records. (An inner join combines records from two tables whenever there are matching values in a common field.) The number of records (and therefore results) is less than the total number of call records. For example, ISDN data calls do *not* have entries in a modem_call_rec table. The modem report *narrows* the scope of data passed into the report query, limiting the data to what is relevant to modem calls only.

The second difference concerns the definition of the report specification, that is, the number of modem-specific fields (attributes) in modem_call_rec that are selected for possible conditions and grouping. In this case, a modem report *widens* the scope of available fields, thus increasing the breadth of the resulting table. This is because modem calls have more fields (all the columns in modem_call_rec) that are irrelevant to nonmodem calls.

Export File Format

The export function exports each item in the report result. The export file format is as follows:

```
ReportResultName,<name of the report result>
ReportSpecName,<report spec name of the result>
ReportTypeName,<report type name of the result>
Description,<description>
TimeRangeCoverage From,<the from time of the time coverage>
TimeRangeCoverage Till,<the till time of the time coverage>
ExecutionStartTimeDate,<time when report execution started>
ExecutionEndTimeDate,<time when report execution ended>
Status,<report result status>
```

```
QueryResult Description,<description of the query result>
QueryResult SQLQuery,<the SQL query used to generate this result>
QueryResult ResultType,<result type of the query result>
QueryResult NumOfColumns,<number of columns>
QueryResult NumOfRows,<number of rows>
<col name 1>,<col name2>, ......,<col name n>
<value of col 1 row 1>,<value of col 2 row 1>,......,<value of col n row 1>
......
<value of col 1 row m>,<value of col 2 row m>,......,<value of col n row m>
```

Role of Cisco IOS Call Tracker

For every call that is terminated, the gateway sends one record—a Cisco IOS Call Tracker call record—to the configured logging host through the syslog logging protocol. If the call is a modem call, the gateway sends another record—a Cisco IOS Call Tracker modem record—to the logging host. The CDR Loader component of Cisco UGCA reads the syslog file, then parses the call record and modem call record and loads them into the database.

Both Cisco IOS Call Tracker and syslog logging must first be enabled on the gateways. See Enable Cisco IOS Call Tracker and syslog on the Gateways, page 2-4.

Understanding Cisco IOS Call Tracker Outputs

For additional information about Cisco IOS Call Tracker, refer to *Understanding Call Tracker Outputs* at the following URL:

http://www.cisco.com/warp/public/471/calltracker_view.html

The above document provides useful links to the following documents that are useful in troubleshooting disconnect reason codes (cause codes) for MICA and NextPort modems:

- *MICA Modem States and Disconnect Reasons*

  http://www.cisco.com/warp/public/76/mica-states-drs.html

- *Interpreting NextPort Disconnect Reason Codes*

  http://www.cisco.com/warp/public/471/np_disc_code.html

CHAPTER 2

Installing Cisco UGCA

For Cisco UGCA to work as a system, a variety of requirements must be met in addition to installing the software. This chapter presents the following major sections:

- Prerequisites
- Installing Cisco UGCA Components on the Server

Note You must have root privileges to install Cisco UGCA and administer its functions.

Prerequisites

Cisco UGCA System Requirements

This section summarizes the hardware and software requirements for the Cisco UGCA server.

Note Make sure that the server is properly connected to a network.

Hardware Requirements

Table 2-1 lists the hardware requirements for the Cisco UGCA server.

Table 2-1 Hardware Requirements for the Cisco UGCA Server

Resource	Cisco UGCA Server
Hardware	Sun Ultra 10 workstation (minimum)
Processor	UltraSparc
Operating system	Solaris 8
Memory	512 MB
Free disk space	4.6 GB (see Free Disk Space Requirements, page 2-6). See Note below.

Cisco UGCA was tested to varying degrees on a number of Sun Microsystems platforms, including the Ultra 10 workstation, the Ultra 60 workstation, the Enterprise 420R server, and the SunFire 280R server. Although Cisco UGCA was optimized for a low-end Ultra 10 and was most thoroughly tested on that platform, the application should run successfully in most Sun Solaris 8 environments.

Note Cisco UGCA does not perform file management. It is the responsibility of the user to ensure that there is always enough disk space in the system. Disk space is consumed most when CDRLoader is running, reports are being executed, and backups are being performed.

The value of 4.6 GB is calculated to provide enough space for seven backups. During installation, the user has the option of determining whether backups are to be performed nightly. If this is elected, a new backup file is created every night. If desired, the backup directory configuration in the file *ugca.xml* (Appendix A, "Cisco UGCA Configuration File: ugca.xml") can be edited to reroute the backup files to a different location.

Software Requirements

The following table lists the software requirements:

Table 2-2 Software Requirements for the Cisco UGCA Server

Software	Version
Sun Solaris patches	Latest recommended patch cluster Solaris 8 and J2SE. Refer to the following URL: http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access
J2SE recommended patch cluster for Solaris 8 patches	
Client browser (recommended)	Microsoft Internet Explorer (IE) 5.5 or 6.0 with Microsoft Java VM on Windows 2000. See Browser Requirements, below.

Note This document does not detail the installation of the Sun Solaris operating system and patches. Refer to Sun documentation.

Browser Requirements

Before you can use Cisco UGCA, you must ensure that a web browser is installed. Cisco UGCA release 1 supports Microsoft Internet Explorer (IE) only. This document does not address the installation of the browser.

Note Both Microsoft IE version 5.5 and version 6.0 were tested for this release. Cisco recommends that you use those versions. You also need to enable the Microsoft Java Virtual Machine and cookies.

Do not use the Java Virtual Machine (JVM) provided by Sun. This is installed if the JDK (Java Developer's Kit) is installed. You also need to enable cookies and Java Script. Finally, you must restart the client machine. The following procedures are detailed below:

- Disable Sun JVM
- Enable Cookies and Java Script
- Restart the Client Computer

Disable Sun JVM

Microsoft IE Version 5.5 and 6.0

If the Sun JVM has already been enabled in either version, you must do the following to disable it in Microsoft IE version 5.5 and 6.0:

Step 1 Choose **Tools > Internet Options** and click the **Advanced** tab.

Step 2 In the Settings list, look for **Java (Sun)**.

Step 3 If **Java (Sun)** is not present, you are finished with this procedure. Proceed to Enable Cookies and Java Script, below.

Step 4 If **Java (Sun)** is present, do the following:

a. Ensure that the option **Use Java 2.v.1.4.0_01 for <applet> (requires restart)** is *not* selected.

b. Click **Apply**. Do not leave the **Internet Options** window.

c. Proceed to Enable Cookies and Java Script, below.

Enable Cookies and Java Script

To enable cookies and Java Script in Microsoft IE versions 5.5 and 6.0, do the following:

Microsoft IE Version 5.5

To enable cookies and Java Script in Microsoft IE version 5.5, do the following:

Step 1 Still within the Internet Options window, click the **Security** tab.

Step 2 Depending on where you are accessing Cisco UGCA from, click either the **Internet** or **Local intranet** icon.

Step 3 Click **Custom Level**. The **Security Settings** dialog box opens.

Step 4 Look for **Cookies**.

Step 5 Ensure that **Allow per-session cookies (not stored)** is enabled.

Note You can also enable **Allow cookies that are stored on your computer**, although this is not mandatory.

Step 6 Still within the **Security Settings** dialog box, look for **Scripting**.

Step 7 Ensure that the following are enabled: **Active scripting** and **Scripting of Java applets**.

Step 8 Click **OK** to apply changes.

Step 9 Proceed to Restart the Client Computer, below.

Microsoft IE Version 6.0

Do the following to enable cookies in Microsoft IE version 6.0.

Step 1 Choose **Tools** > **Internet Options** > **Privacy** and click the **Advanced** tab.

Step 2 There are two sets of options: First-party Cookies and Third-party Cookies. Under both columns, ensure that Accept is selected.

Restart the Client Computer

For the above changes to take effect in Microsoft IE, you must restart your computer.

Gateway Provisioning Requirements

You must enable logging and the Cisco IOS Call Tracker feature on all gateways from which you want to receive reports. Additional configurations you must make are discussed below in the following sections:

- Enable Cisco IOS Call Tracker and syslog on the Gateways
- Configure Timing on Gateways

Enable Cisco IOS Call Tracker and syslog on the Gateways

To enable the Cisco IOS Call Tracker feature and syslog logging, enter the following in global configuration mode. See Role of Cisco IOS Call Tracker, page 1-9.

Step 1 Enable time stamps, with logging and debugging:

```
service timestamps log datetime
service timestamps debug datetime
```

Note See Configure Timing on Gateways, page 2-5, for a discussion of network timing issues.

Step 2 Enable the Cisco IOS Call Tracker feature.

a. Enable Cisco IOS Call Tracker:

```
calltracker enable
```

b. Enable the terse option:

```
calltracker call-record terse
```

Note The **terse** option generates two records: a call record and a modem call record. If the call is not a modem call, only a call record is generated. If both **modem call-record terse** and **calltracker call-record terse** have been enabled on a gateway, Cisco UGCA ignores the modem call record and just parses and processes the Cisco IOS Call Tracker call record and modem call record.

Step 3 Enable syslog to the IP address of the Cisco UGCA server:

```
logging <Cisco_UGCA_server_IP_address>
```

Configure Timing on Gateways

Ensuring proper timing is essential to the establishment of accurate start and stop intervals for any reporting scheme. The Network Time Protocol (NTP) provides a common time base for networked routers, servers, and other devices. A synchronized time enables you to correlate syslog and Cisco IOS debug output to specific events. For example, you can find call records for specific users within one millisecond.

The steps below are an example of how to enable NTP on a single Cisco gateway.

Step 1 Locate an authoritative clock source. For example, you can use a Cisco router or an atomic clock that is attached to a time server.

Step 2 On the Cisco gateway, specify the primary NTP server's IP address and automatic calendar updates on other gateways that will be slaves to the master clock:

a. Set the time.

```
clock set 14:00:00 30 sept 2002
```

Note To ensure proper timing across extensive networks, Cisco recommends that you set all gateways to UTC (Coordinated Universal Time, also known as Greenwich Mean Time, or GMT). However, local requirements may vary.

b. Establish calendar updates.

```
ntp update-calendar
```

c. Assign the IP address of the time server.

```
ntp server <NTP_server_IP_address> prefer
```

Step 3 Verify that the clock is synchronized to the NTP server.

a. Inspect the status and time association. Clock sources are identified by their stratum levels.

```
show ntp status
```

b. You can see how often the network access server is polling and updating to the stratum clock.

```
show ntp association
```

Tip An asterisk (*) next to the NTP server's IP address indicates successful synchronization with the stratum clock.

You can configure a variety of optional NTP features. For details, refer to Performing Basic System Management at the following URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/fun_c/fcprt3/fcgenral.htm

Installing Cisco UGCA Components on the Server

The Cisco UGCA application is currently available only on CD. Do the following to install the application software from the CD. The following commands are executed at the command line of the target Cisco UGCA server.

Free Disk Space Requirements

During an installation, only 100 MB of free disk space is required to accommodate the system software. However, the installation process checks to see that a minimum of approximately 4.6 GB is available, to accommodate backups. If this check fails, you can continue the installation, but if the check for 100 MB fails, the system will not install.

Determine Free Disk Space

To determine available disk space, use the following UNIX command:

```
# df -b /opt
```

Output like the following appears:

```
Filesystem             avail
/dev/dsk/c0t0d0s0      913758
```

The value is in kilobytes. To accommodate our backup requirement of 4.6 GB, look for a value of 4804841.

Install the Application Package

Caution To install the software application package, you must be logged in as root.

Step 1 Install the package:

```
# pkgadd –d /cdrom/cdrom0
```

Step 2 Follow the prompts to install the package.

Note The system checks for sufficient free disk space during installation. If the required free disk space (see Free Disk Space Requirements, above) is not available, you get a warning. You can continue the installation, but you may have problems later.

Step 3 You are asked during installation if you want to install the application in */opt*. This is not required, but is recommended. Press **Return** to enable all defaults.

Step 4 You are asked during installation to supply new database passwords for the system and user accounts. Key presses are not echoed back to the screen so you must enter the passwords twice to confirm them.

Step 5 You are asked during installation if you want to enable daily backups and purges. If you choose not to elect these options now, you can enable them later. See Managing crontab Entries for Backups and Purges, page 3-22.

Step 6 You are asked during installation if you want the application to restart automatically when the server reboots. This is optional.

Step 7 Following the installation, you are prompted to select the other packages to install. Type **q** to quit at this point.

Cisco UGCA starts automatically and you are informed that the installation was successful.

CHAPTER 3

Starting, Using, and Maintaining Cisco UGCA

This chapter presents the following major topics:

- Launching Cisco UGCA from a Browser
- Using Cisco UGCA: Quick Tour
- Maintaining Cisco UGCA
- Using Cisco UGCA Utility Tools
- Troubleshooting

Note This chapter does not cover the details of using the Cisco UGCA browser-based application, except for launching it and performing basic housekeeping. Instructions for using that application to generate and view reports are available in the online help.

Launching Cisco UGCA from a Browser

To access Cisco UGCA from a Web browser, do the following:

Note Initial releases of this application require Microsoft Internet Explorer. Make sure you have read Chapter 2, "Installing Cisco UGCA," in particular Prerequisites, page 2-1.

Step 1 Start Microsoft Internet Explorer and go to the following URL:

```
https://<FQDN>
```

where FQDN is the fully qualified domain name of the application server (for example, *myserver.mydomain.com*)

Cisco UGCA launches.

Tip You may get a warning that the security certificate was not issued by a trusted company. You can add the certificate now. This stops future warning messages from appearing when you launch the application.

Step 2 Enter your user ID and password.

The predefined user ID is **admin**.

The predefined password is **admin**.

You are now connected to Cisco UGCA and can create reports.

Maintaining the Cisco UGCA application on the server is discussed in the remainder of this chapter.

Using Cisco UGCA: Quick Tour

A brief introduction to using Cisco UGCA follows. For more details, see the online help. The major tasks, in order, are as follows:

- Task 1: Load CDR Records
- Task 2: Create a Report Specification
- Task 3: Generate a Report
- Task 4: View a Report
- Task 5: Modify a Report Specification

Task 1: Load CDR Records

To load CDR records in the form of syslog files, do the following:

Step 1 Identify a syslog file to load.

Refer to Loading CDR Data, page 3-23. The syslog records are saved in /tmp/syslog.log.

Step 2 Open a UNIX terminal window and login as root.

Step 3 Enter the following to run CDRLoader and load the syslog file into the application:

```
# install/CSCOugca/bin/loadcdr_file.sh /tmp/syslog.log /tmp/syslog.log.result
```

where *install* is the user-selected installation directory. The following messsage should appear:

```
Job started successfully, please check job information for details.
```

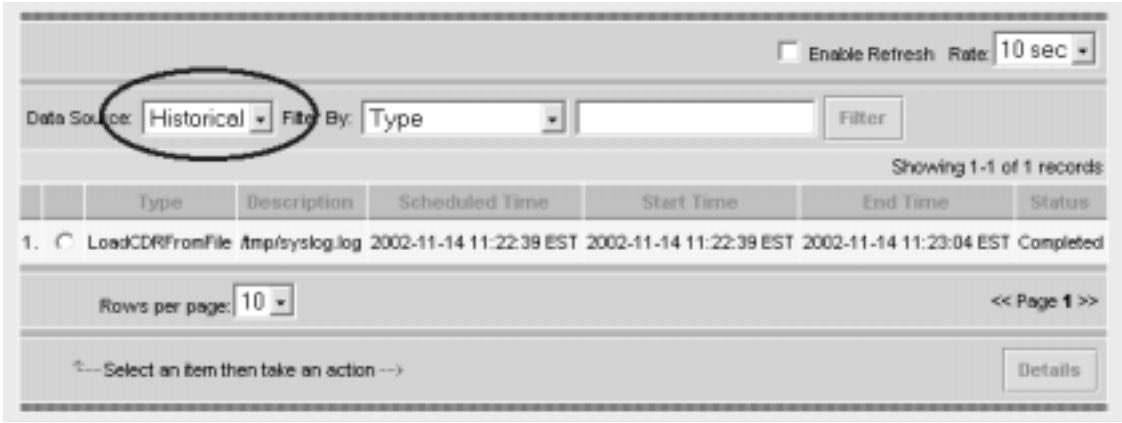
Step 4 After the job is started, you can observe its status in the browser.

a. Job status is initially considered active. To confirm this, choose **Reports > Job Status** from the main window. Choose **Data Source > Active**. See Figure 3-1.

Figure 3-1 Viewing the Status of an Active Job

b. After the job is completed, it is moved to the historical job list. To confirm this, choose **Reports** > **Job Status** from the main window. Choose **Data Source** > **Historical**. See Figure 3-2.

Figure 3-2 Viewing the Status of a Completed Job



An output file containing statistics, /tmp/syslog.log.result, is generated.

c. To see the content of the file, enter the following command in the UNIX terminal window:

```
# cat /tmp/syslog.log.result
```

An example file follows:

```
================ Summary ================
               Status : Completed
           Start time : 2002-11-14 11:22:39 EST
             End time : 2002-11-14 11:23:04 EST
             Time (s) : 24.90
    Total syslog msgs : 14781
       Total CR to DB : 1640
   Total Markup to DB : 1640
      Total MCR to DB : 1599
 Total inserted to DB : 4879
      Total failed CR : 0
     Total failed MCR : 0
 Av. insert rate (/s) : 195.91
 Av. CR ins. rate(/s) : 65.85
      Files processed : /tmp/syslog.log
         Files failed :
```

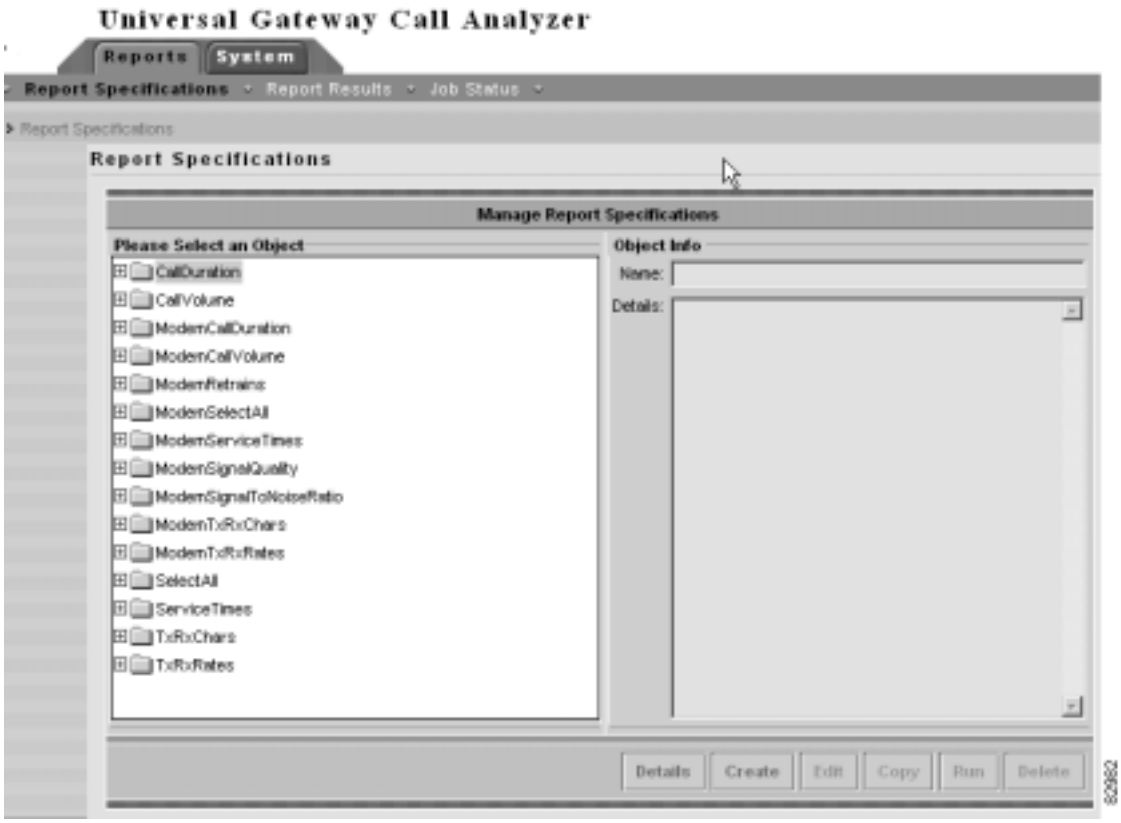
Task 2: Create a Report Specification

To generate a report for call volume distribution across nodes for calls lasting longer than 300 seconds, do the following:

Step 1 Select a report type.

a. Choose **Reports > Report Specifications**. The Manage Report Specifications window appears. See Figure 3-3.

Figure 3-3 Selecting a Report Specification

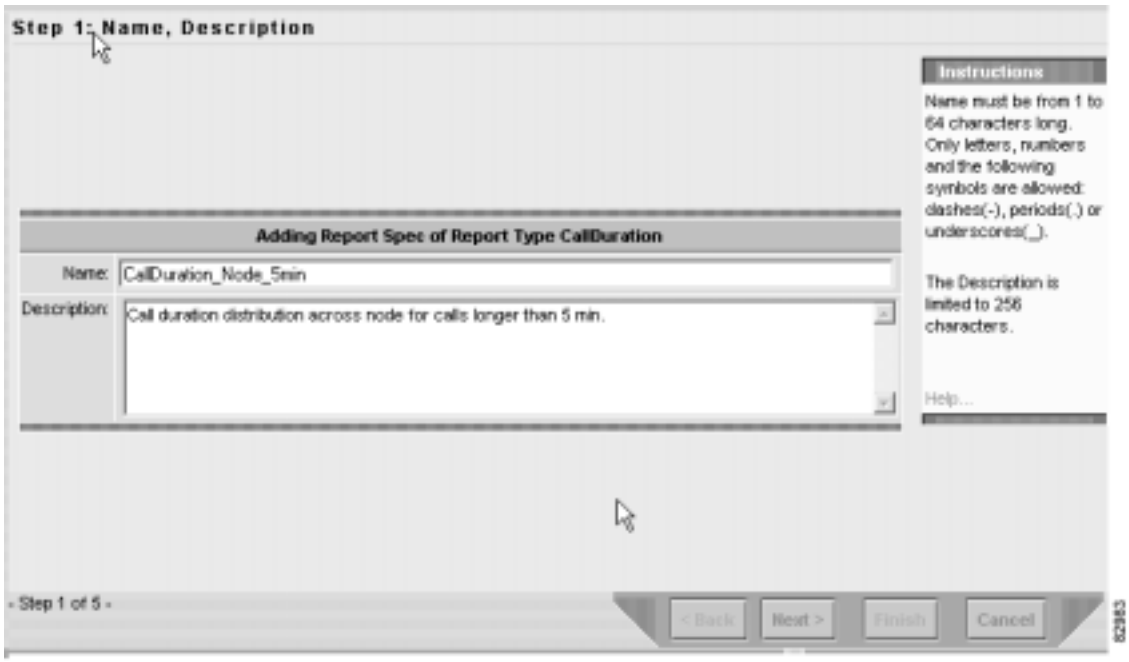


b. Click the folder **Call Duration**.

c. Click **Create**, at the bottom right. The Name, Description window appears. See Figure 3-4.

Step 2 Enter a report name and description.

Figure 3-4 Entering a Report Name and Description



a. In the Name field, enter a name. In our example, we enter **CallDuration_Node_5min**.

Note Use underscores instead of spaces.

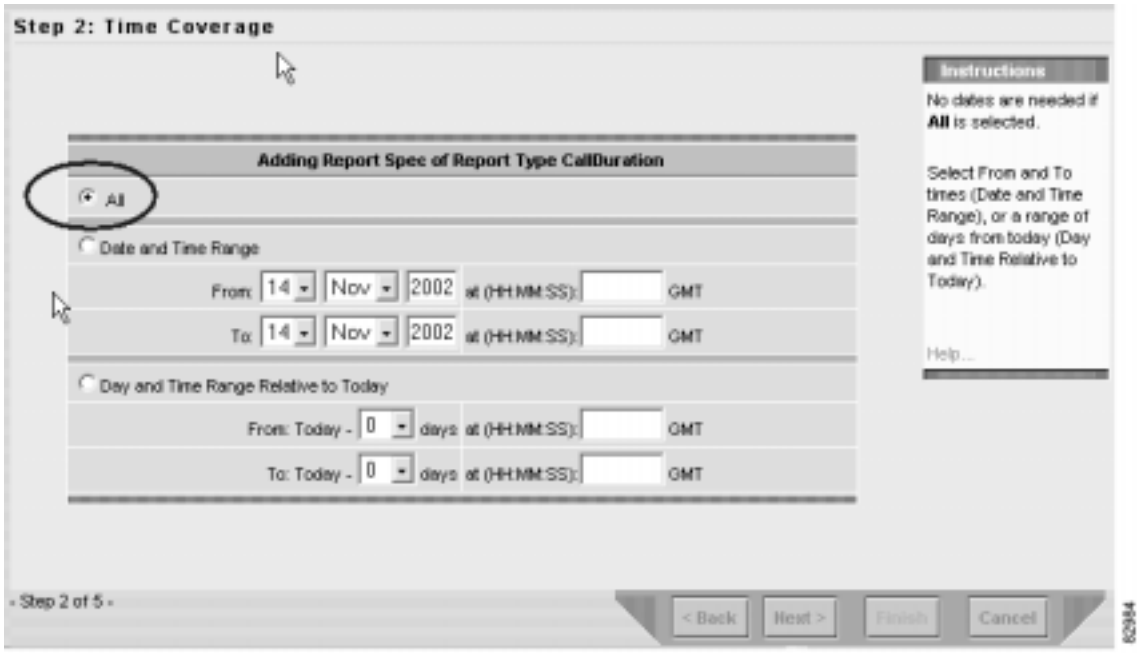
b. In the Description field, enter a description.

Note A good description assists others in understanding report results.

c. Click **Next**. The Time Coverage window appears. See Figure 3-5.

Step 3 Configure time coverage.

Figure 3-5 Configuring Time Coverage

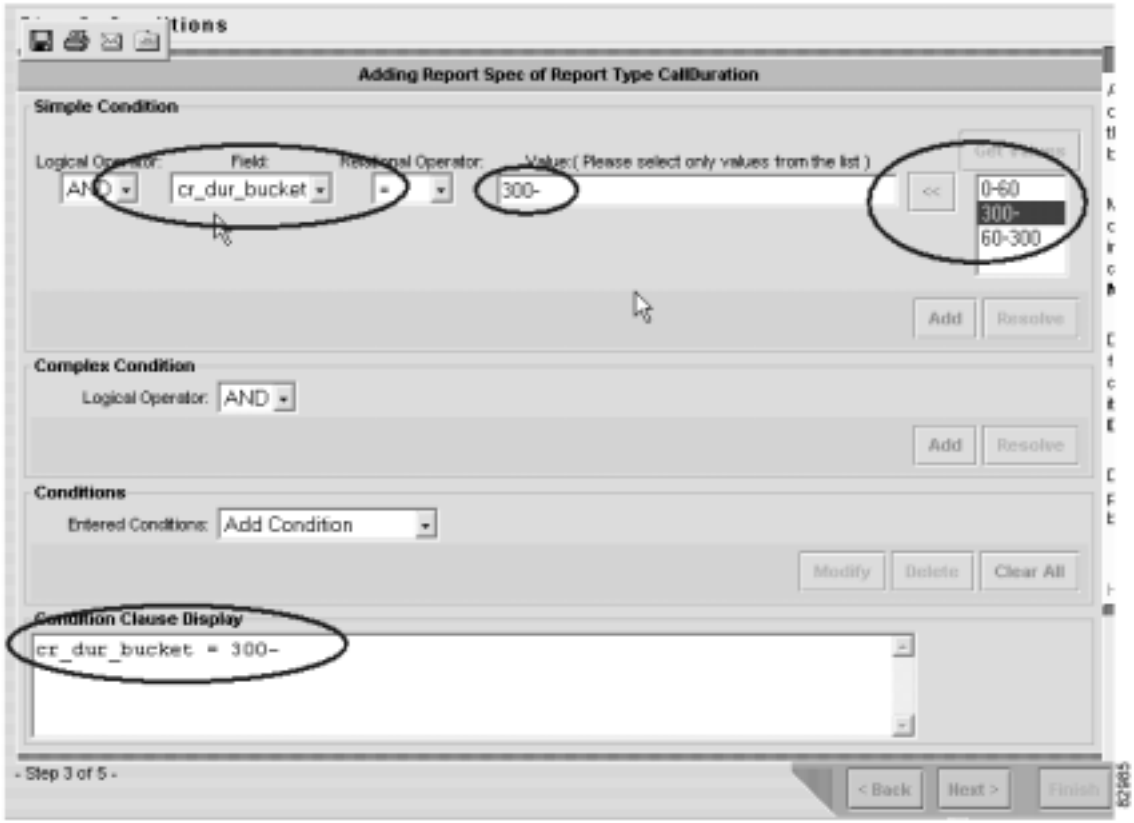


a. Choose **All**.

b. Click **Next**. The Configuring Conditions window appears. See Figure 3-6.

Step 4 Configure conditions.

Figure 3-6 Configuring Conditions



a. Under Field, choose **cr_dur_bucket**.

b. Under Get Values, choose **300-**.

c. Click << to populate the Value field.

d. Under Get Values, click **Add**.

e. Leave other conditions at their default values.

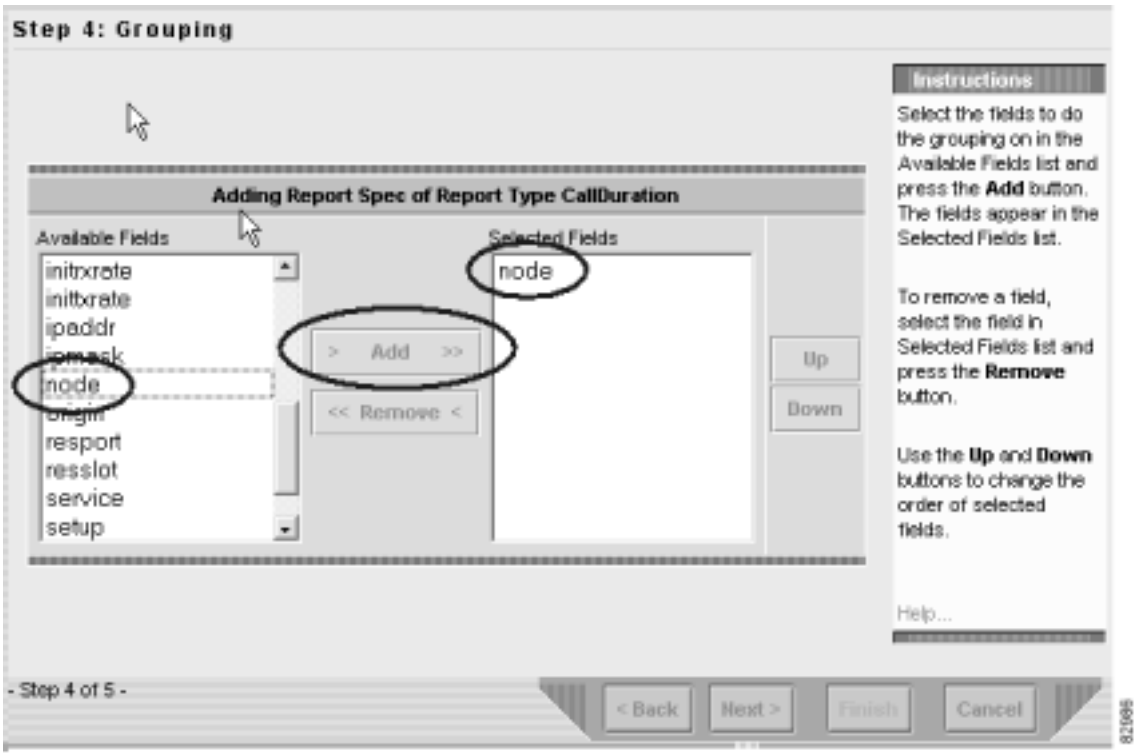
f. Note the Condition Clause Display field at the bottom, and verify that it reads as follows:

```
cr_dur_bucket = 300-
```

g. Click **Next**. The Grouping window appears. See Figure 3-7.

Step 5 Configure grouping.

Figure 3-7 Configuring Grouping



a. Under Available Fields, choose **node**.

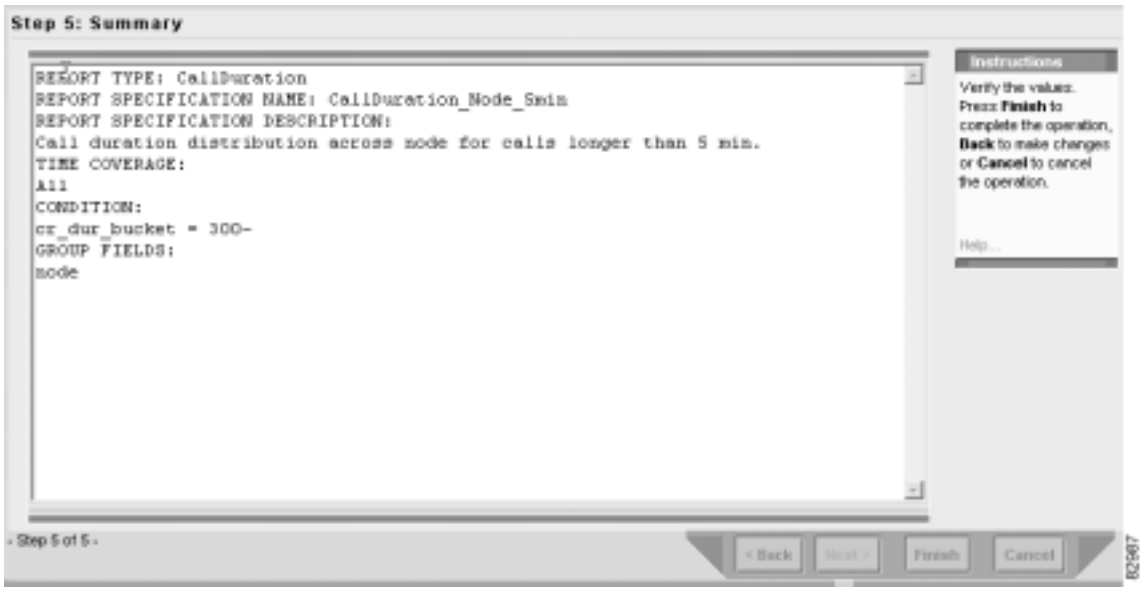
b. Click **Add**.

c. Under Selected Fields, verify that *node* is added.

d. Click **Next**. The Summary window appears. See Figure 3-8.

Step 6 Verify the report specification.

Figure 3-8 Verifying the Report Specification

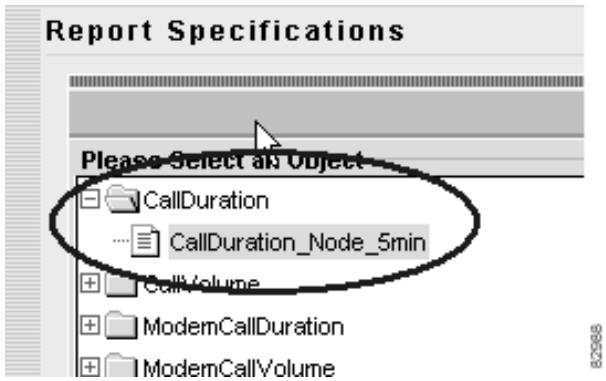


a. In the Summary window, verify the report specification.

b. Click **Finish**. The Report Specifications window appears. See Figure 3-9.

Step 7 Finish creating the report specification.

Figure 3-9 Finishing the Report Specification



a. Look at the Report Specifications window.

b. Verify that the report specification CallDuration_Node_5min is created under CallDuration.

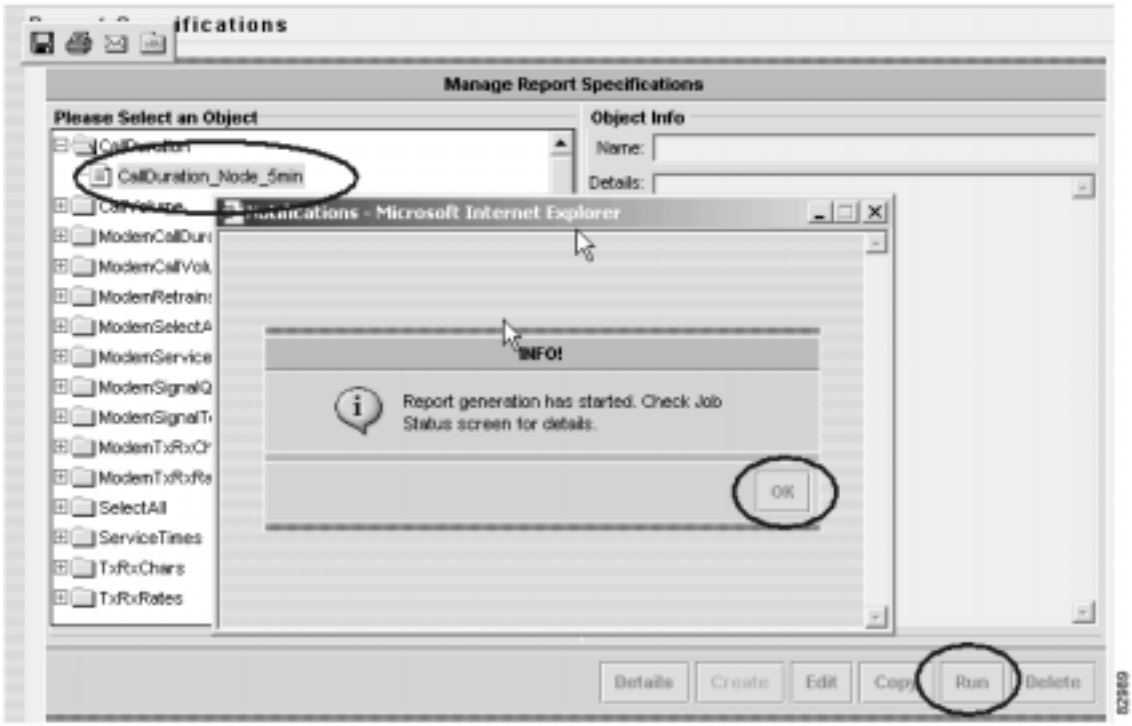
Task 3: Generate a Report

To generate a report, do the following:

Step 1 Run report generation.

a. From the main menu, choose **Reports > Report Specifications**. The Manage Report Specifications window appears. See Figure 3-10.

Figure 3-10 Generating a Report



b. Choose **Call Duration > CallDuration_Node_5min**.

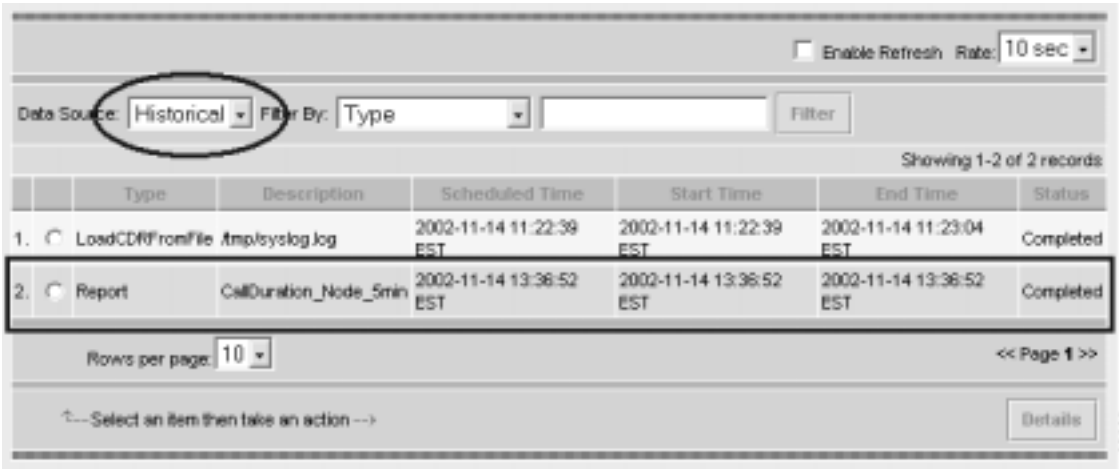
c. Click **Run**, at the bottom of the window. A dialog box confirms that report generation has started.

d. Click **OK**.

Step 2 Verify that report generation has completed.

a. From the main menu, choose **Reports > Job Status**. See Figure 3-11.

Figure 3-11 Verifying that Report Generation Has Completed



b. Choose **Data Source > Active**. Wait until the task CallDuration_Node_5min disappears from the list.

c. Choose **Data Source > Historical**. Look under the Status column to verify that the report generation is completed.

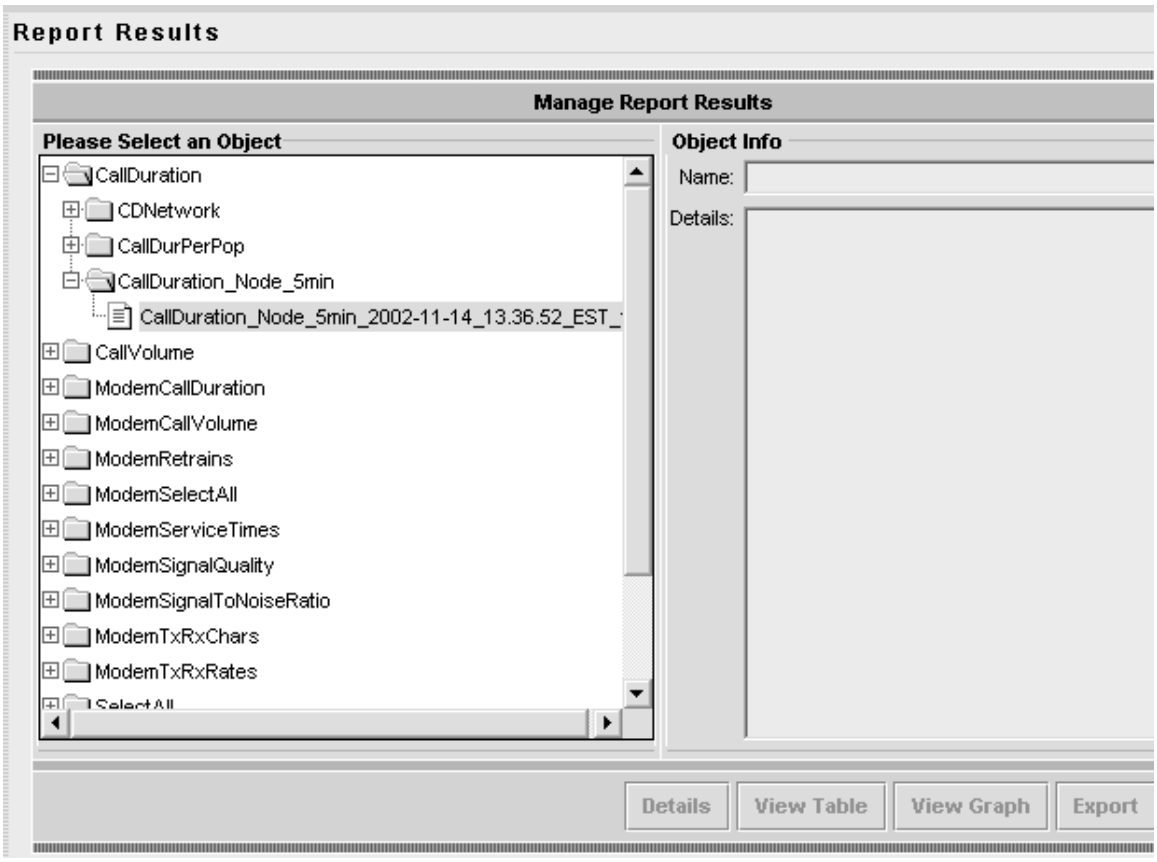
Task 4: View a Report

To view a report, do the following:

Step 1 Select the report.

a. From the main menu choose **Reports > Report Results**. The Report Results window appears. See Figure 3-12.

Figure 3-12 Viewing a Report



b. Double-click the folder **CallDuration** to expand it, and find the folder CallDuration_Node_5min.

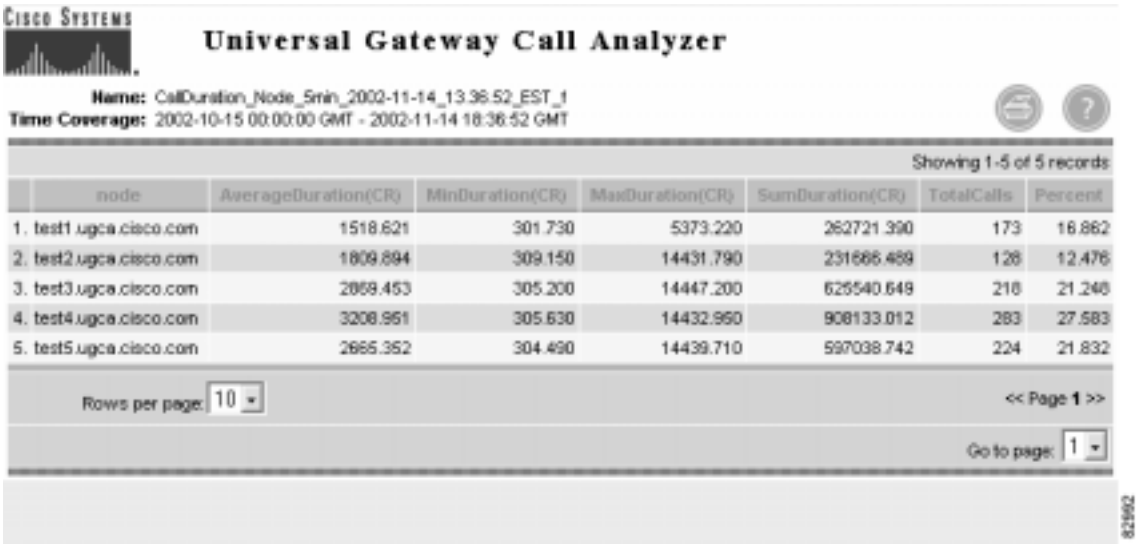
c. Open the folder **CallDuration_Node_5min** and choose the report result that has the appropriate time stamp.

Note The timestamp of a report result is taken from the End Time of the job (see Step 2 in Task 3: Generate a Report, page 3-11. Reports of the same type that are run at different times will have different time stamps appended to the original title. In our example, we are concerned with only a single report.

Step 2 View the report as a table. See Figure 3-12.

a. Click **View Table**. Report details appear. See Figure 3-13.

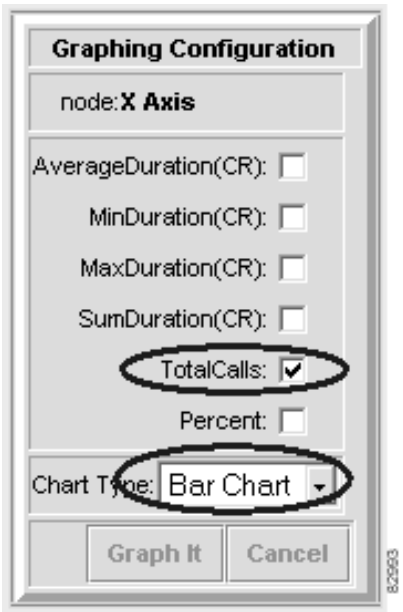
Figure 3-13 Viewing a Report as a Table



Step 3 View the report as a graph. See Figure 3-12.

a. Click **View Graph**. The Graphing Configuration window appears. See Figure 3-14.

Figure 3-14 Viewing a Report as a Graph

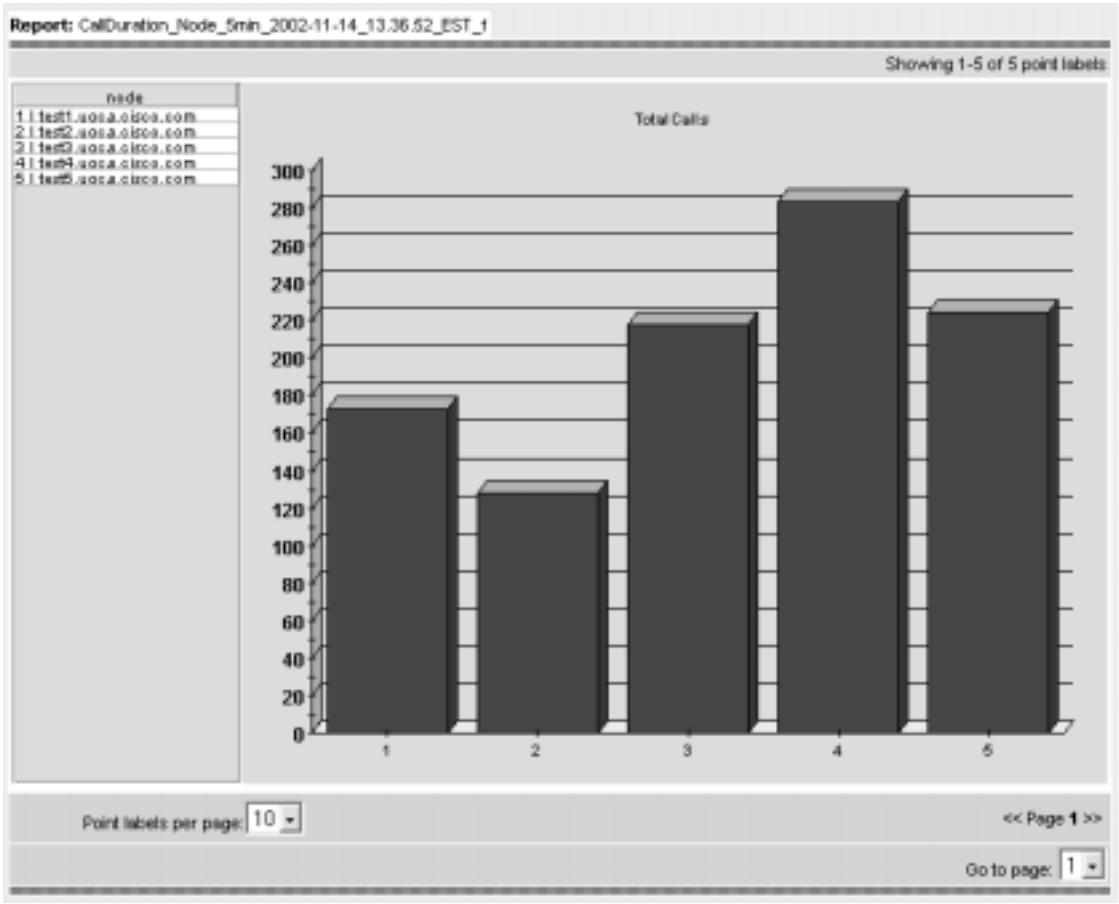


b. Choose **TotalCalls** as the value to be displayed.

c. Choose **Bar Chart** as the type of the graph.

d. Click **Graph It**. In our example, the following bar chart appears. See Figure 3-15.

Figure 3-15 Example Bar Graph



Step 4 Print the report. You can print a report from either table or graph views.

a. To print a report as a table, first view the report as a table. (See Step 2 in this task.)

b. Within the selected view, select the printer icon. A new window with the report in printable format is displayed.

c. Choose **File > Print**.

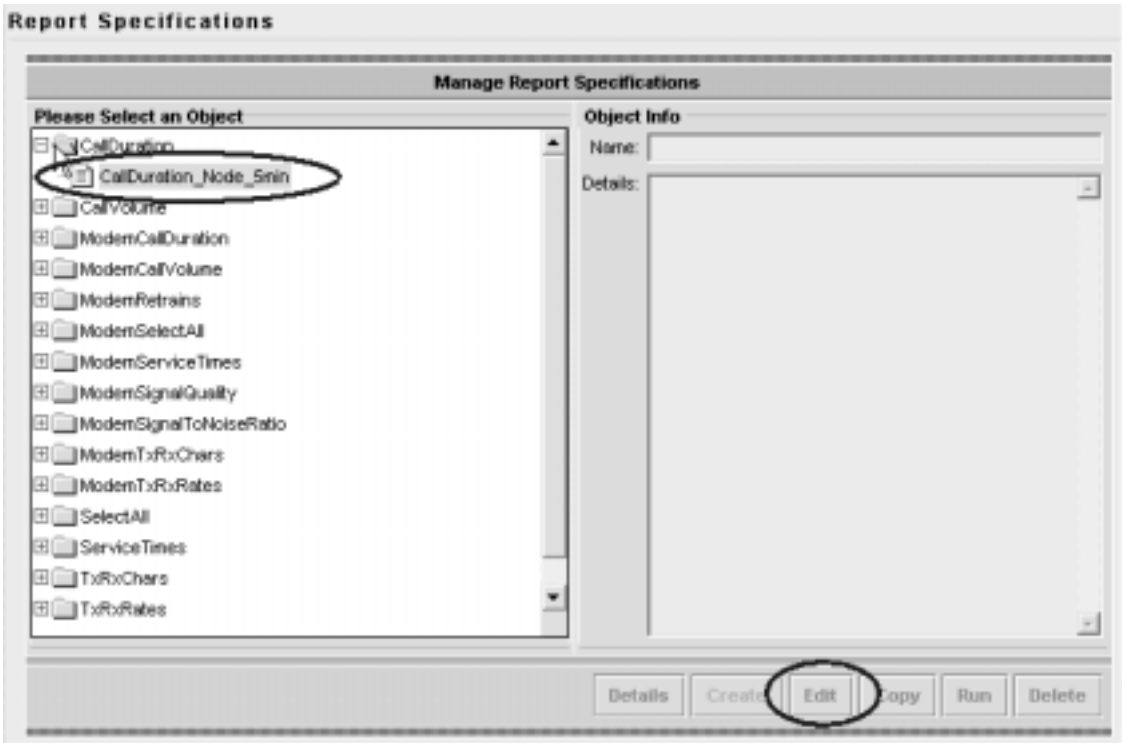
d. To print a report as a graph, first view the report as a graph. (See Step 3 in this task.). Then repeat Step b and Step c immediately above.

Task 5: Modify a Report Specification

Step 1 Select the report to modify.

a. From the main menu, choose **Reports > Report Specifications**. The Manage Report Specifications window appears. See Figure 3-16.

Figure 3-16 Modifying a Report Specification



b. Double-click the folder **CallDuration** to expand it, and find the folder CallDuration_Node_5min.

c. Open the folder **CallDuration_Node_5min** and choose the report result that has the appropriate time stamp.

d. Click **Edit**.

Step 2 Modify the report specification.

The procedure for modifying a report is the same as in Task 2: Create a Report Specification, page 3-5, except that you can choose specific screens to modify directly.

Maintaining Cisco UGCA

Perform the following routine maintenance procedures *as needed* to maintain Cisco UGCA:

Caution Root access is required for the following activities.

- Purging Data
- Backing Up the System
- Restoring the System
- Stopping Cisco UGCA
- Resetting the Cisco UGCA Database
- Starting Cisco UGCA
- Removing Cisco UGCA
- Editing Session Timeout
- Managing crontab Entries for Backups and Purges
- Loading CDR Data
- Maintaining Markup Data
- Managing the Volume of Log Files
- Accessing and Reading Raw Data
- Changing Database Account Passwords

Note The output messages resulting from purging, backing up, restoring, starting, and stopping Cisco UGCA are also logged in syslog messages.

Purging Data

Note In the following discussion, *install* represents the directory in which you installed the Cisco UGCA application. See also Managing crontab Entries for Backups and Purges, page 3-22.

A purge utility purges data older than the specified age and is located in the following directory:

install/CSCOugca/bin/purge.sh.

The utility purges three data sets:

- Loaded syslog data
- Historical job entries
- Apache Tomcat log files (see Note below)

Note Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process. For more information, refer to the Apache Jakarta Project at the following URL:

http://jakarta.apache.org/tomcat/

The purging is controlled by age parameters specified in the configuration file *install*/CSCOugca/etc/config/ugca.xml, under the module "purge."

The parameters are as follows:

- *callrecordsage*—Specifies the age for loaded syslog data and defaults to 30 days. Valid range is from 0 to 2000 days.
- *tasksage*—Specifies the age for historical job entries and defaults to 7 days. Valid range is from 0 to 100 days.
- *tomcatlogsage*—Specifies the age of Apache Tomcat log files. Valid range is from 1 to 100 days.

To purge the data, enter the following at the UNIX command line:

install`/CSCOugca/bin/purge.sh`

For loaded syslog data, data falling into the following range is purged:

(*current date in GMT*) – (*call setup date in GMT*) >= (*callrecordsage* + 1)

For historical job entries, data falling into the following range is purged:

(*current date in local time*) – (*job end date*) >= (*tasksage* + 1)

For Apache Tomcat log files, data falling into the following range is purged:

(*current local time*) – (*last time log file was modified*) >= *tomcatlogsage* * *24 hour*

Note The use of the "+ 1" leaves the current day's data. For *callrecordsage* = 0 (the minimum allowed value), data for the previous day and earlier is purged.

Backing Up the System

Caution Before attempting a backup, you must ensure that sufficient free memory is available to accommodate the files. See Determine Free Disk Space, page 2-6.

Note The backup directory must exist before this utility can be run. See also Managing crontab Entries for Backups and Purges, page 3-22.

The backup utility is located in the following directory:

install/CSCOugca/bin/backup.sh

The utility will back up the database plus the directory *install*/CSCOugca/etc into the specified backup directory. The backup directory is specified in the configuration file *install*/CSCOugca/etc/config/ugca.xml, under the module "backup" with the parameter "backupdir."

Performing a Backup

To back up the system, enter the following at the UNIX command line:

```
install/CSCOugca/bin/backup.sh
```

The backup file is compressed and is named automatically in the following ISO-standard format (HH represents time in 24-hour format):

UGCA_yyyymmddHHmmss.tar.Z

Note If the Cisco UGCA server is up before a backup is run, the backup utility stops the server first, then restarts the server following the backup. If the server is down before the backup is run, it remains down following the backup.

Restoring the System

Note Before restoring the system, shut down the Cisco UGCA server. See Stopping Cisco UGCA, page 3-19.

To restore the system, enter the following:

```
install/CSCOugca/bin/restore.sh filename_and_path
```

This command takes as its parameter the filename and path as discussed in Backing Up the System, above. This restores either the database, the directory *install*/CSCOugca/etc, or both the database and the directory.

Caution Cisco strongly recommends that you run the backup utility first, to back up the current database and the /etc directory. If a restore fails, the system attempts to revert to the previous database files, but this cannot be ensured.

Stopping Cisco UGCA

Caution Do not stop Cisco UGCA while maintenance activities (backup, restore, reset, purge, check_markup) are in progress. If a report is running or a file is being loaded, stopping the application aborts these activities.

To stop Cisco UGCA on the server, enter the following:

```
/etc/init.d/ugca stop
```

Caveats and Examples

Note the following caveats and example reports following a *stop* command. The following cases are considered:

- If No Other Processes Are Running
- If Processes Are Running
- If the System Is Busy

Note User input is highlighted in **bold**.

If No Other Processes Are Running

The *stop* command first checks whether any Cisco UGCA processes are running. If none are running, the application exits.

```
/etc/init.d/ugca stop

Stopping UGCA
No UGCA processes were detected to stop
```

If Processes Are Running

If any Cisco UGCA processes are running, then a *stop* command asks them to shut down and report on success:

```
/etc/init.d/ugca stop

Stopping UGCA
UGCA stopped
```

If the System Is Busy

If the system is busy, it may take more than a minute for the application to exit.

Caution Do not interrupt a *stop* command during this time or attempt to run another *stop*.

If any problems are encountered as the process is shutting down (for example, the process does not reply to the request to shut down), then the *stop* command will report that it encountered problems. This should be taken only as a warning, as it is the eventual success of the *stop* process that really counts. In the example below, a *stop* command encounters some issues but eventually runs successfully:

```
/etc/init.d/ugca stop

Stopping UGCA
Problems encountered whilst shutting down
UGCA stopped
```

The only circumstances under which the user needs to take further action is when a *stop* command reports that some processes remained running. For example:

```
/etc/init.d/ugca stop

Stopping UGCA
```

```
Problems encountered whilst shutting down
UGCA stop failed, the following processes were detected:
 8943 /opt/CSCOugca/j2sdk1.4.0/jre/bin/java
 8942 /opt/CSCOugca/j2sdk1.4.0/jre/bin/java
Please see the troubleshooting section of the user guide for help
```

Note The system may or may not report "Problems encountered whilst shutting down." What counts is the line "UGCA stop failed, the following processes were detected:" followed by a list of processes.

To return the system to a stable state, first try issuing a *stop* command again. If the system still reports that it detected Cisco UGCA processes, do the following:

Step 1 Issue the command **kill <pid>**, where **<pid>** is the process ID (the number at the left of the "following processes were detected" line (see above).

```
kill 8943
kill 8942
```

Step 2 Issue another *stop* command.

```
/etc/init.d/ugca stop
```

Step 3 If processes are still reported, use the **kill** command again, this time with the **–9** flag:

```
kill -9 8943
kill -9 8942
```

Step 4 Issue another *stop* command.

```
/etc/init.d/ugca stop
```

The system is down when a *stop* reports either of the following:

```
UGCA stopped
```

or

```
No UGCA processes were detected to stop
```

At this point you can then either (1) perform a maintenance activity that requires a stopped system, or (2) start the system again. When the process of stopping the system is started, continue until the system is successfully stopped. Do not attempt to restart the system or perform a maintenance activity that requires a stopped system until the system is finally down.

Caution Any maintenance activity (such as *restore*, *reset*, *check_markup*) that requires a stopped system followed by a start refuses to run if the system is not totally stopped.

Resetting the Cisco UGCA Database

Resetting the database returns it to the state it is in immediately following an installation.

Caution A reset removes all data from the database and reverts to the predefined user ID and password of *admin*.

Step 1 Make sure that Cisco UGCA is stopped on the server. See Stopping Cisco UGCA, above.

Step 2 To reset the Cisco UGCA database, enter the following:

```
install/CSCOugca/bin/reset.sh
```

Starting Cisco UGCA

To start Cisco UGCA on the server, enter the following:

```
/etc/init.d/ugca start
```

Removing Cisco UGCA

Step 1 To remove Cisco UGCA from the server, enter the following:

```
pkgrm CSCOugca
```

Step 2 Follow the prompts. The system stops CSCOugca first, before removing it from the server.

Editing Session Timeout

Session timeout is a mechanism that will invalidate a user session after a predefined number of minutes of inactivity. The invalidation of a user session also causes all locked resources to be unlocked.

Note The ReportSpec is locked whenever a user goes to "Edit mode." In this case no other session can delete the ReportSpec or edit it. However, session timeout unlocks the ReportSpec.

The default timeout is 10 minutes.

Step 1 To modify the default, edit the following configuration file:

install/CSCOugca/jakarta-tomcat-4.0.3/webapps/ugca/WEB-INF/web.xml

Step 2 Find the following line and change the default timeout value, taking care to write and save the file:

```
<session-timeout>10</session-timeout>
```

Managing crontab Entries for Backups and Purges

By default, the crontab file is set to add entries to run a backup at 10:00 p.m. and purge data at 5:00 a.m. daily. If you chose not to enable backups and purges during installation, you can use the **crontab** command to enable them now, as well as to change the default times.

When the system is removed, these entries are removed from the crontab file for the root user. The following is a sample crontab file.

```
#ident  "@(#)root       1.19    98/07/06 SMI"   /* SVr4.0 1.1.3.1       */
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * 0,4 /etc/cron.d/logchecker
10 3 * * 0   /usr/lib/newsyslog
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#0 22 * * * /opt/CSCOugca/bin/backup.sh > /dev/null 2>&1
#0 5 * * * /opt/CSCOugca/bin/purge.sh > /dev/null 2>&1
```

The last two lines belong to the application, and are always added when the package is installed and removed when the package is removed. The "#" signs mean that these lines are commented out. If you answer yes to the enabling of backup or purge during installation, then the "#" is removed automatically from the beginning of the appropriate line. The backup line says to run the backup script at 0 minutes past 22 hours on the * (every) day of the month, on * (every) month of the year and * (every) day of the week. The purge script is similar, this time running at 5 a.m. instead of at 10 p.m. The format of this file and how to manipulate it is described if you type the following, to launch the UNIX man page for **crontab**:

```
man crontab
```

Note The rest of the file was established by the Sun Solaris operating system during installation.

Loading CDR Data

Cisco UGCA uses a program called CDRLoader to load CDR data into the database for report generation. The CDRLoader script prints out usage information such as the following. Below is the default without special parameters set.

Note User input is highlighted in **bold**. The value of *install* is the user-selected installation directory.

```
# install/CSCOugca/bin/loadcdr_file.sh

Usage: loadcdr_file.sh <input> <result>
  Where
    input   file name to be loaded, or directory name
            containing files to be loaded.
    result  file name, to which results will be written.
  Exit code
    0       job scheduled or started successfully.
    1       otherwise.
```

For example, to load a syslog file with the filename /tmp/syslog.msg, the resulting file is generated in the same directory with the following name:

```
# cd /opt/CSCOugca/bin
# ./loadcdr_file.sh /tmp/syslog.msg /tmp/syslog.msg.rst
```

```
Job started successfully, please check job information for details.
```

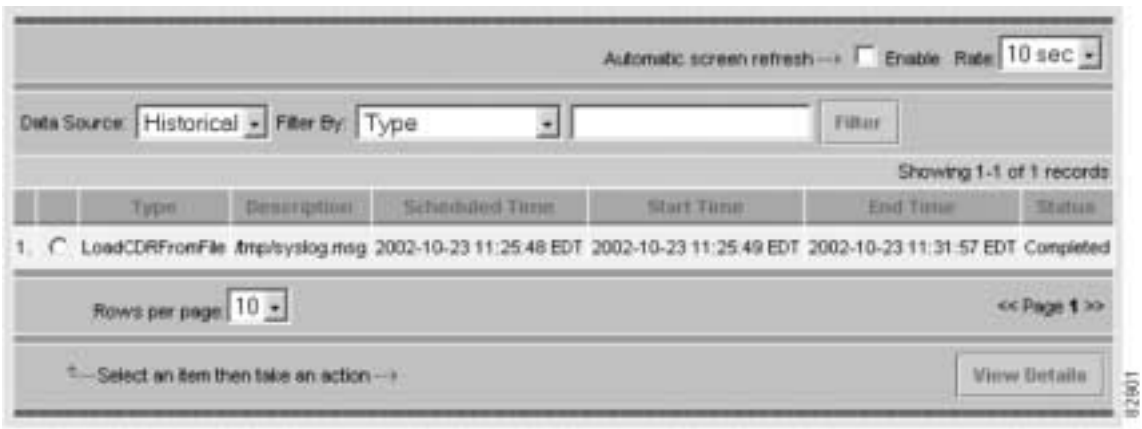
Internally, the CDR loading request was translated into a job. CDRLoader then executes this job. During execution, if you open the "Job Status" page and select "Active" jobs on the browser, the status of this job is displayed. After the job is executed, you can display it by choosing "Historical" as the data source.

Figure 3-17 illustrates the status of a running LoadCDRFromFile job.

Figure 3-17 A Running LoadCDRFromFile job

Figure 3-18 illustrates the status of the completed job.

Figure 3-18 A Completed LoadCDRFromFile job



The result file is always generated. If the operation was successful, it contains the statistics of the execution. Otherwise, it contains error messages. An additional purpose of the result file is to automate CDR loading. An external script can create the syslog file, invoke the loadcdr_file.sh script, and wait for the creation of the result file. By analyzing the contents of the result file, the script can also determine whether the file loaded successfully.

CDRLoader Configuration

The configuration of CDRLoader is stored in the following files:

- ugca.xml
- cdrloader.xml

During system startup, the backend server reads the file ugca.xml. This file is the master source of configuration for the Cisco UGCA system. Then, the backend server starts a process that reads only the file cdrloader.xml. When an attribute in the configuration is queried, CDRLoader looks locally first, to the file cdrloader.xml on the client machine. If the attribute is found, it is returned. Otherwise, it forwards the query to the backend server, which has the content of the file ugca.xml.

Note In other words, if a configuration aspect is configured in ugca.xml and not in cdrloader.xml, CDRLoader uses the configuration from ugca.xml. If the aspect is also configured in cdrloader.xml, CDRLoader uses the configuration from cdrloader.xml. See the following Tip.

Tip For example, you may want to change only the logging level configuration in cdrloader.xml. If you change it in ugca.xml, it affects all processes.

The format of ugca.xml and cdrloader.xml is identical. In most cases, it is sufficient to use only *ugca.xml*. However, some situations require cdrloader.xml. For example, the user may want to change only the logging level of CDRLoader. Table 3-1 lists and describes CDRLoader configuration attributes, with their defaults.

Table 3-1 CDRLoader Configuration Attributes

Name	Description	Default
queueSize	Maximum size of the message queue used internally in the CDR message loader. In general, a larger queue size makes the loader better able to accept large numbers of messages in a short time. This may cause higher memory consumption.	4000
queueSampleThreshold	Threshold used in monitoring the message queue. A snapshot of queue activity is taken after this number of messages is inserted to the queue.	800
mcrMatchQueueSize	Maximum size of the message queue used for matching a call record with its corresponding modem call record in the case of a modem call. A larger queue size may cause higher memory consumption.	100
UnmatchedCRQueueSize	Maximum size of the message queue used for unmatched call records.	250
maxProcessors	Maximum number of CDR message processors. Each processor consumes one database connection. If the maximum number of processors is constantly observed in the statistics, it is beneficial to reduce this number (for example, to 5).	10

Table 3-1 CDRLoader Configuration Attributes (continued)

Name	Description	Default
statisticsEnabled	Enable or disable statistics. This is mostly used for tuning. It should be disabled under normal operation. Statistics are written to the log file with the logging level "INFO."	false
sampleInterval	Interval between statistics, in seconds.	60
dbInsertDisabled	Processing without a database insert (used internally). This should not be changed under normal circumstances.	false

Note A result file is not created when the CDR loader process is not shut down properly. An example of an improper shutdown is the use of the UNIX command **kill –9** or the unplugging of the power cord.

Maintaining Markup Data

Markup data is generated from raw CDR data. Markup data (table `markup`) is stored in the database . The primary purpose of markup data is to increase the efficiency of report generation, as well as to provide some level of normalization of raw data. For a better understanding of this topic, some knowledge of SQL and high-level programming languages is required.

Markup is generated automatically during the loading of CDR data, and no direct user involvement is required. However, user interaction is required for configuration and maintenance.

This section presents the following major topics:

- Markup Configuration Attributes
- Markup Generation Rules
- Maintaining Markup Data

Markup Configuration Attributes

General configurations are stored in the file markup.xml. Table 3-2 lists and describes markup configuration attributes, with their defaults.

Table 3-2 General Markup Configurations

Name	Description	Default
regenChunkSize	Markup data is regenerated chunk by chunk. This variable specifies the chunk size. A larger chunk size may cause a shorter markup-data regeneration time, but increase memory consumption. Too large a chunk size may cause more frequent paging, and an even longer markup-data regeneration time.	5000
regenThreshold	When inconsistency in markup data is detected, the existing data is either repaired or regenerated, depending on the number of inconsistent entries in the database. If the number is greater than this variable, existing markup data is simply removed and regenerated. Otherwise, existing markup data is repaired.	1000

The database schema and rules used for markup generation are totally controlled by the configuration file markup.xml (defined by markup.dtd). Changes in this file can be detected automatically during system startup, but the user must manually invoke a utility for the changes to take effect. Below is the file markup.dtd:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT MarkupSetup (MarkupAttribute)+ >
<!ELEMENT MarkupAttribute (MarkupAttrEnumType)?>
<!ATTLIST MarkupAttribute
    Name     CDATA                  #REQUIRED
    Type     CDATA                  #REQUIRED
    Unique   (TRUE|FALSE|true|false) #REQUIRED
    Default  CDATA                  #IMPLIED
    Rule     CDATA                  #REQUIRED >
<!ELEMENT MarkupAttrEnumType (Pair)+>
<!ATTLIST MarkupAttrEnumType
    DefaultName  CDATA              #REQUIRED
    DefaultValue CDATA              #REQUIRED >
<!ELEMENT Pair EMPTY>
<!ATTLIST Pair
    Name     CDATA                  #REQUIRED
    Value    CDATA                  #REQUIRED >
```

The content of the file markup.xml must be 100 percent compliant with its DTD, markup.dtd. Internally, all markup data is stored in a table called markup. As defined in the DTD, the root element MarkupSetup consists of one or more MarkupAttributes. Each has the following attributes:

- *Name*—Mandatory. Name of the MarkupAttribute. It must be a string without spaces. Internally it is translated into the name of a column in the markup table.
- *Type*—Mandatory. SQL data type of this column.
- *Unique*—Mandatory. It has the value *true* when this column is the unique key in the markup table (column *recid*). Additional markup columns should always set this attribute to *false*.
- *Default*—Optional. This value is used when no value is provided for this column during the insertion of a new row. If it is not presented in the configuration file, the default value is NULL.
- *Rule*—Mandatory. Rules used for markup generation.

A MarkupAttribute can have 0 or 1 MarkupAttrEnumTypes. If an enumeration type is presented in the definition of a markup attribute, this attribute is interpreted internally as an enumeration type. This type must be unique across the system. A MarkupAttrEnumType has the following attributes:

- *DefaultName*—Mandatory. It specifies the default return name if a query cannot find the name from an input value.
- *DefaultValue*—Mandatory. It specifies the default return value if a query cannot find the value from an input name.

A MarkupAttrEnumType consists of one or more Pairs (name–value mappings).

The following is a simplified example of markup.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MarkupSetup SYSTEM "file://localhost//opt/CSCOugca/etc/dtd/markup.dtd">

<MarkupSetup>
  <MarkupAttribute Name = "recid"
                   Type = "BIGINT"
                   Unique = "true"
                   Rule = "
      value = call_record.recid;
          ">
  </MarkupAttribute>
</MarkupSetup>
```

Markup Generation Rules

The rules used to generate markup data are similar to those of a simple programming language. Note that the XML specification requires the use of character sequences for certain characters. For better readability, most examples in the text will not substitute the following:

Character	String Used in XML File
&	&amp;
‘	&apos;
“	&quot;
<	&lt;
>	&gt;

Note The full BNF grammar for markup generation rule can be found in Appendix B, “BNF for Markup Interpreter: Terminals and Nonterminals.” The file makup.xml is shown in Appendix D, “Cisco UGCA Markup: markup.xml and call_record.discid.”

The return value of the markup logic should always be saved in the variable *value*. After execution, if this variable cannot be found in the internal symbol table, the default value defined in previous section is used.

The rules are applied to call record and modem record pairs; all values in these records are accessible from the markup interpreter. Details about these tables can be found in Appendix C, “Cisco UGCA Database Schema.” For example, the following are call record and modem record pairs:

Variable	Explanation
call_record.recid	Record ID for a call record defined in table call_record
modem_call_rec.sq	Signal quality defined in table modem_call_rec

The following additional topics are illustrated below:

- Comments
- Data Types
- Calculation Operations
- Boolean Operations
- If Statements
- Assignment
- Markup Attributes

Comments

The following comment styles are supported:

- Some comments
- /* Some comments */

Data Types

Only two data types are supported: string and number. Strings are character sequence quoted by '"'.

Example: "A string"

Note In the XML file, '"' must be substituted by '&quote;').

Any number is interpreted as a float internally (range: –2139095040 to 2139095039).

Example: 0, 1.2, –54226.

Calculation Operations

Operation	Explanation
+	Addition. *Example*: 2.2+3; a+2; b+c
–	Substitution. *Example*: 2.2-3; a-2; b-c
*	Multiplication. *Example*: 2.2*3; a*2; b*c
/	Division. *Example*: 2.2/3; a/2; b/c
~	Matching. Returns the first match of a regular expression (right-hand side) from the left-hand side. *Example*: "kaaabc" ~ "a*b" returns "aaab".

Boolean Operations

Operation	Explanation
&&	And. *Example*: (3>2) && (3<2) returns false.
||	And. *Example*: (3>2) || (3<2) returns true.
==	Equal to. *Example*: 3 == 2 returns false.
>	Greater than. *Example*: 3>2 returns true.
>=	Greater than or equal to. *Example*: 2>=3 returns false.
<	Less than. *Example*: 3<2 returns false.
<=	And. *Example*: 3<=2 returns false.
~~	Contains. Operation returns *true*, at least one match of the regular expression (right-hand side) can be found in left-hand side. *Example*: “kaaabc”~~ “a*b” returns true. “kaaacb” ~~ “a*b” returns false.

If Statements

Syntax:

"if" "(" Expression ")" Statement ("else" Statement)?

Example:

```
a = 5;                    /* Assign 5 to variable a        */
if (a == 5) {                 /* Check, if a equals 5          */
       value = a;             /* If true, assign 5 to value    */
}
else {
       value = -1;        /* Otherwise, assign -1 to value */
}
```

Assignment

Syntax:

PrimaryExpression "=" Expression

Example:

```
a = 5;                    // Assign 5 to variable a
a = b + c;                    // Assign sum of b and c to a
```

Markup Attributes

Table 3-3 illustrates a fully commented section from markup.xml. See also Appendix D, "Cisco UGCA Markup: markup.xml and call_record.discid."

Table 3-3 Markup Attributes and Explanations

Markup Attribute	Explanation

<MarkupAttribute Name = "cr_dur_bucket"
Type = "INTEGER"
Unique = "false"
 	
Default = "1"
Rule = "
if(call_record.duration > 300) {
value = 3;
}
 	
else if(call_record.duration > 60) {
value = 2;
}
else {
value = 1;
}
">
<MarkupAttrEnumType DefaultName = "0-60"
DefaultValue = "1">
<Pair Name="0-60" Value="1"/>
<Pair Name="60-300" Value="2"/>
<Pair Name="300-" Value="3"/>
</MarkupAttrEnumType>
</MarkupAttribute>
	Open an new MarkupAttribute element and set Name to "cr_dur_bucket"
	Set Type to SQL INTEGER
	Set Unique to false, this column is then not used as a unique key
	Set default value to 1
	
	If call duration defined in table call_record is greater than 300, set cr_dur_bucket value to enumeration value 3 (300-).
	
	Otherwise, if call duration defined in table call_record is greater than 60, set cr_dur_bucket value to enumeration value 2 (60-300).
	
	Otherwise, set cr_dur_bucket value to enumeration value 1 (0-60)
	
	Open an new MarkupAttrEnumType element with the default name "0-60" and default value 1.
	Add pair "0-60" - 1
	Add pair "60-300" - 2
	Add pair "300-" - 3
	Close MarkupAttrEnumType element
	Close MarkupAttribute element

The first column represents bucketized call duration in the call_record according to the following criteria:

- 0 < duration <= 60, value = 1
- 60 < duration <= 300, value = 2
- 300 < duration, value = 3

Maintaining Markup Data

You can manually invoke the script check_markup.sh to perform markup data maintenance.

Caution You must stop the system first before performing maintenance. See Stopping Cisco UGCA, page 3-19.

Under normal circumstances, no maintenance is required. During system startup, the system performs checks on markup data and determines if maintenance of markup data is required. If system decides, that maintenance of markup data must be performed, it prints out a warning message. However, it does not perform markup data maintenance automatically.

Note the following example:

```
# /etc/init.d/ugca start

Starting UGCA
[1] 22901
[1] 22906
[1] 22908
UGCA started
# WARNING:
Inconsistency in markup data has been detected. It's strongly
suggested to stop the system and run the command-line utility
/opt/CSCOugca/bin/check_markup.sh.
```

The script *install*/CSCOugca/bin/check_markup.sh tries to detect the following:

- Changes in the markup configuration
- Inconsistencies in existing markup data

If maintenance of the markup data is required, the script performs one of the following actions:

- If the markup configuration is changed, it regenerates markup data completely.
- If only a small portion of the markup data is inconsistent, it simply repairs this portion of the data.
- Otherwise, it regenerates markup data completely.

The following example illustrates markup data being repaired:

```
# /etc/init.d/ugca stop

Stopping UGCA
UGCA stopped

# /opt/CSCOugca/bin/check_markup.sh

Start checking markup data ...
Detected markup data inconsistency.
Repairing markup data ...
Regenerated markup data for 150 row(s).
Markup checking completed successfully.

# /etc/init.d/ugca start

Starting UGCA
[1] 23047
[1] 23052
[1] 23054
UGCA started
```

Note Currently, Cisco UGCA provides mostly error checking of the file markup.xml at the syntax level. Logical errors such as invalid enumeration values may not be detected.

Managing the Volume of Log Files

The logging facility of Cisco UGCA captures information about the state of the system. This information is stored as text files in the directory *install*/CSCOugca/logs. As these files increasingly consume space on the hard disk, memory consumption must be managed.

Two parameters, or "pair keys," in the configuration file *install*/CSCOugca/etc/config/ugca.xml control the logging files volume:

- *count*—Logging file rotation counter. Defines how many recyclable logging files the system uses to store the latest logging information
- *limit*—Logging file size, specified in bytes.

These pair keys are found under the section "Module logger" in the file. See Appendix A, "Cisco UGCA Configuration File: ugca.xml."

For example, if *limit* is set to 500000 (500 KB) and *count* is set to 5, the total cumulative size of the log files will be no more than 2.5 MB.

Editing Logging Parameters

To edit the limit and count log file parameters (pair keys), do the following:

Step 1 Edit and save the file *install*/CSCOugca/etc/config/ugca.xml.

Step 2 Stop and restart the system.

Accessing and Reading Raw Data

Cisco UGCA provides a limited read-only access to the historical data the application stores. For more information and a presentation of the database schema, see Appendix C, "Cisco UGCA Database Schema."

Changing Database Account Passwords

Cisco UGCA contains two database accounts, a system account that is used by Cisco UGCA and a user account for read-only access to the raw data. See the Accessing and Reading Raw Data section in this chpater.

Note Before changing the database account passwords, shut down the Cisco UGCA server. See the Stopping Cisco UGCA section in this chapter.

To change the system password, enter the following:

```
install/CSCOugca/bin/db_passwd.sh -system
```

To change the user password, enter the following:

```
install/CSCOugca/bin/db_passwd.sh -user
```

In both cases, follow the prompts to complete the changes.

Using Cisco UGCA Utility Tools

Cisco UGCA provides a variety of utility tools, or scripts, to assist you in general maintenance as well as in troubleshooting:

- Script to Correct Certain Cisco Call Tracker Messages
- Example Scripts to Schedule the Loading of syslog Files
- Script to Schedule Report Generation
- Script to Repair Database Corruption

Script to Correct Certain Cisco Call Tracker Messages

Some releases of Cisco IOS have Cisco Call Tracker syslog messages that need repair. The following are currently known issues:

- Commas may be missing between some fields.
- Some 32-bit counters are signed and hence may show negative values for values greater than 2,147,483,647 (the maximum positive value for an integer).

Note A missing comma has been identified after *authen=aValue*. Another has been identified in *CALLTRKR-6-CALL_RECORD* before the *init-rx* field. Negative values for numbers greater than the maximum positive value of an integer have been identified for 32-bit counters of *rx/tx chars* in *CALLTRKR-6-CALL_RECORD* and *rx/tx: chars* in CALLTRKR-6-MODEM_CALL_REC. The utility script deals only with these defects. For any other defects you can edit the script to make the desired corrections to the syslog records.

A scripting tool has been developed to address these issues, by correcting malformed Cisco Call Tracker syslog messages into correct ones that can be loaded into the Cisco UGCA application. The scripting tool is on the application CD-ROM, in the directory /utils/tools. The scripting file name is repairSyslog.awk.

Launching the Script

You can launch the script from the directory /utils/tools, or you can copy it to a working directory and launch from there.

Usage is as follows:

```
# ./repairSyslog.nawk <input_file_name> <output_file_name>
```

where

input_file_name is the input syslog file to be fixed, and *output_file_name* is the output (fixed) syslog file that can be loaded into the Cisco UGCA application.

The script provides a summary containing information about the following:

- The number of input and output records
- The number of modem and call records processed
- The number of modem and call records that have been changed
- The input file and output file names provided by the user

Example

```
# ./repairSyslog.nawk /opt/newcisco.log.1.new /opt/newcisco.log.1.new.out

The script is running. Please wait...

Running Script Results
==============================================
  Total Input Records:  160373
 Total Output Records:  160373
  Total Modem Records:  79998
   Total Call Records:  80375
Modem Records Changed:  757
 Call Records Changed:  769
    Input Syslog File:  /opt/newcisco.log.1.new
   Output Syslog File:  /opt/newcisco.log.1.new.out
```

Example Scripts to Schedule the Loading of syslog Files

Two example scripts, push_syslog.sh and pull_syslog.sh, are provided to schedule the loading of syslog files by means of the UNIX cron utility. These scripts address the following scenario:

- A syslogd (syslog daemon) is running on a UNIX machine, and is able to receive syslog messages. It dumps these messages to /var/log/syslog7.log.
- Cisco UGCA is running on a UNIX machine. (This can be different from the machine on which the syslogd is running.)
- We have two objectives:
  - Pull the dumped syslog files, verify them, and put them where Cisco UGCA can access them.
  - Run CDRLoader to load those syslog files into Cisco UGCA.

Caution The environment setup and configuration may vary considerably from customer to customer. Consequently, these scripts should be used only as a starting point, and should be modified according to the actual environment.

push_syslog.sh

This script takes two parameters: *input_file_name* and *output_file_name*. It reads the input file (in this example, /var/log/syslog7.log), repairs the syslog files, and saves the verified files to the output file. It then clears the input file after the result file is created, in preparation for the next invocation. The output file can be accessed by the Cisco UGCA system.

pull_syslog.sh

This script takes one parameter: *input_file_name*. It invokes CDRLoader to load the input file. CDRLoader automatically provides the output file name, which is the input file name plus the time stamp of the invocation. The output file is created in the same directory as the input file.

Note Reserve sufficient time between the input and output files, so that push_syslog.sh can create its output file completely.

Both scripts can be invoked as cron tasks.

Example

The following example creates a cron entry for each script to process syslog once a day at 01:00 UTC, with 15 minutes between the input and output files:

```
0 1 * * * /opt/CSCOugca/bin/push_syslog.sh /var/log/local7.log /tmp/local7.log >/dev/null
2>&1
15 1 * * * /opt/CSCOugca/bin/pull_syslog.sh /tmp/local7.log >/dev/null 2>&1
```

The scripts are located on the application CD-ROM, in the directory /utils/sample.

Script to Schedule Report Generation

A sample script, run_sch_report.sh is provided to schedule report generation by means of the UNIX cron utility. By default, the content of this script is completely commented out. You can modify or add new entries to this script, as well as create a cron entry for it.

Caution Because the environment setup and configuration may vary considerably from customer to customer, this script should be used only as a starting point.

Example

```
0 2 * * * /opt/CSCOugca/bin/run_sch_report.sh >/dev/null 2>&1
```

Note If you want to generate a report after CDR data is loaded to the system, make sure to leave enough time between CDRLoader's loading of the cron entry and the report generation. Also, during report generation, make sure no other maintenance activities or CDR loading are running.

The script is located on the application CD-ROM, in the directory /utils/sample.

Script to Repair Database Corruption

In case the database becomes corrupt, it is possible to repair it.

Determining Database Corruption

If the database is corrupt, console and syslog messages so indicate during a backup. In addition, the following error message appears when the application starts. The message also appears in the log files.

```
com.cisco.nm.ugca.common.db.DatabaseException: java.sql.SQLException: General error: Can't
open file: 'call_record.MYD'. (errno: 144)
```

Also, other database error messages may appear in the log files. Check these files for messages if all other possible causes of database corruption have been eliminated. However, the absence of messages is not an indication of database integrity.

Repairing Database Corruption

Once you have determined that the database is corrupted, you must repair it. To do so, run the script *myisamchk script*, as indicated below.

Caution Stop the system before proceeding. See Stopping Cisco UGCA, page 3-19. If you do not stop the system first, bogus errors are reported. If you try to fix these bogus errors, further corruption results.

Step 1 To check for database corruption, use the following command:

```
install/CSCOugca/mysql/bin/myisamchk --silent --fast --key_buffer=128M --sort_buffer=128M
--read_buffer=2M --write_buffer=2M <install>/CSCOugca/mysql/data/ugca/*.MYI
```

This checks for errors in all *.MYI* files.

Step 2 If errors are reported, repeat Step 1, but add *--recover* to the list of options only for the *.MYI* file reported to have errors.

Troubleshooting

This section contains known techniques that can resolve problems and improve performance.

Tip See also Using Cisco UGCA Utility Tools, page 3-36.

Optimizing System Performance

Exploiting Parallelism

The report engine can execute multiple reports in parallel. In turn, the maximum number of reports that are executable in parallel is determined by the configuration parameter *nThreads* in the following file:

install/CSCOugca/etc/config/ugca.xml

The default value for *nThreads* has been set to 1. This number optimizes performance on the minimum recommended hardware, the Sun Ultra 10. However, in some cases, on powerful multi-CPU machines with large memory and fast disks, it is beneficial to increase this default value to run parallel report-engine threads.

Where the machine is sufficiently capable, the user is free to experiment with the configuration parameter *nThreads*. If performance appears not to be affected, you can adjust this parameter and evaluate its effects. The file, presented in Appendix A, “Cisco UGCA Configuration File: ugca.xml,”, should provide sufficient comments to enable you to understand the various parameters.

Note Although the system has been tuned for the hardware specified, a Sun Ultra 10, it scales well to higher-end hardware. Because CDR loading is multithreaded and limited by CPU speed, using multiple or faster processors speeds up loading. However, report running is disk bound, so using a machine with sufficient memory (2 GB or more) for the operating system to cache the database files can dramatically improve the performance of running individual reports. Once a report is running in memory and not off the hard disk, it is limited by CPU speed. In this case, faster processors help, and adding more processors and increasing the parameter *nThreads* increases parallel processing.

It is recommended that you do not run multiple reports in parallel (change *nThreads* > 1) while reports are still running off the disk. The system is disk bound, and this slows down report generation.

Cisco UGCA Configuration File: ugca.xml

The following is a listing of the annotated configuration file, *install*/CSCOugca/etc/config/ugca.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Master configuration file for Cisco UGCA -
  Universal Gateway Call Analyzer

  N.B. Whilst this file is in XML format, certain UGCA utilities
       use simple string matching to find entries in this file.
       These utilities will fail if formatting (e.g. whitespace)
       changes are made. Please refer to CSCea45974.
  -->

<!DOCTYPE ugca SYSTEM "file://localhost/_BASEDIR_/CSCOugca/etc/dtd/config.dtd">

<ugca>
  <!-- Module common -->
  <module name="common">
    <!-- Two following values must be kept in sync:
         serverDateFormat is Date (Year to Day) format in MySQL notation;
         reportDateFormat must be exactly the same but in Java SDF notation.
         These formats define the exact format of 'date' fields used
         in GROUP BY clauses in ReportResults files instances.
         -->
    <pair key="reportDateFormat"  value="yyyy/MM/dd"/>
    <pair key="serverDateFormat"  value="%Y/%m/%d"/>
    <!--
    The next value defines the number of positions after decimal
    points to be used in the ReportResults floating point data values
    -->
    <pair key="floatDecimalPositions"  value="3"/>
  </module>

  <!-- Module db -->
  <module name="db">
    <!-- Parameters used by database access using JDBC driver.
         The URL will be created from these parameters, for
         examples: jdbc:mysql://localhost/ugca_db?user=myuser.

         driver:   full Java class name of the JDBC driver,
                   default: org.gjt.mm.mysql.Driver
         server:   database server host name, default: localhost
         prefix:   protocol name for JDBC, default: jdbc:mysql
         database: database name, default: ugca
         seed:     encryption key
         username: database user name, default: root
         password: encrypted form of password
```

```
   rousername: read-only user
       ropassword: encrypted form of read-only password
   N.B. Do NOT edit seed, username, password, rousername nor
            ropassword. These are managed by the db_passwd utility.
       -->
  <pair key="driver"            value="org.gjt.mm.mysql.Driver"/>
  <pair key="server"            value="localhost"/>
  <pair key="prefix"            value="jdbc:mysql"/>
  <pair key="database"          value="ugca"/>
  <pair key="seed"              value="40dfcef1b5ab5ba8"/>
  <pair key="username"          value="root"/>
  <pair key="password"          value="6f6a036f4abf5c6bc3658b15eebe1382"/>
  <pair key="rousername"        value="ugca"/>
  <pair key="ropassword"        value="d6a6a51240c8d577"/>

  <!-- If this variable is set to true, the database driver will
       try to reconnect to the database after connection are lost.
       Default value: true
       -->
  <pair key="autoReconnect"     value="true"/>

  <!-- Size of the database connection pool.
       Default value: 30
       -->
  <pair key="poolSize"          value="30"/>

  <!-- Maximal waiting time (in seconds) for a thread, when
       there is no database connection available in the connection
       pool. If no connections become available during this time,
       the request to retrieve a connection fails.
       Default value: 10
       -->
  <pair key="poolMaxWaitTime"   value="10"/>
</module>

<!-- Module cdrloader -->
<module name="cdrloader">

  <!-- Maximal size of the message queue used internally
       in the CDR message loader. In general, a larger
       queue size provides the loader better capability
       to accept large numbers of messages in short time,
       but this may cause higher memory consumption.
       Default value: 4000
       -->
  <pair key="queueSize"         value="4000"/>

  <!-- Threshold used for monitoring of the message queue.
       A snapshot of the queue activity is taken after this
       number of message is inserted to the queue.
       Default value: 20% of queueSize
       -->
  <pair key="queueSampleThreshold" value="800"/>

  <!-- Maximal size of the message queue used for matching
       between a call record and its corresponding modem call
       record in the case of a modem call. A larger queue
       size may cause higher memory consumption.
       Default value: 100
       -->
  <pair key="mcrMatchQueueSize" value="100"/>

  <!-- Maximal size of the message queue used for unmatched
       call records.
```

```
        Default value: 250
        -->
  <pair key="UnmatchedCRQueueSize" value="250"/>

  <!-- Maximal number of CDR message processors. Each
        processor consumes a database connection. If the
        maximal number of processors is constantly observed
        in the statistics, it's beneficial to reduce this
        number, for example 5.
        Default value: 10
        -->
  <pair key="maxProcessors"     value="10"/>

  <!-- Enable/disable statistics. This is mostly used for
        tuning, it should be disabled under normal operation.
        Statistics will be written to the log file with the
        logging level "INFO".
        Default value: false
        -->
  <pair key="statisticsEnabled" value="false"/>

  <!-- Interval between statistics in second.
        Default value: 60
        -->
  <pair key="sampleInterval"    value="60"/>
</module>

<!-- Module backup -->
<module name="backup">
  <pair key="backupdir"         value="_BASEDIR_/CSCOugca/backup"/>
</module>

<!-- Module markup -->
<module name="markup">
  <!-- Regeneration of markup data is done chunk by chunk.
        This variable specifies the chunk size. A larger chunk
        size may cause shorter markup data regeneration time, but
        increase memory consumption. And a too large chunk size
        may cause more frequent paging, and even longer markup
        data regeneration time.
        Default value: 5000
        -->
  <pair key="regenChunkSize"    value="5000"/>

  <!-- When inconsistency in markup data is detected, the existing
        data is either repaired or regenerated depending on the
        number of inconsistent entries in the database. If the number
        is greater than this variable, existing markup data is simply
        removed and regenerated; otherwise existing markup data is
        repaired.
        Default value: 1000
        -->
  <pair key="regenThreshold"    value="1000"/>
</module>

<!-- Module processmgr -->
<module name="processmgr">
  <!-- WARNING: Please DON'T modify any attributes in this seciton!
                Improper changes may cause failure of system startup.
        -->
  <pair key="reportengine_prefix"         value="/usr/bin/nice -n 10 "/>
  <pair key="reportengine_jvmoption"       value=" "/>
  <pair key="cdrfromfile_prefix"    value="/usr/bin/nice -n 10 "/>
  <pair key="cdrfromfile_jvmoption" value=" "/>
```

```
</module>

<!-- Module logger -->
<!-- This section contans configuration for topmost logger, logging
     file handler and logging levels
     -->
<module name="logger">
  <!-- The topmost logger retains logger properties defined for the process.
        All descendant loggers inherit these properties.
        Default: com.cisco.nm.ugca
        -->
  <pair key="topmost_logger"  value="com.cisco.nm.ugca"/>

  <!-- Log output directory
       Default: _BASEDIR_/CSCOugca/logs
       -->
  <pair key="dir"             value="_BASEDIR_/CSCOugca/logs/"/>

  <!-- Suffix with generation number to distinguish rotated logs and extension
       Default: _%g.log
       -->
  <pair key="suffix"          value="_%g.log"/>

  <!-- Log messages append to existing file after system restarts
       Default: false
       -->
  <pair key="append"          value="true"/>

  <!-- Rotation counter. User can manage this parameter
       Default: 5
       -->
  <pair key="count"           value="5"/>

  <!-- Log file size limit in bytes. User can manage this parameter
       Default: 500000
       -->
  <pair key="limit"           value="500000"/>

  <!-- Log file format XML or simple, false means simple
       Default: false
       -->
  <pair key="xml_format"      value="false"/>

  <!-- Logging levels for packages
       Default: INFO    com.cisco.nm.ugca
        -->
  <pair key="OFF"             value=""/>
  <pair key="SEVERE"          value=""/>
  <pair key="WARNING"         value=""/>
  <pair key="INFO"            value="com.cisco.nm.ugca"/>
  <pair key="CONFIG"          value=""/>
  <pair key="FINE"            value=""/>
  <pair key="FINER"           value=""/>
  <pair key="FINEST"          value=""/>
  <pair key="ALL"             value=""/>
</module>

<!-- Module export -->
<module name="export">
  <!-- The directory for exporting report result
       -->
  <pair key="exportdir"         value="_BASEDIR_/CSCOugca/export"/>
</module>
```

```
<!-- Module purge -->
<module name="purge">
  <!-- Parameters for purge utility.
       callrecordsage is for purging table call_record,
       modem_call_rec and markup. Its value must be a
       number >=0 and <=2000.
       i.e If callrecordsage is 30, then in the above
       tables, any data whose date older than 30 days
       will be purged.

       tasksage is for purging table task_rt, task_def.
       Its value must be a number >=0 and <=100.
       i.e If tasksage is 7, then in the above tables
       data whose date older than 7 days will be purged.

       tomcatlogsage is for purging TOMCAT log files under
       directory jakarta-tomcat-4.0.3/logs.Its value must be
       a number >=1 and <=100. i.e If tomcatlogsage is 7,
       then files whose modification time(precise to seconds)
       older than 7 days will be purged.
       -->
  <pair key="callrecordsage"         value="30"/>
  <pair key="tasksage"               value="7"/>
  <pair key="tomcatlogsage"          value="7"/>
</module>

<!-- Module report -->
<module name="report">
  <!-- Parameters for ReportEngine part of BackEnd.

       Number of threads defines number of reports the engine
       is capable of running simultaneously. If thread is
       active it uses exclusively its own database connection.
       Therefore the total number of database connections for
       ReportEngine is nThreads+1 (extra one is for the main thread).
       If being inactive, the threads wakes up every threadWaitInterval ms.
       The database connection is retained for at least threadHoldInterval ms
       before being release by the thread that ceased to be active.
   The default value of nThreads is set to 1, which is optimal for
   small installations (U10 with 512M memory). Machines with more memory
   and faster disks can get better ammortized performance with multiple
   parallel report engine threads.
       -->
  <pair key="nThreads"               value="1"/>
  <pair key="threadHoldInterval"     value="600000"/>
  <pair key="threadWaitInterval"     value="60000"/>

  <!-- Result Limit for number of rows in report results.
       Every reporting query uses this value as LIMIT.
       If number of actual results exceeds this resultLimit,
       the reporting task will still be successful but details
       will contain indication of truncation to respective number of rows.
       To work around this limit it is recommended first
       to adjust report's conditions and grouping values to
       produce queries working with data segments separately.
       If nevertheless decided to increase resultLimit the user
       should take precaution to watch for 'Out of memory' exceptions
       in possible dailure scenarios and must be aware that reports
       in general will take more time to complete.
       It is not recommended to exceed value of 10000 anyway
       -->
  <pair key="resultLimit"            value="5000"/>
</module>
```

```
  <!-- Module fe -->
  <module name="fe">
    <pair key="feRRCacheEnable" value="true"/>
    <pair key="RRAskBackEnd"    value="true"/>
    <pair key="memoryRatioLimit"    value="40"/>
    <!--  maxMemoryBugAdjust value is required as a workaround for JDK1.4 bug 4686462 -->
    <pair key="maxMemoryBugAdjust"    value="67108864"/>
  </module>

  <!-- Module getfieldvalues -->
  <module name="getfieldvalues">
    <!-- Parameters for "Get Values" action in page "Conditions"
         of Create/Modify report spec wizard.
         "resultLimit" means: For all the non-enum field that
         "Get Values" supports, the maximum number of values to
         to be displayed. The value of "resultLimit" must be
         >= 0. In case of an invalid value, default value 100
         will be used.
         -->
    <pair key="resultLimit"             value="100"/>
  </module>
</ugca>
```

BNF for Markup Interpreter: Terminals and Nonterminals

This appendix lists terminals and nonterminals used by the markup interpreter. Backaus Naur Form (BNF) is a formal compiler-syntax notation system. A terminals is a symbol that is to be written exactly as represented (for example, a token). A nonterminals is a sequence of alternatives consisting of terminals or nonterminals separated by a metasymbol such as "|".

Terminals

```
<IDENTIFIER> ::= <LETTER> (<LETTER>|<DIGIT>|<SYMBOL>)*
    <LETTER> ::= ["a"-"z", "A"-"Z"]
    <SYMBOL> ::= ["_", "."]
  <STR_CHAR> ::= (~["\"","\\","\n","\r"])
               | ("\\"
                   ( ["n","t","b","r","f","\\","'","\""]
                   | ["0"-"7"] ( ["0"-"7"] )?
                   | ["0"-"3"] ["0"-"7"] ["0"-"7"]
                  )
     <DIGIT> ::= ["0"-"9"]
<BOOL_VALUE> ::= "TRUE"|"FALSE"|"true"|"false"
```

Nonterminals

```
           CompilationUnit ::= ( Statement )* <EOF>
               Expression  ::= Assignment
                             | ConditionalMatchExpression
               Assignment  ::= PrimaryExpression "=" Expression
ConditionalMatchExpression ::= ConditionalOrExpression ( "~~" ConditionalOrExpression )*
    ConditionalOrExpression ::= ConditionalAndExpression ( "||" ConditionalAndExpression
)*
  ConditionalAndExpression  ::= EqualityExpression ( "&&" EqualityExpression )*
       EqualityExpression  ::= RelationalExpression ( "==" RelationalExpression |
                                                      "!=" RelationalExpression )*
     RelationalExpression  ::= MatchingExpression ( "<" MatchingExpression |
                                                    ">" MatchingExpression |
                                                    "<=" MatchingExpression |
                                                    ">=" MatchingExpression )*
       MatchingExpression  ::= AdditiveExpression ( "~" StringLiteral )?
       AdditiveExpression  ::= MultiplicativeExpression ( "+" MultiplicativeExpression |
                                                          "-" MultiplicativeExpression )*
  MultiplicativeExpression ::= PrimaryExpression ( "*" PrimaryExpression |
                                                   "/" PrimaryExpression )*
         PrimaryExpression ::= Literal
                             | Variable
```

```
                    | "(" Expression ")"
        Variable  ::= <IDENTIFIER>
          Literal ::= ( <NUMBER_LITERAL> )
                    | ( <BOOL_LITERAL> )
                    | StringLiteral
  StringLiteral ::= ( <STRING_LITERAL> )
      Statement ::= ";"
                    | Block
                    | StatementExpression
                    | IfStatement
          Block ::= "{" ( Statement )* "}"
StatementExpression ::= Assignment ";"
    IfStatement ::= "if" "(" Expression ")" Statement ( "else" Statement )?
```

Cisco UGCA Database Schema

Cisco UGCA provides a very limited read-only access to the historical data the application stores. The database used is MySQL, which listens to the UNIX socket */tmp/mysql.sock* and the *port localhost:3306*. The username is *ugca*. The password is the user database account password that was set during installation or by following the steps in the Changing Database Account Passwords section in Chapter 3, "Starting, Using, and Maintaining Cisco UGCA". The database is *ugca*.

Note For added security, the port is available only to applications on the localhost . If you want to connect to the database from another machine (for example, with Access and MySQL ODBC connector on a Windows platform), then you must configure some form of secure port forwarding (such as *ssh*) between the remote machine and the server on which Cisco UGCA is installed. You will also need to forward the local TCP port 3306 to the remote machine's localhost port 3306.

A sample of clients that can be used to connect to the database is provided on the Cisco UGCA CD, in the directory */utils/clients* . This is only a sample of clients that can be used, and they are provided on the CD only to save the user having to download them. Cisco does not provide any guarantees or support for these clients.

This appendix presents the following examples:

- Example of call_record Table
- Example of modem_call_rec Table

Example of call_record Table

Every successfully loaded CALLTRKR-6-CALL_RECORD is stored in the *call_record* table, with the schema as follows:

```
+------------+---------------------+
| Field      | Type                |
+------------+---------------------+
| recid      | bigint(20)          |
| logid      | int(11) unsigned    |
| date       | datetime            |
| node       | char(100) binary    |
| ct_hndl    | int(11) unsigned    |
| service    | tinyint(3) unsigned |
| origin     | tinyint(3) unsigned |
| category   | tinyint(3) unsigned |
| ds0slot    | smallint(6)         |
| ds0port    | smallint(6)         |
| ds1        | smallint(6)         |
```

```
| channel    | smallint(6)         |
| called     | char(64) binary     |
| calling    | char(64) binary     |
| resslot    | smallint(6)         |
| resport    | smallint(6)         |
| userid     | char(50) binary     |
| ipaddr     | char(15) binary     |
| ipmask     | char(15) binary     |
| accountid  | char(64) binary     |
| setup      | datetime            |
| conntime   | float               |
| phystime   | float               |
| servtime   | float               |
| authtime   | float               |
| initrxrate | int(11)             |
| inittxrate | int(11)             |
| rxchars    | int(11) unsigned    |
| txchars    | int(11) unsigned    |
| duration   | float               |
| discid     | smallint(6)         |
| disccode   | smallint(6)         |
+------------+---------------------+
```

Discussion

Note the following definitions:

- *recid*—A unique key identifying the record and partner records in other tables.
- *logid*—The syslog ID of the message.
- *date*—The timestamp of the syslog message.
- *node*—The host field of the syslog message.

For definitions of all other fields, see the reference provided in Understanding Cisco IOS Call Tracker Outputs, page 1-9, with the following modifications:

- *service*, *origin*, and *category*—These are numerated fields with the enumerations in *install*/CSCOugca/etc/config/enum.xml.
- *discid—This* is an enumeration formed from the *disc subsys* and *disc text* fields in the call record. The enumerations are stored in the table disc_syscode, shown below:

```
+-------------+------------------+
| Field       | Type             |
+-------------+------------------+
| discid      | int(11)          |
| disc_subsys | char(20) binary  |
| disc_text   | char(255) binary |
+-------------+------------------+
```

Example of modem_call_rec Table

Every successfully loaded CALLTRKR-6-CALL_RECORD is stored in the modem_call_rec table, with the schema as follows:

Field	Type
recid	bigint(20)
logid	int(11) unsigned
last_prot	tinyint(3) unsigned
attempt_prot	tinyint(3) unsigned
last_comp	tinyint(3) unsigned
last_std	tinyint(3) unsigned
attempt_std	tinyint(3) unsigned
init_std	tinyint(3) unsigned
snr	tinyint(3) unsigned
sq	tinyint(3) unsigned
rxchars	int(11) unsigned
txchars	int(11) unsigned
rxecframes	int(11)
txecframes	int(11)
rxecbad	int(11)
lastrxrate	int(11)
lasttxrate	int(11)
lowrxrate	int(11)
lowtxrate	int(11)
highrxrate	int(11)
hightxrate	int(11)
desclient_rxrate	int(11)
desclient_txrate	int(11)
deshost_rxrate	int(11)
deshost_txrate	int(11)
local_retr	smallint(6)
remote_retr	smallint(6)
fail_retr	smallint(6)
local_shift_up	smallint(6)
local_shift_down	smallint(6)
remote_shift_up	smallint(6)
remote_shift_down	smallint(6)
fail_shift	smallint(6)
v90_stat	tinyint(3) unsigned
v90_client	tinyint(3) unsigned
v90_fail	tinyint(3) unsigned
duration	float
disc_reason_type	smallint(6)
disc_reason	char(6) binary

Discussion

The parameters *recid* and *logid* are the same as in the call_record table, above.

For definitions of all other fields, see the reference provided in Understanding Cisco IOS Call Tracker Outputs, page 1-9. Note also that *last_prot*, *attempt_prot*, *last_comp*, *last_std*, *attempt_std*, *init_std*, *v90_stat*, *v90_client*, and *v90_fail*, as well as the enumerations, are in the same file as for call_record.

Enumerations

For both call_record and modem_call_rec, enumerations are used on certain fields, as a number is stored in the database only when an actual string is received. This is because those fields can have only a limited set of values (for example, *origin* can be only *Originate* or *Answer*). The mappings between the values stored in the database and the actual text are found in the file *install/*CSCOugcs/etc/config/enum.xml. It does not matter whether they are used in call_record or modem_call_record. The file enum.xml is listed in Appendix E, "Cisco UGCA Enumeration Types: enum.xml."

Markup Data

For more information, see Maintaining Markup Data, page 3-26. The file markup.xml, in *install/*CSCOugca/etc/config/markup.xml, is listed in Appendix D, "Cisco UGCA Markup: markup.xml and call_record.discid."

Cisco UGCA Markup: markup.xml and call_record.discid

This appendix presents the text of the file *install*/CSCOugca/etc/config/markup.xml. “Color” markup is detailed here.

In addition, the values for call_record.discid, which are used to mark up color, are presented here as well. call_record.discid is actually the table disc_syscode in the database to which the user has read-only access.

markup.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Markup setup file for Cisco UGCA -
  Universal Gateway Call Analyzer

  For special characters, the following have to used:

    &amp;    '&'
    &apos;   '''
    &quot;   '"'
    &lt      '<'
    &gt      '>'
  -->

<!DOCTYPE MarkupSetup SYSTEM "file://localhost/_BASEDIR_/CSCOugca/etc/dtd/markup.dtd">

<MarkupSetup>
  <MarkupAttribute Name = "recid"
                   Type = "BIGINT"
                   Unique = "true"
                   Rule = "
      value = call_record.recid;
          ">
  </MarkupAttribute>

  <MarkupAttribute Name = "call_class"
                   Type = "TINYINT UNSIGNED"
                   Unique = "false"
                   Default = "1"
                   Rule = "
      value = 1;
      if(call_record.discid ==  8 ||
         call_record.discid == 10 ||
```

```
   call_record.discid == 11 ||
   call_record.discid == 12 ||
   call_record.discid == 13 ||
   call_record.discid == 14 ||
   call_record.discid == 15 ||
   call_record.discid == 16 ||
   call_record.discid == 17 ||
   call_record.discid == 18 ||
   call_record.discid == 19 ||
   call_record.discid == 20 ||
   call_record.discid == 21 ||
   call_record.discid == 22 ||
   call_record.discid == 23 ||
   call_record.discid == 25 ||
   call_record.discid == 26 ||
   call_record.discid == 27 ||
   call_record.discid == 28 ||
   call_record.discid == 29 ||
   call_record.discid == 30 ||
   call_record.discid == 31 ||
   call_record.discid == 32 ||
   call_record.discid == 33 ||
   call_record.discid == 34 ||
   call_record.discid == 35 ||
   call_record.discid == 36 ||
   call_record.discid == 35 ||
   call_record.discid == 36 ||
   call_record.discid == 37 ||
   call_record.discid == 39 ||
   call_record.discid == 41 ||
   call_record.discid == 42 ||
   call_record.discid == 43 ||
   call_record.discid == 56 ||
   call_record.discid == 57 ||
   call_record.discid == 60 ||
   call_record.discid == 61 ||
   call_record.discid == 62) {
  value = 4;
}
if(call_record.duration &lt;= 60 ||
   call_record.rxchars &lt; 800 ||
   call_record.txchars &lt; 800) {
  value = 5;
}
if( value &lt; 2 ) {
    if( modem_call_rec.remote_retr &gt;= 2 ||
        call_record.servtime &gt;= 40) {
      value = 2;
    }
}
if( value &lt; 3 ) {
    if( (modem_call_rec.snr &gt; 0 &amp;&amp; modem_call_rec.snr &lt;= 30 ) ||
        modem_call_rec.sq &lt;= 3 ||
        modem_call_rec.local_retr &gt;= 2) {
      value = 3;
    }
}
if( value &lt; 4 ) {
  if(modem_call_rec.fail_retr &gt;= 1) {
    value = 4;
  }
}
    ">
<MarkupAttrEnumType DefaultName  = "Normal"
```

```
                    DefaultValue = "1">
    <Pair Name="Normal"                Value="1"/>
    <Pair Name="Slow_Throughput"       Value="2"/>
    <Pair Name="Noisy_Line"            Value="3"/>
    <Pair Name="Abnormal_Disconnect"   Value="4"/>
    <Pair Name="No_Logon"              Value="5"/>
  </MarkupAttrEnumType>
</MarkupAttribute>

<MarkupAttribute Name = "cr_dur_bucket"
                 Type = "TINYINT UNSIGNED"
                 Unique = "false"
                 Default = "1"
                 Rule = "
    if(call_record.duration &gt; 300) {
      value = 3;
    }
    else if(call_record.duration &gt; 60) {
      value = 2;
    }
    else {
      value = 1;
    }
        ">
  <MarkupAttrEnumType DefaultName  = "0-60"
                      DefaultValue = "1">
    <Pair Name="0-60"      Value="1"/>
    <Pair Name="60-300"    Value="2"/>
    <Pair Name="300-"      Value="3"/>
  </MarkupAttrEnumType>
</MarkupAttribute>

<MarkupAttribute Name = "mcr_dur_bucket"
                 Type = "TINYINT UNSIGNED"
                 Unique = "false"
                 Default = "1"
                 Rule = "
    if(modem_call_rec.duration &gt; 300) {
      value = 3;
    }
    else if(modem_call_rec.duration &gt; 60) {
      value = 2;
    }
    else {
      value = 1;
    }
        ">
  <MarkupAttrEnumType DefaultName  = "0-60"
                      DefaultValue = "1">
    <Pair Name="0-60"      Value="1"/>
    <Pair Name="60-300"    Value="2"/>
    <Pair Name="300-"      Value="3"/>
  </MarkupAttrEnumType>
</MarkupAttribute>

<MarkupAttribute Name = "user_gr"
                 Type = "TINYINT UNSIGNED"
                 Unique = "false"
                 Default = "1"
                 Rule = "
    if(call_record.userid ~~ &quot;^aolnet/&quot;) {
      value = 2;
    }
    else if(call_record.userid ~~ &quot;^aolip&quot;) {
```

```
        value = 3;
      }
      else if(call_record.userid ~~ &quot;^aol&quot;) {
        value = 4;
      }
      else {
        value = 1;
      }
          ">
    <MarkupAttrEnumType DefaultName  = "Other"
                        DefaultValue = "1">
      <Pair Name="Other"       Value="1"/>
      <Pair Name="aolNet"      Value="2"/>
      <Pair Name="aolip"       Value="3"/>
      <Pair Name="aol"         Value="4"/>
    </MarkupAttrEnumType>
  </MarkupAttribute>

  <MarkupAttribute Name = "disc_reason_gr"
                   Type = "TINYINT UNSIGNED"
                   Unique = "false"
                   Default = "1"
                   Rule = "
      if(modem_call_rec.disc_reason == &quot;0x1F06&quot;) {
        value = 2;
      }
      else if(modem_call_rec.disc_reason == &quot;0x1F03&quot;) {
        value = 3;
      }
      else if(modem_call_rec.disc_reason == &quot;0x0220&quot;) {
        value = 4;
      }
      else {
        value = 1;
      }
          ">
    <MarkupAttrEnumType DefaultName  = "Other"
                        DefaultValue = "1">
      <Pair Name="Other"       Value="1"/>
      <Pair Name="Carrier"     Value="2"/>
      <Pair Name="IOS"         Value="3"/>
      <Pair Name="Normal"      Value="4"/>
    </MarkupAttrEnumType>
  </MarkupAttribute>

  <MarkupAttribute Name = "calling_prefix"
                   Type = "CHAR(6)"
                   Unique = "false"
                   Rule = "
      value = call_record.calling ~ &quot;[0-9]{6}&quot;;
          ">
  </MarkupAttribute>

  <MarkupAttribute Name = "called_prefix"
                   Type = "CHAR(6)"
                   Unique = "false"
                   Rule = "
      value = call_record.called ~ &quot;[0-9]{6}&quot;;
          ">
  </MarkupAttribute>

</MarkupSetup>
```

Cisco UGCA Enumeration Types: enum.xml

The following is a listing of the configuration file, *install*/CSCOugca/etc/config/enum.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Definition of enumuration types
  -->
<!DOCTYPE enums SYSTEM "file://localhost/_BASEDIR_/CSCOugca/etc/dtd/enum.dtd">

<enums>
  <type name="service"
     default="None">
    <pair name="None"       value="1"/>
    <pair name="Other"      value="2"/>
    <pair name="Exec"       value="3"/>
    <pair name="L2F"        value="4"/>
    <pair name="L2TP"       value="5"/>
    <pair name="MP"         value="6"/>
    <pair name="PPP"        value="7"/>
    <pair name="Slip"       value="8"/>
    <pair name="TcpClear"   value="9"/>
    <pair name="Telnet"     value="10"/>
  </type>
  <type name="origin"
     default="Answer">
    <pair name="Answer"     value="1"/>
    <pair name="Originate"  value="2"/>
  </type>
  <type name="category"
     default="None">
    <pair name="None"       value="1"/>
    <pair name="Other"      value="2"/>
    <pair name="CasDigital" value="3"/>
    <pair name="IsdnSync"   value="4"/>
    <pair name="Modem"      value="5"/>
    <pair name="V110"       value="6"/>
    <pair name="V120"       value="7"/>
  </type>
  <type name="prot"
     default="None">
    <pair name="None"       value="1"/>
    <pair name="Other"      value="2"/>
    <pair name="ARA1.0"     value="3"/>
    <pair name="ARA2.0"     value="4"/>
    <pair name="ASYNC Mode" value="5"/>
    <pair name="Direct"     value="6"/>
    <pair name="Fax Mode"   value="7"/>
```

```
  <pair name="Isdn Mode"  value="8"/>
  <pair name="LAP-M"      value="9"/>
  <pair name="MNP"        value="10"/>
  <pair name="SYNC Mode"  value="11"/>
  <pair name="Type 7"     value="12"/>
</type>
<type name="comp"
   default="Other">
  <pair name="None"       value="1"/>
  <pair name="Other"      value="2"/>
  <pair name="MH"         value="3"/>
  <pair name="MMR"        value="4"/>
  <pair name="MNP5"       value="5"/>
  <pair name="MR"         value="6"/>
  <pair name="V.42bis-Both" value="7"/>
  <pair name="V.42bis-RX" value="8"/>
  <pair name="V.42bis-TX" value="9"/>
  <pair name="V.44-Both"  value="10"/>
  <pair name="V.44-RX"    value="11"/>
  <pair name="V.44-TX"    value="12"/>
</type>
<type name="std"
   default="Unknown">
  <pair name="Unknown"    value="2"/>
  <pair name="Bell103"    value="3"/>
  <pair name="Bell212"    value="4"/>
  <pair name="K56Flx"     value="5"/>
  <pair name="V.17"       value="6"/>
  <pair name="V.21"       value="7"/>
  <pair name="V.22"       value="8"/>
  <pair name="V.22bis"    value="9"/>
  <pair name="V.23"       value="10"/>
  <pair name="V.27terbo"  value="11"/>
  <pair name="V.29"       value="12"/>
  <pair name="V.32"       value="13"/>
  <pair name="V.32bis"    value="14"/>
  <pair name="V.32terbo"  value="15"/>
  <pair name="V.33"       value="16"/>
  <pair name="V.34"       value="17"/>
  <pair name="V.34+"      value="18"/>
  <pair name="V.90"       value="19"/>
  <pair name="V.110"      value="20"/>
  <pair name="VFC"        value="21"/>
</type>
<type name="v90_stat"
   default="No Attempt">
  <pair name="Failure"    value="1"/>
  <pair name="No Attempt" value="2"/>
  <pair name="Success"    value="3"/>
</type>
<type name="v90_client"
      default="Unknown">
  <pair name="Unknown"    value="1"/>
  <pair name="Others"     value="2"/>
  <pair name="Rockwell"   value="3"/>
  <pair name="USR"        value="4"/>
  <pair name="Lucent"     value="5"/>
  <pair name="PCTel"      value="6"/>
</type>
<type name="v90_fail"
      default="None">
  <pair name="None"              value="1"/>
  <pair name="ClientFallBack"    value="3"/>
  <pair name="ClientNonPCM"      value="4"/>
```

```
    <pair name="ServerV90Disabled" value="5"/>
  </type>
  <type name="disc_reason_type"
     default="Unknown">
    <pair name="None"               value="0"/>
    <pair name="Unknown"            value="1"/>
    <pair name="Other"              value="2"/>
    <pair name="InCallSetup"        value="3"/>
    <pair name="InDataMode_RxOk"    value="4"/>
    <pair name="InDataMode_RxNotOk" value="5"/>
    <pair name="InDataMode_TxOk"    value="6"/>
    <pair name="InDataMode_TxNotOk" value="7"/>
  </type>
</enums>
```