Cisco Meeting Server Call Detail Reporting System Guide

Contents

[Overview 2](#_Toc77237720)

[Architecture 2](#_Toc77237721)

[Cisco Meeting Server 2](#_Toc77237722)

[CMS Meeting Manager 2](#_Toc77237723)

[Web Server 2](#_Toc77237724)

[CMS CDR 2](#_Toc77237725)

[CMS Reports 3](#_Toc77237726)

[Database Server 3](#_Toc77237727)

[Record Processing – spProcess Stored Procedures 3](#_Toc77237728)

[Reporting - spRpt\_ Stored Procedures 3](#_Toc77237729)

[Installation 4](#_Toc77237730)

[Cisco Meeting Server 4](#_Toc77237731)

[Database Server 4](#_Toc77237732)

[Web Server 4](#_Toc77237733)

[Create web apps 4](#_Toc77237734)

[Customize database connection strings 4](#_Toc77237735)

[Maintenance 4](#_Toc77237736)

[Validating Operation 4](#_Toc77237737)

[CMS Upgrades 4](#_Toc77237738)

[Database Size 4](#_Toc77237739)

[References 5](#_Toc77237740)

[Appendix 1 – Structure of CMSCDR\_Reports.xml 6](#_Toc77237741)

[XML Schema Definition (XSD) (In progress) 6](#_Toc77237742)

[Appendix 2 – Software Used 7](#_Toc77237743)

[Database Server 7](#_Toc77237744)

[Web Server 7](#_Toc77237745)

[Development Tools 7](#_Toc77237746)

# Overview

The Cisco Meeting Server Call Detail Reporting System (CMS CDR) receives, stores, processes, and provides reporting on data generated by Cisco CMS.

# Architecture

## Cisco Meeting Server

The Cisco Meeting Server (CMS) generates Call Detail Records (CDRs) internally for key call-related events, such as a new SIP connection arriving at the server, or a call being activated or deactivated. The server can be configured to send these records to a remote system to be collected and analyzed. There is no provision for records to be stored on a long-term basis on CMS, nor any way to browse CDRs. The CDR system can be used in conjunction with the Meeting Server API, with the call ID and call leg IDs values being consistent between the two systems to allow cross referencing of events and diagnostics. The Meeting Server supports up to four CDR receivers enabling you to deploy different management tools or multiple instances of the same management tool.

CDRs are sent out by CMS over HTTP or HTTPS as a series of XML documents. When new records are generated, a connection is made to the receiving system and the receiving system should expect to receive one or multiple records on this connection. When CMS has successfully sent a group of records to the receiver, those records are purged, and responsibility for their long-term storage then moves to the receiving device. CMS considers the records to have been successfully sent to the receiver if the HTTP or HTTPS connection has been successfully established, the XML record data has been sent by CMS, and the receiver has acknowledged the data with a "200 OK" HTTP response. The Call Bridge supports keepalive connections to allow it to send multiple (batches of) records on one TCP or TLS connection to a receiver.

CMS Meeting Manager

Cisco Meeting Management (CMS MM) is a management tool for Cisco's on-premises video meeting platform, Cisco Meeting Server. You can use the tool to monitor and manage meetings that are running on the platform, and it also provides information about which Cisco licenses you are using. Meeting Management, with its current feature set, is included within existing Cisco Meeting Server licensing.

CMS MM is used for real-time meeting management and has no historical reporting capability.

## Web Server

The web server hosts two IIS applications, cmscdr and cmsreports. They are separated for security and availability reasons, however, they both use the SQL Server local account ‘cmscdr’ for database access.

### CMS CDR

As described by the Cisco CMS CDR reference guide, CMS CDR receives the call detail records from CMS via HTTP POST. If recorded successfully, it will respond with a “200 OK”.

The VB controller ReceiveController.vb code simply takes the POSTed XML and uses it as a parameter for the Stored Procedure (SP) ‘spLoadRawRecords’, which INSERTs the XML into tblRawRecords.

### CMS Reports

This webapp is responsible for the reporting interface that meeting administrators use. Its resources are as follows:

* site.css – Provides HTML styling for the user-facing pages
* CMSCDR\_Reports.xml – Provides metadata for all reports, which is used to build the report list and collect the data required for each report. Metadata here must match the SP in SQL. See Appendix 1 for structure of this file.
* default.vbhtml – Data collection form built in Razor. This page dynamically builds the list of reports on the left column, and the form fields in the Main section. All forms are pre-rendered and are revealed with click events from the report menu. The “Generate Report” button performs a POST to the report.vbhtml page, including all the specified parameters of the current report form, plus a hidden ID field.
* report.vbhtml – Report Generator built in Razor. Operates as follows:
  1. Reads the POSTed ID integer field
  2. Assembles the SP command based on the parameters defined in CMSCDR\_Reports.xml. This is safer than relying on all user-provided parameters.
  3. Calls SQL Server execution of the SP and populates a dataset with 1 or more tables
  4. Renders a separate HTML table for each table that the SP returns, enabling a richer report capability.
  5. Enhances certain columns for formatting and links

## Database Server

Platform: Microsoft SQL Server

Database: CMS\_CDR

Login used: cmscdr (dbo role)

### Record Processing – spProcess Stored Procedures

As stated, all records are initially deposited into table tblRawRecords as XML blobs. Each XML blob can contain more than one call detail record. Each record is of a certain type among 10, and each record type has a different field structure. (This is all documented in the Cisco CDR Guide.) The job of parsing these records rests on the SPs prefixed with “spProcess”. spProcessAll is only a parent SP that launches all the children SPs, but it is the one that is scheduled to run every 15 minutes. In order to ensure relational data integrity, spProcessAll is programmed to fail as a whole if any of the children SPs fail.

As long as the XML makes it into tblRawRecords, the other tables can always be rebuilt. In fact, spResetAll was designed to empty all *other* tables of their data so that various forms of maintenance could be performed. As long as tblRawRecords is populated, spProcess\* will parse the XML and redistribute the data into all the respective tables.

### Reporting - spRpt\_ Stored Procedures

SPs with the “spRpt\_” prefix handle the data retrieval for reporting. An administrator can write any type of SQL query that returns rows, as long as it is reflected properly in CMSCDR\_Reports.xml.

# Installation

## Cisco Meeting Server

As a basic step for setup, CMS must be configured to send CDRs to this application. To configure the recipient of the CDRs:

1. Open the CMS Web Admin Interface.
2. Go to Configure > CDR settings.
3. In the CDR Receiver Settings section, enter the web application server’s HTTPS URI (e.g. https://server.fqdn/cmscdr/receiver)

For more information, see the official CMS CDR Guide.

## Database Server

Using Microsoft SQL Server Management Studio, or your preferred SQL Server client, load and run ‘CMS CDR Database Creation Script.sql’ while connected to the chosen SQL Server.

Set a password for the login ‘cmscdr’ and record in a safe location.

## Web Server

Pull

### Customize database connection strings

Rename ConnectionStrings.template to ConnectionStrings.config and replace variables \_\_SERVERNAME\_\_ and \_\_PASSWORD\_\_

### Create web apps

Load the SLN file into Visual Studio and deploy both projects to your selected IIS web server using Web Deploy. Web Deploy simplifies deployment of Web applications and Web sites to IIS servers, and must be installed as an application on the server. Use the [Web platform installer](https://www.microsoft.com/web/downloads/platform.aspx) to install it.

# Maintenance

## Validating Operation

* Inspecting tblRawRecords for most recent records
* Inspecting details of SQL Server Agent job history “CMS\_CDR\_Process\_All\_Records”
* Inspecting the bitProcessed field of tblRawRecords

## CMS Upgrades

Attention must be paid when upgrading CMS, as sometimes Cisco will introduce XML schema changes to the CDR records. Administrators should read the Release Notes and CDR Guide of the new product to determine what has changed. It is unlikely that the changes will break the reception and storage of the raw records, and due to the inherent resiliency of XML, it is also likely that the record processing will not break. However, it is still important to reflect and coordinate any schema changes into the following areas:

* SQL Server - call record tables
* SQL Server - spProcess\* stored procedures
* Web server - CMSCDR\_Reports.xml file

## Database Size

As of this writing, the database size is ~26MB after two months, and is estimated that the database will be 1GB after 5 years. This is considered sustainable, meaning that considering storage, processing, and memory factors, we should not have to archive/delete records from the database.

# References

* CMS Reporting URL: <https://ngiac2dcsimsv03.ng.ds.army.mil/cmsreports>
* CMS CDR URL: <https://ngiac2dcsimsv03.ng.ds.army.mil/cmscdr/receive>
* Cisco Meeting Server Release 3.2 Call Detail Records Guide @ <https://www.cisco.com/c/en/us/support/conferencing/meeting-server/products-programming-reference-guides-list.html>
* Transact-SQL Reference (Database Engine) @ <https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15>

# Appendix 1 – Structure of CMSCDR\_Reports.xml

<?xml version="1.0" encoding="utf-8"?>

<reports>

<report id="[unique integer]" name="[Friendly Name]" sp="[Stored Procedure Name]"

description="[Lengthy Description]">

<parameters>

<parameter p="[SQL SP Parameter Name]" name="[Friendly Form Name]" description="[Description]" type="[HTML Input Type]" />

.

.

</parameters>

</report>

</reports>

## XML Schema Definition (XSD) (In progress)

<?xml version = "1.0" encoding = "UTF-8"?>

<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

<xs:element name = "report">

<xs:complexType>

<xs:sequence>

<xs:element name = "id" type = "xs:int" />

<xs:element name = "name" type = "xs:string" />

<xs:element name = "sp" type = "xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

# Appendix 2 – Software Used

## Database Server

Microsoft SQL Server 2019 15.0.2000.5

running on Microsoft Windows Server 2019 Datacenter

## Web Server

IIS running on Microsoft Windows Server 2019

Publishing mechanism: Web Deploy

## Development Tools

Microsoft Visual Studio Community 2019 Version 16.10.0

VisualStudio.16.Release/16.10.0+31321.278

Microsoft .NET Framework

Version 4.8.04084

Installed Version: Community

Visual C++ 2019 00435-60000-00000-AA594

Microsoft Visual C++ 2019

.NET Core Debugging with WSL 2 1.0

ASP.NET and Web Tools 2019 16.10.525.31942

ASP.NET Web Frameworks and Tools 2019 16.10.525.31942

C# Tools 3.10.0-4.21269.26+029847714208ebe49668667c60ea5b0a294e0fcb

IntelliCode Extension 1.0

Microsoft Library Manager 2.1.113+g422d40002e.RR

Microsoft MI-Based Debugger 1.0

Microsoft Visual C++ Wizards 1.0

Microsoft Visual Studio VC Package 1.0

ProjectServicesPackage Extension 1.0

Razor (ASP.NET Core) 16.1.0.2122504+13c05c96ea6bdbe550bd88b0bf6cdddf8cde1725

SQL Server Data Tools 16.0.62105.04180

TypeScript Tools 16.0.30429.2002

Visual Basic Tools 3.10.0-4.21269.26+029847714208ebe49668667c60ea5b0a294e0fcb

Visual Studio Code Debug Adapter Host Package 1.0