Call Concentrator 7

Reference Manual

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library CD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44 (0) 118 974 7002	support@genesyslab.co.uk
Asia Pacific	+61 7 3368 6868	support@genesyslab.com.au
Japan	+81-3-5649-6821	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the *Genesys Technical Support Guide* for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys 7 Licensing Guide*.**

Released by

Genesys Telecommunications Laboratories, Inc.

http://www.genesyslab.com

Document Version: 70cc_ref_11-2004_v3.00

Table of Contents

Chapter

1 About This Document

Welcome to the *Call Concentrator 7 Reference Manual*. This manual explains how Call Concentrator structures the data it collects, provides a detailed layout of each default database table, and gives instructions for modifying the customizable tables and fields.

Call Concentrator 7 is a Reporting product that collects and processes call-based data on activity in your enterprise. It draws on information from Configuration Server and T-Server® to create detailed representations of each leg of a call and of the call as a whole. Call Concentrator then stores this data in various tables, several of which you can customize to best suit your interaction management environment. These tables provide data that your reporting applications can take up, perform additional processing on, and present.

This manual is valid only for the 7.0 release(s) of this product.

Note: For releases of this manual created for other releases of this product, please visit the Genesys Technical Support website, or request the Documentation Library CD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

This chapter provides an overview of this manual , identifies the primary audience, introduces document conventions, and lists related reference information:

Intended Audience

This manual, primarily intended for database administrators, assumes that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.

You should also be familiar with the data collection and storage requirements for your enterprise and procedures for setting up and maintaining your database.

Chapter Summaries

In addition to this opening chapter, this manual contains these chapters:

- Chapter 2, "Data Overview" on page 11, introduces Call Concentrator data sources, their representation, and data storage.
- Chapter 3, "Global Data Records" on page 23, discusses the GCDR (Global Call Details Records) table which stores information about the call as a whole.
- Chapter 4, "Single Call Details Records" on page 37, discusses the SCDR (Single Call Details Records) table which stores information about the individual call legs that constitute global calls.
- Chapter 5, "User Data Tables" on page 51, discusses the tables that Call Concentrator uses to store attached user data (that is, custom data attached to telephony events coming to and from T-Server).
- Chapter 6, "Associated Records" on page 69, discusses the AREC (Associated Records) table which collects information about Directory Number (DN) states.
- Chapter 7, "Performance Measurements" on page 79 discusses Call Concentrator 7 performance measurements.

Document Conventions

This document uses some stylistic and typographical conventions with which you might want to familiarize yourself.

Version Number

A document version number appears at the bottom of the inside front cover of this manual. Version numbers change as new information is added to this manual. Here is a sample version number:

70fr_ref_09-2003_v1.00

You will need this version number when you are talking with Genesys Technical Support about this product.

Type Styles

Italic

In this document italic is used:

- When a term is being defined.

  Example

  - *A customary and usual practice* is one that is widely accepted and used within a particular industry or profession.
- For emphasis. For example, "Do *not* use this value for this option."
- For variables, for example, $x + 1 = 7$ where x stands for . . .

Monospace

A monospace font, which is shown in the following examples, is used for:

- All programming identifiers and GUI elements—*except* for instances of these occurring in tables and figures. This convention includes the *names* of directories, files, folders, paths, scripts, dialog boxes, options, fields, text and list boxes, all buttons including radio buttons, check boxes, commands, tabs, CTI events, and error messages; the values of options; logical arguments and command syntax; and code samples.

  Examples

  - Select the `Show variables on screen` check box.
  - Click the `Summation` button.
  - On the `Properties` dialog box, enter the value for the host server in your environment.
  - In the `Operand` text box, enter your formula.
  - Click `OK` to exit the `Properties` dialog box.
  - The following table presents the complete set of error messages T-Server distributes in `EventError` events.
  - If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls.
- For any text the user must manually enter during a configuration or installation procedure:

Example

- Enter `exit` at the command line.

Correction of Errors in Screen Captures

Screen captures taken from the product GUI (graphical user interface) and used in this document may sometimes contain a minor spelling, capitalization, or grammatical error. The text accompanying and explaining the screen captures corrects such errors.

Use of Square Brackets

In any logical arguments, commands, and programming syntax presented in this document, square brackets are used to indicate that a particular parametric value is optional. That is, the value is not required to resolve a command, argument, or programming syntax. The customer/user decides whether to supply a value and what that value is. Here is a sample:

```
smcp_server -host [/flags]
```

Use of Angle Brackets

Angle brackets are used to indicate that a value in a logical argument, command, or programming syntax is required, but that the user must supply the data for the value. Because the value is specific to an individual enterprise—for example, DNs or port numbers—the program cannot predict (that is, program in) what the value is. Here is a sample:

```
smcp_server -host <confighost>
```

Related Resources

Consult these additional resources as necessary:

- *Call Concentrator 7 Getting Started Guide,* which provides an overview of Call Concentrator features and functionality, describes the architecture, and offers deployment planning recommendations.
- *Call Concentrator 7 Deployment Guide,* which provides detailed instructions for installing and configuring Call Concentrator. It also includes instructions for starting and stopping Call Concentrator using any of the available methods.
- *Technical Reference Guide for the Reporting 6.5 Release*, which contains useful information regarding call flows in the Genesys environment.
- The documentation provided for your Genesys T-Server.
- The *Framework 6.5 Load Distribution Server User's Guide*, for questions about using Call Concentrator with LDS.

- The *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library CD and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.
- The *Genesys Migration Guide*, also on the Genesys Documentation Library CD, which contains a documented migration strategy for Genesys product releases 5.x and later. Contact Genesys Technical Support for additional information.
- The Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at `http://genesyslab.com/support`.

Information on supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys 7 Supported Operating Systems and Databases*
- *Genesys 7 Supported Media Interfaces*

Genesys product documentation is available on the:

- Genesys Technical Support website at `http://genesyslab.com/support`.
- Genesys Documentation Library CD, which you can order by e-mail from Genesys Order Management at `orderman@genesyslab.com`.

Making Comments on This Document

If you especially like or dislike anything about this document, please feel free to e-mail your comments to `techpubs.webadmin@genesyslab.com`.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the information in this document only and to the way in which the information is presented. Speak to Genesys Technical Support if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Chapter

2 Data Overview

Call Concentrator draws call data from various Genesys Framework sources, transforms it, and then populates a number of database tables with call details. Call Concentrator uses conventions to represent different key kinds of data, including times, durations, and references to configuration objects. This chapter presents an overview of the data representation conventions, the database tables, and table interrelationships. It contains these sections:

Data Tables Overview

The tables deployed by Call Concentrator can be categorized into three groups corresponding to the kind of information they store:

- Call details tables: GCDR (Global Call Details Records) and SCDR (Single Call Details Records)
- User data tables: EVREF, EVDATA, EVREFEX, and GDATAEX
- Attached data table: AREC

Note: You create these tables during or just following Call Concentrator installation using the scripts described in the "Deploying SQL Scripts" chapter of the *Call Concentrator 7 Deployment Guide*.

Use of the tables in their default form is optional. You can use Call Concentrator configuration settings to customize the tables that fit your needs. Call Concentrator then writes data only in the custom tables. For more detailed

information, see "Customizing User Data Tables" on page 62 and Chapter 6, "Associated Records" on page 69).

This document assumes a standard deployment of Call Concentrator SQL scripts and describes the tables and relationships accordingly.

Data Representation

Call Concentrator uses various conventions to identify different kinds of data, including times, durations, and references to configuration objects. These conventions are discussed in the following subsections:

- Configuration Objects and their DBIDs
- Time Measurement
- Data Sources

Configuration Objects and their DBIDs

All configuration objects, that is, objects defined in the Configuration Database such as Tenants, Switches, DNs, Places, Agents, and so on, are referenced by the database identifiers (DBIDs) assigned to them by Configuration Server.

Note: Call Concentrator does not process any information about configuration objects if the objects are not configured in the Configuration Database. For example, all DNs on monitored switches that may be involved in calls should be defined in the Configuration Layer.

Time Measurement

Absolute time values are always measured as the number of *seconds* elapsed since 00:00:00 01/01/1970 UTC (Universal Coordinated Time). Call Concentrator attaches its own time information to data records rather than using time data from T-Server.

Note: Stat Server also uses its own timestamps. You must take this into consideration when combining Call Concentrator and Stat Server data in reports.

When processing data to create reports, Genesys recommends that you do not make any conversions involving local time zones until presenting the data to the user. Some processing may involve Call Concentrator(s), switches, and users located in different time zones. By keeping the data in a format

independent of a local time zone, you can minimize problems related to time zone differences.

Data Sources

The data that Call Concentrator uses to populate the fields of call records, user data, and agent tables comes from several different sources:

- **DBID**—A unique identifier assigned by Configuration Layer to a contact center object (DN, agent, and so on). Call Concentrator retrieves this DBID from the Configuration Database.
- **CCon**—The object identifiers and counters generated by Call Concentrator.
- **T-Server**—Values taken from T-Server and applied to a call or call segment (leg) without any additional processing.
- **Derived**—A value derived from other fields of this or other tables.
- **Processed**—Values taken from T-Server and processed according to the topology of calls and their segments or derived entirely from the topology of calls and their segments.

In the following chapters, these abbreviations are used to denote the source of a field value in discussing the structure of those tables:

Call Details Records and Tables

As a rule, Call Concentrator represents a call using one record in the GCDR table and one or more records in the SCDR table. This section describes the type of data stored in each table and includes a list of fields for each table:

- The GCDR Table
- The SCDR Table

GCDR Table

The record in the GCDR table stores global information about the call as a whole and the records in the SCDR table store details about call legs or segments that compose the total call. The GCDR record (and the call represented by it) is uniquely identified by the call's connection ID, which is stored in the `ConnID` field. This field is also present in the SCDR table and points to the parent GCDR record that represents the call as a whole. Table 1 on page 14 shows the fields found in the GCDR table. For a detailed description of the GCDR table, see Chapter 3, "Global Data Records" on page 23.

Table 1: GCDR Table Fields

Field Name	Description
agwttime	The time between the first appearance of a call in a call center and the first answer by a logged-in agent.
Calls	The number of calls segments constituting the call.
CallType	Call type—internal, outbound, inbound, consultation, or unknown.
ConnID	A primary key that is an identification number assigned by T-Server to a call (for example, 392732680203).
Customer	A reference to a tenant.
Destination	The value taken from the ThirdPartyDN attribute of the EventRouteUsed event.
DestLabel	Refers to the DN present in the ThirdPartyDN attribute of the EventRouteUsed event.
DN	Refers to first DN that received or initiated the call.
Dur_Cons	Total duration of consultation call segments constituting the call.
Dur_Inb	Total duration of inbound call segments constituting the call.
Dur_Int	Total duration of internal call segments constituting the call.
Dur_Outb	Total duration of outbound call segments constituting the call.
Duration	Total call time from initial connection to termination.
First_CallID	CallID from the first call segment.
First_CDIG	Reserved.
First_DNIS	DNIS (Dialed Number Identification Service) from the first call segment.
First_PHONE	The ANI (Automatic Number Identification) service or an outbound dialed number for the first call segment.
First_QUEUE	The queue DN of the first queue that the call entered.
FirstTime_Delivery	Reserved.
FirstTime_Park	The duration of the treatment (if any) applied to the first call segment.
FirstTime_Queue	The amount of time the first call segment spent in queue.
FirstTime_Ring	The amount of time the first call segment spent ringing.

Table 1: GCDR Table Fields (Continued)

Field Name	Description
FirstTime_Rout	The amount of time the first call spent segment being routed from the first routing point from which it was successfully routed.
Flag_Abandoned	A flag indicating whether the call was abandoned at the physical telephony object.
HAgent	Refers to the first agent that participated in the call.
HResult	Not used.
MediaType	Captures any media type information attached by T-Server.
N_conf	The number of call segments that have been in a conference.
N_Cons	The number of consultation call segments among those constituting the call.
N_Inb	The number of inbound call segments among those constituting the call.
N_Int	The number of internal call segments among those constituting the call.
N_Outb	The number of outbound call segments among those constituting the call.
N_park	The number of call segments given a treatment.
N_queue	The number of call segments among those constituting the call that passed through a queue.
N_trans	The number of transferred call segemtns among those constituting the call.
Project	Reserved.
StParkTime	The timestamp of the time that treatment was applied to the first call segemtn (if any).
StQueueTime	The timestamp of the first call segment that entered a queue.
StRoutTime	The timestamp of the first EventRouteRequest message.
StTime	The start date and time of the call.
Switch	Refers to the switch where the call was initiated or to the first switch that received the call.
TimeIn_conf	Total time spent by the call segments in a conference state.
TimeIn_park	Total time spent by the call segments in a treatment state.
TimeIn_queue	Total time spent by the call segments waiting in a queue.
TimeIn_trans	Total duration of the call segments that were transferred.

Table 1: GCDR Table Fields (Continued)

Field Name	Description
Tot_DialTime	Total time spent by the call segments in the dialing state. `DialTime` ends when the call is either answered or abandoned.
Tot_RingTime	Total time spent by the call segments in the ringing state.

SCDR Table

Call Concentrator creates a new SCDR record for each segment of a call that has (or can have) a connection, that is, each segment that can transmit voice, audio, or digital information. No new SCDR record is created if one endpoint cannot transmit information, such as a Queue, Route Point, Virtual DN, Virtual Queue, or Virtual Route Point. For a summary list of the fields found in the SCDR table, see Table 2. For a detailed description of the SCDR table, refer to Chapter 4, "Single Call Details Records" on page 37.

Table 2: SCDR Table Fields

Field Name	Description
ConnID	A foreign key that is an identification number assigned by T-Server to a call.
DialTime	The total time from when the call segment was first dialed until it made a connection or was abandoned.
DNIS	Dialed Number Identification Service.
EndTime	The timestamp when this call segment was terminated.
F_aban	A flag indicating whether the call segment was abandoned in queue.
F_conf	A flag indicating whether the call segment was conferenced.
F_dial	A flag indicating whether the call segment was dialed.
F_estb	A flag indicating whether the call segment was established.
F_inconf	A flag indicating whether the call segment was in conference.
F_obsrv	Not used
F_que	A flag indicating whether the call segment was in queue.
F_rels	A flag indicating whether the call segment was released.
F_retr	A flag indicating whether the call segment was retrieved.
F_ring	A flag indicating whether the call segment was rung.

Table 2: SCDR Table Fields (Continued)

Field Name	Description
F_tran	A flag indicating whether the call segment was transferred.
LocAgent	The DBID of the local agent.
LocDN	The DBID of the local agent.
LocLQ	The last local queue that the call segment passed through.
LocQueue	The first local queue the call segment passed through.
LocSwitch	The local switch.
LocTrunk	Optional number for the local trunk provided by some switches.
ParkTime	The duration of a call in a parked (or held) state.
Phone	The ANI service value for the inbound call segment or the dialed number for the outbound call segment.
RingTime	The total time from the time the call segment first rang.
RmtAgent	The remote agent.
RmtDN	The remote DN.
RmtLQ	The last remote queue that the call segment passed through.
RmtQueue	The first remote queue that the call segment passed through.
RmtSwitch	The remote switch.
RmtTrunk	The optional number for a remote trunk, provided by some switches.
RoutTime	The amount of time that the call segment spent being routed from the first routing point from which it was successfully routed.
SCallID	A primary key that is the call-segment identification; a unique identifier for a call segment that is generated by Call Concentrator.
SCallType	Call type—internal, outbound, inbound, consultation, unknown.
SCSequence	The ordinal number of the SCDR table in the sequence of SCDR records having the same value ConnID.
SDuration	The duration of the call segment from the moment it was created to the moment it was terminated.
SHResult	Not used.
SSwitchCallID	The call ID assigned to the call segment by the physical switch.

Table 2: SCDR Table Fields (Continued)

Field Name	Description
SProject	Not used.
SResult	Call status.
StParkTime	The timestamp when the first treatment was applied to the call.
StQueueTime	The timestamp when the call segment entered the queue.
StRoutTime	The timestamp when the first routing of the call segment began.
StTime	The start date and time of the call segment.
WtTime	The duration of time that the call segment spent in all queues before being established or abandoned.

User Data Tables

User data tables store the data that different components and tools, both Genesys and third-party, may attach to the messages coming to and from T-Server. The attached data can be associated with a call or with an individual call segment. Depending on the specific deployment environment, the structure and content of the attached data can vary widely. Because of that, Call Concentrator provides configuration options allowing you to customize the structure of the user data tables.

GDATAEX Table

The GDATAEX table can be used by Call Concentrator to store the attached data that a user would like to associate with a whole call (that is, with a record in the GCDR table). You are responsible for creating a GDATAEX structure that fits your needs. The only predefined and required field in the structure is the `ConnID` that ties a GDATAEX record to the respective record in the GCDR table. You must specify what data should be stored, and in what format.

Note: By default, Call Concentrator does not write data to the GDATAEX table. You can override this behavior by setting the `GDATAEX` configuration option to `on`. For more information about how to create the GDATAEX table and configure Call Concentrator to use it, see “GDATAEX Table” on page 61.

EVREF, EVDATA, and EVREFEX Tables

The attached data associated with the individual leg calls can be stored in two ways:

- Using the EVREF and EVDATA tables.
  - The EVREF table is an auxiliary table. It records the links between an EVDATA record and the SCDR record representing the associated call leg. Table 3 lists the fields in the EVREF table.
  - The EVDATA table stores the attached data itself. Several EVDATA records can be associated with a call leg. Table 4 on page 20 lists the fields in this table.
- Using a single EVREFEX table. The attached data is stored in this table as one record per call leg. Table 5 on page 20 lists the fields in the EVREFEX table.

Note: By default, Call Concentrator does not write data to the EVREFEX table. It uses EVREF and EVDATA instead. This behavior can be overridden by setting the `evrefex` configuration option to `on`. For more information about how to create the EVREFEX, EVREF and EVDATA tables, and to configure Call Concentrator to use them, see Chapter 5, "User Data Tables" on page 51.

Table 3: EVREF Table Fields

Field Name	Description
Agent	The agent object in the Configuration Database.
ConnID	If not zero, the field ties the EVREF record to the record in the GCDR table representing the call associated with this EVREF record.
DN	The DN object in the Configuration Database.
ESequence	An integer counter generated by Call Concentrator. The value ties the EVREF record to one or more records in the EVDATA table having the same value as the EVDATA `ESequence` field.
EventType	Not used.
SCSequence	The ordinal number of the call segment in chronological order.
Time	The start date and time of the call segment. This value is the same as that in the `StTime` field of the SCDR record associated with the EVREF record in question.

Table 4: EVDATA Table

Field Name	Description
DataType	The type of user data stored in the EVDATA record.
ESequence	An integer counter generated by Call Concentrator.
KeyName	The key name retrieved from a key-value pair that was sent by an agent from a desktop phone application.
ValChar	The text value retrieved from a key-value pair that was sent by an agent from a desktop phone application.
ValInt	The integer value retrieved from a key-value pair that was sent by an agent from a desktop phone application.

Table 5: EVREFEX Table

Field Name	Description
Agent	The agent object in the Configuration Database.
ConnID	If not zero, the field ties the EVREFEX record to the record in the GCDR table representing the call associated with this EVREFEX record.
DN	The DN object in the Configuration Database.
ESequence	An integer counter generated by Call Concentrator.
EventType	Not used.
SCSequence	The ordinal number of the call segment in chronological order.
Time	The start date and time of the call segment. This value is the same as that in the `StTime` field of the SCDR record associated with the EVREFEX record in question.

AREC Table

The AREC table stores data about the states of Agents and DNs, as shown in Table 6. These states may be related or unrelated to a call. If a call segment passes through multiple DNs, the AREC table will contain records corresponding to each one of those DNs that belong to a monitored switch and the time the call arrived and left each of those DNs.

AREC Data vs. SCDR Data

In contrast with the AREC table, the SCDR table only records the origination, final destination, first queue, and last queue the call passed through. To track down the path of the call through all queues and routing points, you must get

the information from the AREC table. However, the SCDR table provides sufficient data to analyze the final topology of the call segments.

Note: By default, Call Concentrator always writes data to the AREC table. This behavior can be overridden by setting the `arec` configuration option to `off`. For more information about how to create the AREC table, and how to configure Call Concentrator to use it, see "Customizing the AREC Table" on page 76.

For more detailed information about the AREC table, see Chapter 6, "Associated Records" on page 69.

Table 6: AREC Table Fields

Field Name	Description
Agent	The agent involved in the state.
ConnID	The field refers to the record in the GCDR table representing the call (if any) involved in the state.
ConnSwitch	The Field refers to the switch where the call involved in the state (if any) originated.
DN	The field refers to the DN involved in the state.
Duration	Duration of the state.
Login	If an agent is involved in the state, the field refers to the agent login object in the Configuration Layer.
Place	The field refers to the place (if any) the agent involved in the state is logged on. A place may include multiple DNs; however only one agent may be logged onto the place.
SCallID	The field refers to the identifier of the call segment (if any) involved in the state (see description of the SCallID field in "SCDR Table Fields" on page 39). Call Concentrator generates this identifier.
SCallSwitch	The field refers to the switch where the call segment (if any) involved in the state resided (see also the description of the LocSwitch and RmtSwitch fields in "SCDR Table Fields" on page 39).
Sequence	The value of the Field is a counter generated by Call Concentrator. It may be used as the primary key for the AREC table.
Status	The value of the field is the code of the state. The code may be a predefined codes or of a custom state. See "Customizing the AREC Table" on page 76.

Relationship Between Call Concentrator Tables

Call Concentrator tables relate to one another by the key fields that tie each record of a table to the appropriate record in another table.

- The `ESequence` field ties the attached data stored in an EVDATA record to the EVREF record containing general information about the attached data.
- The `ConnID` and `SCSequence` fields tie each related record of the EVREF table to records in the SCDR table.
- The `ConnID` field ties SCDR records to GCDR records.
- In addition, AREC records are related to the `ConnID` field in SCDR and GCDR tables for AREC records that correspond to calls. However, some AREC records are unrelated to calls and, therefore, they have no `ConnID` and do not correspond to any SCDR or GCDR record.

These table interrelationships are shown in Figure 1 on page 22.

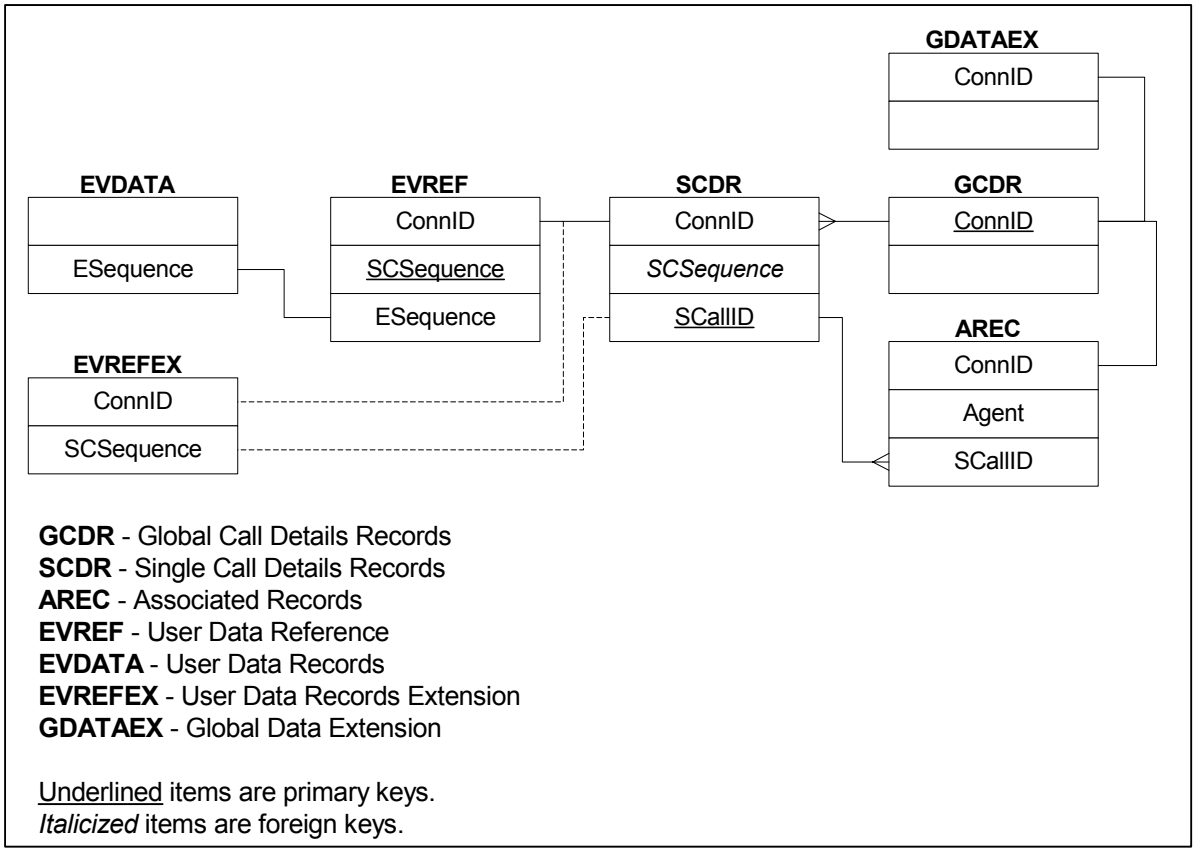

Figure 1: Call Concentrator Table Relationships

Chapter

3 Global Data Records

This chapter describes the GCDR table in detail. It includes these sections:

GCDR Overview

Once a call is completed Call Concentrator writes a record to the GCDR table. That record represents the call as a whole, while the information about individual call segments is written to the SCDR table. The GCDR table is sometimes referred to as the parent record for those SCDR records associated with it. Each GCDR record is identified by the value of its `ConnID` field. A parent GCDR record is linked to its associated SCDR records by this field, which is present in both those tables.

Note: By default, Call Concentrator always writes data to the GCDR table. This behavior can be overridden by setting the `gcdr` configuration to `off`. Refer to "Customizing Your Configuration" in the *Call Concentrator 7 Deployment Guide* for information about this configuration option.

GCDR Fields

This section describes each GCDR table field. The fields are listed in alphabetical order. Each description of a field in the GCDR table consists of four parts:

- Explanation
- Field Type

- Field Source
- Configuration

Explanation describes the meaning of the field's value, how the value depends (if it does) on a specific Call Concentrator configuration, and any other factors that may be relevant.

Field Type describes the *data type* used for the field in a database in a conceptual, platform-independent way. The actual data type of the field depends on the platform deployed and on the SQL script that has been used to create the Call Concentrator tables. For example, a field marked as `Small Integer` may be of data type `tinyint` on the Sybase platform and `smallint` on Informix. Numbers enclosed in parentheses represent the maximum number of digits or characters allowed. For example, `Varchar(20)` means that a maximum of 20 characters is allowed in the field of type variable character.

Note: These constraints on the data length are valid only if the database administrator has deployed a default SQL script provided with Call Concentrator. If the script has been modified, the actual constraint may differ.

Field Source indicates the *data source* used by Call Concentrator to populate the field with a value.In the following chapters, the following abbreviations will be used to denote the source of a field value in discussing the structure of those tables:

- **DBID**

  A unique identifier assigned by Configuration Layer to a contact center object (DN, Agent, and so on). Call Concentrator retrieves this DBID from the Configuration Database.

- **CCon**

  Object identifiers and counters generated by Call Concentrator.

- **Derived**

  A value derived from other fields of this or other tables. For a derived field, this description also includes an expression that can be used to calculate the value of the field. The expression is written using generic SQL syntax or by using a self-explanatory arithmetic notation.

- **T-Server**

  Values taken from T-Server and applied to a call or call segment (leg) without any additional processing.

- **Processed**

  Values taken from T-Server and processed according to the topology of calls and their segments or derived entirely from the topology of calls and their segments.

The *Configuration* part of a field description indicates the way the field's value depends on the Call Concentrator configuration settings. `None` means that the configuration settings do not affect the value.

GCDR Table Fields

agwttime

Explanation: The time, in seconds, between the first appearance of a call in a call center and the first answer by a logged-in agent.
Field Type: `Integer`
Field Source: Processed
Configuration: The field is present in the GCDR table only if the `newtables` option is set to `dart`, and the appropriate SQL script has been deployed to create the table.

Calls

Explanation: The number of call segments constituting the call. In other words, the number of SCDRs having the same value of the `ConnID` field as this GCDR.
Field Type: `Small Integer`
Field Source: Derived

```
Calls = SELECT COUNT(*) FROM SCDR
WHERE GCDR.ConnID = SCDR.ConnID
```

Configuration: None

CallType

Explanation: Which category the call belongs to. May be one of the following values:

Table 7: Call Types

Value	CallType
0	CALL_TYPE_UNKNOWN
1	CALL_TYPE_INTERNAL
2	CALL_TYPE_INBOUND
3	CALL_TYPE_OUTBOUND
4	CALL_TYPE_CONSULT

Field Type: `Small Integer`
Field Source: Processed
Configuration: None

ConnID

Explanation: An identification number assigned by T-Server to a call (for example, `392732680203`). If this original call is transferred or becomes part of a conference, Call Concentrator creates a new call segment (represented by a separate SCDR record having the same `ConnID` value). Multiple GCDR records with different values of the `ConnID` field, along with different SCDR sequences, may be created if Call Concentrator is unable to properly recognize the call.
Field Type: `Decimal(20)`
Field Source: Processed
Configuration: None

Customer

Explanation: A reference to a Tenant. A Tenant is a Customer Interaction Network (a call center or business) using the network services of a provider. In a single-site system, there is only one customer or Tenant ID.
Field Type: `Integer`
Field Source: DBID
Configuration: The value of the field may be affected by the `EnvironmentDBID` configuration option. For configuration option values, see the "Customizing Your Configuration" chapter in *Call Concentrator 7 Deployment Guide*.

Destination

Explanation: The value taken from the `ThirdPartyDN` attribute of the `EventRouteUsed` T-Event in cases when this value does not correspond to a DN specified in Configuration Database.
Field Type: `Varchar(30)`
Field Source: T-Server
Configuration: The field may be filled with a nonempty string value only if the `NetworkCallFlow` configuration option is set to `true`.

DestLabel

Explanation: Refers to the DN number present in the `ThirdPartyDN` attribute of the `EventRouteUsed` T-Event in case this DN has been specified in Configuration Database as belonging to a DN group.
Field Type: `Integer`
Field Source: DBID
Configuration: The field may be filled with a nonempty string value only if the `NetworkCallFlow` configuration option is set to `true`.

DN

Explanation: Refers to first DN that received or initiated the call.
Field Type: Integer
Field Source: DBID
Configuration: None

Dur_Cons

Explanation: Total duration, in seconds, of consultation call segments constituting the call.
Field Type: Integer
Field Source: Derived

```
Dur_Cons = SELECT SUM(SDuration) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Consult";
```

Configuration: None

Dur_Inb

Explanation: Total duration, in seconds, of inbound call segments constituting the call.
Field Type: Integer
Field Source: Derived

```
Dur_Inb = SELECT SUM(SDuration) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Inbound";
```

Configuration: None

Dur_Int

Explanation: Total duration, in seconds, of internal call segments constituting the call.
Field Type: Integer
Field Source: Derived

```
Dur_Int = SELECT SUM(SDuration) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Internal";
```

Configuration: None

Dur_Outb

Explanation: Total duration, in seconds, of outbound call segments constituting the call.
Field Type: Integer
Field Source: Derived

```
Dur_Outb = SELECT SUM(SDuration) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Outbound";
```

Configuration: None

Duration

Explanation: Total call time, in seconds, from initial connection to termination.

> **Note:** The value stored in this field may not be equal to the sum of `SDuration` fields in all SCDRs associated with this GCDR because of an overlap in the time. For example, a consultation call may coexist with the initial inbound call. The original party may leave the call before it terminates (if some consultation call or conference call outlives the original call), yet `Duration` is calculated from start to finish of the whole call sequence.

Field Type: `Integer`
Field Source: Processed
Configuration: None

First_CallID

Explanation: `CallID` generated by Call Concentrator for the first call segment. This is not the `Call ID` used by the switch.
Field Type: `Integer`
Field Source: Derived

```
First_CallID = SELECT SCallID FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

First_CDIG

Explanation: Reserved
Field Type: `Varchar(40)`
Field Source: None
Configuration: None

First_DNIS

Explanation: DNIS (Dialed Number Identification Service) from the first call segment.
Field Type: `Character(20)`
Field Source: Derived

```
First_DNIS = SELECT DNIS FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

First_PHONE

Explanation: A call's ANI (Automatic Number Identification service) or an outbound dialed number for the first call segment. If a T-Server message contains data in its `OtherDN` attribute that is not recognized as a known DN, the

data in the OtherDN attribute is recorded as the dialed phone number in the SCDR Phone field. For more information about T-Server messages (events) and the OtherDN attribute, refer to *Framework 7 T-Server Developer's Guide*.
Field Type: Varchar(40)
Field Source: Derived

```
First_PHONE = SELECT Phone FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

First_QUEUE

Explanation: This field is only populated if the call has entered at least one queue during its lifetime (that is, if an EventQueued message has been received for the call). The field then contains the Queue DN of the first queue that the call entered.
Field Type: Integer
Field Source: Derived

```
First_Queue = SELECT LocQueue FROM SCDR
WHERE SCSequence = SELECT MIN(SESequence) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.LocQueue > 0;
```

> **Note:** If the embedded SELECT clause returns NULL, then First_Queue is set to 0.

Configuration: None

FirstTime_Delivery

Explanation: Reserved
Field Type: Integer
Field Source: None
Configuration: None

FirstTime_Park

Explanation: The duration of the treatment (if any) applied to the first call segment.
Field Type: Integer
Field Source: Derived

```
FirstTime_Park = SELECT ParkTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

FirstTime_Queue

Explanation: The amount of time, in seconds, the first call segment spent in a queue.
Field Type: Integer
Field Source: Derived

```
FirstTime_Queue = SELECT WtTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

FirstTime_Ring

Explanation: The amount of time, in seconds, the first call segment spent ringing.
Field Type: Integer
Field Source: Derived

```
FirstTime_Ring = SELECT RingTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

FirstTime_Rout

Explanation: The amount of time, in seconds, the first call segment spent being routed from the first routing point from which it was successfully routed.
Field Type: Integer
Field Source: Derived

```
FirstTime_Rout = SELECT RoutTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

Flag_Abandoned

Explanation: Set to 1 if the call was abandoned at a physical telephony object (as opposed to an abstract DN, such as a virtual queue); otherwise set to 0 (zero).
Field Type: Small Integer
Field Source: Derived

```
Flag_Abandoned = SELECT F_aban FROM SCDR WHERE SCDR.ConnID =
ConnID AND SCDR.CallType != Consult
AND SCDR.SCSequence = SELECT MAX (SCSequence)
FROM SCDR WHERE SCDR.SCDR.ConnID = ConnID;
```

Configuration: None

HAgent

Explanation: Refers to the first agent that participated in the call.
Field Type: `Integer`
Field Source: DBID, Processed.
Configuration: None

HResult

Explanation: Not used
Field Type: `Integer`
Field Source: None
Configuration: None

MediaType

Explanation: Records any media type information attached by T-Server.
Field Type: `Integer`
Field Source: T-Server
Configuration: The field is present in the GCDR table only if the `MediaType` option is set to `on` and the appropriate SQL script has been used to create the table.

N_conf

Explanation: The number of call segments that have been in a conference.
Field Type: `Small Integer`
Field Source: Derived

```
N_conf = SELECT SUM(F_inconf) FROM SCDR
WHERE SCDR.ConnID = ConnID;
```

Configuration: None

N_Cons

Explanation: The number of consultation call segments among those constituting the call.
Field Type: Small Integer
Field Source: Derived

```
N_conf = SELECT COUNT(*) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Consult";
```

Configuration: None

N_Inb

Explanation: The number of inbound call segments among those constituting the call.
Field Type: `Small Integer`
Field Source: Derived

```
N_Inb = SELECT COUNT(*) FROM SCDR
```

```
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Inbound";
```

Configuration: None

N_Int

Explanation: The number of internal call segments among those constituting the call.
Field Type: Small Integer
Field Source: Derived

```
N_Int = SELECT COUNT(*) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Internal";
```

Configuration: None

N_Outb

Explanation: The number of outbound call segments among those constituting the call.
Field Type: Small Integer
Field Source: Derived

```
N_Outb = SELECT COUNT(*) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.CallType = "Outbound";
```

Configuration: None

N_park

Explanation: The number of call segments given a treatment.
Field Type: Small Integer
Field Source: Processed
Configuration: None

N_queue

Explanation: The number of call segments, among those constituting the call, that passed through a queue.
Field Type: Small Integer
Field Source: Derived

```
N_queue = SELECT SUM(F_queue) FROM SCDR
WHERE SCDR.ConnID = ConnID;
```

Configuration: None

N_trans

Explanation: The number of transferred call segments among those constituting the call.
Field Type: Small Integer
Field Source: Derived

```
N_Outb = SELECT SUM(F_tran) FROM SCDR
WHERE SCDR.ConnID = ConnID;
```

Configuration: None

Project

Explanation: Reserved
Field Type: Integer
Field Source: None
Configuration: None

StParkTime

Explanation: The time a treatment was applied to the first call segment (if any).
Field Type: Integer
Field Source: Derived

```
StParkTime = SELECT StParkTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

StQueueTime

Explanation: The time (in seconds) at which the first call segment entered a queue.
Field Type: Integer
Field Source: Derived

```
StQueueTime = SELECT StQueueTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

StRoutTime

Explanation: Represents the date and time of the first EventRouteRequest message, that is, the time at which the call was routed.
Field Type: Integer
Field Source: Derived

```
StRoutTime = SELECT StRoutTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

StTime

Explanation: Represents the call's start date and time.
Field Type: Integer
Field Source: Derived

```
StTime = SELECT StTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.SCSequence = 0;
```

Configuration: None

Switch

Explanation: Refers to the switch where the call was initiated or to the first switch that received the call.
Field Type: `Integer`
Field Source: Processed
Configuration: None

TimeIn_conf

Explanation: Total time, in seconds, spent by the call segments in a conference state.
Field Type: `Integer`
Field Source: Derived. This is the total duration of all call segments that have been in a conference state, excluding the overlap in time. This value cannot be calculated by means of a pure SQL expression.
Configuration: None

TimeIn_park

Explanation: Total time, in seconds, spent by the call segments in a treatment state.
Field Type: `Integer`
Field Source: Processed
Configuration: None

TimeIn_queue

Explanation: Total time, in seconds, spent by the call segments waiting in a queue.
Field Type: `Integer`
Field Source: Derived

```
TimeIn_queue = SELECT SUM(WtTime) FROM SCDR
WHERE SCDR.ConnID = ConnID
```

Configuration: None

TimeIn_trans

Explanation: Total duration, in seconds, of the call segments that have been transferred.
Field Type: `Integer`
Field Source: Derived

```
TimeIn_trans = SELECT SUM(SDuration) FROM SCDR
WHERE SCDR.ConnID = ConnID AND SCDR.F_tran = 1
```

Configuration: None

Tot_DialTime

Explanation: Total time, in seconds, spent by the call segments in the dialing state. `DialTime` includes the time a call spent in the dialing state before it was either answered or abandoned.
Field Type: `Integer`
Field Source: Derived

```
Tot_DialTime = SELECT SUM(DialTime) FROM SCDR
WHERE SCDR.ConnID = ConnID
```

Configuration: None

Tot_RingTime

Explanation: Total time, in seconds, spent by the call segments in the ringing state.
Field Type: `Integer`
Field Source: Derived

```
Tot_RingTime = SELECT SUM(RingTime) FROM SCDR
WHERE SCDR.ConnID = ConnID
```

Configuration: None

Chapter

4 Single Call Details Records

This chapter describes in detail the structure of the Single Call Details Records (SCDR) table. It includes these sections:

SCDR Overview

Call Concentrator uses the SCDR table to store information about the call segments (legs) that constitute the global calls represented in the GCDR table. Each SCDR record corresponds to such a segment and refers to the parent GCDR record via the value of the `ConnID` field that is present in both tables.

Note: By default, Call Concentrator always writes data into the SCDR table. This behavior can be overridden by setting the `scdr` configuration option to `off`.

The description of a field consists of four parts:

- Explanation
- Field Type
- Field Source
- Configuration

Explanation describes the meaning of the field's value, how the value depends (if it does) on a specific Call Concentrator configuration, and any other factors that may be relevant.

Field Type describes the *data type* used for the field in a database in a conceptual, platform-independent way. The actual data type of the field depends on the platform deployed and on the SQL script that has been used to create the Call Concentrator tables. For example, a field marked as `Small`

`Integer` may be of data type `tinyint` on the Sybase platform and `smallint` on Informix. Numbers enclosed in parentheses represent the maximum number of digits or characters allowed. For example, `Varchar(20)` means that a maximum of 20 characters is allowed in the field of type variable character.

Note: These constraints on the data length are valid only if the database administrator has deployed a default SQL script provided with Call Concentrator. If the script has been modified, the actual constraint may differ.

Field Source indicates the *data source* used by Call Concentrator to populate the field with a value.In the following chapters, the following abbreviations will be used to denote the source of a field value in discussing the structure of those tables:

- **DBID**

  A unique identifier assigned by Configuration Layer to a contact center object (DN, agent, and so on). Call Concentrator retrieves this DBID from the Configuration Database.

- **CCon**

  The object identifiers and counters generated by Call Concentrator.

- **Derived**

  A value derived from other fields of this or other tables. For a derived field, this description also includes an expression that can be used to calculate the value of the field. The expression is written using generic SQL syntax or by using a self-explanatory arithmetic notation.

- **T-Server**

  Values taken from T-Server and applied to a call or call segment (leg) without any additional processing.

- **Processed**

  Values taken from T-Server and processed according to the topology of calls and their segments or derived entirely from the topology of calls and their segments.

The *Configuration* part of a field description indicates the way the field's value depends on the Call Concentrator configuration settings. `None` means that the configuration settings do not affect the value.

SCDR Table Fields

The SCDR table includes these fields:

ConnID

Explanation: This is an identification number assigned by T-Server to a call. If this original call segment is transferred or becomes part of a conference call, Call Concentrator creates a new leg of the call (represented by a separate SCDR record having the same value as `ConnID`). Multiple GCDRs with different values of the `ConnID` field, along with different SCDR sequences, can be created in case Call Concentrator is unable to properly recognize the call as a whole.
Field Type: `Decimal(20)`
Field Source: Processed
Configuration: None

DialTime

Explanation: The total time, in seconds, from the time the call segment is first dialed until it makes a connection or is abandoned.
Field Type: `Integer`
Field Source: Processed
Configuration: None

DNIS

Explanation: The DNIS (Dialed Number Identification Service) field gives the first DNIS attribute mentioned in a T-Server message related to the call segment.
Field Type: `Varchar(20)`
Field Source: T-Server
Configuration: None

EndTime

Explanation: Date and time that the leg of the call stored in this record was terminated.
Field Type: `Integer`
Field Source: Derived

```
EndTime = StTime + SDuration
```

Configuration: Call Concentrator fills the `EndTime` field with data only if the `newtables` configuration option is set to `dart`. If the option is set to `on`, then the appropriate SQL script should be deployed to ensure the physical presence of the `EndTime` column in the SCDR table.

F_aban

Explanation: Call Concentrator sets this flag to 1 if the call was abandoned (defined as a call that has been placed in a queue and was terminated without being distributed to any destination).
Field Type: Small Integer
Field Source: Derived. See description of SResult .

```
F_aban = SResult & CALL_FLAG_ABANDONEDQUEUE
```

Configuration: None

F_conf

Explanation: Call Concentrator sets this flag to 1 if the call was conferenced.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_conf = SResult & CALL_FLAG_CONFERENCED
```

Configuration: None

F_dial

Explanation: Call Concentrator sets this flag to 1 if the call was dialed.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_dial = SResult & CALL_FLAG_DIALED
```

Configuration: None

F_estb

Explanation: Call Concentrator sets this flag to 1 if the call was established.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_estb = SResult & (CALL_FLAG_ESTABLISHED |
CALL_FLAG_AUDIO_PATH)
```

Configuration: None

F_inconf

Explanation: Call Concentrator sets this flag to 1 if the call was in conference.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_inconf = SResult & CALL_FLAG_INCONFERENCE
```

Configuration: None

F_obsrv

Explanation: Not used
Field Type: Integer
Field Source: None
Configuration: Call Concentrator fills the F_obsrv field with data only if the newtables configuration option is set to dart. If the option is set to on, then

the appropriate SQL script should be deployed to ensure the physical presence of the F_obsrv column in the SCDR table.

F_que

Explanation: Call Concentrator sets this flag to 1 if the call was in the queue.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_que = SResult & CALL_FLAG_QUEUE
```

Configuration: None

F_rels

Explanation: Call Concentrator sets this flag to 1 if the call was released.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_rels = SResult & CALL_FLAG_RELEASED
```

Configuration: None

F_retr

Explanation: Call Concentrator sets this flag to 1 if the call was retrieved.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_retr = SResult & CALL_FLAG_RETRIEVED
```

Configuration: None

F_ring

Explanation: Call Concentrator sets this flag to 1 if the call was ringing.
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_ring = SResult & CALL_FLAG_RINGED
```

Configuration: None

F_tran

Explanation: Call Concentrator sets this flag to 1 if the call type is not a consultation and the call segment was transferred (defined as any call, transferred from a local agent, a place, or an agent group to another local agent, place, or agent group, or place group).
Field Type: Small Integer
Field Source: Derived. See description of SResult.

```
F_tran = CallType != Consult & (SResult &
CALL_FLAG_TRANSFERRED | CALL_FLAG_H_TRANSFERRED)
```

Configuration: None

LocAgent

Explanation: Refers to the DBID of the local agent. The local agent is the agent (if any) that was logged on the local DN (see description of the LocDN field below).
Field Type: `Integer`
Field Source: DBID, Processed.
Configuration: None

LocDN

Explanation: Refers to the DBID of the local DN. The local DN is the first agent DN (that is, not a queue or routing point) mentioned with the `ThisDN` attribute in a T-Server message related to the call segment. For an internal call or outbound call, it is the DN of the caller. For an inbound call, it is the DN that received the call.
Field Type: `Integer`
Field Source: DBID, Processed.
Configuration: None

LocLQ

Explanation: Refers to the last local queue that the call segment passed through. The local queue is mentioned with the `ThisQueue` attribute in a T-Server message related to the call segment.
Field Type: `Integer`
Field Source: DBID,T-Server.
Configuration: Call Concentrator fills the `LocLQ` field with data only if the `newtables` configuration option is set to `dart`. If the option is set to `dart,` the appropriate SQL script should be used to ensure the physical presence of the `LocLQ` column in the SCDR table.

LocQueue

Explanation: Refers to the first local queue the call segment has passed through. The local queue is a queue mentioned with the `ThisQueue` attribute in a T-Server message related to the call segment.
Field Type: `Integer`
Field Source: DBID, T-Server.
Configuration: None

LocSwitch

Explanation: The local switch where the DN referred by the `LocDN` field is located.
Field Type: `Integer`
Field Source: DBID, T-Server.
Configuration: None

LocTrunk

Explanation: Optional number for the local trunk provided by some switches, that is, the first trunk mentioned as local trunk. Taken from T-Server messages without any processing.
Field Type: `Integer`
Field Source: T-Server
Configuration: None

ParkTime

Explanation: The duration, in seconds, of a call in a parked (or held) state. `ParkTime` exists only if the call segment has been queued or routed while a treatment was applied. Any T-Server message that ends the queuing or routing is considered the end of `ParkTime`. The message that begins `ParkTime` is `EventTreatmentApplied`. The message that ends `ParkTime` is either `EventRouteUsed` or `EventDiverted`.
Field Type: `Integer`
Field Source: Processed
Configuration: None

Phone

Explanation: The Automatic Number Identification (ANI) service value for the inbound call segment or the dialed number for the outbound call segment. Taken from the ANI attribute of a T-Server message related to the call segment (or from the `OtherDN` attribute of the message), if the value of the `OtherDN` attribute has not been recognized as a reference to a DN defined in the Configuration Database.
Field Type: `Varchar(40)`
Field Source: T-Server
Configuration: None

RingTime

Explanation: The total time, in seconds, from the time the call segment first rings. If the call is never established but the `F_ring` field is set to `1`, see the value in the `SDuration` field for the ring time.
Field Type: `Integer`
Field Source: Processed
Configuration: None

RmtAgent

Explanation: Refers to the remote agent. The remote agent is the agent logged on DN referred by the `RmtDN` field (see description of the `RmtDN` field below). For an internal call, this is called an Agent.
Field Type: `Integer`
Field Source: Processed
Configuration: None

RmtDN

Explanation: Refers to the remote DN. The remote DN is the agent DN (that is, not a queue or routing point) mentioned with the `OtherDN` attribute in a T-Server message related to the call segment. For an internal call it is called DN.
Field Type: `Integer`
Field Source: Processed
Configuration: None

RmtLQ

Explanation: Refers to the last remote queue that the call segment has passed through. The remote queue is a queue mentioned with the `OtherQueue` attribute in a T-Server message related to the call segment.
Field Type: `Integer`
Field Source: DBID, T-Server.
Configuration: Call Concentrator fills the `RmtLQ` field with data only if the `newtables` configuration option is set to `dart`. If the option is set to `on`, then the appropriate SQL script should be deployed to ensure the physical presence of the `RmtLQ` column in the SCDR table.

RmtQueue

Explanation: Refers to the first remote queue that the call segment passed through. The remote queue is a queue mentioned with the `OtherQueue` attribute in a T-Server message related to the call segment.
Field Type: `Integer`
Field Source: DBID
Configuration: None

RMTSwitch

Explanation: The remote switch where the DN referred by the `RmtDN` field is located. Currently the remote switch is the same as `LocSwitch`. Calls between switches may one day be represented as a single record with different switches on both ends.
Field Type: `Integer`
Field Source: DBID
Configuration: None

RmtTrunk

Explanation: Optional number for a remote trunk, provided by some switches. This is the first remote trunk mentioned in a T-Server message related to the call segment.
Field Type: `Integer`
Field Source: T-Server
Configuration: None

RoutTime

Explanation: The amount of time, in seconds, that the call segment spent being routed from the first routing point from which it was successfully routed.
Field Type: `Integer`
Field Source: T-Server
Configuration: None

SCallID

Explanation: Call-segment identification; a unique identifier for a call segment. This is not the Call ID used by the switch but a value generated by Call Concentrator.
Field Type: `Integer`
Field Source: CCON
Configuration: None

SCallType

Explanation: Call type. May take one of the values listed in Table 88.
Field Type: `Small Integer`
Field Source: Processed
Configuration: None

Table 8: SCallType Values

Value	SCallType
0	CALL_TYPE_UNKNOWN
1	CALL_TYPE_INTERNAL
2	CALL_TYPE_INBOUND
3	CALL_TYPE_OUTBOUND
4	CALL_TYPE_CONSULT

SCSequence

Explanation: Ordinal number of the call segment record in the sequence of SCDR records having the same `ConnID`. In other words, the ordinal number of the call leg in chronological order. The numbering starts with `0` (zero).
Field Type: `Small Integer`
Field Source: CCON
Configuration: None

SDuration

Explanation: The duration, in seconds, of the call segment from the moment it was created to the moment it was terminated.
Field Type: `Integer`
Field Source: Processed
Configuration: None

SHResult

Explanation: Not used (obsolete).
Field Type: `Integer`
Field Source: None
Configuration: None

SSwitchCallID

Explanation: Call ID assigned to the call segment by the physical switch.
Field Type: `Integer`
Field Source: None
Configuration: Call Concentrator fills the `SSwitchCallID` field with data only if the `AddSwitchCallID` configuration option is set to `on`. If the option is set to `on`, then the appropriate SQL script should be deployed to ensure the physical presence of the `SSwitchCallID` column in the SCDR table.

SProject

Explanation: Reserved for future use.
Field Type: `Integer`
Field Source: None
Configuration: None

SResult

Explanation: Call status, expressed as a bit mask. The meaning and symbolic names used to refer to individual bits are listed in Table 9.

> **Note:** This is the field from which all `F_<---->` fields (now deprecated) are derived. This field should be used instead whenever possible.

Field Type: `Integer`
Field Source: Processed
Configuration: None

Table 9: SResult Bit Mask Values

Value	Call Status
1	CALL_FLAG_DIALED
2	CALL_FLAG_RINGED
4	CALL_FLAG_ESTABLISHED
8	CALL_FLAG_TRANSFERRED
16	CALL_FLAG_CONFERENCED
32	CALL_FLAG_RETRIEVED
64	CALL_FLAG_RELEASED
128	CALL_FLAG_QUEUE
256	CALL_FLAG_ABANDONEDQUEUE
512	CALL_FLAG_INCONFERENCE
1024	CALL_FLAG_ROUTED
2048	CALL_FLAG_TREATMENT
4096	CALL_FLAG_HELD
8192	CALL_FLAG_DESTINATION_BUSY
16384	CALL_FLAG_AFTER_TRANSFER
32768	CALL_FLAG_AUDIO_PATH
65536	CALL_FLAG_OBSERVE
131072	CALL_FLAG_CONSULT_TRANSFER
262144	CALL_FLAG_H_TRANSFERED
524288	CALL_FLAG_DELETED_BY_REQUEST
1048576	CALL_FLAG_PICKUP

The bit values listed in Table 9 are cumulative. For example, when a typical inbound call begins, the integers `2` (ringing) then `4` (established) and `32768` (audio path) are stored as `32774`. When the call is finished, the integer `64` (released) is added to the `32774`. By the end of the call, the SResult field stores the integer `32838`, which is equal to `2 + 4 + 32768 + 64`.

Note: `CALL_FLAG_H_TRANSFERRED` records the bit mask `262144` in the `SResult` field when a call is released from a transfer.

Table 10: SResult Integer Values

Type of Call	Integer Typically Stored in SResult Field	Description of Integer
Inbound	32838	2 + 4 + 32768 + 64
Outbound	32837	1 + 4 + 32768 + 64
Internal	32839	1 + 2 + 4 + 32768 + 64
First call segment in Conference	36919	2 + 4 + 16 + 32 + 4096 + 32768
First call segment in Transfer	36943	2 + 4 + 8 + 64 + 4096 + 32768

The values in Table 10 are for illustration purposes only. More bits may be added in the future. Bit masks should never be compared as integers. Only after using bit-wise operations to extend bits are they relevant for comparison. When bit-wise operations are not directly available, integer expressions with powers of 2 can be used to implement them (for example, integer *n*, *n*& `4` can be calculated as *n*/`4*4` - *n*/`8*8`).

StQueueTime

Explanation: Date and time when the call segment entered the queue.
Field Type: `Integer`
Field Source: Processed
Configuration: None

StRoutTime

Explanation: Date and time when the first routing of the call segment began.
Field Type: `Integer`
Field Source: Processed
Configuration: None

StTime

Explanation: Start date and time of the call segment.
Field Type: `Integer`
Field Source: Processed
Configuration: None

WtTime

Explanation: The duration of time, in seconds, that the call segment spent in all queues before being established or abandoned.
Field Type: `Integer`
Field Source: Processed
Configuration: None

Chapter

5 User Data Tables

This chapter gives a detailed description of the tables that Call Concentrator uses to store the user attached data. The chapter includes these sections:

User Data Overview

The Genesys Framework allows you to attach custom data to telephony events coming to and from T-Server. The attached data takes the form of key-value pairs. In the key-value pair, the key acts as a name for the attached data, for example, `CallComments`; the value represents the data itself, for example, the string `The customer would like a follow-up call.`

Multiple Key-Value Pairs

You can have multiple keys for a call segment, but only one value for each key. For example, if a message arrives containing the key-value pair `Price, 50`, Call Concentrator adds the pair to the previously collected list of the key-value pairs attached to the call segment. If, in a couple of minutes, another message arrives containing the key-value pair `Price, 40`, only the most recently received value for this key, `40`, is saved in the user data list.

The most generic way to attach the custom data is provided by the T-Server API but it is usually a human agent who attaches the data using a desktop application, like Genesys Contact Navigator or an application created using the Genesys Interaction SDK. The attached data may be associated with a call that the agent participates in or it can relate to after-call activity. For more information about attached user data, refer to *Framework 7 T-Server Developer's Guide*.

Call Locking

To ensure that the data will be attached to the call after the call has already ended (post-call attached data), the agent must send the attached data from the desktop application before the time specified with the `DeleteTime` option has expired.

An alternative is to use call locking. The key-value pair with the `LockCall` key tells Call Concentrator that the call the pair is related to should be contained in the memory (or locked) during the amount of time specified by the value in the pair. This feature allows an agent desktop application to keep the call readily available while all user data related to the call will be attached to the call.

To do this, the agent (that is, the agent desktop application) should send the key-value pair:

```
"LockCall", 100
```

immediately after the agent receives the call. Later, after the call ends, the agent will have 100 seconds to attach the user data:

```
"LogAfterCall", "The customer would like a followup call"
```

and immediately after that sends another key-value pair:

```
"LockCall", "-"
```

This user data may be recorded in a properly-configured GDATAEX table as well as in the EVREFEX or/and EVDATA table. In the last two cases, the the `SCSequence` field for the database record is set to `0`, and the `EventType` field to `96`.

When `EventUserEvent` is sent with the `ConnID` of the call, `LockCall` as the key, and with a value that is an integer or a string that represents a decimal integer, the call will remain locked after its termination at least for the time (in seconds) specified by that value.

Note: You cannot use `EventAttachedDataChanged` to write data to a table using call locking. You must use `EventUserEvent`.

This means that the call record will not be written until the expiration of this timeout, and it will still be possible to attach some user data to the call. The call can be unlocked before the timeout expires by sending the user event with the `LockCall` key and the `"-"` value. Multiple locks can be set for the same call. In this case, the longest timeout applies, and the `"-"` value decreases the number of locks by one.

Warning! If the value for `DeleteTime` is longer than the delay time set for `LockCall`, all SCDR rcords for the locked call could be deleted and attached data might appear only in the GDATAEX or EVDATA tables, not the SCDR or EVREFEX tables.

For example, the system administrator of a Customer Interaction Network decides to create the `LogAfterCall` key to allow agents to store data about calls. To do this, the system administrator sets the `EventData` option to the value:

```
...,char,LogAfterCall,...
```

An agent finishes an interaction with a customer and decides to attach some data about that call for storing in the database. In this example, the agent sends the key-value pair,

```
"LogAfterCall", "The customer would like a followup call"
```

to be stored in the EVDATA table.

The resulting record in the EVDATA table may appear as shown in Table 11.

Table 11: Lock Call EVDATA Table

ESequence	DatType	KeyName	ValInt	ValChar
1589062784	1	LogAfterCall	0	The customer would like a followup call

The User Data Tables

The Call Concentrator user data tables are these:

- EVREF Table (User Data Reference)
- EVDATA Table (User Data Records)
- EVREFEX Table (User Data Reference Extension)
- GDATAEX Table (Global User Data Extension)

The next four subsections of this chapter describe the structure of these tables explaining the purpose of each table and its fields. Field descriptions consist of four parts:

- Explanation
- Field Type
- Field Source
- Configuration

Explanation describes the meaning of the field's value, how the value depends (if it does) on a specific Call Concentrator configuration, and any other factors that may be relevant.

Field Type describes the *data type* used for the field in a database in a conceptual, platform-independent way. The actual data type of the field depends on the platform deployed and on the SQL script that has been used to create the Call Concentrator tables. For example, a field marked as `Small Integer` may be of data type `tinyint` on the Sybase platform and `smallint`

on Informix. Numbers enclosed in parentheses represent the maximum number of digits or characters allowed. For example, `Varchar(20)` means that a maximum of 20 characters is allowed in the field of type variable character.

Note: These constraints on the data length are valid only if the database administrator has deployed a default SQL script provided with Call Concentrator. If the script has been modified, the actual constraint may differ.

Field Source indicates the *data source* used by Call Concentrator to populate the field with a value.In the following chapters, the following abbreviations will be used to denote the source of a field value in discussing the structure of those tables:

- **DBID**

  A unique identifier assigned by Configuration Layer to a contact center object (DN, agent, and so on). Call Concentrator retrieves this DBID from the Configuration Database.

- **CCon**

  The object identifiers and counters generated by Call Concentrator.

- **Derived**

  A value derived from other fields of this or other tables. For a derived field, this description also includes an expression that can be used to calculate the value of the field. The expression is written using generic SQL syntax or by using a self-explanatory arithmetic notation.

- **T-Server**

  Values taken from T-Server and applied to a call or call segment (leg) without any additional processing.

- **Processed**

  Values taken from T-Server and processed according to the topology of calls and their segments or derived entirely from the topology of calls and their segments.

The *Configuration* part of a field description indicates the way the field's value depends on the Call Concentrator configuration settings. `None` means that the configuration settings do not affect the value.

EVREF Table

EVREF table records are of two kinds: those linked to a call segment and those linked to a call as whole. These two kinds can be distinguished by the values of the `EventType` field.

EVREF Records for Call Segments

In a record linked to a call segment, the `EventType` field value is `0` (zero). The EVREF record is linked to the SCDR record representing the call segment

through the values of the ConnID and SCSequence fields, which are present in both the EVREF and SCDR tables.

This type of EVREF record is simply a reference point that connects all the user data for the call segment, which may be stored in multiple records in the EVDATA table. The EVDATA records, representing the user data attached to the call segment, are linked to the parent EVREF record through the value of the ESequence field, which is present in both tables, EVREF and EVDATA.

EVREF Records for Calls

In an EVREF record for an entire call, the EventType field has a non-zero value. To be exact, the value is 93 (an arbitrarily-assigned figure) and the value of the SCSequence field is 0 (zero). This type of record collects the user data delivered with an EventUserEvent that has been sent to a locked call after all segments of the call are completed. For details on call locking, see "Call Locking" on page 52.

EVREF and EVDATA Table Relationships

Call Concentrator stores the user data list attached to a call segment, or to an EventUserEvent, as a single record in the EVREF table and several records in the EVDATA table. Each EVDATA record corresponds to a single key-value pair. The key is stored in the KeyName field, and the value in the ValInt or ValChar field.

Note: The ConnID and SCSequence fields of the EVREF table refer to the record in the SCDR table that represents the call segment.

For more information about the way Call Concentrator fills the EVREF and EVDATA tables, refer to "Customizing User Data Tables" on page 62.

Below is an alphabetical list of EVREF fields and their descriptions.

EVREF Table Fields

Agent

Explanation: Refers to an agent object in the Configuration Database.
Field Type: Integer
Field Source: Derived

```
If condition DN equals SELECT LocDN FROM SCDR
   WHERE SCDR.ConnID = ConnID AND
   SCDR.SCDR.SCSequence = SCSequence
holds for the DN field, then
   Agent = SELECT LocAgent FROM SCDR
   WHERE SCDR.ConnID = ConnID AND
   SCDR.SCDR.SCSequence = SCSequence
Otherwise,
   Agent = SELECT RmtDN FROM SCDR
   WHERE SCDR.ConnID = ConnID AND
```

```
   SCDR.SCDR.SCSequence = SCSequence;
```

Configuration: None

ConnID

Explanation: If not zero, the field ties an EVREF record to the record in the GCDR table representing the call this EVREF record is associated with. For more information about `ConnID` values, see Chapter 3, "Global Data Records" on page 23.

Field Type: `Decimal(20)`

Field Source: T-Server

Configuration: None

DN

Explanation: Refers to a DN object in the Configuration Database to which DNs this EVREF record is associated.

Field Type: `Integer`

Field Source: Derived

```
Let DN1 = SELECT LocDN FROM SCDR
   WHERE SCDR.ConnID = ConnID AND
   SCDR.SCDR.SCSequence = SCSequence
If DN1 is not zero, then DN = DN1.
Otherwise DN = SELECT RmtDN FROM SCDR
   WHERE SCDR.ConnID = ConnID AND
   SCDR.SCDR.SCSequence = SCSequence
```

Configuration: None

ESequence

Explanation: An integer counter generated by Call Concentrator tying the EVREF record to one or more records in the EVDATA table having the same value as the EVDATA. `ESequence` field (see "EVDATA Table" on page 57).

Field Type: `Decimal(10)`

Field Source: CCON

Configuration: None

EventType

Explanation: The value of the field distinguishes the EVREF records linked to a call segment from those that are linked to the call as a whole (the GCDR record). See "EVREF Table" on page 54 for more on how this field is used.

Field Type: `Small Integer`

Field Source: None

Configuration: None

SCSequence

Explanation: The ordinal number of the call segment in chronological order. It is the same as the ordinal number of the SCDR representing this call segment,

in the sequence of the SCDR records having the same value ConnID as the GCDR record representing the whole call. The numbering starts with 0 (zero). The pair (ConnID, SCSequence) ties the EVREF record to a record in the SCDR table representing the call leg the EVREF record is associated with.
Field Type: Small Integer
Field Source: CCON
Configuration: None

Time

Explanation: For the EVREF record that is associated with an SCDR record, the field contains the start date and time of the call segment. The value is the same as in the StTime field of the SCDR record associated with the EVREF record in question. For the EVREF record produced by an EventUserEvent, this field records the time that the event arrived. See "Time Measurement" on page 12 for information about the time measurements used.
Field Type: Integer
Field Sources:
For the EVREF record associated with a SCDR record—Derived

```
Time = SELECT StTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND
SCDR.SCSequence = SCSequence
```

For the EVREF record produced by an EventUserEvent—T-Server
Configuration: None

EVDATA Table

Each record in the EVDATA table contains a key-value pair that is attached to a call segment. The ESequence field in each record refers to the parent record for that call segment in the EVREF table. For more information, see "EVREF Table" on page 54.

Configuring the EVDATA Table Options

Two configuration options, EventData and DataFilter, affect the way Call Concentrator populates the EVDATA table. In the absence of the EventData option or if its value is set to off, Call Concentrator records all attached key-value pairs into the EVDATA table. But if the EventData option is set to on and the DataFilter option is absent or is set to on, then Call Concentrator records only those key-value pairs that correspond to the keys listed in the EventData option into the EVDATA table.

The fields of the EVDATA table are listed in alphabetical order and described.

EVDATA Table Fields

DataType

Explanation: The type of user data stored in the EVDATA record, that is, whether the data is stored as an integer or as text. Possible values for this field are `0` and `1`. The value `0` (zero) means that the data is an integer and should be stored in the `ValInt` field. The value `1` means that the data is textual and should be stored in the `ValChar` field.
Field Type: `Small Integer`
Field Source: Processed
Configuration: None

ESequence

Explanation: An integer counter generated by Call Concentrator. The value of the `ESequence` field ties the EVDATA record to the record in the EVREF table having the same value as the EVREF `ESequence` field.
Field Type: `Decimal(10)`
Field Source: CCON
Configuration: None

KeyName

Explanation: The key name retrieved from a key-value pair that was sent by an agent from a desktop phone application.
Field Type: `VarChar(255)`
Field Source: Processed
Configuration: None

ValInt

Explanation: If the `DataType` field value equals `0`, specifying the integer data, this field will contain the integer value retrieved from a key-value pair that was sent by an agent from a desktop phone application.
Field Type: `Integer`
Field Source: T-Server
Configuration: Call Concentrator stores a user data value in this field only if the user data key (that is stored in the `KeyName` field) is specified with the `EventData` option as having the `integer` data type.

ValChar

Explanation: If the `DataType` field value equals `1`, specifying the textual data, this field will contain the character value retrieved from a key-value pair that was sent by an agent from a desktop phone application.
Field Type: `VarChar(255)`
Field Source: T-Server
Configuration: Call Concentrator stores a user data value in this field only if the user data key (that is stored in the `KeyName` field) is specified with the

EventData option as having the textual data type, or if the data type of the key has not been specified at all.

EVREFEX Table

The EVREFEX table stores attached user data as a single record for each call segment instead of splitting the data into several records separately stored in two tables, EVREF and EVDATA. The structure of the EVREFEX table is essentially the same as the structure of the EVDATA table except that the EVREFEX table contains one or more custom fields.

Just as in the EVREF table, the records in EVREFEX table are of two kinds: those associated with a call segment (that is, with an SCDR record) and those associated with with the call as whole.

EVREFEX Records for Call Segments

In records associated with call segments, the value of the EventType field is 0 (zero). The record is linked to the associated SCDR record through the values of the ConnID and SCSequence fields, which are present in both the EVREFEX and SCDR tables.

EVREFEX Records for Calls

In an EVREFEX record for an entire call, the EventType field has a non-zero value. To be exact, the value is 93 (an arbitrarily-assigned figure) and the value of the SCSequence field is 0 (zero). This record type collects the user data delivered with an EventUserEvent that has been sent to a locked call after all segments of the call are completed. For information on call locking, see “Call Locking” on page 52.

Using the EVREFEX Table

To use the EVREFEX table, set the evrefex configuration option to on and follow the procedure described in “Customizing User Data Tables” on page 62.

The EVREFEX field names are listed in alphabetical order and described below.

EVREFEX Table Fields

Agent

Explanation: The field refers to an Agent object in the Configuration Database. It is the Agent object the EVREFEX record is associated with.
Field Type: Integer
Field Source: Derived (see the description of the Agent field in the EVREF table)
Configuration: None

ConnID

Explanation: If not zero, the field ties the EVREFEX record to the record in the GCDR table representing the call this EVREFEX record is associated with.

For more information about ConnID values, see Chapter 3, "Global Data Records" on page 23.
Field Type: Decimal(20)
Field Source: T-Server
Configuration: None

DN

Explanation: The value of the field refers to a DN object in the Configuration Database. It is the DN this EVREFEX record is associated with.
Field Type: Integer
Field Source: Derived (see the description of the Agent field in the EVREF table).
Configuration: None

ESequence

Explanation: The value of the field is an integer counter generated by Call Concentrator.
Field Type: Decimal(10)
Field Source: CCON
Configuration: None

EventType

Explanation: The value of the field distinguishes the EVREFEX records linked to a call segment from those that are linked to the call as a whole (the GCDR record). See "EVREFEX Table" on page 59 for more on how this field is used.
Field Type: Small Integer
Field Source: None
Configuration: None

SCSequence

Explanation: The call segment ordinal number is in chronological order. It is the same as the ordinal number of the SCDR representing this call segment, in the sequence of SCDR records having the same value ConnID as the GCDR record representing the whole call. The numbering starts with 0 (zero). The pair (ConnID, SCSequence) ties the EVREFEX record to a record in the SCDR table representing the call leg the EVREFEX record is associated with.
Field Type: Small Integer
Field Source: CCON
Configuration: None

Time

Explanation: For the EVREFEX record that is associated with an SCDR record, the field contains the start date and time of the call segment. The value is the same as in the StTime field of the SCDR record associated with the EVREFEX record

in question. For the EVREFEX record produced by an `EventUserEvent`, this field records the time that the event arrived. See "Time Measurement" on page 12 for information about the time measurements used.

Field Type: `Integer`

Field Sources:

For the EVREFEX record associated with a SCDR record—Derived

```
Time = SELECT StTime FROM SCDR
WHERE SCDR.ConnID = ConnID AND
SCDR.SCSequence = SCSequence
```

For the EVREFEX record produced by an `EventUserEvent`—T-Server

Configuration: None

GDATAEX Table

The GDATAEX table can be used to store attached user data that you would like to associate with the call as a whole instead of with a call segment. This capability may simplify reporting when a call has undergone multiple transfers and each leg of the call has user data attached to it. Instead of dealing with several records in the EVREFEX table (or EVREF and EVDATA tables), you can use a single set of user data collected into a GDATAEX record.

Note: If a call is transferred and agents attach data at each segment of the call, only the last value for each key is stored in the GDATAEX table.

GDATAEX provides a way to store the last attached data associated with a call and link that attached data with the GCDR record for that call. You must specify the fields you want to use. To associate the GDATAEX record with the correct GCDR record, you must create the `ConnID` field when you create the GDATAEX table. This `ConnID` field is the only predefined and mandatory field in the GDATAEX table.

To use the GDATAEX table, set the `GDATAEX` configuration option to `on` and follow the procedure described in "Customizing User Data Tables" on page 62.

Customizing User Data Tables

To customize the EVREFEX or GDATAEX tables:

1. Specify the custom fields for the table.
2. Create, or re-create the table with an appropriate structure.

The next section explains these steps in detail.

Specifying Custom Fields

To customize a user data table to your needs, first specify the custom fields that are to be added to the table. You can do that by using the `EventData` configuration option for the EVREFEX table or by using the `GlobalData` configuration option for the GDATAEX table.

Configuring the EventData Option

The value of the `EventData` option should be set in the form of a comma-separated list and must not contain blanks:

```
type1,key1,type2,key2,...,typeN,keyN
```

where `key1,key2,...,keyN` are the names of user data keys, and `type1,type2,...,typeN` specify the data types of the respective user data values. That is, `type1` specifies the data type for `key1` values, `type2` specifies the data type for `key2` values, and so on.

The key name is arbitrary, except that it must not contain blanks and must be unique among other key names.

The type specification may be given as `char(length)`, specifying the value of the key as having textual data type, or as `int`, specifying the data type as integer. The `char(length)` setting enables you to specify how many characters Call Concentrator should record.

Note: You also may omit the `length` specification, and specify the data type just as `char`. In this case, Call Concentrator records up to the default amount of data, which is 255 characters.

Below is an example of correctly specified `EventData` value:

```
char,CustomerName,char(125),ServiceType,int,ChargeCode,int,
Amount
```

Using Multiple EventData Options

In this example, the value of the `EventData` option is rather short. In a real environment, the value may be much longer. This sometimes raises a problem because of restrictions on the length of the option value that can be set with

older versions of Configuration Manager. If the required value of the `EventData` option is too long, you may set it by using several options named as `EventData, EventData_1, EventData_2,` and so on. The resulting value used by Call Concentrator will be the concatenation of the values that have been set by the `EventData, EventData_1, EventData_2,` ... options (in this order).

For Call Concentrator to read these `EventData` options correctly, you must construct them so that the values, when concatenated, form a syntactically valid list. For example, if you take the `EventData` value given above:

```
char,CustomerName,char(125),ServiceType,int,ChargeCode,int,
Amount
```

and break it apart, the result is:

```
EventData=char,CustomerName,
EventData_1=char(125),ServiceType,
EventData_2=int,ChargeCode,
EventData_3=int,Amount
```

Notice that all segments except the last end with a comma. If you omit the commas, Call Concentrator reads the resulting value as:

```
char,CustomerNamechar(125),ServiceTypeint,ChargeCodeint, Amount
```

which is invalid.

Configuring the GlobalData Option

The `GlobalData` option is configured in the same way as the `EventData` option described in the previous section. Just substitute `GlobalData` for `EventData` when specifying option parameters.

Creating Customized User Data Tables

After specifying custom fields for a user data table, create the table itself and ensure that the structure of the table fits the custom field specifications. The procedure is essentially the same for both the EVREFEX and GDATAEX tables.

If the specifications given with the `EventData` (or the `GlobalData`) option has the form:

```
type1,key1,type2,key2,...,typeN,keyN
```

then create an EVREFEX (or GDATAEX) table that, in addition to the predefined fields described in the previous sections, also contains *N* custom fields designed to hold the user data values supplied with the respective keys.

These additional fields should follow the predefined ones. The first custom field is designed to hold the values supplied with `key1`, the second is designed to hold the values supplied with `key2,` and so on.

Note: The name of a custom field may differ from the name of the key it corresponds to; however, the data type of the field should comply with the data type specified for the values supplied with this key.

If the data type for a key is specified as `int`, then the data type of the field that corresponds to the key may be any data type that does not cause the SQL server to report an error while processing SQL statements such as this:

```
INSERT INTO EVREFEX(...,FNAME, ...) VALUES (...,1,...);
```

where `FNAME` stands for the field's name and `1` is put in the position corresponding to the field. Also, the data type should be chosen in such a way that any value supplied with the key fits in the field. For example, for Oracle, the data type may be chosen as `char(10)`, but cannot be chosen as `smallint` if the value supplied with the key may be too big.

The type specification may be given as `char(length)`, specifying the value of the key as having textual data type, or as `int`, specifying the data type as integer. The `char(length)` setting enables you to specify how many characters Call Concentrator should record.

Note: You also may omit the `length` specification, and specify the data type just as `char`. In this case, Call Concentrator records the default amount of data, which is 255 characters.

For example, if the specification of the custom fields set with the `GlobalData` option has the form:

```
char,CustomerName,int,ChargeCode,int,Amount
```

then, in Oracle, the custom fields may be defined as:

```
Customer character(32)
Code smallint
Charge int
```

When defining the custom fields, the only requirement is that a field's data type be consistent with that defined in the `GlobalData` option. For a textual field, it means also that the length of the field in the database should not be *less* than that defined in the `GlobalData` option. For example, the `Customer` field in the above example could be also defined as `character(N)` where *N* is any integer number greater than 32.

How to Create a Custom Table

The recommended way to create a customized user data table containing additional custom fields is by modifying one of the SQL scripts provided on the Call Concentrator Installation CD. For example, in the `makecdr_oracle_new.sql` script, the GDATAEX table is created by means of the following SQL statements:

```
DROP TABLE GDATAEX;
CREATE TABLE GDATAEX (
ConnID DECIMAL(20),
key3 VARCHAR(30),
key4 INT
;
```

This part of the script may be edited and transformed into:

```
DROP TABLE GDATAEX;
CREATE TABLE GDATAEX(
ConnID DECIMAL(20),
CUSTOMER CHARACTER(32),
CODE SMALLINT,
CHARGE INT
;
```

For the EVREFEX table these statements may appear as:

```
DROP TABLE EVREFEX;
CREATE TABLE EVREFEX(
ESequence INT,
ConnID DECIMAL(20),
SCSequence INT,
EventType INT,
Time INT,
DN INT,
Agent INT,
CUSTOMER CHARACTER(32),
CODE SMALLINT,
CHARGE INT
;
```

Migration to the EVREFEX table

If user data is already stored in existing EVREF and EVDATA tables, you can migrate data to the EVREFEX table format via one of the migration scripts provided with Call Concentrator 7. The scripts are named:

- `migrate_to_evrefex_oracle.sql`
- `migrate_to_evrefex_mssql.sql`
- `migrate_to_evrefex_sybase.sql`

Note: Call Concentrator does not support migration for existing DB2 and Informix databases.

Migration for Oracle

The `migrate_to_evrefex_oracle.sql` script creates a new EVREFEX table and replaces any existing EVREFEX table. The scripts take existing data in the EVREF and EVDATA tables, and during conversion, will combine these into a single table called EVREFEX. If the EVREFEX table already exists, the table will be replaced.

Warning! Before running an EVREFEX migration script, back up all your data.

The migration script first scans the key names contained in the `KeyName` column of your existing EVDATA table. Then it prefixes each of those keys with the `KEY_` string and creates the EVREFEX table using these prefixed key names as the names for the custom fields in the EVREFEX table. The `KEY` prefix is designed to avoid conflict with the names of the predefined fields of the EVREFEX table, like `ConnID`, `ESequence`, and so on.

The migration script then rescans the EVREF and EVDATA tables and joins them into a single EVREFEX table.

After you complete the migration of the tables:

1. Set the `evrefex` configuration option to `on`.
2. Set the `EventData` configuration option to specify the custom fields just created. For example, the `EventData` option can be set to the value:

   `char,Customer,int,ChargeCode,int,Amount`

Migration for MS SQL Server and Sybase

In the case of MS SQL and Sybase, the migration to the EVREFEX table is two-step process. The migration script (`migrate_to_evrefex_mssql.sql` or `migrate_to_evrefex_sybase.sql`, respectively) acts in essentially the same way as the Oracle script described in the preceding sub-section. However, instead of populating the EVREFEX table it creates a new SQL script that should be run separately afterwards.

If you are using MS SQL, use this command for the first script:

```
isql -U username -P password -i migrate_to_evrefex_mssql.sql -w
500 -n -o create_EVREFEX.sql
```

Then use this command for the second script:

```
isql -U username -P password -i create_evrerefx.sql
```

Note: You can use any name you prefer for the resulting script, instead of `create_everefx.sql`.

If you are using Sybase, use this command for the first script:

```
isql -U username -P password -i migrate_to_evrefex_sybase.sql -w
500 -o create_EVREFEX.sql
```

Then use this command for the second script:

```
isql -U username -P password -i create_evrerefx.sql
```

Migration Example

To illustrate the way the migration works, assume you start with the EVREF and EVDATA tables shown in Table 12 and Table 13. The migration script merges the tables into a single EVREFEX table, shown in Table 14.

Table 12: Existing EVREF Table

ESequence	ConnID	SCSequence	EventType	Time	Agent
E1	C1	S1	0	T1	A1
E2	C2	S2	0	T2	A2
E3	C3	S3	0	T3	A3

Table 13: Existing EVDATA Table

ESequence	KeyName	DataType	ValInt	ValChar
E1	Customer	1	0	Adam Smith
E1	ChargeCode	0	11	
E1	Amount	0	50	
E2	Customer	1	0	John Smith
E2	ChargeCode	0	7	
E2	Amount	0	100	
E3	Customer	1	0	Chris Lee
E3	ChargeCode	0	13	
E3	Amount	0	200	

Table 14: EVREFEX Table Following Migration

ESequence	ConnID	SCSequence	EventType	Time	Agent	Key_Customer	Key_Charge Code	Key_ Amount
E1	C1	S1	0	T1	A1	Adam Smith	11	50

Table 14: EVREFEX Table Following Migration (Continued)

ESequence	ConnlD	SCSequence	EventType	Time	Agent	Key_Customer	Key_Charge Code	Key_ Amount
E2	C2	S2	0	T2	A2	John Smith	7	100
E3	C3	S3	0	T3	A3	Chris Lee	13	200

Chapter

6 Associated Records

This chapter describes the Associated Records (AREC) table, which collects information about DN states. This chapter includes these sections:

AREC Table Overview

A record in the AREC table corresponds to a single state of a single DN. The next section gives the nomenclature of the DN states used by Call Concentrator and the following section describes the AREC table structure. As with user data tables, you can customize the structure to fit the needs of your specific deployment environment. The last section in this chapter describes the customization procedure.

Call Concentrator enables you to filter statuses recorded into the AREC table so that only custom statuses and/or the held statuses of an agent are recorded, as defined by the values of the `AgentStatuses`, `CustomAgentStatusesOnly`, and `HeldAgentStatusesOnly` configuration options.

Note: For detailed information on these configuration options, see the "Customize Your Configuration" chapter in the *Call Concentrator 7 Deployment Guide*.

DN States

Call Concentrator uses states to describe the dynamics of DN activity. A state is characterized by its state code, its duration, and by the objects, such as a call and an agent, that were involved in the state.

The *state code* specifies the meaning of the state. Table 15 shows the list of state codes used by Call Concentrator, what each code means, and what kind of DN generates it.

The *duration* of a state is measured by the number of seconds that elapse from the moment Call Concentrator receives a telephony event signaling that the DN went into a state until the moment Call Concentrator receives another event signaling that the DN went into another state.

Note: A DN state may or may not be associated with a call. If it is associated with a call, then the record in the AREC table has a reference to the record in the GCDR table representing the call.

Besides the predefined states listed in Table 15, you can specify custom DN states that Call Concentrator recognizes and records. For more information on the subject, see the "Customizing the AREC Table" section on page 76.

Table 15: DN States

State Code	State Denotation	Any DN	Agent DN Only	Queue
1	LOGIN Indicates that an agent has been logged in the agent DN. The state ends at the moment when the agent logs out.		X	
2	READY Indicates that the agent is ready to receive new calls. To be in this state, the agent should also be in the LOGIN state.		X	
3	BUSY Indicates the Do-Not-Disturb feature has been turned on.	X		
4	OFFHOOK Indicates that T-Server has reported an OffHook event.	X		

Table 15: DN States (Continued)

State Code	State Denotation	Any DN	Agent DN Only	Queue
5	ONCALL Indicates that a call or a call leg is in progress on a DN. • The sum of all duration times associated with an agent's ONCALL status for a call can be greater than the actual total time the agent spent talking. For example, consult calls can result in overlapping ONCALL states for single agent.	X		
6	MAIL_LOGIN Indicates that an agent is logged into her/his mail-box.		X	
7	FORWARD Indicates that the Forwarding feature has been turned on for the DN involved in the state.	X		
8	ONHOLD Indicates that a call was abandoned or released while on hold. If the call is retrieved, Call Concentrator changes this state to ONHOLD_RETRIEVED before recording to the AREC table.	X		
9	VIRTUAL_QUEUE Indicates that a call is in a virtual queue. It usually is not recorded, being replaced by VIRTUAL_QUEUE_POSITIVE or by VIRTUAL_QUEUE_NEGATIVE when the call leaves the queue.			X
10	VIRTUAL_QUEUE_POSITIVE Indicates that a call left a virtual queue to be diverted to another telephony object.			X
11	VIRTUAL_QUEUE_NEGATIVE Indicates that a call left a virtual queue to be abandoned.			X
100	WAITNEXTCALL Indicates the agent is waiting for the next call. To be in the state, the agent also should be in states LOGIN and READY.		X	

Table 15: DN States (Continued)

State Code	State Denotation	Any DN	Agent DN Only	Queue
101	ONANYCALL Indicates that an agent is talking. The sum of all duration times associated with this state for an agent on a call is the actual total time the agent talked on a call. This state is deprecated.	X		
102	QUEUE_DIVERTED Indicates that the call leg was queued on an ACD queue or a route point and then diverted or routed.			X
103	QUEUE_ABANDONED Indicates that the call leg was queued on an ACD queue or a route point and then abandoned while still in the queue.			X
104	ONHOLD_RETRIEVED Indicates that the call leg was placed on hold and then retrieved.	X		
110	QUEUE_TRANSFERRED Indicates that the call leg was queued on an ACD queue or a route point and then transferred.			X
111	VIRTUAL_QUEUE_TRANSFERRED Indicates that the call leg was queued on a virtual queue or a route point and then transferred.			X
Greater than 200	Reserved for custom fields specified as described in the section “Customizing the AREC Table” on page 76			

AREC Table Fields

This section explains the structure of the AREC table. The AREC fields are listed alphabetically below. The description of each field consists of four parts:

- Explanation
- Field Type
- Field Source
- Configuration

Explanation describes the meaning of the field's value, how the value depends (if it does) on a specific Call Concentrator configuration, and any other factors that may be relevant.

Field Type describes the *data type* used for the field in a database in a conceptual, platform-independent way. The actual data type of the field depends on the platform deployed and on the SQL script that has been used to create the Call Concentrator tables. For example, a field marked as `Small Integer` may be of data type `tinyint` on the Sybase platform and `smallint` on Informix. Numbers enclosed in parentheses represent the maximum number of digits or characters allowed. For example, `Varchar(20)` means that a maximum of 20 characters is allowed in the field of type variable character.

Note: These constraints on the data length are valid only if the database administrator has deployed a default SQL script provided with Call Concentrator. If the script has been modified, the actual constraint may differ.

Field Source indicates the *data source* used by Call Concentrator to populate the field with a value.In the following chapters, the following abbreviations will be used to denote the source of a field value in discussing the structure of those tables:

- **DBID**

  A unique identifier assigned by Configuration Layer to a contact center object (DN, agent, and so on). Call Concentrator retrieves this DBID from the Configuration Database.

- **CCon**

  The object identifiers and counters generated by Call Concentrator.

- **Derived**

  A value derived from other fields of this or other tables. For a derived field, this description also includes an expression that can be used to calculate the value of the field. The expression is written using generic SQL syntax or by using a self-explanatory arithmetic notation.

- **T-Server**

  Values taken from T-Server and applied to a call or call segment (leg) without any additional processing.

- **Processed**

  Values taken from T-Server and processed according to the topology of calls and their segments or derived entirely from the topology of calls and their segments.

The *Configuration* part of a field description indicates the way the field's value depends on the Call Concentrator configuration settings. `None` means that the configuration settings do not affect the value.

AREC Table Fields

Agent

Explanation: The agent, if any, involved in the state.
Field Type: `Integer`
Field Source: DBID
Configuration: None

ConnID

Explanation: The record in the GCDR table representing the call (if any) involved in the state.
Field Type: `Decimal(20)`
Field Source: T-Server
Configuration: None

ConnSwitch

Explanation: The switch where the call involved in the state (if any) originated.
Field Type: `Integer`
Field Source: DBID
Configuration: None

DN

Explanation: The DN involved in the state.
Field Type: `Integer`
Field Source: DBID
Configuration: None

Duration

Explanation: This duration, in seconds, of the state.
Field Type: `Integer`
Field Source: Processed
Configuration: None

Login

Explanation: The agent login object in the Configuration Database if an agent is involved in the state.
Field Type: `Character(21)`
Field Source: DBID
Configuration: None

Place

Explanation: The place (if any) the agent involved in the state is logged on. A place may include multiple DNs, however, only one agent may be logged on to the place.
Field Type: `Integer`
Field Source: DBID
Configuration: None

SCallID

Explanation: The identifier of the call segment (if any) involved in the state (see description of the `SCallID` field in Chapter 4, "Single Call Details Records" on page 37). Call Concentrator generates this call segment identifier.
Field Type: `Integer`
Field Source: CCON
Configuration: None

SCallSwitch

Explanation: The switch where the call leg (if any) involved in the state resided (see also the description of the `LocSwitch` and `RmtSwitch` fields in Chapter 4, "Single Call Details Records" on page 37).
Field Type: `Integer`
Field Source: Processed
Configuration: None

Sequence

Explanation: The value of the field is a counter generated by Call Concentrator. It may be used as the primary key for the AREC table.
Field Type: `Decimal(10)`
Field Source: CCON
Configuration: None

Status

Explanation: The code of the state which may be one of the predefined codes listed in Table 15, or of a custom state (for more information about custom states, see the section "Customizing the AREC Table" on page 76).
Field Type: `Integer`
Field Source: Processed
Configuration: None

StTime

Explanation: Start date and time of the state. See "Time Measurement" on page 12 for information about time measurements used.
Field Type: `Integer`
Field Source: Processed
Configuration: None

Customizing the AREC Table

You can define custom agent state type and custom agent fields by adding the `AgentRecordUserTypes` and `AgentUserFields` configuration options to the Call Concentrator application in Configuration Manager.

Custom agent fields can have any name not already used as a field name. After you define the custom agent fields in Configuration Manager, your database administrator must add the fields to the AREC table.

AgentRecordUserTypes

The `AgentRecordUserTypes` configuration option is used to define a custom agent state. The custom agent state can be started and finished from the application that sends the data. After the state is started and before it is finished, data can be sent in user events to be stored in the custom fields corresponding to the state.

To add custom agent states, the system administrator defines the states in Configuration Manager using the `AgentRecordUserTypes` option. The format is:

```
AgentRecordUserTypes <state code>,<key>
```

For example:

```
AgentRecordUserTypes 207,AfterCallWork
```

Multiple states can be listed, separated by a comma:

```
AgentRecordUserTypes 207,AfterCallWork,208,Break
```

The state code must be a number greater than `199` and the key is the name of the custom state that the application will send as the key in the user event.

AgentUserFields

The `AgentUserFields` configuration option is used to define the custom agent record (AREC) fields to store data sent while DN was in a custom agent state. The corresponding custom fields must be added to the AREC table in order to operate properly. :

To add custom agent fields:

1. The system administrator defines the field in Configuration Manager.

The format is:

```
AgentUserFields <type>,<custom field name>,<key>
```

- The *type* specification may be given as `char(length)`, specifying the value of the key as having textual data type, or as `int`, specifying the data type as integer. The `char(length)` setting enables you to specify how many characters Call Concentrator should record.

Note: You also may omit the `length` specification, and specify the data type just as `char`. In this case, Call Concentrator records the default amount of data, which is 255 characters.

- The *custom field name* is the name of the custom field in the AREC table.
- The *key* is the key used by the application.

For example:

```
AgentUserFields char(length),LogAfterCall,Comment
```

You can define multiple fields separated by commas:

```
AgentUserFields
char(length),LogAfterCall,Comment,char,LogData,AnotherComment
```

Note: The order of the fields in the `AgentUserFields` table must be the same as the order of the fields in the AREC table.

2. The database administrator adds the custom fields to the AREC creation script and re-creates the table.

Warning! Running the table creation script shipped with Call Concentrator deletes all existing data in the tables mentioned in the script.

Sample Custom Script

For the Sybase database, you can edit the AREC section of the `makecdr_sybase.sql` file as follows:

```
DROP TABLE AREC;
CREATE TABLE AREC(
Sequence INT,
Status INT,
StTime INT,
Duration INT,
DN INT,
Place INT,
Login CHARACTER(21),
Agent INT,
SCallID INT,
SCallSwitch INT,
ConnID INT,
ConnSwitch INT,
LogAfterCall CHARACTER <=== added custom field
```

```
;
```

Using Custom States

Starting the Custom State

To start recording a custom state, the application should send the custom state name as the key and a plus sign as the value:

```
"AfterCallWork" , "+"
```

where `AfterCallWork` is an example of a custom state name.

Sending Custom State Data

To send the data to be stored in a custom agent field, the application should send a user event with the name of the user field as the key, and value made as a formatted string consisting of the custom state number, commas, and the data:

```
"Comment" , "207,This is the data about the call"
```

Closing the Custom State

To finish the state, the application should send the custom state name as the key and a minus sign as the value:

```
"AfterCallWork" , "-"
```

Using Multiple States Simultaneously

Only one state per type can be active for a DN at a time even though multiple different states can be simultaneously handled independently from each other. For example, two different states can be active on one DN, with different data corresponding to each:

```
"AfterCallWork" , "+"
"Break" , "+"
"Comment" , "207,This is the data about the call"
"Comment" , "208,This is the data about the break"
"Break" , "-"
"AfterCallWork" , "-"
```

For each state only one value of each field can be sent. If multiple key/value pairs for the same state and field are sent, only the last value is stored. For example, the following sequence will result in only the last key/value pair (`More data` in the field `Comment` for state `207`) being stored:

```
"Comment" , "207,This is data"
"Comment" , "207,More data"
```

Note: It is mandatory for the application to finish all the states it starts.

Chapter

7 Performance Measurements

This chapter describes performance measurements for Call Concentrator 7.0. This chapter contains these sections:

Performance Test Environments

The following describes the test environments used for measuring Call Concentrator performance:

- **Test Machine 21**, **Test Machine 22**—Micron Netframe 2101 2 CPU Pentium 2 400 MHz 512 MB NT
- **Sunlab**—Sun Enterprise 250 2 CPU UltraSPARC II 400 MHz 1Gb Solaris 2.8

Note: If you use more powerful hardware than was used in the testing environments, you can expect improved results.

The test environment was set up to as 4 sites—1 IVR site and 3 agent sites. It used a Load Balancing strategy to handle the information distribution.

Test Results on Microsoft SQL 7

Table 16 shows the percentage of hardware resources consumed to operate Call Concentrator using a Microsoft SQL database. The resulting number of calls

(per second) being handled is based on a 10-second TalkTime and 100 bytes of attached data.

Table 16: Microsoft SQL 7 Database

TalkTime = 10 seconds		Attached Data = 100 bytes				
Application/Calls		**4**	**8**	**12**	**16**	**20**
CCon 7.0.000.01 on Windows NT	CPU%	9	17	27.5	37	47
	Mem Mb	143	145	148	150	152
CCon 7.0.000.01 on Solaris	CPU%	8.5	17.4	27.2	35.5	41
	Mem Mb	82.4	83.9	85.7	87.3	88.8
Microsoft SQL	CPU%	8.5	16	21.6	31	40

Figure 2 shows the results presented in Table 16 in a graphical format. It shows CPU usage results for Call Concentrator on both Windows NT and Solaris when using a Microsoft SQL database.

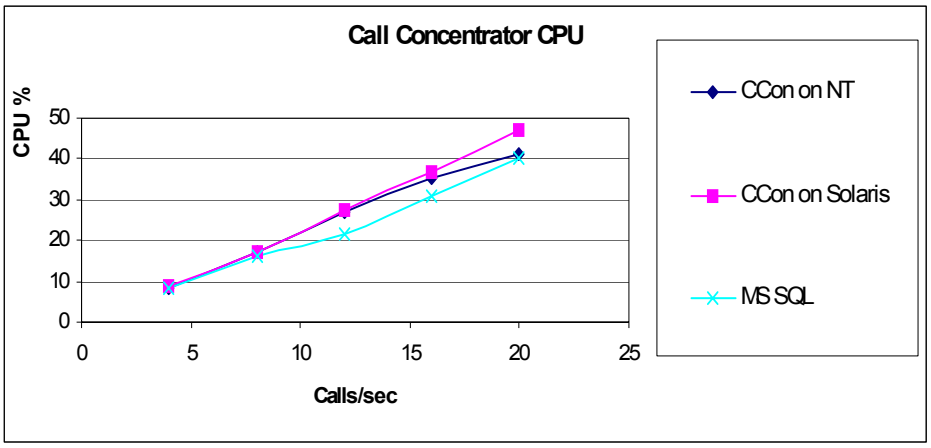

Figure 2: CPU Usage Results—Microsoft SQL Database

Test Results on Oracle

Table 17 shows the percentage of hardware resources consumed to operate Call Concentrator using an Oracle database. The resulting number of calls (per

second) being handled is based on a 10-second TalkTime and 100 bytes of attached data.

Table 17: Oracle Database

TalkTime = 10 seconds		Attached Data = 100 bytes						Using Binding
Application/Calls		**4**	**8**	**12**	**16**	**20**	**24**	**28**
CCon 7.0.000.01 on Solaris	CPU%	7	13	21	29	37	44	53
	Mem Mb	143	145	148	151	154	159	168
Oracle	CPU%	0.5	1	1.5	2	3	3.2	3.5

Figure 3 on page 81 shows the results presented in Table 17 in a graphical format. It shows CPU usage results for Call Concentrator on Solaris when using an Oracle database.

Note: Using Binding mode increases Call Concentrator performance by approximately 20%.

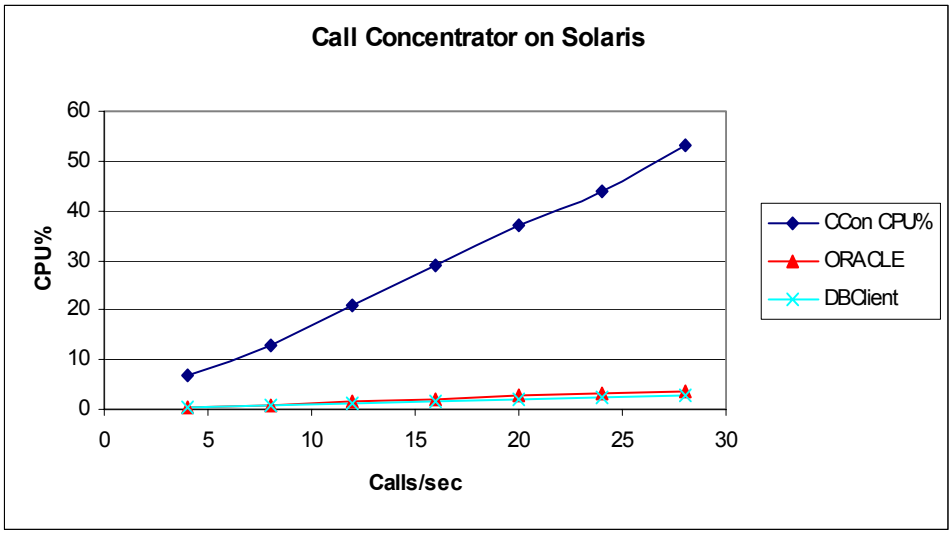

Figure 3: CPU Usage Results—Oracle Database

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W