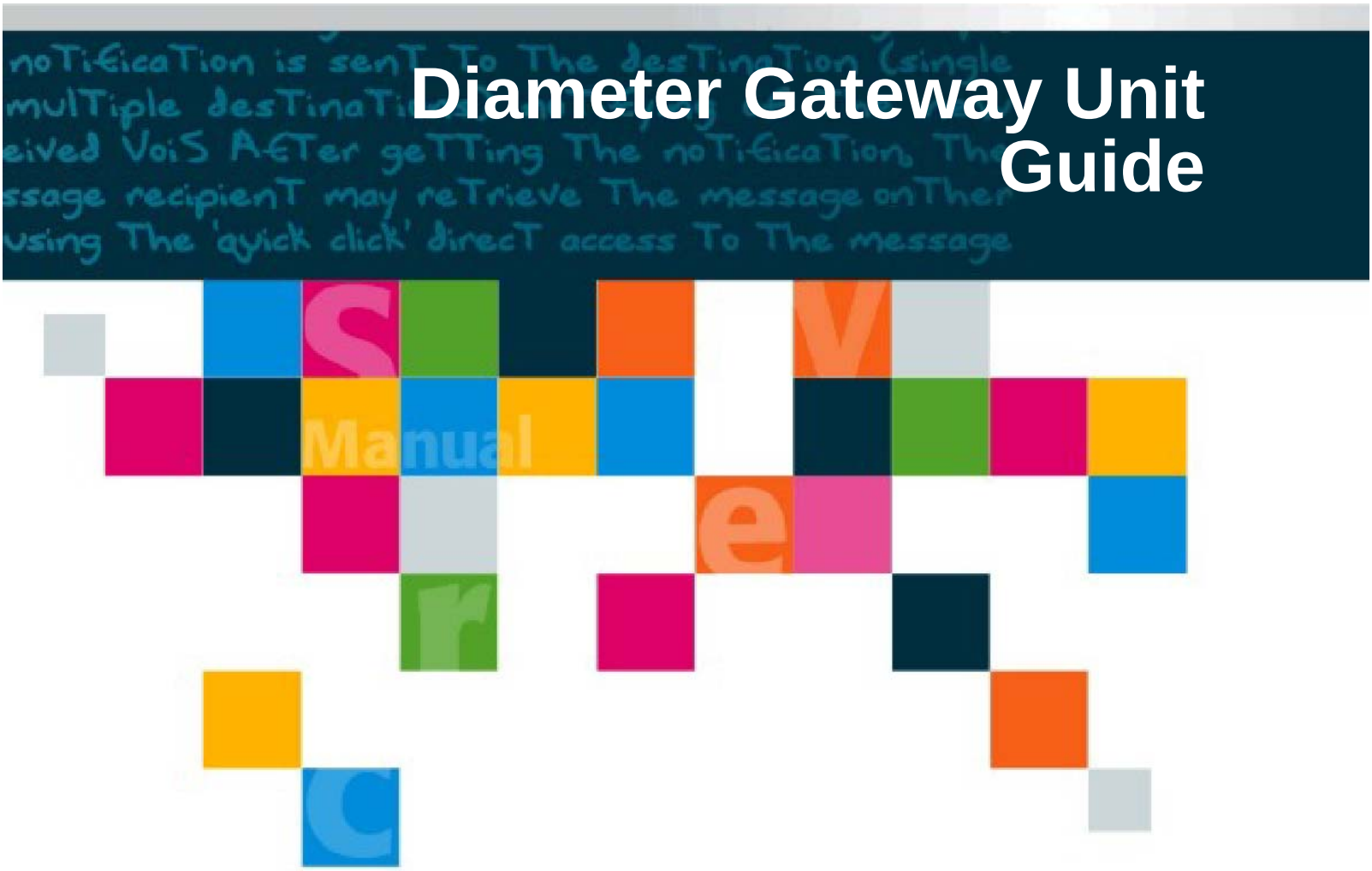
COMVERSE

ComverseONE™

3.5 RT TR 2.0

Diameter Gateway Unit Guide



ComONE-3.5-RT-TR-2.0-DGUG-2011-10-04

Notice

Corporate Headquarters
200 Quannapowitt Parkway
Wakefield, MA 01880 USA
Tel: (781) 246-9000
Fax: (781) 224-8143
www.comverse.com

Revision History

The following table lists the document changes since the initial publication:

Date	Chapter	Description
10/04/2011		Initial publication.

Contents

Figures

Tables

Notational Conventions

Useful information appears in this format.

Provides direction to important information

Important information appears in this format.

Indicates possible risk of damage to data, software, or hardware.

Indicates serious risk of damage to data, software, or hardware.

Table 1 Notational Conventions

Notation	Explanation of Convention
References to printed documents	*Helvetica italic* **Example:** See *Database Reference Volume 2*.
<KEYS>	UPPERCASE HELVETICA, in angle brackets **Example:** Press <CTRL><Q><SHIFT><P> to create an em dash.
`User-entered text`	**`Courier bold`** **Example:** Enter **`Total Charges`** in the field.
`Placeholders for user-determined text`	*`Courier italic`*, in angle brackets **Example:** Enter your <*`password`*>.
`Code samples, TABLE_NAMES, field_names, file and directory names, file contents, user names, passwords, UNIX ENVIRONMENT_VARIABLES`	`Courier`
Placeholders for system-generated text	*Helvetica italic* **Example:** Messages appear in this form: *timestamp messageID >> text.*
Buttons, Icon Names, and Menu items	**Helvetica bold** **Example:** Choose **Reports** from the main menu.

Special Markers

The Comverse ONE Billing and Active Customer Management solution has the three derivatives shown in Table 2, "Labels in Markers." For user convenience, any content that is specifically included in a derivative is highlighted with special markers so that it can readily be distinguished.

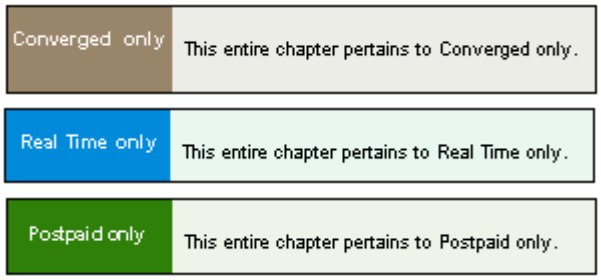
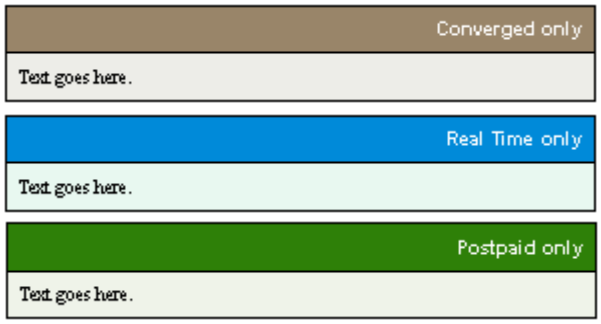
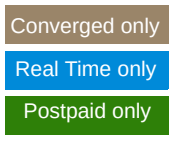
Table 2 Labels in Markers

Derivative	Label Shown in Markers
Comverse ONE Converged Billing derivative	**Converged only**
Comverse ONE Real-Time Charging derivative	**Real Time only**
Comverse ONE Postpaid Billing derivative	**Postpaid only**

Each derivative has a set of three color-coded markers, as shown in Table 3, "Types of Markers." The markers are used individually or in combination to highlight derivative-specific content by:

- Entire chapters
- Selected portions of chapters
- Tables, either entire or partial

Table 3 Types of Markers

Marker	Example	Description
Alert		■ Placed at the beginning of an entire chapter that pertains only to a specific derivative. ■ Placed just before a table that partially or entirely pertains only to a specific derivative.
Block		A shaded box that encloses sections of documentation that pertain only to a specific derivative.
Flag		■ Designates a shaded table row whose contents pertain only to a specific derivative. ■ In a bulleted list, designates an item that pertains only to a specific derivative.

Comverse ONE Documentation List

this is a comprehensive list. As such, it may include documentation for products which you have not licensed.

The documents described below reference the Comverse ONE solution products. All documentation available with the Comverse ONE solution is described in the following pages, organized by the following categories:

- Infrastructure Domain
- Rating, Charging, and Promotions Domain
- Billing and Financials Domain (Converged only)
- Customer and Order Management Domain (Converged only)
- Mediation and Roaming Solutions Domain
- Self-Service Solutions Domain

Read the relevant Solution Description first to get an overview of *your* Comverse ONE solution. It gives an overview of the functionality in each product domain and also includes cross-references to the user documentation that provides more detailed information about the functionality.
There are two such documents and they are listed under the Infrastructure Domain heading below.

- *Converged Billing & Active Customer Management Solution Description*
- *Real-Time Billing & Active Customer Management Solution Description*

Infrastructure Domain

Download every document in the Infrastructure domain if you purchase the Comverse ONE solution. Documentation for this domain includes the following (in alphabetical order):

- *Alarms Reference*
  Contains tables of alarm IDs, descriptions, likely causes, and recommended resolutions for systems and components of the Unified Platform.
- *Back Office Administration GUI Guide*
  Provides information about the BackOffice subsystems for Inventory Administration, Address Management and Bulk Operations.
- *Converged Billing & Active Customer Management Solution Description*
  General overview of the Comverse ONE Converged Offer and the functionality available in each domain.
- *Database Reference*
  Describes all database tables and fields in detail.

- *Disaster Recovery Operations Guide* (Optional Module)
  The Disaster Recovery Operations Guide serves as both a technical overview of the optional Disaster Recovery solution and as a guide which details the operational procedures for failover, switchover and switchback provided by the solution.
- *Glossary*
  Provides a list of terms used specifically for the Comverse ONE solution
- *Investigation Units and Financial GUIs Guide*
  Describes the GUI-based tools used for investigating and troubleshooting various financials related processes: payments, bill invoices, refunds, and incomplete data work entries
- *Operation Reference*
  Describes the processes in the Comverse ONE solution.
- *Platform Operations Guide*
  Describes the back-end operations and maintenance functionality of the core Comverse ONE solution components. Includes AIX/HACMP platform and cluster operations, Linux/Veritas platform and cluster operations, backup/recovery, shared storage and fiber switch operations, and tape backup operations.
- *Product Catalog Overview*
  Provides a high-level description of the Comverse ONE solution Product Catalog, which is the primary mechanism for creating, configuring, managing, and propagating Product Catalog versions.
- *Product Catalog User Guide*
  Instructions on using the Product Catalog application to define and manage all aspects of Service provisioning.
- *Real-Time Billing & Active Customer Management Description*
  General overview of the Comverse ONE Real-Time Offer and the functionality available in each domain.
- *Schedulable Entity Reference Manual*
  Documents all the jobs, monitors, and workflows, for each component in the Unified Platform.
- *Security Platform Operations Guide*
  Technical overview of the security platform and information on how to provision and administer the platform.
- *Security Server API Guide*
  Provides an overview of the interfaces exposed by the Java-based Security SDK API, which client applications can leverage to access various security services, such as authentication, authorization, auditing, key management, and credentials management. Also provides information on the Security Web Services API, which provides interfaces to a subset of Security Server commands (Identity Management commands).
- *Signaling Gateway Unit Guide*
  Describes the hardware, installation, configuration, and maintenance of the Signaling Gateway Unit (SGU) used to connect Comverse real-time systems to the SS7 signaling network using either traditional SS7 protocols or Sigtran (SS7 over IP).
- *System Measurements Guide*
  The Comverse ONE Solution automatically collects statistical data from the Service Logic Unit (SLU) and the Service Gateway Unit (SGU). This includes service statistics on the SLF layer and platform data on the IPF layer.
  This guide describes the format and location of this measurement information and provides a description of the meaning of the data.The measurement data can be used to create reports. It can also be imported into other applications (such as Excel) to be viewed.
- *System Validation Check Reference*
  Details all the system validation checks performed by the Comverse ONE Unified Platform on its components.
- *Unified API Guide*
  General overview of the Unified API, a brief description of its architecture, and information about:

- Framework classes and the functionality they provide
- Two standard interfaces provided with the Unified API (client SDK and web services)
- A subset of Unified API business methods most commonly used

- *Unified Platform Guide*
  Technical overview of the Unified Platform and information on the procedures to manage core systems operations in the Comverse ONE solution.

Rating, Charging, and Promotions Domain

Documentation for this domain includes the following (in alphabetical order):

- *Bulk Provisioning Guide*
  - The *CC Batch* utility enables bulk creation of recharge vouchers and subscribers.
  - The *Bulk Provisioning* Utility enables bulk creation of anonymous accounts to support the pre-activation of pre-paid SIM cards.
- *Charging Interfaces Guide*
  Describes the four interfaces that enable external services to support real-time authorization, rating, and charging for transactional usage: (1) the Event Charging Interface, a simple TCP/IP-based interface, (2) Open Services Access (OSA), (3) a Diameter-based interface version enhanced to take advantage of features of the Comverse ONE solution, and (4) a Diameter-based interface packet-switched version.
- *Customer Care Client Provisioning Guide* — Real-Time
  Detailed task-oriented instructions for using Customer Care Client.
- *Diameter Gateway Unit Guide*
  Describes the hardware, installation, configuration and maintenance of the Diameter Gateway Unit (DGU) used to connect Comverse real-time systems to external services, using the diameter protocol over IP.
- *IVR Call Flows Reference*
  These all flows detail the logic flow of specific scenarios. Multiple access numbers can map to the same call flow. Different resellers have the option to publish different numbers but share the same logic.
- *Network Interfaces and Notifications Guide*
  Describes the operation, features, and provisioning of notifications, CAMEL-enabled services, and USSD-enabled services.
- *Network Self-Care Guide*
  Describes the configuration, structure, and features.
- *Operational Reports and Data Warehouse Utility Guide*
  Describes the real-time Operational Reports Interface (ORI) and the Data Warehouse Extract Utility.
- *Rating Technical Reference*
  Describes the Unified Rating Engine, which is the subsystem responsible for gathering incoming CDRs and processing them for billing.
- *Recurring–Non-Recurring Charges Server Guide*
  Describes all processes commonly available through the Recurring —Non-Recurring Charges Server.
- *Voucher and Recharge Guide*
  Describes the process by which subscribers add funds to accounts using recharge vouchers through IVR, interaction with Customer Service, and other methods. Provides details of the Recharge Control Table, which allows resellers to provision the effects of recharges so that bonuses, discounts, and other changes to offers can result from a successful recharge. Also describes the Card Generator software used to create batches of recharge vouchers.

Billing and Financials Domain (*Converged* only)

Documentation for this domain includes the following (in alphabetical order):

- *Advanced Invoice Numbering Guide*
  Describes how to configure and use Advanced Invoice Numbering.
- *Billing Reports and File Layouts User Guide*
  Describes control reports and other file formats.
- *Billing Technical Reference*
  High-level descriptions of billing architecture, administration, bill generation and formatting, and system parameters
- *Collections Guide*
  Contains information on configuring Collections database tables, running the Collections module, and using the Collections interface.
- *Invoice Designer Strings and Filters Reference*
  Describes the static strings, dynamic strings, and filters in the Invoice Designer.
- *Invoice Designer Technical Reference*
  Describes how to configure and run Invoice Designer.
- *Invoice Designer User Guide*
  Describes the Invoice Designer and how to perform the tasks needed to create an invoice template.
- *Journals Guide*
  Describes the theory, configuration, and running of Journals processes.
- *Miscellaneous Configurable Entities*
  Instructions for configuring late fees, adjustments, and several other database entities used in postpaid and converged billing.
- *Process Workflow Orchestration Guide*
  Describes the command-line entries and the default queries for running billing-related processes via the Unified Platform.
- *Taxation Guide*
  Describes the configuration, operation, structure, and features of Taxation.

Customer and Order Management Domain (*Converged* only)

Documentation for this domain includes the following (in alphabetical order):

- *Customer Center User Guide*
  Detailed task-oriented instructions for using Customer Center.
- *Inventory Guide*
  Describes the configuration, operation, structure, and features of Inventory.
- *Inventory Replenishment Guide*
  Describes the operation, structure, and features of Inventory Replenishment.
- *Orders Services Guide*
  Describes the structure and features of Orders Services.
- *Request Handling and Tracking and Service Fulfillment User Guide*
  Describes the configuration, operation, structure and features of Request Handling and Tracking and Service Fulfillment.
- *Workflow Developers Guide*
  Helps new users understand the rules-based business process management system so users can create solutions and integrate Workpoint within those solutions.
- *Workflow User Guide*
  Describes the configuration, operation, structure, and features of Workpoint.

Customer Relationship Management

- *Campaign Management Data Mapping Reference*
  Describes how the data in DataMart is mapped to information in the Comverse ONE Customer database, the Comverse ONE ODS, and the Comverse ONE Sales and Service database.
- *Campaign Management DataMart Implementation Guide*
  Contains in-depth technical information on how to configure and populate the data mart used by all Campaign Management applications.
- *Campaign Management Outbound Marketing Manager Reference*
  Describes how to use the Campaign Management Outbound Marketing Manager features and guides you through the program's basic functionality.
- *Campaign Management Quick Implementation Guide*
  Helps novice users get started with implementing Campaign Management. It contains an overview of the product architecture, information on data mart design and creation, an explanation of how extraction works, and procedures for creating web pages, reports, lists, and campaigns.
- *Campaign Management Topic Implementation Guide*
  Provides information for implementers and professional services personnel who are creating applications that will run on an Campaign Management EpiCenter. Summarizes the Campaign Management functionality, architecture, and administration and contains in-depth technical information for configuring the Campaign Management topics required for Campaign Management and analysis.
- *Campaign Management User Guide*
  Provides you with basic information about the Campaign Management applications.
- *Case Management User and Administration Guide*
  Contains detailed information about GUI screens and form fields that appear in the Case Management application. Also provides information on performing general procedures in the GUI and administrative tasks.
- *Customer Center User Guide*
  Detailed task-oriented instructions for using Customer Center.
- *Sales and Service Admin Console User Guide*
  Provides supervisors, managers, and executives with the information to use the Case Management and Sales Force Automation Admin Console application.
- *Sales and Service Application Reference*
  Contains technical reference information relevant to implementers involved in implementing and customizing CRM applications at customer sites. This book provides the reference context for the procedural information available in the Implementation Guide.
- *Sales and Service Architecture Reference*
  Provides technical information relevant to individuals involved in implementing the Open Architecture and the applications built on the architecture
- *Sales and Service Data Dictionary Reference*
  Includes a listing and description of the tables and columns used to store CRM operational business data. It also includes a description of the naming conventions for the tables. The target audience includes database administrators, application developers, and implementers.
- *Sales and Service Dialog Designer User Guide*
  Describes the Sales & Service Dialog Designer, a web-based graphical application for defining and editing dialogs. Includes procedures for using it.
- *Sales and Service IBR Designer User Guide*
  Describes how to use the IBR Designer to create Intelligent Business Rules, which can be used to implement rule-based behavior within your CRM applications.
- *Sales and Service Implementation Guide*
  Provides procedural information relevant to individuals involved in implementing and customizing the core and the Sales and Service applications built on the core.

- *Sales and Service Integration Guide*
  Provides overview and configuration information for the set of tools used to exchange data with a variety of back-end data sources, including generic SQL sources, Java and EJB-based sources, Web services, and other database types.
- *Sales and Service Workflow Designer*
  Explains how to use Workflow Designer, a web-based graphical tool for defining and editing workflows
- *Sales Force Automation User and Administration Guide*
  Contains detailed information about GUI screens and form fields that appear in the Sales Force Automation application. Also provides information on performing general procedures in the GUI and administrative tasks.

Mediation and Roaming Solutions Domain

Documentation for this domain is subdivided into Mediation/Roaming and Revenue Settlements.

Mediation and Roaming

Mediation and Roaming documentation includes the following (in alphabetical order):

- *API Guide*
  Provides the concepts and functions for the Collection Application Programming Interface (CAPI), Mediation API, and Socket-Based Transmission API.
- *Data Manager GUI Reference*
  Contains detailed information about GUI screens and form fields that appear in the Data Manager interface
- *GRID Mapping Language Developer Guide*
  Describes the mediation feature components, semantics, and general syntax of the GRID Mapping Language (GML).
- *Installation Guide for HP*
  Describes how to install and configure the application, components, and some third-party applications associated with the HP platform.
- *Installation Guide for HP Itanium*
  Describes how to install and configure the application, components, and some third-party applications associated with the HP Itanium platform.
- *Installation Guide for HP PA-RISC*
  Describes how to install and configure the application, components, and some third-party applications associated with the HP PA-RISC platform.
- *Installation Guide for IBM*
  Describes how to install and configure the application, components, and some third-party applications associated with the IBM platform.
- *Installation Guide for SUN*
  Describes how to install and configure the application, components, and some third-party applications associated with the SUN platform.
- *Mediation and Roaming User Guide*
  Provides information on how to use the GUI interface, including information on using the Data System Manager application pages.
- *Roaming Database Reference*
  Provides reference information on the Roaming database.
- *Roaming Setup Guide*
  Describes how to configure the Roaming Setup application pages. It also provides information on working with TAP, RAP, and CIBER statistics.

- *Scripts Guide*
  Provides information on script files, which contain additional instructions on functions for data collection and transmission.
- *System Manager GUI Reference*
  Contains detailed information about GUI screens and form fields that appear in the System

Self-Service Solutions Domain

The Comverse ONE Self-Service Solutions domain consists of the core products plus the optional separately licensed premium products. The core products consist of the following:

- Self-Service Solutions Platform
- Self-Service Solutions Applications

Self-Service Solutions Platform Documentation

The Self-Service Solutions Platform has a comprehensive set of documentation covering the installation, configuration, and use of our products. The documentation set is divided into the following categories:

- **Manuals**: These manuals cover installing and using the platform.
- **Reference**: These reference documents contain information about APIs, databases, configuration files, and so on. These documents are delivered in HTML.

Self-Service Solutions Platform Manuals

Self-Service Solutions Platform manuals include the following (in alphabetical order):

- *Self-Service Platform Administration Guide*
  Provides operations and maintenance instructions for Web applications using the Self-Service Solutions Platform.
- *Self-Service Platform Catalog Loader Reference*
  Provides information about the Catalog Loader, including a functional description as well as installation, configuration, and use instructions.
- *Self-Service Platform Communications Billing and Usage Reference*
  Provides detailed descriptions of the data models and structure of the Self-Service Solutions Platform Communications Billing and Usage (CBU) database.
- *Self-Service Platform Connectors Development Guide*
  Provides instructions for developing and customizing Connectors of the Self-Service Solutions Platform.
- *Self-Service Platform Core Module Development Guide*
  Provides instructions for configuring and developing features of the core module of the Self-Service Solutions Platform.
- *Self-Service Platform Customer Interaction Datastore Reference*
  Provides detailed descriptions of the data models and the structure of the Self-Service Solutions Platform Customer Interaction Datastore (CID).
- *Self-Service Platform Database Modules Development Guide*
  Provides instructions for configuring, customizing, and developing features of the database module of the Self-Service Solutions Platform.
- *Self-Service Platform Installation Guide*
  Provides installation and configuration instructions for the Self-Service Solutions Platform.
- *Self-Service Platform Services Guide*
  Provides instructions for configuring, customizing, and developing features that use the services provided by the Self-Service Solutions Platform.

- *Self-Service Platform Processors Development Guide*
  Provides instructions for developing and customizing Processors of the Self-Service Solutions Platform.
- *Self-Service Platform Reports Development Guide*
  Provides instructions for developing and customizing Reports of the Self-Service Solutions Platform.
- *Self-Service Platform Web Applications Development Guide*
  Provides instructions for configuring, developing, and deploying Web applications that use the Self-Service Solutions Platform.
- *Self-Service Solutions Overview Guide*
  Provides a high-level architectural and functional description of the Comverse ONE Self-Service Solutions. It also includes a detailed description of the concepts and development process to create and deploy Self-Service Solutions.

Self-Service Solutions Platform Reference

Self-Service Solutions Platform reference documentation includes the following (in alphabetical order):

- *Base Logic Manager Reference*
  Describes usage syntax and configuration files for the Base Logic Manager (BLM) APIs. These APIs are the core services of the Self-Service Solutions Platform.
- *CID2CBU Object Mapping Reference*
  Describes the default mapping of Customer Interaction Datastore (CID) and Communications Billing and Usage (CBU) objects.
- *Communications Billing and Usage Reference*
  Provides detailed descriptions of fields and tables in the Communications Billing and Usage (CBU) database.
- *Customer Interaction Datastore Reference*
  Provides detailed descriptions of fields and tables in the Customer Interaction Datastore (CID).
- *Integration Services Framework API Reference*
  Describes usage syntax of the set of APIs to program connectors and other components of the Intelligent Synchronization Framework (ISF).
- *Integration Services Framework Message Cache Reference*
  Provides detailed descriptions of fields and tables in the Intelligent Synchronization Framework (ISF) Message Cache.
- *Integration Services Framework Script API Reference*
  Describes usage syntax of the Intelligent Synchronization Framework (ISF) script APIs to program the ISF connectors.
- *JavaServer Page Framework for Internet Application API Reference*
  Describes usage syntax for the JavaServer Page Framework for Internet Application (JFN) APIs. These APIs are used to build JSPs using the JFN. This framework provides basic application functions and services as the foundation of user interfaces.
- *Logger Message Reference*
  Provides detailed descriptions of the Self-Service Solutions Platform log messages.
- *QRA API Reference*
  Describes usage syntax for the Query, Reporting, and Analysis (QRA) Engine APIs. These APIs are used to build reports.
- *UTIL API Reference*
  Describes usage syntax for the UTIL package used by different components of the Self-Service Solutions Platform. This package contains a set of utilities including the logger.

Self-Service Solutions Applications Documentation

Each Self-Service Solutions Application comes with a comprehensive set of documentation covering the installation, configuration, and use of the product. The application documentation expands and complements the Self-Service Solutions Platform documentation.

The documentation set is divided into the following categories:

- **Manuals**: These manuals cover installing and using the application.
- **Reference**: These reference documents contain information about APIs, databases, configuration files, and so on. These documents are delivered in HTML.

Self-Service Solutions Application Manuals

A full set of these manuals is available for each Self-Service Solutions Application (Business, Channel, Consumer, andCSR Portal). The documentation set includes the following (in alphabetical order):

- *Business Objects Model Reference*
  Provides a detailed description of the models and entities that make up the Self-Service Solutions Application.
- *Configuration and Development Guide*
  Provides instructions for configuring and developing Self-Service Solutions Application features.
- *Introduction*
  Provides a high-level architectural and functional description of the Self-Service Solutions Application. It covers common features, order management, account management, and bill presentment.
- *Feature Reference*
  Describes the logic and provides use cases for the functional domains of the application.
- *Out-of-the-Box Reference Guide*
  Describes the Self-Service Solutions Application Out-of-the-Box release.
- *Self-Service Installation Guide for Comverse ONE*
  Provides detailed installation, configuration, and deployment instructions for the Self-Service Solutions Application alongside other elements of the Comverse ONE solution.
- *Self-Service Installation and Deployment Guide*
  Provides detailed installation, configuration, and deployment instructions for the Self-Service Solutions Application.
- *User Guide*
  Provides instructions for navigating and using the Self-Service web application. For Business Self-Service and CSR Portal only.

Self-Service Solutions Application References

A full set of these references is available for each Self-Service Solutions Application.The reference documentation set includes the following (in alphabetical order):

- *API Reference*
  Describes usage syntax for the Self-Service Solutions Application APIs. These APIs are used to program the user interface and manage data.
- *Invoice Schema Reference*
  Describes the invoice schema reference of the Self-Service Solutions Application.
- *Presentation Layer Page Flow Reference*
  Describes the page flows of the Self-Service Solutions Application.
- *Specification Entity Relationship Diagrams*
  Provides diagrams describing the actors, use cases, user activity, and storyboard in IBM Rational Rose format.

Self-Service Solutions - Separately Licensed Products

Documentation available with optional, separately-licensed premium products in the Comverse Self-Service Solutions is listed below.

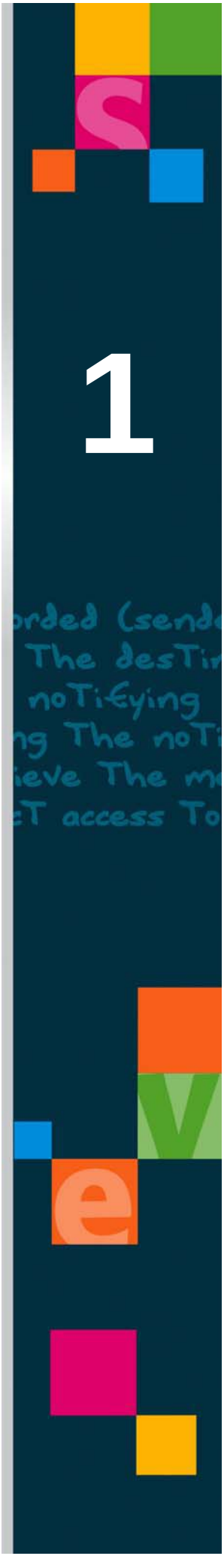
Online Catalog Manager

Online Catalog Manager (OCM) documentation includes the following (in alphabetical order):

- *Introduction to the Online Catalog Manager*
  Provides a high-level architectural and functional description of the Online Catalog Manager.
- *Online Catalog Manager Getting Started Guide*
  Describes the best way to build product catalogs in the Online Catalog Manager. This manual is a template for creating end-user documentation.
- *Online Catalog Manager Installation and Configuration Guide*
  Provides installation and configuration instructions for the Online Catalog Manager.
- *Online Catalog Manager User Documentation Template*
  Describes the use of the Online Catalog Manager. This manual is a template for creating end-user documentation. This manual covers many common concepts and procedures of the OCM.
- *Online Catalog Manager User Guide*
  Provides a detailed description of the concepts and use of the Online Catalog Manager. The topics include:
  - Managing Media Files
  - Managing Offers
  - Managing Prices
  - Managing Products
  - Managing Properties
  - Managing Reference Data
  - Publishing

Chapter 1

Diameter Gateway Unit Platform Overview



New Features for This Release

The following features for the Comverse ONE 3.5 RT TR 2.0 release affect the Diameter Gateway Unit Guide:

- Create a Mediator/loader into Ported Numbers

Overview

The Diameter Gateway Unit (DGU) is part of the Comverse Diameter Online Charging Server (OCS). The DGU provides an interface to the Comverse OCS for Diameter clients. The DGU interfaces with back-end Diameter Service Logic Units (DSLUs).

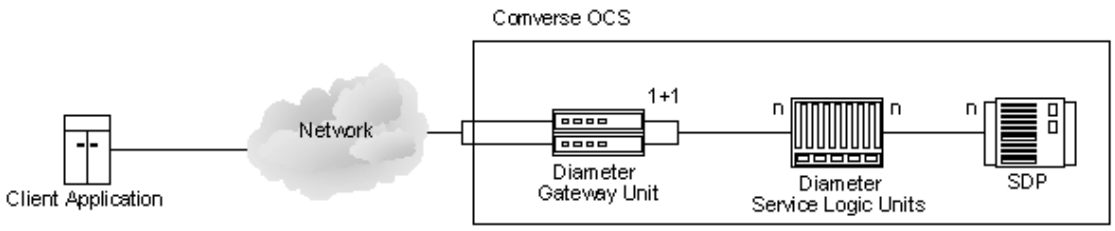
The DGU provides the following functions:

- Distributes administration of systems, including the maintenance of security associations, to a configurable grouping
- Concentrates requests from a number of distributed client sets to a set of like servers
- Provides value-added processing to the requests or responses
- Supports load balancing

The OCS design is based on a distributed architecture with a limited number of exposed Diameter interface points to client applications and some number of processing logic devices behind these interface points.

The following illustration shows the distributed OCS system.

Figure 1 Comverse Diameter Online Charging Server



The Comverse Diameter OCS implements Diameter Server functionality and comprises the following components:

- Diameter Gateway Unit (DGU)
- Diameter Service Logic Unit (DSLU)
- Service Data Point (SDP)

The design supports high availability and a scaleable architecture.

The DGU provides an interface between the outside network and the Comverse OCS. Only Transmission Control Protocol (TCP) as transport and IP security (IPsec) are supported.

The DGU provides all of the necessary components of the Diameter protocol. The DGU alone does not implement all functions of the protocol. A portion of the protocol is implemented at the DGU and the DSLU implements the remainder. The division of functionality between the DGU and DSLU is as follows:

- DGU implements Diameter Base protocol, routing, load distribution, and handling of a limited set of non-session messages.

- DSLU implements the Diameter application, for example, the Diameter Credit Control Application (DCCA) and service logic.

The interface between the DGU and the DSLU uses the Diameter encoding of message formats but is enhanced to support DGU to DSLU connectivity management. Comverse-specific AVPs, using Comverse Vendor ID (1619), are added to Diameter messages to pass proprietary information (for example, congestion indication) on this interface. Only TCP is supported on this interface. No security method is required or supported on this interface.

The DGU maintains a Diameter Base protocol information database that includes:

- The peer table and realm table used in routing outbound Diameter messages
- Origin-host(s) and Origin-realm, inserted in Answer messages to clients
- Various values for AVPs necessary to support Diameter Base protocol

The DGU also maintains a database of known DSLUs and their current status. The DGU updates this status dynamically on DSLU availability. The DSLUs also provide information indicating their ability to accept traffic.

To support reliability, the DGU is deployed using a 1+1 load sharing configuration. The DGU interfaces to *n* DSLUs, which provide the scalability. As processing requirements increase, DSLUs are added to increase capacity of the system. Capacity increases of the DGU is only achieved by adding CPUs or upgrading the machine. The DSLUs are organized in a N+1 scheme. If a DSLU fails, all sessions handled by the failed DSLU are lost. The remaining DSLUs share the traffic load.

The SDP contains the subscriber and provisioning database for the Comverse ONE solution. The DSLU interfaces to this database through the existing Charging API developed for other Comverse ONE solution applications.

The Diameter library is threads based, with one thread per connection. The Diameter Gateway Application (DGA) process, which uses the Diameter library, is the main process at the DGU. The Diameter library establishes multiple TCP connections with the Diameter clients, and there are multiple TCP connections to the back-end DSLUs. In such an environment, a multi-threaded application architecture enables better utilization of the multi-CPU hardware.

DSLU Interface

Interaction with the DSLU is based on “RFC 3588 – Diameter Base Protocol,” but is not a fully compliant Diameter interface and has the following limitations:

- Dynamic discovery of peers is not supported.
- Only TCP is supported as transport; the DGU acts as a server and waits for Capabilities Exchange Request (CER) from the DSLU.
- No security method is supported.
- Comverse-specific AVPs are used for congestion handling.
- On DSLU failure, messages destined for that DSLU are sent to another DSLU within the realm handling that application.

Client Interface

Interaction with Diameter clients conforms to “RFC 3588 - Diameter Base Protocol” with the following limitations:

- Dynamic discovery of peers is not supported.
- TLS security is not supported.
- Load sharing is not supported for outgoing requests.

Network Interface

The DGU is deployed in a 1+1 CE configuration. Each DGU CE supports at least four Network Interface Cards (NICs).

Two interfaces from each DGU CE connect to a client IP zone. From the client perspective, these are the only interfaces to the Comverse OCS. IPsec provides security on these interfaces. The interface to the client IP zone is behind a firewall in an isolated segment. This is referred to as a Demilitarized Zone or DMZ.

To communicate with the DSLUs, at least two interfaces from each DGU CE connect to a Secure and Storage zone, protected by a firewall from the client DMZ.

Failure Modes

The Diameter protocol has built-in capabilities for error detection and recovery including message-related, connection-related, and node-related errors. The Diameter protocol works in a Request/Answer mode of operation and any lost message results in retransmission of the request. Diameter uses a reliable transport mechanism (TCP) and all connections between Diameter entities are redundant.

The Comverse implementation of Diameter is always deployed as a dual node system with redundant connections. Each node is capable of handling the full traffic load, enabling uninterruptible operation even after the total loss of a node.

DGU Subsystem

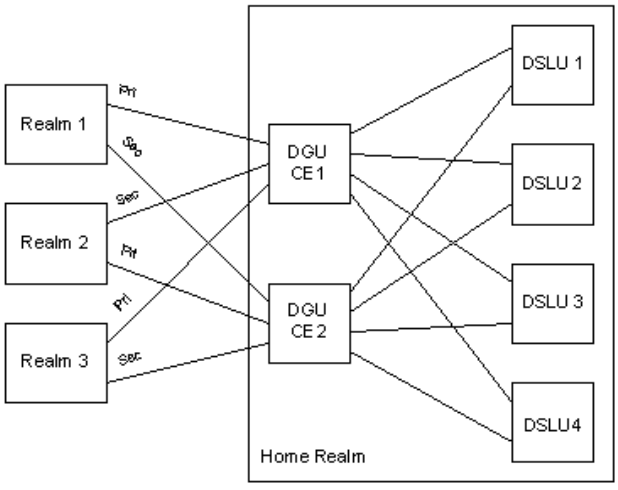
The DGU operating system is Linux.

The DGU is always deployed in a 1+1 configuration with primary and secondary connections distributed between the two CEs. Any single CE failure has no impact on the DGU functionality.

The backend DSLUs are organized in an N+1 scheme. Any failure in the DSLU causes all messages destined for that DSLU to be forwarded to another DSLU. The other DSLU is expected to send the appropriate response back to the client.

Figure 2, “DGU Interconnection”shows the interconnection of the various components of the OCS. At a minimum, there are established connections with two peers per realm, known as the primary and secondary peers. If necessary, additional connections are established.

Figure 2 DGU Interconnection

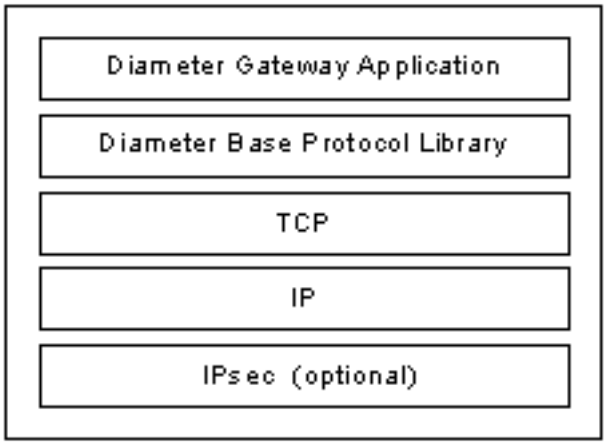


Typically, all messages for a realm are sent to the primary peer. If failover procedures are invoked, any pending requests are sent to the secondary peer. Implementations can load balance requests between a set of peers. Also, a given peer can act as a primary for a given realm, while acting as a secondary for another realm. TCP is the only transport protocol supported and only IPsec security is supported.

On the DSLU side, each DSLU sets up a connection to each DGU CE. Only TCP transport is supported. No security method is used on this interface.

Figure 3, “DGU Protocol Stack” shows the components of the DGU protocol stack.

Figure 3 DGU Protocol Stack



Diameter Gateway Application

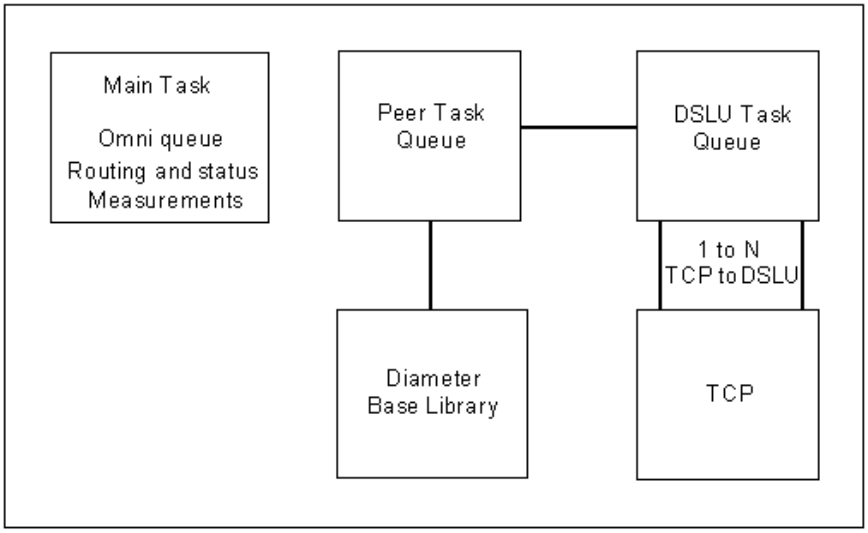
The DGA process is functionally similar to a Diameter Proxy Agent. Peer connections established between the Diameter clients and the DGA follow the Diameter protocol requirements strictly. Peer connections are also established between the DGA and the back-end DSLUs, but these do not conform strictly to the Diameter protocol requirements. Only TCP is supported for the DSLU connections.

The DGA is implemented as a single OMNI process. Ulticom provides the Diameter Base Protocol library. The TCP and IP protocols are part of the Linux operating system.

The DGA application is implemented using POSIX threads. The Diameter Base Protocol library assigns each TCP connection to a separate thread.

Figure 4, “Diameter Gateway Application Processes”shows the internal organization of the DGA process. The following sections describe each block in more detail.

Figure 4 Diameter Gateway Application Processes



Main Task

The Main Task is responsible for process initialization and OMNI message handling (send and receive).

The responsibilities of the Main task are:

- Optionally attach to OMNI
- Register with OMNI
- Initialize debug
- Configure based on configuration file
- Build global database based on configuration and maintain status
- Create Peer Task
- Create DSLU Task
- Process OMNI messages

Peer Task

The Peer Task initializes and configures the Diameter Base library and processes messages from the library.

On receiving a message from an external client, the library issues a callback defined in this task. The callback function, running in the context of the library thread, queues a pointer to the message received and related information to the task queue. On getting the queued message, the main service routine selects a DSLU based on the routing algorithm, encodes the message, and sends it using the DSLU Task. Information related to this request is also saved in a pending requests list maintained for each DSLU, using hop-id as the primary key.

The task receives messages from the DSLU via the DSLU Task. For the answer message, it matches the pending request entry based on the hop-id and sends the answer on the peer connection the request arrived on.

The library also issues the callback defined in this process for notifications. Depending on the notification, it results in an alarm, event, measurement update or status.

The responsibilities of the Peer task are:

- Initialize Diameter library

- Create Peers
- Create Realms
- Create Service within the Realm
- Add Peer to handle the Service
- Activate Peer
- Set library receive callback
- Set library notify callback
- Receive messages from external client, then encode and route to DSLU based on destination host or destination realm, using a modified round robin algorithm
- Receive messages from DSLU, then decode and send to external client. The answer message is sent back on the same peer connection on which the request arrived.

DSLU Task

The DSLU task interfaces to the DSLUs using TCP. It listens on a well-defined port for DSLUs to connect and starts a handler for each DSLU connection.

The responsibilities of the DSLU task are:

- Accept connection from DSLU and create new handler
- DSLU connection state event
- Watchdog state event
- Process message from Peer Task and route to specified DSLU
- Process application messages from DSLU and send to Peer Task
- Process DWR, DWA, and CER locally
- Process congestion indication from DSLU and modify routing algorithm

The DSLU Task maintains the following tables:

- Peer Table
- Service Table
- DSLU Table
- DSLU Routing Table

Each of these tables is decribed in detail in the sections that follow.

Peer Table

The Peer Table contains information on all configured peers. Configuration information is read at startup and passed to the Diameter Base library. This information is static. The Peer Table also maintains the current status of the peer connection.

The Peer Table entry fields are:

- Peer Name – Logical name that identifies this peer
- Peer FQDN – The Fully Qualified Domain Name of the Peer host (same as Origin-Host AVP in the received CER or Capabilities Exchange Answer (CEA))
- Local FQDN
- Peer Port
- Local Port
- Transport (TCP or SCTP)
- Security (IPsec, TLS or NONE)

- Connection (Client or Server)
- Peer Activated – true or false
- Peer Status – Current status of the peer connection
- Message Counters

Service Table

The Service Table contains information on the services configured and the associated peers. The configuration part of this information is read at startup and passed to the Diameter Base library. This information is static. The Service Table also maintains the current status of the service.

The Service Table entry fields are:

- Service Name – Logical name that identifies this service (realm/application)
- Realm Name – Primary key for lookups based on longest-match-from-the-right on the realm
- Application Identifier – Secondary key, identified by a vendor ID and an application ID
- List of Peer Table Entries
- Service Status

DSLU Table

The DSLU Table contains information on all DSLUs configured. The configuration part of this information is read at startup and is static. The DSLU Table also maintains the current status of the DSLU connection.

The DSLU Table entry fields are:

- DSLU Name – Logical name that identifies this DSLU.
- DSLU FQDN – Fully Qualified Domain Name of the DSLU host (same as Origin-Host AVP in the received CER or CEA).
- Local FQDN
- DSLU Port
- Local Port
- DSLU Activated – true or false
- DSLU Status – Current status of the DSLU connection.
- Pending Requests Count
- Roundtrip Delay Values
- Message Counters

DSLU Routing Table

The DSLU Routing Table contains information on the services configured and the associated DSLUs. The configuration part of this information is read at startup and is static. The DSLU Routing Table is used to select the next DSLU to route the new session. The DSLU Routing Table also maintains the current status of the service.

The DSLU Routing Table entry fields are:

- Service Name – Logical name that identifies this service (realm/application)
- Realm Name – Primary key for lookups based on longest-match-from-the-right on the realm
- Application Identifier – Secondary key, identified by a vendor ID and an Application ID
- Service Status
- List of DSLU Table Entries
- Last used DSLU Table Entry – used for round robin

Diameter Base Library

The Diameter Base protocol provides the minimum features required to deliver AVPs, capability negotiation, error notification, and so on. The Diameter Base protocol is defined by RFC 3588. The Diameter applications extend the Base protocol by adding new command codes and AVPs.

Diameter uses TCP for Diameter message transport over IPv4. Management of the underlying TCP sockets for connection establishment, failover procedures, and so on, is handled internally by the Diameter library.

Peer Connections

When no transport connection exists with a peer, a connection attempt is made periodically. The Linux Tc timer handles this behavior with a default value of 30 seconds. There are certain exceptions to this rule, such as when a peer has terminated the transport connection stating that it does not wish to communicate.

If Diameter receives data from TCP that cannot be parsed or identified as a Diameter error made by the peer, the stream is compromised and cannot be recovered. The transport connection is closed. The DGU periodically attempts to reestablish the connection until successful. The Linux Tc timer controls these periodic reconnection attempts.

The DGA listens for TCP connection from the DSLU on a well-defined port that does not conflict with other applications. The DSLU attempts to establish a connection with the DGA on startup and repeats the connection attempt periodically until it is successful.

Capabilities Exchange

When two Diameter peers establish a transport connection, they exchange Capabilities Exchange messages. This message allows the discovery of a peer's identity and its capabilities, including protocol version number, supported Diameter applications, security mechanisms, and so on.

A Diameter entity that receives a CER message and does not have any applications in common with the sender returns a CEA with the Result-Code AVP set to DIAMETER_NO_COMMON_APPLICATION and disconnects the transport layer connection.

Similarly, a Diameter entity that receives a CER)message that does not have any security mechanisms in common with the Diameter client returns a Capabilities-Exchange-Answer (CEA) with the Result-Code AVP set to DIAMETER_NO_COMMON_SECURITY and disconnects the transport layer connection. The DGA raises an alarm when Capabilities-Exchange fails.

The DGA does not support dynamic discovery of peers. Only manual configuration of peers is supported.

The exchange of Capabilities Exchange messages is also implemented on the DSLU interface . On startup, the DSLU connects to the DGA and sends a CER message. The DGA validates the CER message against the configuration. If the validation is successful, a CEA with Result-Code of success is returned. If the validation fails, a CEA with Result-Code indicating error is returned and the connection is closed. The DGA also raises an alarm indicating DSLU interface failure.

Disconnecting Peer Connections

When a Diameter node must disconnect one of its transport connections, it sends a Disconnect-Peer-Request message with the Disconnect-Cause AVP set to inform its peer of its intent. Upon receipt of the message, the Disconnect-Peer-Answer is returned containing an error if messages have recently been forwarded and are likely in flight (which would otherwise cause a race condition).

The receiver of the Disconnect-Peer-Answer initiates the transport disconnect.

The Disconnect-Cause AVP supports the following values:

- REBOOTING – A scheduled reboot is imminent.
- BUSY – The peer's internal resources are constrained and it has determined that the transport connection needs to be closed.

- DO_NOT_WANT_TO_TALK_TO_YOU – The peer has determined that it does not see a need for the transport connection to exist, since it does not expect any messages to be exchanged in the near future.

The DEACTIVATE-PEER MML command forces a Diameter client peer disconnect from the DGA side. The DEACTIVATE-SLU MML command forces a DSLU disconnect from the DGA side.

Transport Failure Detection

The Device-Watchdog-Request (DWR) and Device-Watchdog-Answer (DWA) messages proactively detect transport failures.

The DWR is sent to a peer when no traffic has been exchanged between two peers. Either peer can send this message. The watchdog interval is controlled by a single timer (Tw) with an initial default value of 30 seconds. The DWA is sent as a response to the DWR message.

The DWR and DWA messages are also used on the DSLU interface to check for transport failures. When a transport failure to a DSLU is detected, the transport connection is closed.

Failover and Failback Procedures

If a transport failure is detected with a peer, all pending request messages are forwarded to an alternate agent, if possible. This is called failover.

The DGA maintains a pending message queue for each peer. When an answer message is received, the corresponding request is removed from the queue. The Hop-by-Hop Identifier field matches the answer with the queued request.

The DGA does not retransmit any message and depends on the client and DSLU to retransmit the request message. When a transport failure to a peer is detected, the DGA drops all messages in the pending message queue for that peer.

Multiple identical requests or answers can be received as a result of a failover. The End-to-End Identifier field in the Diameter header, along with the Origin-Host AVP, identifies duplicate messages.

The DGU periodically sends a connection request to the failed peer to reestablish the transport connection. Once a connection has been successfully established, messages are again forwarded to the peer. This is called failback.

High Availability

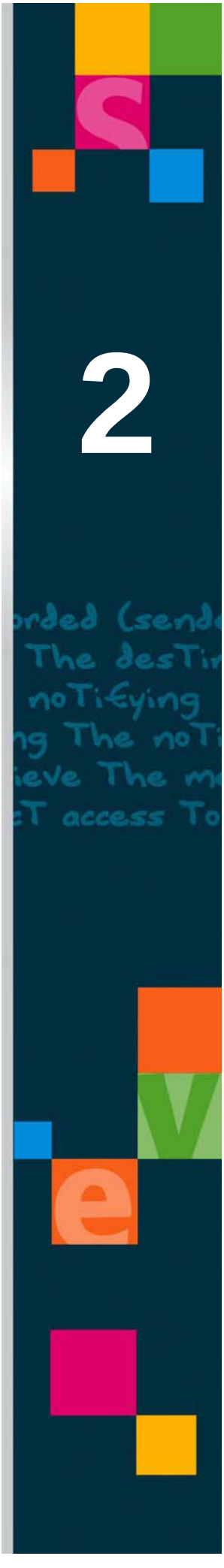
The DGU operates in a multi-CE configuration. Each DGU CE establishes a peer connection to a Diameter Client realm. For a given Diameter client, one CE is considered the primary connection and the other CE is considered the secondary connection. In the event of primary failure, the Diameter client invokes the Diameter failover procedure and sends any pending requests to the secondary. The two connections can also be configured as load sharing.

When the peer connections are configured as primary and secondary, the primary connections are distributed across the two CEs to ensure proper load balancing. On any DGU CE failure, the remaining CE handles all traffic.

The DGA process implements the OMNI automatic restart mechanism to recover from application process crashes. On application process crash, all the transport connections on this CE terminate and are reestablished immediately on application restart. The Diameter clients invoke the failover procedures for the failed connections. Once the connections are reestablished, failback procedures are invoked.

Chapter 2

DGU Hardware



Overview

This chapter describes the hardware components that support the DPM-based DGU platform, including:

- DGU Components
- Module LED Indicators and Connectors
- Module Replacement

DGU Components

DGU components are mounted in a DCP-6 or DCP-10 chassis and can include the following:

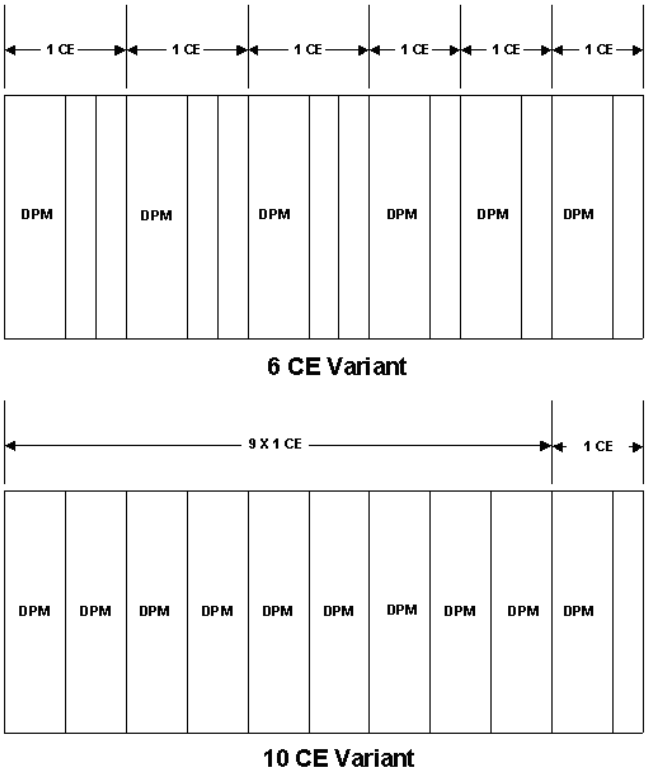
- Dual Processor Module 2 (DPM 2) and associated Rear Transition Module (DPM2-RTM)
- Dual Processor Module 3 (DPM 3) and associated Rear Transition Module (DPM3-RTM)

Chassis Configurations

The DGU DPM platform is configured in the DCP chassis. The DCP chassis implements exactly 21 slots distributed over multiple Compact PCI busses. The 21 slots on the DCP chassis are partitioned into contiguous groups and each group is associated with one PCI bus. Supporting boards include the power supply module

Figure 5, “DCP Chassis Configurations,” shows the six DPM and 10 DPM configurations. The DCP chassis implements one Compact PCI Bus for each DPM supported. No slots are allocated for hard drive bays or the power supply. The hard drive mounts directly to the DPM board. The power supply is located within the fan units below the chassis.

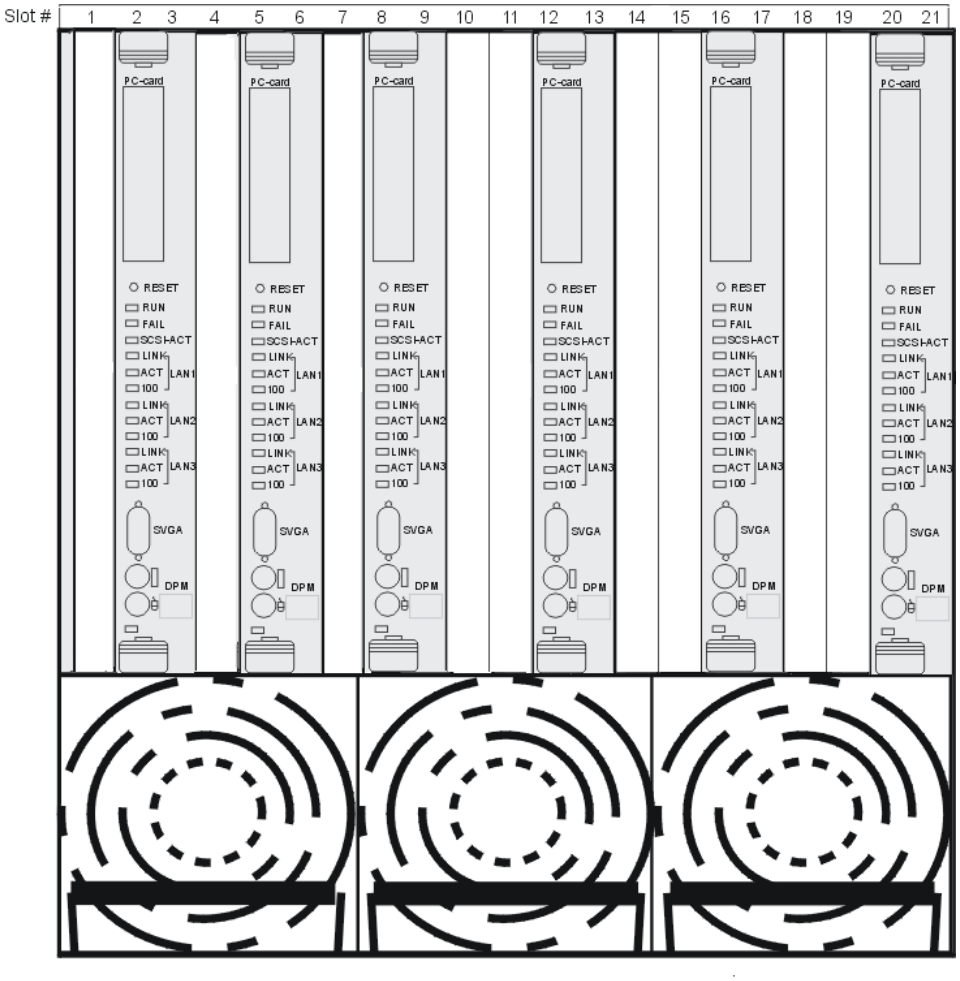
Figure 5 DCP Chassis Configurations



Internal sensors in the DCP chassis continually monitor the unit's internal temperature. The sensor output regulates the speed of the internal fans.

Figure 6, "DGU Modules in DCP-6 Chassis (Rear View)" shows a DCP-6 chassis with six DPM 2 cards.

Figure 6 DGU Modules in DCP-6 Chassis (Rear View)



Dual Processor Module Variants (DPM 2, DPM 3)

The Dual Processor Module has the following two variants:

- Dual Processor Module 2 (DPM 2)
- Dual Processor Module 3 (DPM 3)

Each DPM consists of two modules: the DPM itself and the Rear Transition Module (RTM) that supports the array of connectors required for I/0 for DPM I/O connections: SGVA, keyboard and mouse, two COM ports, three (DPM 2) or five (DPM 3) LAN connectors, USB port, and SCSI connector (DPM 2).

NOTE The DPM 2 occupies two slots in the DCP chassis. The DPM 3 occupies only a single slot.

DCP-6 Chassis

A single DCP-6 chassis shelf accommodates one dual-CE DGU configuration consisting of two DPM 2 boards. Three self-regulating fans are included in the DCP-6 chassis to cool and ventilate the platform.

Module LED Indicators and Connectors

This section describes the connectors, controls, and LED indicators on the DGU module panels.

Dual Processor Module 2 (DPM 2) Front Panel

Table 4, “DPM 2 Front Panel Controls, Connectors, and LED Indicators” describes the DPM 2 front panel RESET button, LED indicators, and connectors. Figure 7, “DPM 2 Front Panel Controls, Connectors, and LED Indicators” shows the DPM 2 front panel.

Table 4 DPM 2 Front Panel Controls, Connectors, and LED Indicators

LED	Purpose
RESET	Push the button with the tip of a pencil to reset the hardware.
RUN	Lights green when power is supplied to the drawer.
FAIL	Lights amber when a failure is detected.
SCSI-ACT	Lights green when SCSI drive is active.
SCSI-IDE	Lights green when IDE drive is active.
LINK	Light green to indicate LAN 1, 2, or 3 is up.
ACT	Activity indicator for LAN 1, 2, or 3.
100	100 Mbps indicator for LAN 1, 2, or 3.
1000	1000 Mbps indicator for LAN 1 or 2
USB	USB connector for keyboard.
HS	Hot Swap indicator; lights blue when module is safe to be swapped

Figure 7 DPM 2 Front Panel Controls, Connectors, and LED Indicators

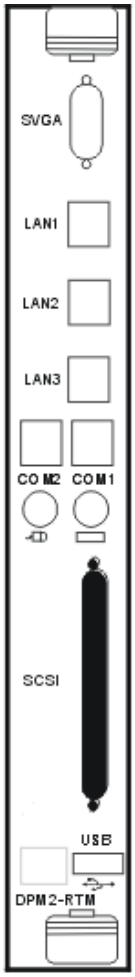
DPM 2 Rear Transition Module

Table 5, “DPM 2 Rear Transition Module (DPM2-RTM) Connectors” describes the DPM2-RTM connectors. Figure 8, “DPM 2 Rear Transition Module (DPM2-RTM) Connectors” shows these connectors.

Table 5 DPM 2 Rear Transition Module (DPM2-RTM) Connectors

Connector	Purpose
SVGA	Connector for monitor
LAN1-LAN3	LAN connectors 1, 2, and 3 (RJ-45)
COM1-COM2	Two serial port connectors
Keyboard	Connector for keyboard
Mouse	Connector for mouse
SCSI	68 pin SCSI connector for external disk drive
USB	USB port

Figure 8 DPM 2 Rear Transition Module (DPM2-RTM) Connectors



DPM 3 Front Panel

Table 6, “DPM 3 Front Panel Controls, Connectors, and LED Indicators” describes the DPM 3 front panel RESET button, LED indicators, and connectors. Figure 9, “Dual Processor Module 3 (DPM 3) Front Panel” shows the DPM 3 front panel.

Table 6 DPM 3 Front Panel Controls, Connectors, and LED Indicators

LED	Purpose
RESET	Push the button with the tip of a pencil to reset the hardware.
RUN	Lights green when power is supplied to the drawer.
FAIL	Lights amber when a failure is detected.
ACT / LINK	Lights green when floppy or CD drive is active.
HDD	Lights green when hard disk drive drive is active.
LAN 1-5 ACT / LINK	Activity indicator / Light green to indicate LAN 1-5 is up
LAN 1-5 100 / 1000	100 / 1000 Mbps indicator for LAN 1 - 5.
USB	USB connector for keyboard.

Figure 9 Dual Processor Module 3 (DPM 3) Front Panel

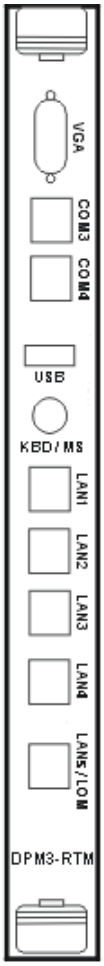
DPM 3 Rear Transition Module

Table 7, “DPM 3 Rear Transition Module (DPM3-RTM) Connectors” describes the DPM3-RTM connectors. Figure 10, “DPM 3 Rear Transition Module (DPM3-RTM) Connectors” shows these connectors.

Table 7 DPM 3 Rear Transition Module (DPM3-RTM) Connectors

Connector	Purpose
VGA	Connector for monitor
COM3-COM4	Two serial port connectors
KBD / MS	Connector for keyboard or mouse
USB	USB port
LAN1-LAN4	LAN connectors 1 - 4 (RJ-45)
LAN5 / LOM	LAN 5 and Lights Out Monitor Connector (RJ-45)

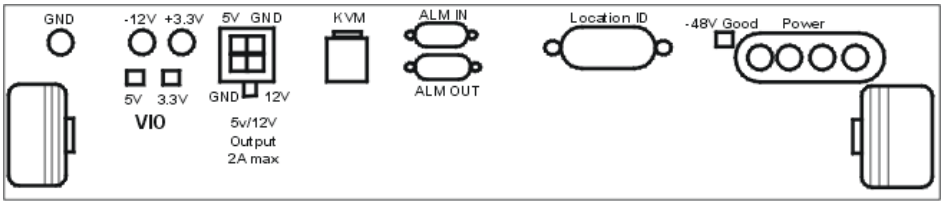
Figure 10 DPM 3 Rear Transition Module (DPM3-RTM) Connectors



Power Supply Module

The power supply module is located at the lower portion of the DCP-6 chassis behind the fan units. Typically, there is a main power supply (A) and a backup (B). The power supply panel is accessible from the rear of the chassis. It contains interface connectors, power supply jacks, and LED voltage status indicators. Figure 11, “DGU Power Supply Panel” shows the power supply panel and its connectors.

Figure 11 DGU Power Supply Panel



Power Supply Panel Jacks and LED Indicators

Table 8, “Power Supply Rear Panel Connectors and LEDs” describes the power supply rear panel connectors and LED indicators.

Table 8 Power Supply Rear Panel Connectors and LEDs

LED	Purpose
GND	Chassis ground jack
-12	-12V test jack
+3.3V	+3.3V test jack
+12V	+12V output jack
+5V	+5V output jack
5V LED	5V LED Indicator
3.3V LED	3.3V LED Indicator
KVM	Mini-Console Bus connector.
ALM IN	Alarm circuit input connector (for customer use)
ALM OUT	Alarm circuit output connector (for customer use)
Location ID	KVB Location ID jack (for thumbwheel card)
-48V Good LED	-48V LED indicator; lights green to indicate -48V input power is available and in the correct input range
POWER	Input power connector

Table 9, “PSM Front Panel LED Indicators”describes the Power Supply Module front panel LED indicators. The power supply LEDs are located on the front panel of each of the three fan assemblies and indicate the status of the main (A) and backup (B) power supplies.

Table 9 PSM Front Panel LED Indicators

LED	Purpose
-48VA[1]	Lights green to indicate -48V input power is available and in the correct input range.
-48VB[1]	Lights green to indicate -48V input power is available and in the correct input range.
RUN[2]	Lights green to indicate that power is supplied to the unit and no faults have been detected.
FAIL[2]	Lights amber when a PSM failure is detected.

1. A: designates power supply on the right side of the chassis (viewed from the rear). B: designates power supply on the left side of the chassis (viewed from the rear).
2. Fail (Amber) and Run (Green) LEDs are ON for 1 second after power turn on to indicate proper status of the LEDs and indicated PSM status.

Module Replacement

The DPM modules are secured in the DCP shelf by screws and latches. Perform the following procedure to replace modules in the DGU shelf:

There is an Electrical Shock Hazard when servicing this system. Switch off the power before opening system cabinets or attempting to remove or adjust boards, components, or other electrical subassemblies.

These modules contain Electrostatic Discharge (ESD) sensitive components. Improper handling can damage components. When working with modules, always attach an ESD wrist strap.

Do not remove a module from its protective wrapper from the module drawer unless you have taken ESD precautions. When installing or removing modules, place them component-side up, on a grounded, static-free surface. Put the modules on a special ESD foam pad if available. Do not slide the modules over a surface of any kind.

During disassembly procedures, be careful to retain all hardware for later use during reassembly procedures.

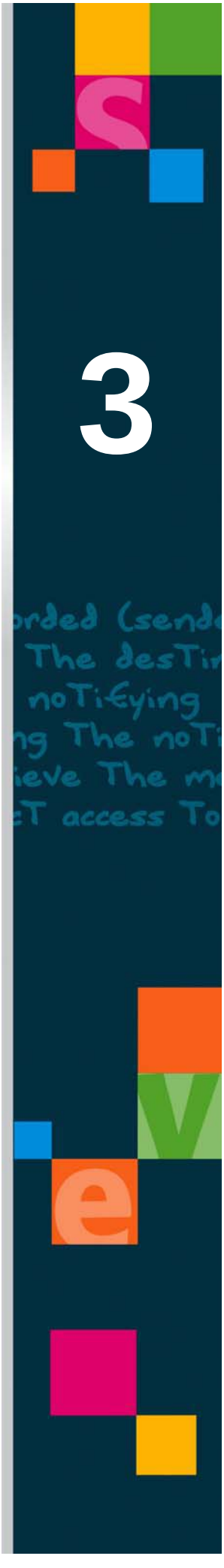
Before installing a disk or board or other electrical component, always verify that required jumpers and DIP switches are properly set. Refer to the following sections for relevant information.

To replace a faulty DGU board in the DCP shelf:

1. Perform an orderly shutdown of the operating system.
2. Turn off the appropriate switch on the distribution panel.
3. Unscrew the screws on the top and bottom latches that secure the module to the shelf frame.
4. Unlatch the top and bottom handles by pushing in on the red buttons and then pulling on the latches.
5. Gently pull the board out of the chassis shelf.
6. Insert the replacement board.
7. Push the card back into the shelf, then press the two top and bottom latches to secure the card in place.
8. Turn on the appropriate switch on the distribution panel.

Chapter 3

DGU Configuration



Overview

The DGU and the associated SLUs are configured using MML commands saved in configuration files. Some MML commands are also issued when the system is up to add, delete, change the configuration, display system information, shut down the system, and so on.

The DGA process configuration is specified in the form of MML commands stored in the db.DGA.206 configuration file in the /home/omni/conf directory. At startup, the DGA process reads this configuration file followed by a recent change file. If there are errors in the configuration file, the process terminates with an error message.

After the DGA process is running, only activate, deactivate, shutdown, and display commands are issued. Any activate and deactivate commands issued are saved by the DGA process in the rc.DGA.206 recent change file.

There are no user-configurable command line flags or environment variables.

Configuration Files

The DGA process uses the following configuration files:

- **db.DGA.206:** Configuration file in /home/omni/conf directory. This file is created by the user and is read at system startup. The file contains a sequence of MML commands.
- **rc.DGA.206:** Configuration file in /home/omni/conf director. This file is created by the DGA process and contains only activate and deactivate MML commands.

Provisioning Hierarchy

The DGU is configured in a hierarchical manner as follows:

1. The peer connections are configured.
2. The network realms are configured.
3. The network service applications are configured for the realms.
4. The peer connections are added to the network service applications.
5. The SLU connections are configured.
6. The SLU realms are configured.
7. The SLU service applications are configured for the realms.
8. The SLU connections are added to the SLU service applications.
9. The SLU connections are activated.
10. The peer connections are activated.

Timers

Table 10 DGU Timer Values

Timer	Description	Default Value	Allowed Range
Tw	Watchdog timer on the Client interface	<TBD>	<TBD>
Tcc	Connection control timer on the Client interface	<TBD>	<TBD>
Tdpr	DPR message timeout value	<TBD>	<TBD>
Tws	Watchdog timer on the SLU interface	30 seconds	6 – 60 seconds
Tas	Answer timer on the SLU interface	10 seconds	6 – 60 seconds

Optional Parameters

Table 11 Optional DGU Parameters

Parameter	Description	Default	Allowed Values
LISTEN-PORT-FOR-SLU	Listen port for SLU connections	10101	Any valid value
MEAS-INTERVAL	Measurement collection interval.	30 minutes	5, 10, 15 or 30 minutes
DISCARD-ON-CONGESTION	Discard initial requests from Client when all SLUs are congested	FALSE	TRUE or FALSE

DGU Configuration

Before configuring the DGU, you must obtain the DGU IP address, gateway IP address, netmask, and domain name for the specific system from the system administrator.

Host name dgu1 and values used in the following examples are for illustration purposes only.

Prerequisites for DGU Configuration

1. In directory /home/omni/conf, copy file hosts.DGU to hosts.<hostname> where hostname is the name of the DGU being configured. For example, if the DGU hostname is dgu1, issue the following command:

   ```
   cp hosts.DGU hosts.dgu1
   ```

   Open hosts.dgu1 and edit the last octet of the CE1 IP address to match the value of the last octet of the IP address for dgu1. For example, if the IP address for dgu1 is 10.230.16.140, edit the first line in the DGU section to read:

   ```
   <HSBN_CLASS_C_NETID>.140 <CE1_HOST> <CE1_HOST2> <CE1_HOST>.<UNIT_DOMAIN>
   ```

The HSBN primary network must be configured as eth0 and entered as HSBN_CLASS_C_NETID in the dguPlatform file. The SS7 LAN network must be configured as eth1 and entered as SS7LAN1_CLASS_C_NETID in the dguPlatform file.

2. In directory /home/omni/ipf/conf, copy file dguPlatform to file dguPlatform.<hostname> where hostname is the short host name (without domain) of the DGU being configured. From the above example, using hostname dgu1, do the following:

   ```
   cp dguPlatform dguPlatform.dgu1
   ```

   Since, in this example, dgu1's IP address is 10.230.16.140, edit the following lines in the newly created file dguPlatform.dgu1 to read as follows:

   ```
   HSBN_CLASS_C_NETID="10.230.16"     # HSBN primary network (eth0)
   HSBN_NETMASK="255.255.248.0"   # HSBN netmask
   GATEWAY="10.230.23.254"        # HSBN gateway
   SS7LAN1_CLASS_C_NETID="192.9.202"  # first SS7 LAN network (eth1)
   SS7LAN1_NETMASK="255.255.248.0"    # netmask of SS7 LAN 1
   CE1_NAME=dgu1              # unix name of CE1
   ```

```
CE1_BOARDS=0          # number of communication boards [0-4]
CE2_BOARDS=0          # number of communication boards [0-4]
UNIT_DOMAIN="comverse.com"      # domain name
```

HSBN_NETMASK, HSBN_GATEWAY and SS7LAN1_NETMASK and UNIT_DOMAIN values in this example are for illustration only. Obtain values for your specific network from your network administrator.

Configuring Transport Services and Ports in /etc/services File

DGU supports the TCP/IP transport service only. Services are defined as either a service name or port number. For each service, an entry must exist in the /etc/services file listing the service name, port number, and transport protocol.

For example, if the DGU requires a TCP service called diameter on port 3868 then /etc/services file must contain the line:

```
diameter 3868/tcp
```

Configuring IPsec

IP security (IPsec) is a standardized framework for securing Internet Protocol. It is an optional feature of DGU for IP communication between DGU and peer. This feature requires the peer to support and be configured for IPsec.

IPsec is configured to connect the DGU and peers using a host-to-host connection. A host-to-host connection uses the network to which each host is connected to create the secure tunnel to each other. The configureDGU script is read from **/home/omni/conf/ipsec.conf** file and based on the content of the file, configures one or more host-to-host IPSec tunnels to the destination host.

The following two steps must be completed before running configureDGU for IPsec:

1. In /home/omni/ipf/conf/dguPlatform.<dgu> edit line CONF_IPSEC = yes.
2. Create the file /home/omni/conf/ipsec.conf using a text editor. The file must have one line per IPSEC tunnel. The parameters within the line are delimited by tab or space(s). The line contains the following fields:

   ```
   <Host> <Host domain name> <Host IP address> <IKE_METHOD> <IKE_PSK>
   ```

   Where:

   Host: Destination host name, to which IPsec channel has to be created.

   Host domain name: Destination domain name.

   Host IP address: IP address of the destination host.

   IKE_METHOD: A pre-shared authentication key used to initiate the connection and exchange encryption keys during the session.

   IKE_PSK: A fixed encryption key, this value must be identical on both the machines to create IPsec tunnel between the hosts.

For example, using dgu2 and dgu5 as the destination, **/home/omni/conf/ipsec.conf** has entries such as:

dgu2 dgu2.comverse-in.com 10.230.18.54 PSK "abcdefg"

dgu5 dgu5.comverse-in.com 10.230.18.77 PSK “dgu1-dgu2”

Running the DGU Configuration Program (configureDGU)

Running configureDGU successfully requires that the user log in as superuser using the su – command (su minus). Using su – changes the environment to the root environment, which is required for successful configuration. Using the su command without the minus inherits the environment from the previous login.

Before executing the configureDGU, verify that OMNI is not running. If OMNI is running, by issue the following command as superuser to terminate OMNI :

```
/sbin/service omni stop
```

Occasionally, the command /sbin/service omni stop, does not terminate the port_daemon task and causes configureDGU to fail, even though the message configureDGU completed OK displays.

Before running configureDGU, issue the following command to verify that the port_daemon is not running:

```
ps  -ef  | grep port_daemon
```

If port_daemon appears in output, issue the following command to terminate the task manually:

```
kill -9 <PID of port_daemon>
```

In directory /home/omni/ipf/conf, run configureDGU <hostname> where hostname is the name of the DGU. For example, if the name of the DGU is dgu1, issue the following commands:

```
cd /home/omni/ipf/conf
./configureDGU dgu1
```

DGU Provisioning Files – Description and Examples

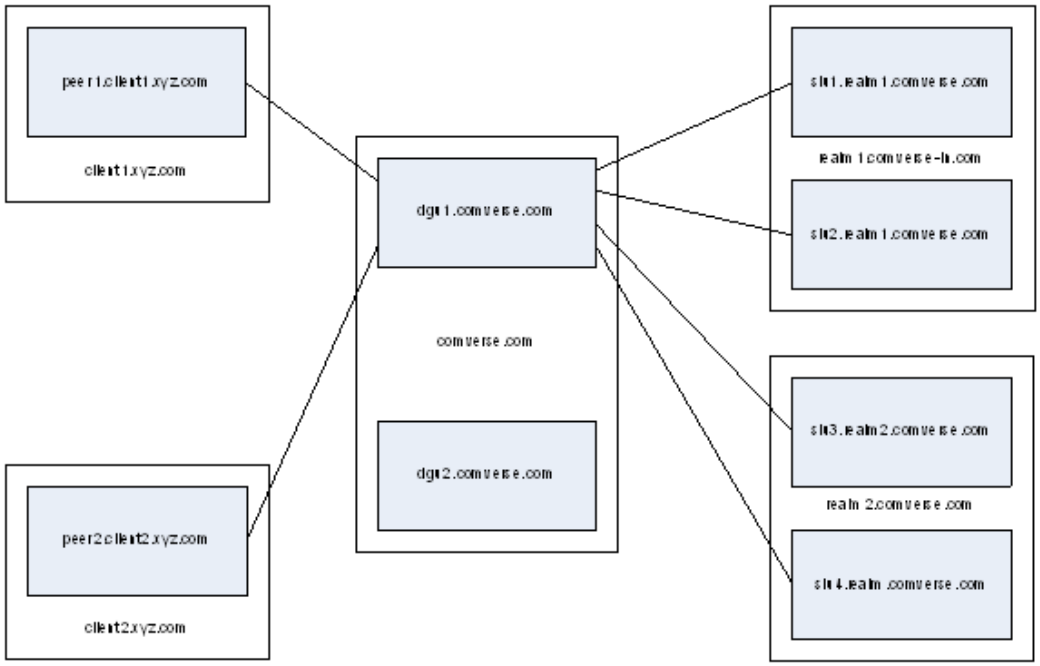
The DGU task, called dga, is located in directory /home/omni/bin. This task is provisioned through the primary configuration file **db.DGU.dga.206** located in directory /home/omni/conf. This file contains MML commands that define the basic elements of the system (Peers, SLUs, DGUs, and services) and is read by the dga task at startup.

In addition, a secondary configuration file, **rc.DGU.dga.206,** is located in directory /home/omni/conf. This is the recent change file that contains certain dynamic MML commands issued via OMNI Debugger (OMD) and termhandler utility while the dga task is up and running. These commands deal with the activation and deactivation of Peers and SLUs. The file, if present, is also read by the dga task at startup.

The MML configuration file db.DGU.dga.206 defines the Peers, SLUs, and DGUs of the system and the characteristics of these entities.

The following sections describe in detail each MML command used to provision this system, and refer to Figure 12, “Sample DGU Configuration,” and the sample db.DGU.dga.206 configuration file that follows. Figure 12 shows a sample DGU configuration with two peers (PEER1, PEER2), one DGU (DGU1), and two SLUs (SLU1, SLU2).

Figure 12 Sample DGU Configuration




```
DGU-SET-LOCAL-REALM: REALM-NAME=”comverse.com”;
DGU-CREATE-REALM: REALM-NAME="client1.xyz.com";
DGU-CREATE-PEER: PEER-NAME="PEER1",
    PEER-FQDN="peer1.client1.xyz.com", LOCAL-FQDN="dgu1.comverse.com",
    PEER-PORT=”diameter”, LOCAL-PORT=”diameter”,
    TRANSPORT=TCP, SECURITY=IPSEC, CONNECTION=SERVER;
DGU-CREATE-SERVICE: SERVICE-NAME="IMS1",
    REALM-NAME="client1.xyz.com", APPLICATION-ID=4, TRAFFIC-MODE=PRIMARY_SECONDARY;
DGU-ADD-PEER-SERVICE:SERVICE-NAME="IMS1",
    PEER-NAME="PEER1", CLUSTER-ID=0, TRAFFIC-ROLE=PRIMARY;
DGU-CREATE-REALM:REALM-NAME="client2.xyz.com";
DGU-CREATE-PEER: PEER-NAME="PEER2,
    PEER-FQDN="peer2.client2.xyz.com", LOCAL-FQDN="dgu1.comverse.com",
    PEER-PORT=”diameter”, LOCAL-PORT=”diameter”,
    TRANSPORT=TCP, SECURITY=IPSEC, CONNECTION=SERVER;
DGU-CREATE-SERVICE: SERVICE-NAME="IMS2",
    REALM-NAME="client2.xyz.com", APPLICATION-ID=4, TRAFFIC-MODE=PRIMARY_SECONDARY;
DGU-ADD-PEER-SERVICE:SERVICE-NAME="IMS2",
    PEER-NAME="PEER2", CLUSTER-ID=0, TRAFFIC-ROLE=PRIMARY;
DGU-CREATE-SLU-REALM: REALM-NAME="realm1.comverse.com";
DGU-CREATE-SLU: SLU-NAME="SLU1", SLU-FQDN="slu1.realm1.comverse.com", SLU-PORT=”d_
ocs”;
DGU-CREATE-SLU: SLU-NAME="SLU2", SLU-FQDN="slu2.realm1.comverse.com", SLU-PORT=”d_
ocs”;
DGU-CREATE-SLU-SERVICE: SERVICE-NAME="PREPAID1", REALM-NAME="realm1.comverse.com",
APPLICATION-ID=4;
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID1", SLU-NAME="SLU1";
```

```
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID1", SLU-NAME=”SLU2";
DGU-CREATE-SLU-REALM: REALM-NAME="realm2.comverse.com";
DGU-CREATE-SLU: SLU-NAME="SLU3", SLU-FQDN="slu3.realm2.comverse.com", SLU-PORT=”d_
ocs”;
DGU-CREATE-SLU: SLU-NAME="SLU4", SLU-FQDN="slu4.realm2.comverse.com", SLU-PORT=”d_
ocs”;
DGU-CREATE-SLU-SERVICE: SERVICE-NAME="PREPAID2", REALM-NAME="realm2.comverse.com",
APPLICATION-ID=4;
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID2", SLU-NAME="SLU3";
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID2", SLU-NAME="SLU4";
DGU-ACTIVATE-PEER: PEER-NAME="PEER1";
DGU-ACTIVATE-PEER: PEER-NAME="PEER2";
```

Mobile Number Portability in OCS

OCS supports finding PortedNumber via External Interface. Now the PortedNumber can also be found by Internal Interface. “ActualDestinationNumber” or “ActualOriginatingNumber” must be set as PortedNumber and calculated using Internal Interfaced in order to be dumped into UsageRecord.

MML Commands

This section presents the Man Machine Language (MML) commands to configure the DGU. Each section in this chapter describes the actions a command performs and contains the name of the command, abbreviation, the syntax of the command, including alternative abbreviated formats, examples, and the name of the man page for the command and for other related topics.

MML commands take the form:

```
COMMAND-NAME:PARAM1-NAME=<value1>, PARAM2-NAME=<value2>...;
```

MML commands observe the following rules:

- **Command Name**: Uppercase letters, must be followed by colon (:).
- **Parameter Name**: All uppercase letters followed by an equal sign (=). Parameters are separated by commas (,).
- **End of Command**: Commands are always terminated by a semicolon (;).

In addition to the rules for the commands, command syntax in this chapter is shown using the following conventions:

- **Values to be supplied**: Surrounded by angle brackets (< and >).
- **Optional parameters**: Surrounded by square brackets ([and]).
- **Parameter choice or value choice**: Surrounded by curly brackets ({ and }) and separated by a vertical bar (|).
- **Actual command names**: Shown in the first line of the section. Any abbreviations are shown in parentheses following the command name.
- **Abbreviations for parameters**: Shown in parentheses following the parameter name.

Static MML Commands

Static MML commands are contained in the primary DGU configuration file db.DGU.dga.206 and are read at dga startup.

DGU-SET-LOCAL-REALM

Defines the local DGU realm. Must be issued before all other MML commands in the DGU configuration file.

Syntax:

```
DGU-SET-LOCAL-REALM: REALM-NAME = “RealmName”;
```

where *RealmName* is the DNS domain name in which the DGU resides. In the example, dga1 resides in domain comverse.*com.* The command needed to set the domain is:

```
DGU-SET-LOCAL-REALM: REALM-NAME = “comverse.com”;
```

DGU-CREATE-REALM

Creates a remote peer realm. A realm contains services, created by DGU-CREATE-SERVICE. A realm is a domain name as found in DNS.

Syntax:

```
DGU-CREATE-REALM: REALM-NAME = “RealmName”;
```

where *RealmName* is the DNS domain name in which the peer resides.

In the example, PEER1 resides in domain *client1.xyz.com*, and PEER2 resides in domain *client2.xyz.com.* The commands needed to create the required realms are:

```
DGU-CREATE-REALM: REALM-NAME = “client1.xyz.com”;
```

and

```
DGU-CREATE-REALM: REALM-NAME = “client2.xyz.com”;
```

DGU-CREATE-PEER

Creates a Diameter peer, which is a direct neighbor of the DGU and is sometimes called a Client.

Syntax:

```
DGU-CREATE-PEER:
    PEER-NAME ="name",
    PEER-FQDN ="pfqdn",
    LOCAL-FQDN="lfqdn",
    PEER-PORT = {pport|"pservices"},
    LOCAL-PORT = {lport|"lservices"},
    TRANSPORT = {SCTP|TCP},
    SECURITY = {IPSEC|TLS},
    CONNECTION = {CLIENT|SERVER};
```

where:

- *name* is the logical name that identifies the peer
- *pfqdn* is the fully qualified domain name of peer (peer host name + peer domain name)
- *lfqdn* is the fully qualified domain name of the DGU (dgu host name + dgu domain name)
- *pport* is the peer port (typically 3868).
- *pservice* is the service name associated with port 3868 (only “diameter” currently defined)
- *lport is the local port* (typically to 3868)
- *lservice* is the service name associated with port 3868 (only “diameter” currently defined)
- *SCTP or TCP* is transport protocol used (TCP or SCTP)

- *IPSEC or TLS* is the security mechanism used (IPSEC or TLS)
- *CLIENT or SERVER* is the role of the DGU. (Always SERVER)

In the example, there are two peers, PEER1 and PEER2, which communicate with DGU dgu1. Currently defined value for peer and local ports is service name “diameter” (mapped to port 3868). The transport protocol is TCP. Security is IPSEC and the local role is SERVER. The commands needed to create the required peers are:

```
DGU-CREATE-PEER:
    PEER-NAME="PEER1",
    PEER-FQDN="peer1.client1.xyz.com",
    LOCAL-FQDN="dgu1.comverse.com",
        PEER-PORT=”diameter”,
        LOCAL-PORT=”diameter”,
        TRANSPORT=TCP,
        SECURITY=IPSEC,
        CONNECTION=SERVER;

         and

        DGU-CREATE-PEER:
        PEER-NAME="PEER2,
        PEER-FQDN="peer2.client2.xyz.com",
        LOCAL-FQDN="dgu1.comverse.com",
        PEER-PORT=”diameter”,
        LOCAL-PORT=”diameter”,
        TRANSPORT=TCP,
        SECURITY=IPSEC,
        CONNECTION=SERVER;
```

If a service name or port other than *diameter* or 3868 is required, it must be defined in file /etc/services along with a protocol identifier. If, for example, a service name *newservice* using TCP port 3456 is required, then the following line must be added to file /etc/services

```
newservice        3456/tcp        # dew peer service name and dedicated port
```

DGU-CREATE-SERVICE

Creates a service with the given realm (created by DGU-CREATE-REALM). The service must be assigned an application ID and a traffic mode. Currently, only PRIMARY_SECONDARY is supported for TRAFFIC-MODE.

Prerequisite: DGU-CREATE-REALM must be done before DGU-CREATE-SERVICE.

Syntax:

```
DGU-CREATE-SERVICE:
    SERVICE-NAME = "sname",
    REALM-NAME = "rname",
    APPLICATION-ID = id,
    TRAFFIC-MODE = {PRIMARY_SECONDARY|LOAD_SHARE};
```

where:

sname is the logical name of the service.

rname is the DNS domain name for the realm where service resides

id is the numeric application identifier (Always 4 for Credit Control Application)

PRIMARY_SECONDARY or LOAD_SHARE is the traffic mode for the service

In the example, there are two credit control application services, IMS1 for realm client1.xyz.com and IMS2 for realm client2.xyz.com. The commands needed to create these services are:

```
DGU-CREATE-SERVICE:
  SERVICE-NAME="IMS1",
  REALM-NAME="client1.xyz.com",
  APPLICATION-ID=4,
  TRAFFIC-MODE=PRIMARY_SECONDARY;
 and

DGU-CREATE-SERVICE:
  SERVICE-NAME="IMS2",
  REALM-NAME="client2.xyz.com",
  APPLICATION-ID=4,
  TRAFFIC-MODE=PRIMARY_SECONDARY;
```

DGU-ADD-PEER-SERVICE

Adds a peer to the given service. The peer has been created by DGU-CREATE-PEER. The service has been created by DGU-CREATE-SERVICE. A service can have more than one peer. In addition, a peer can be associated with more than one service. When the peer is associated (or mapped) with the service, it must be assigned a traffic role. Currently CLUSTER-ID is not supported and defaults to 0 (zero).

Prerequisite: DGU-CREATE-SERVICE and DGU-CREATE-PEER must be done before DGU-ADD-PEER-SERVICE.

Syntax:

```
DGU-ADD-PEER-SERVICE:
    SERVICE-NAME = "sname",
    PEER-NAME = "pname",
    CLUSTER-ID = id,
    TRAFFIC-ROLE = {PRIMARY|SECONDARY};
```

where:

sname is the logical name of the service.

pname is the logical peer name where service resides

id is the cluster ID (Not supported always 0)

PRIMARY or SECONDARY is the traffic role for the service (Always PRIMARY)

In the example, service IMS1 resides in PEER1 and IMS2 resides in PEER2. The commands needed to add the services to the appropriate peers are:

```
DGU-ADD-PEER-SERVICE:
  SERVICE-NAME="IMS1",
  PEER-NAME="PEER1",
  CLUSTER-ID=0,
  TRAFFIC-ROLE=PRIMARY;
```

and

```
DGU-ADD-PEER-SERVICE:
  SERVICE-NAME="IMS2",
  PEER-NAME="PEER2",
  CLUSTER-ID=0,
  TRAFFIC-ROLE=PRIMARY;
```

DGU-ACTIVATE-PEER

Enables the TCP/SCTP connection to existing peer to be established.

Prerequisite: DGU-CREATE-PEER must be done before DGU-ACTIVATE-PEER

Syntax:

```
DGU-ACTIVATE-PEER=PEER-NAME = "pname";
```

where:

pname is the logical name of an existing peer.

In the example there are two created peers PEER1 and PEER2. The commands needed to activate the peers are:

```
DGU-ACTIVATE-PEER: PEER-NAME="PEER1";
DGU-ACTIVATE-PEER: PEER-NAME="PEER2";
```

DGU-CREATE-SLU-REALM

Creates a realm on the SLU side. A realm contains services, created later by DGU-CREATE-SLU-SERVICE.

Syntax:

```
CREATE-SLU-REALM:REALM-NAME="rname";
```

where *rname* is the DNS domain name for the SLU realm.

In the example, SLU1 and SLU2 reside in domain *realm1.comverse.com* and SLU3 and SLU4 reside in domain **realm2.comverse.com**. The commands needed to create the required SLU realms are:

```
DGU-CREATE-SLU-REALM: REALM-NAME="realm1.comverse.com";
      and
DGU-CREATE-SLU-REALM: REALM-NAME="realm2.comverse.com";
```

DGU-CREATE-SLU

Creates an SLU in an SLU realm .

Syntax:

```
DGU-CREATE-SLU:
   SLU-NAME = "name",
   SLU-FQDN = "fqdn",
   SLU-PORT = {por t|"services"};
```

where:

- *name* is the logical SLU name
- *fqdn* is the SLUs fully qualified domain name (host name + domain name)
- *port* is the remote SLU port (typically 10410)
- *service* is the service name associated with port 10410 (only “d_ocs” currently defined)

In the example, SLU1 and SLU2 reside in domain *realm1.comverse.com* and SLU3 and SLU4 reside in domain *realm2.comverse.com.* All SLUs communicate using service name *d_ocs* (port 10410). The commands needed to create the required SLUs are

```
DGU-CREATE-SLU: SLU-NAME="SLU1",
SLU-FQDN="slu1.realm1.comverse.com",
SLU-PORT=”d_ocs”;

DGU-CREATE-SLU: SLU-NAME="SLU2",
SLU-FQDN="slu2.realm1.comverse.com",
SLU-PORT=“d_ocs”;

DGU-CREATE-SLU: SLU-NAME="SLU3",
SLU-FQDN="slu3.realm2.comverse.com",
SLU-PORT=“d_ocs”;

        DGU-CREATE-SLU: SLU-NAME="SLU4",
SLU-FQDN="slu4.realm2.comverse.com",
SLU-PORT=“d_ocs”;
```

If a service name or port other than *d_ocs* or 10410 is required, it must be defined in file /etc/services along with a protocol identifier. If, for example, a service name *newservice* using TCP port 23456 is required, then the following line must be added to file /etc/services

```
newservice      23456/tcp       # new slu service name and dedicated port
```

DGU-CREATE-SLU-SERVICE

Creates a service within an existing SLU realm. The service must be assigned an application ID.

> **Prerequisite**: DGU-CREATE-SLU-REALM must be done before DGU-CREATE-SLU-SERVICE.

Syntax:

```
DGU-CREATE-SLU-SERVICE:
   SERVICE-NAME = "sname",
   REALM-NAME = "rname",
   APPLICATION-ID = id;
```

where:

sname is the logical service name

rname is the DNS domain name of the SLU realm where service resides

id is numeric application identifier (Always 4 for Credit Control Application)

In the example, there are two credit control application services, PREPAID1 for realm *realm1.comverse.com* and PREPAID2 for realm *realm2.comverse.com*. The commands needed to create these services are:

```
DGU-CREATE-SLU-SERVICE:
  SERVICE-NAME="PREPAID1",
  REALM-NAME="realm1.comverse.com",
  APPLICATION-ID=4;
```

and:

```
DGU-CREATE-SLU-SERVICE:
  SERVICE-NAME="PREPAID2",
  REALM-NAME="realm2.comverse.com",
  APPLICATION-ID=4;
```

DGU-ADD-SLU-SERVICE

Adds an SLU to a given service. The SLU is created by DGU-CREATE-SLU. A service can have more than one SLU. In addition, an SLU can be associated with more than one service.

Prerequisite: DGU-CREATE-SLU-SERVICE must be done before DGU-ADD-SLU-SERVICE.

Syntax:

```
DGU-ADD-SLU-SERVICE:
   SERVICE-NAME="sname",
   SLU-NAME="sluname";
```

where:

sname is logical name of service

sluname is logical name of SLU

In the example, service PREPAID1 resides in SLU1 and SLU2 and service PREPAID2 resides in SLU3 and SLU4. The commands needed to add the services to the appropriate SLUs are:

```
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID1", SLU-NAME="SLU1";
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID1", SLU-NAME="SLU2";
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID2", SLU-NAME="SLU3";
DGU-ADD-SLU-SERVICE: SERVICE-NAME="PREPAID2", SLU-NAME="SLU4";
```

Dynamic MML Commands

The following MML commands are issued dynamically via OMD or termhandler utilities while the DGU is running and operational. The commands are read by the dga task.

Certain dynamic MML commands are persistent and saved in the secondary configuration file rc.DGU.dga.206 to be read by the dga task at subsequent startups. Persistent MML commands are noted in the descriptions below.

DGU-ACTIVATE-PEER

Enables the TCP/SCTP connection to an existing peer to be established. It can also be included in the primary MML configuration file db.DGU.dga.206. When issued dynamically, the command is persistent and saved in the secondary configuration file rc.DGU.dga.206

Prerequisite: Peer must exist before issuing DGU-ACTIVATE-PEER

Syntax:

```
DGU-ACTIVATE-PEER=PEER-NAME = "pname";
```

where:

pname is the logical name of an existing peer.

An example of this command is:

```
DGU-ACTIVATE-PEER: PEER-NAME = “PEER1”;
```

This command is persistent and saved in the secondary configuration file rc.DGU.dga.206.

DGU-DEACTIVATE-PEER

Tears down the TCP/SCTP connection to the peer. A Disconnect Peer Request (DPR) message is sent to the peer and the connection is disconnected after receiving the Disconnect Peer Answer (DPA). The disconnect cause is set to the value specified by CAUSE.

Prerequisite: Peer must exist before issuing DGU-DEACTIVATE-PEER.

Syntax:

```
DGU-DEACTIVATE-PEER:
    PEER-NAME = "pname”,
    CAUSE = {BUSY| DO_NOT_TALK};
```

where:

- *pname* is logical name of peer
- *BUSY or DO_NOT_TALK* is the disconnect cause value set in DPR message.
- *BUSY* indicates peer’s resources are low and connection needs to be closed.
- *DO_NOT_TALK* indicates peer does not have need for peer connection to exist.

An example of this command is:

```
DGU-DEACTIVATE-PEER:PEER-NAME = “PEER1”, CAUSE=DO_NOT_TALK;
```

This command is persistent and saved in the secondary configuration file rc.DGU.dga.206.

DGU-ACTIVATE-SLU

Enables the TCP/SCTP connection to an existing SLU peer to be established. Although by default, all SLU are active at DGU startup, command can be included in the primary MML configuration file db.DGU.dga.206. When issued dynamically, the command is persistent and saved in the secondary configuration file rc.DGU.dga.206

Prerequisite: SLU must exist before issuing DGU-ACTIVATE-SLU.

Syntax:

```
DGU-ACTIVATE-SLU:SLU-NAME = "sname";
```

where:

s*name* is the logical name of an existing SLU.

An example of this command is:

```
DGU-ACTIVATE-SLU:SLU-NAME = “SLU4”;
```

This command is persistent and saved in the secondary configuration file rc.DGU.dga.206.

DGU-DEACTIVATE-SLU

Tears down the TCP/SCTP connection to the SLU. A DPR message is sent to the SLU and the connection is disconnected after receiving the DPA. The disconnect cause is set to the value specified by CAUSE

Prerequisite: SLU must exist before issuing DGU-DEACTIVATE-SLU

Syntax:

```
DGU-DEACTIVATE-SLU:
    PEER-NAME = "sname",
    CAUSE = {BUSY| DO_NOT_TALK};
```

where:

- *sname* is logical name of an existing SLU
- *BUSY or DO_NOT_TALK* is the disconnect cause value set in DPR message.
- *BUSY* indicates peer's resources are low and connection needs to be close.
- *DO_NOT_TALK* indicates peer does not have need for slu connection to exist.

An example of this command is:

```
DGU-DEACTIVATE-SLU:SLU-NAME = "SLU4", CAUSE=DO_NOT_TALK;
```

This command is persistent and saved in the secondary configuration file rc.DGU.dga.206.

DGU-DISPLAY-ALL

Displays the DGU information in a suitably formatted manner.

- Realm Display
  - Realm Name (repeated if multiple realms)
    - Service Name (repeated if multiple services within this realm)
      - Application Id
      - Traffic Mode
      - Peer Map (repeated if multiple peers for this service)
      - Traffic Role
        - Cluster Id
        - Peer Name
- Peer Display
  - Peer Name (repeated if multiple peers)
    - Peer FQDN
    - Peer Port
    - Local FQDN
    - Local Port
    - Transport
    - Security
    - State

Connection Establishment Method

- SLU Realm Display
  - Realm Name (repeated if multiple realms)
    - Service Name (repeated if multiple services within this realm)
      - Application Id
      - Peer Map (repeated if multiple peers for this service)
        - Cluster Id
        - SLU Name
- SLU Display

  - SLU Name (repeated if multiple SLUs)
    - SLU FQDN
    - SLU Port
    - Local FQDN
    - Local Port
    - State
- Parameter values
  - Discard on congestion
  - Measurement interval
- Timer values
  - Watchdog timer
  - Connection timer
  - Answer timer
  - SLU watchdog timer
  - SLU answer timer

Syntax:

```
DGU- DISPLAY-ALL;
```

This command is issued dynamically but is not persistent. It is not saved in the secondary configuration file rc.DGU.dga.206.

DGU-DISPLAY-MEAS

Displays the following information in a suitably formatted manner.

Syntax:

```
DGU-DISPLAY-MEAS;
```

Output to console displays the following:

Message statistics on a peer connection basis with the following information for each message:

- Message request receive count
- Message request receive discard count
- Message answer(success) transmit count
- Message answer(failure) transmit count
- Message answer transmit timeout count
- Message request transmit count
- Message request transmit reject count
- Message answer(success) receive count
- Message answer(failure) receive count
- Message answer receive timeout count

Performance statistics on a SLU connection basis:

- Request to answer delay in ms. – minimum value
- Request to answer delay in ms. – maximum value
- Request to answer delay in ms. – mean value

- Request to answer delay less than 100 ms. – count
- Request to answer delay from 100 to 200 ms. – count
- Request to answer delay from 200 to 400 ms. – count
- Request to answer delay from 400 to 1000 ms. – count
- Request to answer delay from 1 to 2 s. – count
- Request to answer delay from 2 to 4 s. – count
- Request to answer delay from 4 to 10 s. – count
- Request to answer delay greater than 10 s. – count

This command is issued dynamically but is not persistent. It is not saved in the secondary configuration file rc.DGU.dga.206.

DBG-MASK

Entered at OMD prompt. Enables or disables DGA debug information to help in diagnosis of DGU problems.

Syntax:

```
DBG-MASK: ACE-MASK= ERROR | NOTICE | INFO | DEBUG;
```

Where ERROR, NOTICE, INFO, and DEBUG are debug levels that enable progressively more debug information with ERROR being the least and DEBUG being the most.

Miscellaneous MML Commands

DGU-SHUTDOWN

Shuts down a DGU CE. Sends the DPR message on all peer and SLU connections, waits for DPA, closes the connection, and calls exit. The disconnect cause is set to REBOOTING.

Syntax:

```
DGU-SHUTDOWN;
```

This command is issued dynamically but is not persistent. It is not saved in the secondary configuration file rc.DGU.dga.206.

DGU-CHANGE-PARAM

Changes system parameter values from default values.

The three syntax variations for DGU-CHANGE-PARAM are:

```
DGU-CHANGE-PARAM: LISTEN-PORT-FOR-SLU = {port|"services"};
```

where *port* is a numeric TCP/SCTP port (typically 10410)

service is a service name associated with port 10410 (only “d_ocs” currently defined)

If a service name or port other than *d_ocs* or 10410 is required, it must be defined in file /etc/services along with a protocol identifier. If, for example, a service name *newservice* using TCP port 23456 is required, then the following line must be added to file /etc/services:

```
newservice      23456/tcp       # new slu service name and dedicated port
```

In order for the LISTEN-PORT-FOR-SLU command to take effect ,it must be issued before the DGU-CREATE-SLU command is issued.

DGU-CHANGE-PARAM:MEAS-INTERVAL={5|10|15|30};

where:

- 5, 10, 15 or 30 indicate the minute interval for reporting measurement to measurement manager.
- Default measurement interval is 30 minutes.

DGU-CHANGE-PARAM:DISCARD-ON-CONGESTION={TRUE|FALSE};

where:

- TRUE or FALSE indicates whether or not CCR messages are discarded on congestion.
- Default discard value is FALSE.
- This command can only be issued by inclusion in MML configuration file db.DGU.dga.206

DGU-CHANGE-TIMER

Changes system and Diameter timer values from default. These parameters must not be changed from default values. This command is for R&D use ONLY.

The five syntax variations for DGU-CHANGE-TIMER are:

`DGU-CHANGE-TIMER:WATCHDOG-TIMER=<value>;`

where value is time in seconds to wait for a DPA after sending DPR

`DGU-CHANGE-TIMER:CONNECTION-CONTROL-TIMER=<value>;`

where value is the time in seconds to wait for a connection response

`DGU-CHANGE-TIMER:DGU-CHANGE-TIMER:DPR-TIMEOUT=<value>;`

where value is time in seconds to wait for a DPA after sending DPR.

`DGU-CHANGE-TIMER:WATCHDOG-TIMER-SLU=<value>;`

where value is time in seconds to wait for DPA from SLU after sendif DPR.

Default value is 30 seconds.

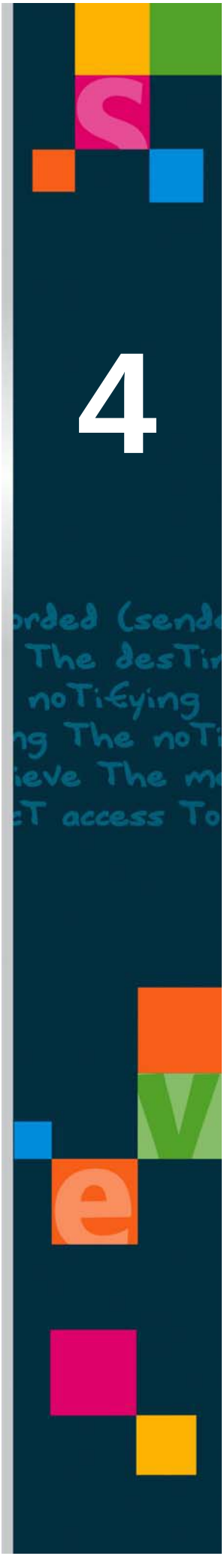
`DGU-CHANGE-TIMER:ANSWER-TIMER-SLU=<value>;`

where value is time in seconds to wait for an answer from SLU.

This command can only be issued by inclusion in MML configuration file db.DGU.dga.206

Chapter 4

DGU Operation



Starting OMNI and DGU

By default, OMNI is configured for auto-startup and requires no special action after powering the DGU on.

If necessary, auto-startup is disabled by executing the following command as root:

```
$OMNI_HOME/conf/configOmniSvc -o off
```

When auto-startup is disabled, OMNI is started by executing the following command as root:

```
service omni start
```

When auto-startup is disabled, OMNI is stopped by executing the following command as root:

```
service omni stop
```

If auto-startup is disabled, OMNI does not start up on Linux reboot.

To enable OMNI auto-startup again, execute the following command as root:

```
$OMNI_HOME/conf/configOmniSvc -o on
```

Measurements

The following measurements are in addition to the normal platform measurements.

Peer Measurements

Table 12, “Peer Measurements” lists the peer measurement counters maintained on a peer connection basis by the DGU.

Table 12 Peer Measurements

Measurement Counter	Description
MSG_RX	Messages received.
MSG_TX	Messages transmitted.
OCT_RX	Octets received.
OCT_TX	Octets transmitted.
ACR_RX	ACR received (all). Includes ACR discarded later in the process.
ACR_RX_DISCARDED	ACR received and discarded.
ACA_TX_SUCCESS	ACA sent with result-code of SUCCESS.
ACA_TX_NOT_SUCCESS	ACA sent with result-code of NOT_SUCCESS.
ACR_TX	ACR sent.
ACR_TMO	ACR timed out.
ACA_RX_DISCARDED	ACA received and discarded.
ACA_RX_SUCCESS	ACA received with result-code of SUCCESS.
ACA_RX_NOT_SUCCESS	ACA received with result-code of NOT_SUCCESS.
ASR_RX	ASR received (all). Includes ASR discarded later in the process.

Table 12 Peer Measurements (Continued)

Measurement Counter	Description
ASR_RX_DISCARDED	ASR received and discarded.
ASA_TX_SUCCESS	ASA sent with result-code of SUCCESS.
ASA_TX_NOT_SUCCESS	ASA sent with result-code of NOT_SUCCESS.
ASR_TX	ASR sent.
ASR_TMO	ASR timed out.
ASA_RX_DISCARDED	ASA received and discarded.
ASA_RX_SUCCESS	ASA received with result-code of SUCCESS.
ASA_RX_NOT_SUCCESS	ASA received with result-code of NOT_SUCCESS.
CCR_RX	CCR received (all). Includes CCR discarded later in the process.
CCR_RX_DISCARDED	CCR received and discarded.
CCA_TX_SUCCESS	CCA sent with result-code of SUCCESS.
CCA_TX_NOT_SUCCESS	CCA sent with result-code of NOT_SUCCESS.
CCR_TX	CCR sent.
CCR_TMO	CCR timed out.
CCA_RX_DISCARDED	CCA received and discarded.
CCA_RX_SUCCESS	CCA received with result-code of SUCCESS.
CCA_RX_NOT_SUCCESS	CCA received with result-code of NOT_SUCCESS.
CER_RX	CER received (all). Includes CER discarded later in the process.
CER_RX_DISCARDED	CRE received and discarded.
CEA_TX_SUCCESS	CEA sent with result-code of SUCCESS.
CEA_TX_NOT_SUCCESS	CEA sent with result-code of NOT_SUCCESS.
CER_TX	CER sent.
CER_TMO	CER timed out.
CEA_RX	CEA received (all).
CEA_RX_DISCARDED	CEA received and discarded. Missing AVP or result-code NOT_ SUCCESS.
DPR_RX	DPR received (all). Includes DPR discarded later in the process.
DPR_RX_DISCARDED	DPR received and discarded.
DPA_TX	DPA sent.
DPR_TX	DPR sent.
DPR_TMO	DPR timed out.
DPA_RX	DPA received.
DWR_RX	DWR received.
DWA_TX	DWA sent.
DWR_TX	DWR sent.
DWR_TMO	DWR timed out.
DWA_RX	DWA received.
RAR_RX	RAR received (all). Includes RAR discarded later in the process.
RAR_RX_DISCARDED	RAR received and discarded.

Table 12 Peer Measurements (Continued)

Measurement Counter	Description
RAA_TX_SUCCESS	RAA sent with result-code of SUCCESS.
RAA_TX_NOT_SUCCESS	RAA sent with result-code of NOT_SUCCESS.
RAR_TX	RAR sent.
RAR_TMO	RAR timed out.
RAA_RX_DISCARDED	RAA received and discarded.
RAA_RX_SUCCESS	RAA received with result-code of SUCCESS.
RAA_RX_NOT_SUCCESS	RAA received with result-code of NOT_SUCCESS.
STR_RX	STR received (all). Includes STA discarded later in the process.
STR_RX_DISCARDED	STR received and discarded.
STA_TX_SUCCESS	STA sent with result-code of SUCCESS.
STA_TX_NOT_SUCCESS	STA sent with result-code of NOT_SUCCESS.
STR_TX	STR sent.
STR_TMO	STR timed out.
STA_RX_DISCARDED	STA received and discarded.
STA_RX_SUCCESS	STA received with result-code of SUCCESS.
STA_RX_NOT_SUCCESS	STA received with result-code of NOT_SUCCESS.
RAR_TMO	
RAA_RX_DISCARDED	Minimum request/answer delay.
RAA_RX_SUCCESS	Maximum request/answer delay.
RAA_RX_NOT_SUCCESS	Average request/answer delay.

DSLU Measurements

Table 13, “DSLU Measurements” lists the SLU measurement counters maintained by the DGU for each SLU.

Table 13 DSLU Measurements

Measurement Counter	Description
MSG_RX	Messages received.
MSG_TX	Messages transmitted.
OCT_RX	Octets received.
OCT_TX	Octets transmitted.
ACR_RX	ACR received (all). Includes ACR discarded later in the process.
ACR_RX_DISCARDED	ACR received and discarded.
ACA_TX_SUCCESS	ACA sent with result-code of SUCCESS.
ACA_TX_NOT_SUCCESS	ACA sent with result-code of NOT_SUCCESS.
ACR_TX	ACR sent.
ACR_TMO	ACR timed out.
ACA_RX_DISCARDED	ACA received and discarded.
ACA_RX_SUCCESS	ACA received with result-code of SUCCESS.
ACA_RX_NOT_SUCCESS	ACA received with result-code of NOT_SUCCESS.
ASR_RX	ASR received (all). Includes ASR discarded later in the process.

Table 13 DSLU Measurements (Continued)

Measurement Counter	Description
ASR_RX_DISCARDED	ASR received and discarded.
ASA_TX_SUCCESS	ASA sent with result-code of SUCCESS.
ASA_TX_NOT_SUCCESS	ASA sent with result-code of NOT_SUCCESS.
ASR_TX	ASR sent.
ASR_TMO	ASR timed out.
ASA_RX_DISCARDED	ASA received and discarded.
ASA_RX_SUCCESS	ASA received with result-code of SUCCESS.
ASA_RX_NOT_SUCCESS	ASA received with result-code of NOT_SUCCESS.
CCR_RX	CCR received (all). Includes CCR discarded later in the process.
CCR_RX_DISCARDED	CCR received and discarded.
CCA_TX_SUCCESS	CCA sent with result-code of SUCCESS.
CCA_TX_NOT_SUCCESS	CCA sent with result-code of NOT_SUCCESS.
CCR_TX	CCR sent.
CCR_TMO	CCR timed out.
CCA_RX_DISCARDED	CCA received and discarded.
CCA_RX_SUCCESS	CCA received with result-code of SUCCESS.
CCA_RX_NOT_SUCCESS	CCA received with result-code of NOT_SUCCESS.
CER_RX	CER received (all). Includes CER discarded later in the process.
CER_RX_DISCARDED	CRE received and discarded.
CEA_TX_SUCCESS	CEA sent with result-code of SUCCESS.
CEA_TX_NOT_SUCCESS	CEA sent with result-code of NOT_SUCCESS.
CER_TX	CER sent.
CER_TMO	CER timed out.
CEA_RX	CEA received (all).
CEA_RX_DISCARDED	CEA received and discarded. Missing AVP or result-code NOT_SUCCESS.
DPR_RX	DPR received (all). Includes DPR discarded later in the process.
DPR_RX_DISCARDED	DPR received and discarded.
DPA_TX	DPA sent.
DPR_TX	DPR sent.
DPR_TMO	DPR timed out.
DPA_RX	DPA received.
DWR_RX	DWR received.
DWA_TX	DWA sent.
DWR_TX	DWR sent.
DWR_TMO	DWR timed out.
DWA_RX	DWA received.
RAR_RX	RAR received (all). Includes RAR discarded later in the process.
RAR_RX_DISCARDED	RAR received and discarded.
RAA_TX_SUCCESS	RAA sent with result-code of SUCCESS.
RAA_TX_NOT_SUCCESS	RAA sent with result-code of NOT_SUCCESS.
RAR_TX	RAR sent.
RAR_TMO	RAR timed out.

Table 13 DSLU Measurements (Continued)

Measurement Counter	Description
RAA_RX_DISCARDED	RAA received and discarded.
RAA_RX_SUCCESS	RAA received with result-code of SUCCESS.
RAA_RX_NOT_SUCCESS	RAA received with result-code of NOT_SUCCESS.
STR_RX	STR received (all). Includes STA discarded later in the process.
STR_RX_DISCARDED	STR received and discarded.
STA_TX_SUCCESS	STA sent with result-code of SUCCESS.

DGU Alarms

The Alarm and Event Monitoring Facility notifies the operator of faults and other events that affect the operation of the Comverse ONE solution.

Alarms are categorized into four priority levels: critical, major, minor, and info.

A critical, major, or minor alarm indicates to the operator that a corrective action is necessary. After the operator has corrected the condition, the system usually recognizes that the problem has been rectified and removes the alarm from the screen.

Alarms with an info priority level require no action. Because info alarms are numerous and often occur in bursts, they are suppressed when duplicated. Because few critical, major, and minor alarms occur, every alarm message is displayed.

Comverse has adapted X.733 standard for alarming. Many of the fault management-related requirements and guidelines in this document derive from that standard.

Alarm Severity Values

In general, an agent determines alarm severity as if the managed object it monitors is responsible for giving a service for its own, regardless of the managed object redundancy. Although many of the system components are operating in N+1 or active/stand-by redundancy mode, in case of a failure, each instance reports severity according to its own capability of supplying the service. The manager upgrades or downgrades the alarm severity when the alarm is correlated with the status and the alarm notifications from other instances of the managed object that are giving the same service.

Table 14 Alarm Severity Levels

Severity Level	Description
Critical	Immediate corrective action is required. Condition that affects service availability has occurred. Such a severity is reported, for example, when a managed object becomes, or is on the verge of becoming totally out of service. For security-related alarms, critical severity is used to block the user after X (usually 5) failed logins or multiple failed logins (or more failed logins from the same device). Failed/successful login is any failed/successful authentication attempt for any protocol (HTTP, SSH, SNMP, or Comverse applications/protocols).
Major	Urgent corrective action is required. Indicates that a service affecting condition has developed. It is likely that normal use of the managed object is impeded. For security-related alarms, major severity is used in the case of network attacks (such as SYN or Land), application attacks (such as directory traversal or buffer overflow), application/protocol input validation (the data or structure of the request is incorrect), and unauthorized activity (includes user attempts to run a functionality that is not allowed or attempts to access a forbidden IP/network/VLAN).
Minor	Corrective action is required. A problem with a relatively low severity level has occurred and a corrective action is required to prevent deterioration to service affecting condition. It is unlikely that normal use of the object is impeded, and the alarm condition is not currently degrading the capacity of the managed object. For security-related alarms, minor severity is used in the case of failed login and to report any security administration change (such as user creation/deletion/edit, password change or encryption level change).
Warning	Diagnostic and corrective actions are recommended. Indicates the detection of potential or impending service affecting fault prior to any significant effects being felt. For security-related alarms, warning severity is used in the case of successful login, successful logout, and reporting on any administration change.
Cleared (Normal)	Message output is as expected. For example, a process begins, a process finishes or status information is displayed. This is used for ending previous problem conditions. Other Normal messages are to be avoided.
Indeterminate (Unknown)	Severity level cannot be determined. Avoid using this value as much as possible.

DGU Alarm Messages

Table 15, “DGU Alarms” lists the DGU alarm messages.

For additional details about DGU alarms, refer to the Comverse ONE solution *Alarm Reference*.

Table 15 DGU Alarms

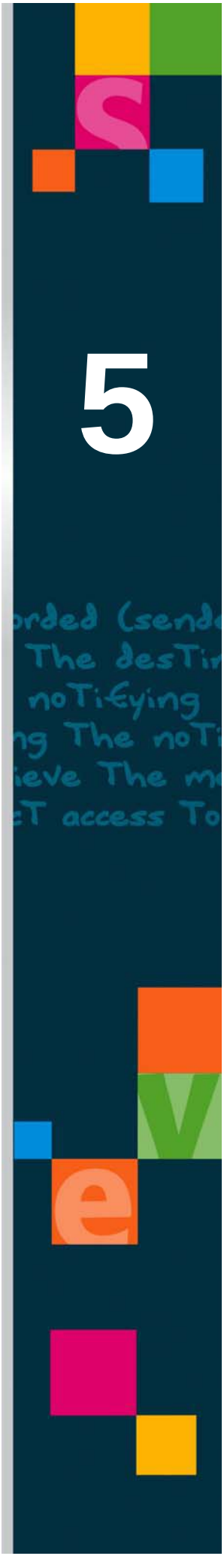
Specific Problem (Alarm ID)	EventID	Alarm Processor ID	Event Type	Additional Information	Severity	Corrective Action/ Instructions
2691303	DGU_DGA_ PEER	8500	2	Connection to Peer <peer_ name> down	Major	Determine why Peer is no longer communicating with DGU. Possible network issue or configuration issues. Restart and monitor Peer CE once determination is made.
2691001	DGU_DGA_ SERVICE	8501	11	Service <service_ name> not available	Critical	Determine why Peer client application is no longer communicating with DGU. Possible network or configuration issues. Check for core dump. Restart and monitor Peer client application once determination is made.
2691203	DGU_DGA_ SLU	8502	11	Connection to SLU <slu_ name> down	Major	Determine why SLU is no longer communicating with DGU. Possible network or configuration issues. Restart and monitor SLU once determination is made.
2691003	DGU_DGA_ SLU_ALL	8503	11	No SLU available	Critical	Determine why SLU is no longer communicating with DGU. Possible network or configuration issues. Restart and monitor SLU once determination is made.
2691103	DGU_DGA_ SLU_ CONGESTIO N	8504	11	SLU <slu_ name> answer delay exceeds <value> seconds	Major	Determine if transaction rate exceeds supported levels. Determine if any SLU process is consuming excessive CPU time.
2691005	DGU_DGA_ SLU_DELAY	8505	11	.SLU <slu_ name> answer delay exceeds <value> seconds	Not determined	Determine if transaction rate exceeds supported levels. Determine if any SLU process is consuming excessive CPU time.
2691006	DGU_DGA_ SLU_ PENDING	8506	11	SLU <slu_ name> request pending queue exceeds <count>	Not determined	Determine if transaction rate exceeds supported levels. Determine if any SLU process is consuming excessive CPU time.
2691012	DGU_DGA_ CONFIG_IP_ ADDR	8507	0	Duplicate connection from SLU <IP address> rejected	Critical	Check IP addresses of all SLU. Remove or reconfigure SLU with duplicate IP address.

Table 15 DGU Alarms (Continued)

Specific Problem (Alarm ID)	EventID	Alarm Processor ID	Event Type	Additional Information	Severity	Corrective Action/ Instructions
2691008	DGU_DGA_ CONFIG_CER	8508	0	CER from SLU <slu_name>, <IP address> rejected: <reason>	Critical	Determine if CER contains invalid or unsupported capabilities for DGU. Reconfigure SLU or DGU to agree on supported capabilities.
2691009	DGU_DGA_ CONFIG_ MML	8509	0	MML <command> failed: <reason>	Critical	Correct MML command specified in Alarm. Edit db file if command originated from db file.
2697318	FTP_ SESSION_<IP ADDRESS>	7318	2			
2698005	HOST_CPU_ LOAD	7301	10	Current Load Average	Critical	Contact your local Comverse Support team
2698111	HOST_ MEMORY_ USE	7302	10	Current Memory Utilization	Critical	Contact your local Comverse Support team.
2698111	HOST_ SWAP_USE	7303	10	Current and Available Swap Space	Critical	Contact your local Comverse Support team.
2698211	ALERT_FS_ <filesystem>	7304	10	<filesystem name>, Utilization	Critical	Contact your local Comverse Support team.
2698211	ALERT_ MON_<id>	7315	10		Critical	Contact your local Comverse Support team.
2698211	ALERT_JOB_ <jobid>	7306	10	<jobname>, <error message>	Major	
2698211	ALERT_ WKF_<id>	7314	10	<workflow name>, <error message>	Major	
2698311	HOST_ PROCESS_ <id>	7305	10	Process failures		
2698311	APP_ PROCESS_ <id>	7305	10	Process failures		

Chapter 5

Diameter Traffic Router



Overview

The Diameter Traffic Router (DTR) is a component of the Comverse One billing system Disaster Recovery feature. DTR supports distribution of Diameter traffic to the DR site based on the subscriber Id identified in the request.

The DTR behaves as a Diameter proxy. This allows the Diameter client to send all requests to a specific realm/peer connection corresponding to the DTR. The DTR inspects the message and determines on which system to route the request.

Two types of determinations must be made for routing the Diameter requests:

- **Requests that are not associated with any previous activity**: The example of this type of request is an immediate event request or the initial request associated with a Diameter session.
- **Follow on requests for an existing Diameter Session**: It is expected that there will be additional routing information embedded in these types of requests (the Destination-Host AVP) that is used for routing to the ultimate destination.

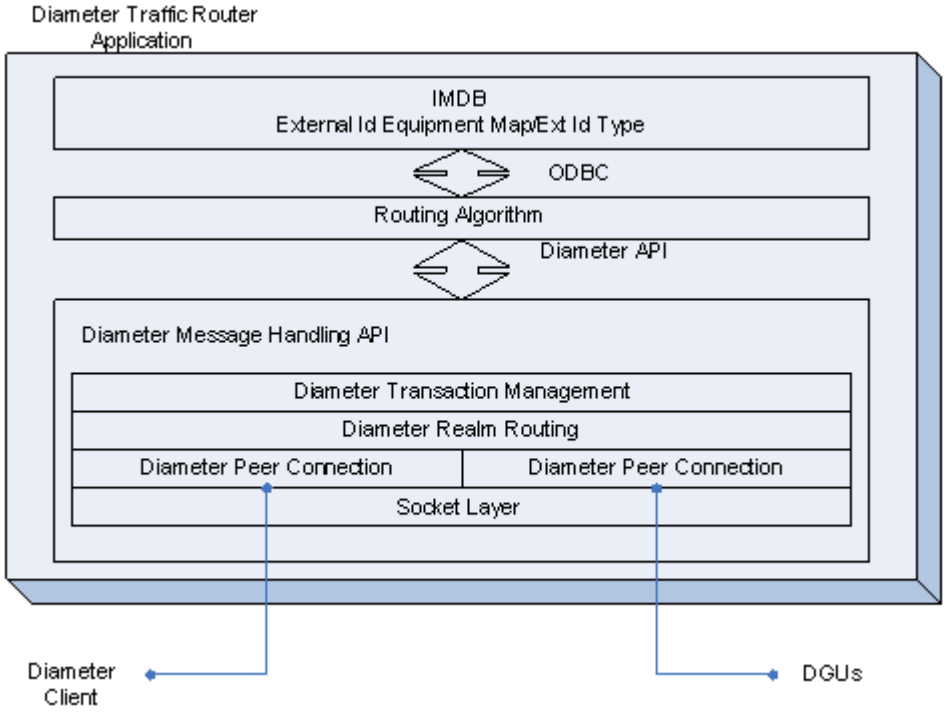
Based on these two scenarios, the algorithm checks for the presence of the Destination-Host AVP and, if found, uses this to make a routing decision. In the case where this AVP is not found, the Subscriber-Id AVP must be evaluated to determine how to route the message.

When using the Subscriber-Id AVP to make a routing decision, the External Id Equipment Map table is consulted. This table identifies the SDP that the subscriber belongs to and the SDP will be associated with a site. Additionally, the status of the SDP on that site is also interrogated to determine the final routing decision.

DTA Application

The DTA application is the hosting application for support of routing within the DTR subsystem. This Application contains several subsystems that are integrated within a single process. The following figure illustrates this relationship.

Figure 13 Diameter Traffic Router Application Subsystems



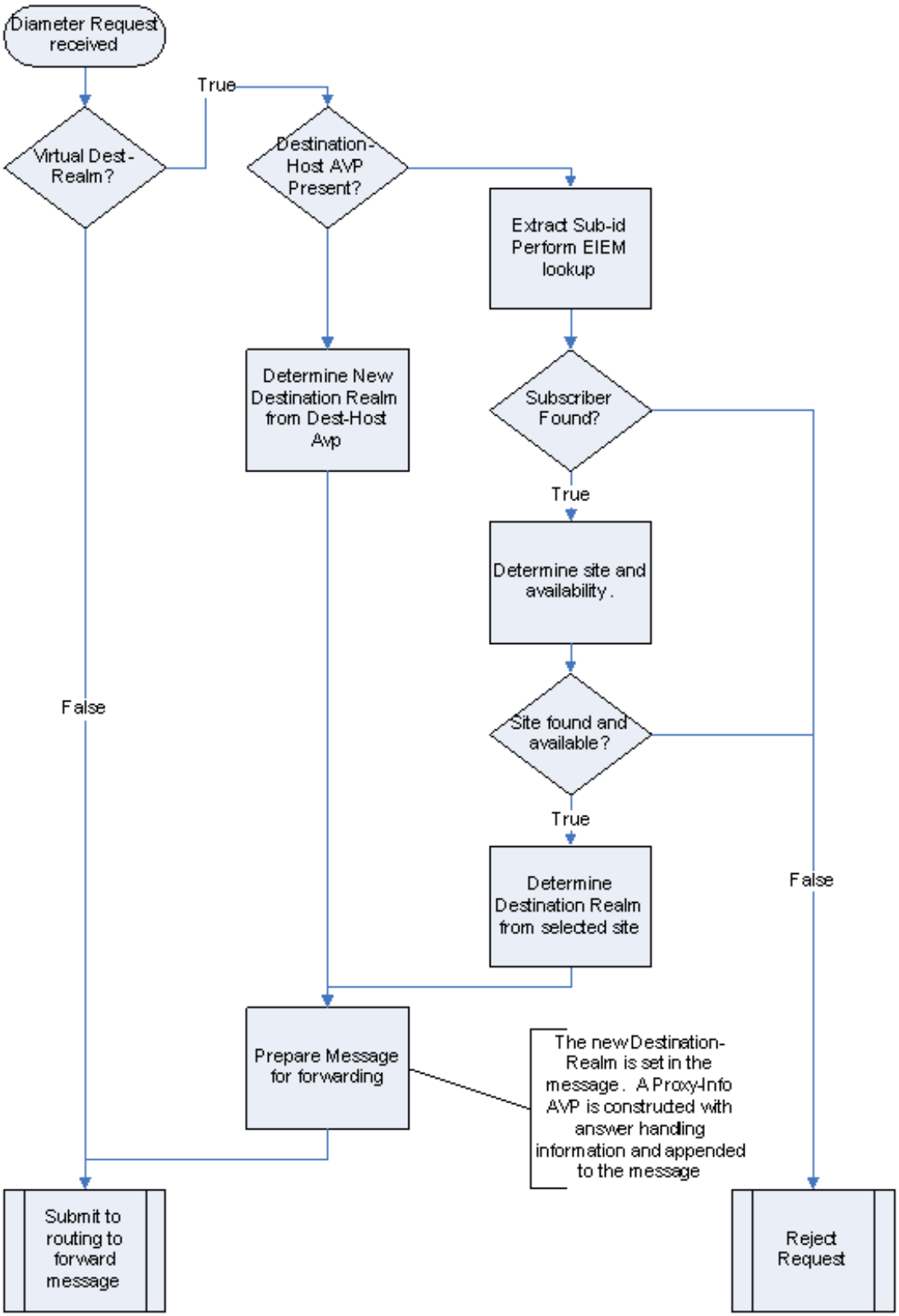
The DTR application interfaces to the IMBD using an ODBC interface which utilizes a shared memory space. The DTA shares its addressable space with the IMDB for faster lookups.

DTR Routing Algorithm

The DTR Routing Algorithm is the subsystem that implements the routing logic described in the following illustration.

The Subscriber Id is used to identify the hosting SDP for the subscriber. The site associated with the SDP is then identified and the Diameter message is routed towards this site using realm-based routing.

Figure 14 DTR Routing Algorithm



The Routing Algorithm processes a received message from Diameter Message Handling API (DiamApi) and applies the above logic to it.

Diameter Request and Answer

There are two types of Diameter messages: Request and Answer. Requests and Answers are always correlated to each other in a transaction. The requirements for routing a Diameter Answer is simply to send it out on the same peer connection that it was received on. In order to support this, the Diameter Message Handling API supports a transaction Id that is part of the original received Request message. To send the Answer, the host needs to identify the transaction that is affiliated with the original Request (the transaction Id) and set the Answer message with this value.

The fundamental mechanism used to determine how to distribute Diameter Request messages is based on a routing mechanism. Diameter supports routing based on Peer Connections or Realm Names. In general, the algorithm is to check for the Destination-Host AVP and, if present, match it to any of the Peer Connections that the Diameter interface is supporting. If the Destination-Host AVP is not present or does not match any of the Peer Connections, then use the Destination-Realm AVP to determine which Peer Connection to transmit the message on.

Virtual Realm

The DTR routes to a site by converting the Destination-Realm AVP in the Diameter Request message to a realm that routes it to a specific site. One of the first tests to apply to the received Diameter Request message is to check if the Destination-Realm AVP contains a string that matches one of the Realms used as a Virtual Realm in the Proxy mechanism. If there is a match then:

- The subscriber ID and address plan is extracted from the message and used to make a lookup in the External_Id_Equip_Map table.
- The lookup returns the primary SDP Id of the subscriber.
- The SDP Id is used to derive a site Id and a new site-specific Destination-Realm value.
- The Diameter Request Message Destination-Realm is updated with the new site-specific Destination-Realm. This message is then resubmitted to the Diameter Message Handling API and it is routed to a peer connection associated with the DGU on the given site.
- When the corresponding answer message is received, the Origin-Realm value is replaced with the virtual Realm Id.

The following figures illustrate message routing based on Virtual Realm and Session-based Transactions.

Figure 15 Basic Message Routing Based on the Virtual Realm

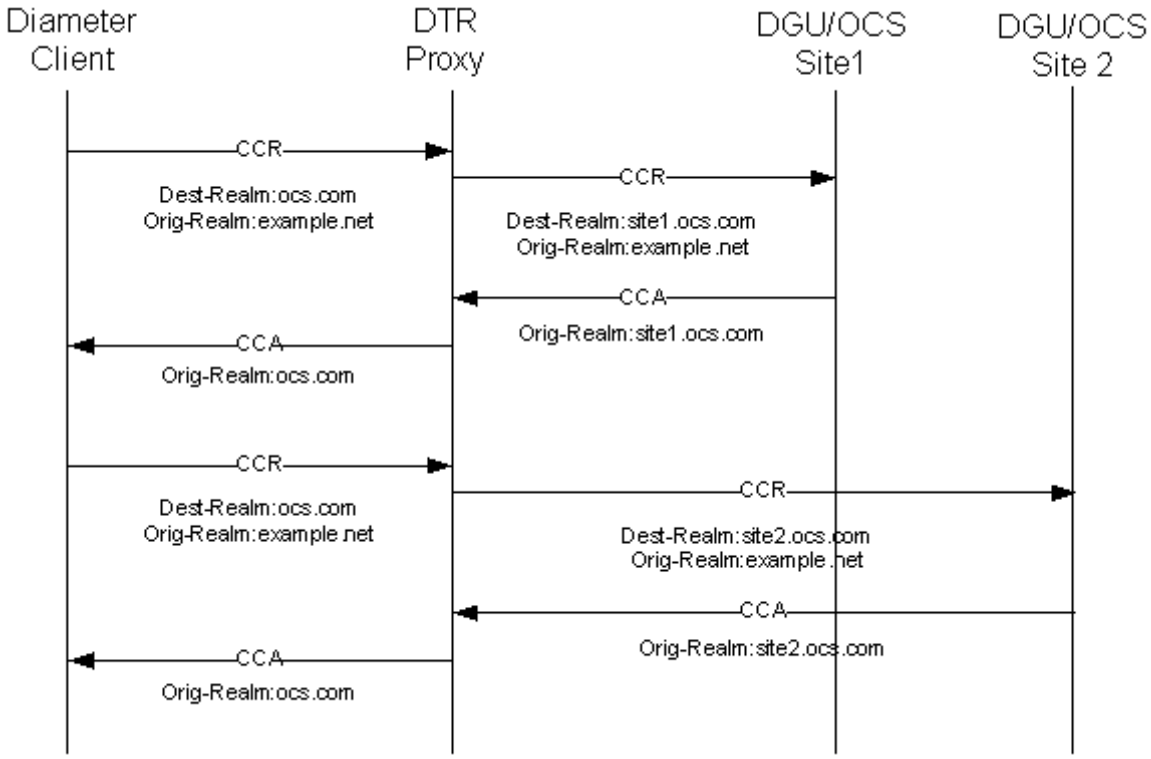
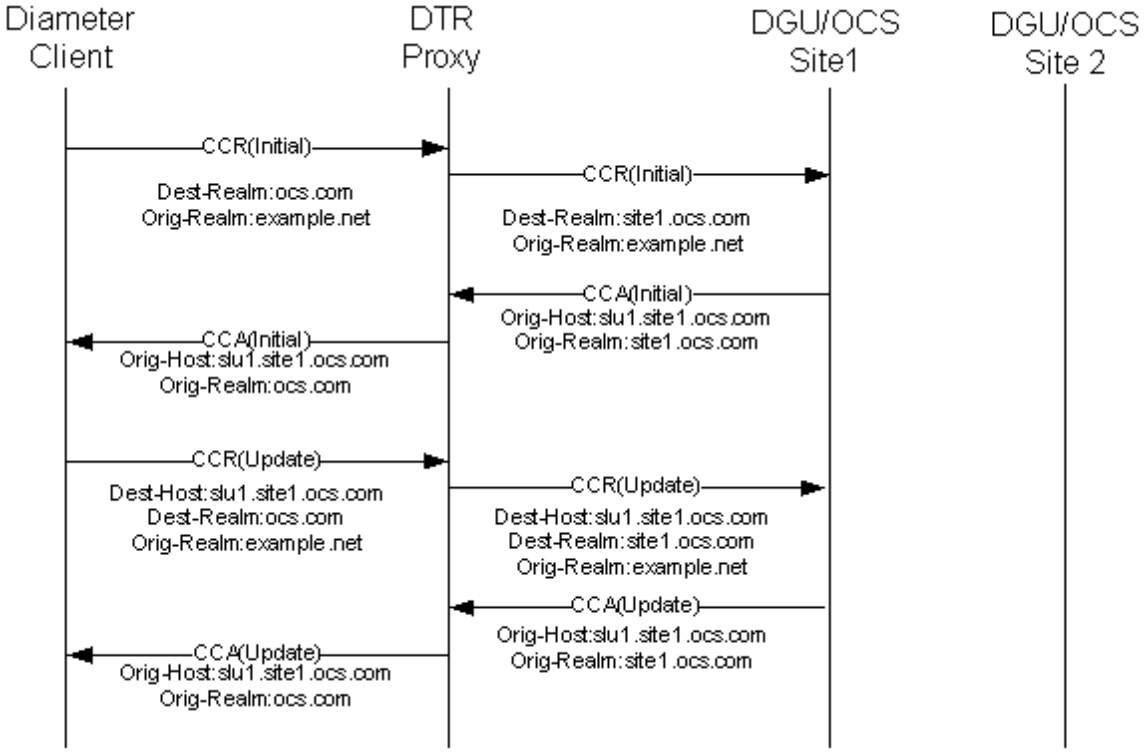


Figure 16 Message Routing for Session-based Transactions



Transaction Management

Diameter specifies requirements on any agent to provide specific information in the answer to any Diameter Request in the Diameter Answer. At the very minimum, the answer should use the same peer connection that the request was received on.

Some information about the status of the transaction is maintained by the Diameter Message Handling API. Other pieces of information (e.g. correlation of inbound requests to the corresponding answer) need to be managed with context information. The DTA application stores a small out of context information in the Proxy-Info AVP and attaches this to the forwarded request. The corresponding answer message contains this AVP, which is then used to determine the original request. The Proxy-Info AVP is stripped from the received Answer message and used to restore the Answer message to be compatible with the original request.

No context information is managed within the DTR.

Diameter Message Handling API

The Diameter Message Handling API (DMH API) is an integrated Diameter Protocol stack realized as a library that links with the hosting application (DTA). This library is comprised of several subsystems that, when taken as a whole, provide an RFC 3588 compliant Diameter protocol implementation.

The DMH API is configured to operate as a Diameter Proxy Agent. This mode logically terminates a transaction at the DTR and initiates a new, corresponding transaction to the ultimate destination. The DMH API is configured to operate in a proxy mode when the following parameters (in <diameter-api-parameters>) are set:

```
<parameter name="diameter-transfer-agent" value = "true" />
<parameter name="retransmission-enabled" value = "false" />
<parameter name="notify-request-error" value = "true" />
```

The parameter diameter-transfer-agent controls the automatic setup of session-Id and proxy info AVPs for transactions.

The parameter retransmission-enabled is set to false to prevent failover retransmissions from being attempted at the proxy server. This is done to force the retransmission policy back to the Diameter Client.

Diameter Credit Control Clients will typically manage retransmission on a per transaction basis and if this is shorted then the failover detection.

The parameter notify-request-error is used to determine if an outbound request fails to complete. This then provides the trigger to cancel the corresponding inbound request.

Subscriber Query

The DTR query takes two parameters ext_id and network_address_plan. The query returns the realm name associated with the site. The realm name is based on the site Id (from UPM OAM database) and the mapping of the rating db to this UPM OAM cluster Id. The rating DB is derived from the external_id_

equip_map using the external Id and another subquery to determine the external_id_types associated with the network_address_plan.

DTA Measurements

Measurements counters are members of measurement sets. Both the set and the counter are identified with a name attribute. The counter element also uses the type attribute to identify the measurement type and the measurement-set element uses the scope attribute to define the scope of the set.

Refer to the System Measurements Guide for additional details.

modDiamConn.pl

The modDiamConn.pl utility is provided to simplify the management of the various Diameter Peer Connections that must be supported at the DTR. The Diameter Message Handling API that is being used by DTR generally defines peer connections in XML. This utility creates the XML to define peer connections using a menu-based interface.

The result of this script is the XML file /home/omni/conf/diameter-conn-cntrl.xml. This file is then imported into the master XML file using an external parsed general ENTITY.

The utility allows the user to add or delete a peer connection or realm route. Adding and deleting the default route is also supported.

Times Ten In Memory Database

The Times Ten In Memory Database (IMDB) provides the ability to load a copy of data from an Oracle DB into local memory. This feature is known as Cache Connect.

The IMDB caches information from the External Id Equipment Map table that is used to make routing decisions. Additional tables are also cached to support this activity.

The IMDB is also used to store status information on the state of each of the SDPs in the system. Central to the DR algorithm is the ability to determine on which site an SDP is currently active.

The Times Ten IMDB is a third party product supplied by Oracle.

SiteMapSetup

The Times Ten In Memory Database for the DTR application caches information from the Rating DB locally to determine the serving SDP for any given subscriber. In DR, the SDPs are replicated pairs across sites that are geographically separated. The replication mechanism can only support one active SDP at a time. For each pair of SDPs, one SDP handles the traffic while the other peer machine is being updated with changes.

DTR needs to know which SDPs at the all sites are currently ‘in production’ mode, meaning willing to accept traffic. Depending on the DR mode, all ‘in production’ SDPs could be at one site or distributed across two sites.

Once the lookup is made to determine which Rating DB contains the entry for the subscriber, another step is taken to determine which site the SDP is currently ‘in production’ on. The information to support the determination of the ‘in-production’ SDP exits in the UPM OAM database (OAM-DB).

DTR caches information from the OAM-DB locally in the SDP_SITE_MAP table. The SDP_SITE_MAP table contains an entry for each ‘in production’ SDP correlated to the Rating DB Id.

Updating the SDP_SITE_MAP table is the job of the SiteMapSetup.pl script. This script queries the Rating DB SERVER_DEFINITION table to determine the DB Ids of all the rating DBs. The script then queries the OAM-DB to determine on which site each of these SDPs is active (in production).

DTA Configuration

DTA is configured from a master configuration file written in XML. The file is split into three sections (see "Configuration Settings" below).

1. The first section contains Diameter Message Handling API parameters (<diameter-message-handling>).
2. The second sections contain DTA system parameters (<system-parameters>). Currently the only parameter that can be set in this section is one to control the measurement collection interval (MeasurementCollectionInterval).
3. The third section is for application parameters (<application-parameters>). This section is passed to the DTA application through the processConfigParameters(xmlNodePtr iParmNode) interface. The argument to this interface is simply a XML node pointer and this allows the embedded application to define its own format for the information within the <application-parameters> element.

DTR Diameter Peer Connections

DTA Diameter peer connections are defined in a separate file which is managed with the modDiamConn.pl utility.

The DTR platform must be configured manually for all Diameter peer connections that it is expected to support. This typically means two peer connections to the DGUs on two sites (two connections to Site-A and two connections to Site-B). In addition each Diameter Client must also be configured.

There is a menu driven utility to define the Diameter peer connections. This utility is called modDiamConn.pl. It is used to configure Diameter Peer connections and the Realm routing table. The configuration updates are made to a file in the DTR environment (/home/omni/conf/ diameter-conn-cntrl.xml). The DTR application must be restarted in order for any changes to the peer connections to take effect.

The peer connection data file for DTR is not delivered in the distribution. This utility must be run at least one time in order for the basic configuration to be created.

To start the modDiamConn.pl utility:

1. Log into the DTR platform as dguuser.
2. Change directory to /home/omni/conf and run the modDiamConn.pl utility.

```
DTR:kdpm324> cd /home/omni/conf
DTR:kdpm324> modDiamConn.pl

Diameter Connection Control Configuration Utility
1) Display Peer Connections
2) Display Realm Routes
3) Add Peer Connection
```

```
4) Delete Peer Connection
5) Add Realm Route
6) Add Default Realm Route
7) Delete Realm Route
8) Display XML output
q) Quit without saving
e) Exit and save changes
DCCU>
```

Adding a Peer Connection

To add a Peer connection to the DTR platform the following information in necessary:

1. The Diameter Peer Host name
2. The Diameter Peer Realm name
3. The hostname (resolved via gethostbyname()) or the IP Address
4. The initiate connection flag (true | false).

   Typically Diameter clients will initiate connections to the server. In the case of DTR the customer client application will initiate connections to the DTR and DTR will initiate connections to the DGUs
5. If DTR is initiating the to the DGU then the TCP port number that the DGU is also provided (typically 3868).

When configuring the peer connections the modDiamConn.pl utility will prompt for this information:

```
Diameter Connection Control Configuration Utility
1) Display Peer Connections
2) Display Realm Routes
3) Add Peer Connection
4) Delete Peer Connection
5) Add Realm Route
6) Add Default Realm Route
7) Delete Realm Route
8) Display XML output
q) Quit without saving
e) Exit and save changes
DCCU> 3
Enter the peer-host: client.customer.com
Peer-host: client.customer.com
Enter the peer-realm [customer.com]:
Peer-realm: customer.com
Enter the peer-port [3868]:
Peer-port: customer.com
Does this entry need to initiate a connection to the peer? [false]:
Initiate connection: false
Enter hostname (or nothing to enter IP address): 10.20.30.40
Hostname: 10.20.30.40
```

```
Confirm: Add Peer client.customer.com [y|n]: y
Adding Primary route for customer.com using peer client.customer.com
```

When adding a peer connection, the modDiamConn.pl utility will also add a realm route for this peer.

Adding a Realm Route

There may be a need to specify a realm route for support of one of the applications in the system. This can be done using the Add Realm Route option of the menu.

To add a Realm Route the following information must be provided:

1. The Diameter Realm that will be specified as the new route
2. The Peer Connection that this route will resolve to.

   The menu interface only allows configuration of routes to existing peer connections.

```
Diameter Connection Control Configuration Utility
1) Display Peer Connections
2) Display Realm Routes
3) Add Peer Connection
4) Delete Peer Connection
5) Add Realm Route
6) Add Default Realm Route
7) Delete Realm Route
8) Display XML output
q) Quit without saving
e) Exit and save changes
DCCU> 5
Enter realm: example.net
Currently defined peer connections:
1) hslu18a.clnt.comverse-in.com
2) hslu18a.clnt2.comverse-in.com
3) kdpm375.svr1.comverse-in.com
4) kdpm375.svr2.comverse-in.com
5) client.customer.com
Enter the peer-connection id: 5
Adding Primary route for example.net using peer client.customer.com
```

Verifying and Saving the Configuration

The status of the modifications can be verified with the Display menu options.

When all modifications are complete entering ‘e’ to exit the utility we generate a new file /home/omni/conf/diameter-conn-cntrl.xml file. If there is a pre-existing version of this file this will be backed up to the filename /home/omni/conf/diameter-conn-cntrl.xml.mmddHHMM where:

mm	Two digit month
dd	Two digit day
HH	Two digit hour

MM Two digit minute

The resultant time stamp is the time stamp that the previous file was saved at (not the current time).

Configure External_Id_Type in PCAT

DTR requires that an extra value be associated with the set of external id types in the Comverse ONE PCAT (and Rating) databases.

All external id types that are expected to support Diameter DTR routing must be configured with a Network Address Plan value.

Maintenance Operations

Starting and Stopping the Times Ten IMDB

The TT IMDB is started as part of the standard Linux boot sequence. This is facilitated by inserting a script file in to /etc/init.d and using chkconfig to enable this script to run when the machine goes to run level 3.

The TT IMDB starts automatically on boot of the DTR platform. The TT IMDB can also be started and stopped manually through a command line script. (/data/TimesTen/dtr_inst/startup/tt_dtr_inst)

To start the TT IMDB:

1. Log into the system as ttuser.
2. Issue the commands:

```
[ttuser@kdpm324 scripts]$ cd $TT_HOME
[ttuser@kdpm324 dtr_inst]$ startup/tt_dtr_inst start
Starting TimesTen Daemon : [  OK  ]
```

To stop the TT IMDB:

1. Log into the system as ttuser.
2. Issue the commands:

```
[ttuser@kdpm324 dtr_inst]$ cd $TT_HOME
[ttuser@kdpm324 dtr_inst]$ startup/tt_dtr_inst stop
Stopping TimesTen Daemon : [  OK  ]
```

Starting and Stopping OMNI and DTR

OMNI is one of the standard execution environments used to configure and maintain the real time applications in the Comverse ONE environment.

OMNI is started via the standard linux startup utilities. It uses a services configuration that places a startup script in the /etc/init.d directory.

The Linux utility chkconfig is used to enable this startup and shutdown script.

OMNI also provides an automatic restart script when there is an unintended shutdown of the OMNI environment.

By default, OMNI is configured for auto-startup and requires no special action.

If needed, auto-startup may be disabled by executing the following command as root:

```
$OMNI_HOME/conf/configOmniSvc -o off
```

When auto-startup is disabled, OMNI can be started by executing the following command as root:

```
service omni start
```

When auto-startup is disabled, OMNI can be stoped by executing the following command as root:

```
service omni stop
```

Please note, When auto-startup is disabled OMNI will not start up upon Linux reboot.

To enable OMNI auto-startup again, execute the following command as root:

```
$OMNI_HOME/conf/configOmniSvc -o on
```

Using OMD to Interface to DTA

The primary task the DTR platform is the Diameter Traffic Application (DTA). This task uses the standard OMD interface to provide real time updates of its operation. There is no formal documentation of the OMD output, however the means to start omd with DTA are outlined here.

OMD provides trace information, information on the OMNI platform and a MML command interface.

1. Log into the DTR platform as dguuser
2. Run the omd command for DTA.

```
DTR:kdpm324> omd DTA
name=omd28035, path=omd, node=, target=DTA
shmkey=0x1280CE, shmid=9011204, shmsize=246320, entries=1200

current time is Fri Aug  7 20:29:07 2009
type '?' for command list
:DTA>>
```

DTA supports OMD trace buffer with various levels of display. Also supported are a series of MML commands to extract information from the application.

Checking the Status of the Times Ten IMDB

The TT IMDB distribution provides several utilities to monitor the status of the application.

Use the ttStatus command to check on the status of the TT IMDB. This can verify that the TT IMDB is running and provide information on the operation of the database.

When the TT IMDB is not running entering ttStatus will provide the following output:

```
DTR:kdpm324> ttStatus
ttStatus: Could not connect to the TimesTen daemon.
If the TimesTen daemon is not running, please start it
by running "ttDaemonAdmin -start".
```

When the TT IMDB is operating entering the ttStatus command will provide out similar to the following:

```
DTR:kdpm324> ttStatus
TimesTen status report as of Fri Aug  7 19:38:18 2009

Daemon pid 21066 port 17000 instance dtr_inst
TimesTen server pid 21078 started on port 17002
No TimesTen webserver running
```

```
------------------------------------------------------------------------
---
Data store /data/TimesTen/dtr_inst/info/dtr
There are 13 connections to the data store
Data store is in shared mode
Shared Memory KEY 0x040580f5 ID 8617989
Type            PID     Context     Connection Name              ConnID
Cache Agent     21469   0x09e9d3a8  Handler                         2
Cache Agent     21469   0x09f34b20  Timer                           3
Cache Agent     21469   0x09fa5e38  Aging                           4
Cache Agent     21469   0x0a010080  timestenorad                    5
Cache Agent     21469   0x0a0d1d00  timestenorad                    6
Process         22266   0x08e685e8  dta-tt                          7
Process         25536   0x0902f398  ttCheckCmd                      1
Subdaemon       21070   0x08c7c748  Worker                       2042
Subdaemon       21070   0x08d33160  Flusher                      2043
Subdaemon       21070   0x08da27d0  Checkpoint                   2044
Subdaemon       21070   0x08df1bd8  Aging                        2045
Subdaemon       21070   0x08e40fe0  HistGC                       2046
Subdaemon       21070   0x08e903e8  Monitor                      2047
RAM residence policy: Always
Replication policy  : Manual
Cache agent policy  : Always
TimesTen's Cache agent is running for this data store
------------------------------------------------------------------------
---
Data store /data/TimesTen/dtr_inst/info/TT_dtr_inst
There are no connections to the data store
Replication policy  : Manual
Cache agent policy  : Manual
------------------------------------------------------------------------
---
Access control enabled.
End of report
```

The TT IMDB also provides a SNMP interface to be integrated with UPM.

This system runs on Linux Redhat 5.1 in a 32-bit mode. It must run on a 4 Gbyte platform.

DTR MML Commands, Measurements and Alarms

The DTR platform provides a variety of commands to interrogate the status of the system. In addition there are some commands that can be used to start and stop Diameter peer connections. No command that changes the Peer Connection state will be persistent across restarts of DTR.

Many of the MML commands can be shortened to a unique value (like DISPLAY to DISPL).

DTR-DISPLAY-DCC;

Display Diameter Connection Control information.

DTR-DISPLAY-DDS;

Display Diameter Default Dataset information

DTR-DISPLAY-DICT;

Display Diameter Dictionary information

DTR-DISPLAY-DSVC;

Display Diameter Service Selection criteria (used to select Diameter dictionaries).

DTR-DISPLAY-DTM;

Display Diameter Transaction Manager information.

DTR-DISPLAY-EVENTS;

Display the reported event history. Each event reported by DTA is recorded locally in a buffer that can be displayed via MML.

DTR-DISPLAY- TCP;

Display information on the TCP connections used by Diameter

DTR-DISPLAY- MEASUREMENTS[:SET="<setname>"];

Display measurement information on the various measurements collected by the platform. Entering the DTR-DISPLAY-MEASUREMENT; MML command without a set name will provide a list of available measurements. For example:

```
:DTA>>DTR-DISPLAY-MEASUREMENTS;
MML sent to kdpm324.PM:
    DTR-DISPLAY-MEASUREMENTS;

DTA measurement set names:
   DTA-BaseDiameter
   DTA-DtaMessage
   DTA-DtaTransaction
   DTA-DtrResponseTimes
   DTA-DtrApplication
   DTA-Peer-hslu18a.clnt.comverse-in.com
   DTA-Peer-hslu18a.clnt2.comverse-in.com
   DTA-Peer-kdpm375.svr1.comverse-in.com
```

```
   DTA-Peer-kdpm375.svr2.comverse-in.com

DTR-DISPLAY-MEASUREMENTS

07-Aug-20 21:05:17
M  COMPLETE
```

Specifying the set name will provide the actual counters

```
:DTA>>DTR-DISPLAY-MEASUREMENTS:SET="DTA-DtaTransaction";
MML sent to kdpm324.PM:
    DTR-DISPLAY-MEASUREMENTS:SET="DTA-DtaTransaction";

DTA measurements for the set DTA-DtaTransaction:
Last collected at: 21:00

   MAX_RX_tps                                  1
   MIN_RX_tps                                  0
   AVE_RX_tps                                395
   Canceled_Trans                              0
   Rejected_Trans                              0

DTR-DISPLAY-MEASUREMENTS

07-Aug-20 21:06:36
M  COMPLETE
```

Refer to the System Measurements Guide for additional details.

DTR-DISPLAY-APPLICATION;

Display information on the current application. The DTR-TT application reports the status of the Times Ten database, the current SDP/Site mapping (which SDPs are active on the specific site) and application parameters

DTR-START-PEER:PEER="<peer-name>";

Start the peer connection to <peer-name>

DTR-START-ALL;

Start all configured peer connections

DTR-STOP-PEER:PEER="<peer-name>",CAUSE=<cause>;

Stop a Diameter Peer connection for <peer-name>. The command causes a Diameter Disconnect Peer Request to be sent by DTR.

The cause attribute can be set to:

```
REBOOTING|BUSY|DNWTTTY
```

This will load the Disconnect-Cause AVP for the DPR. (DNWTTTY translates to the DO_NOT_WANT_TO_TALK_TO_YOU cause value).

DTR-STOP-ALL:CAUSE=<cause>;

Stop all Diameter Peer connections. The command causes a Diameter Disconnect Peer Request to be sent by DTR.

The cause attribute can be set to:

```
REBOOTING|BUSY|DNWTTTY
```

This will load the Disconnect-Cause AVP for the DPR. (DNWTTTY translates to the DO_NOT_WANT_TO_TALK_TO_YOU cause value).

UPA Alarms and Events

Refer to the Unified Platform Guide for additional details.

DTR Alarming Support

Table 16 DTR Alarming Support

Alarm	Severity	Description
DTR_DB_CONNECTION_DOWN	Critical	Internal DB connection is down. Check the status of the IMDB. Most likely the system will require a reboot.
DTR_PEER_CONN_FAILED	Major	The Diameter peer connection to a peer is down. The peer is identified in the alarm attribute. Check the status of the peer with the DTR-DISPLAY-DCC MML command. Verify the peer status and network connectivity to the peer.

DTR UPA and OMNI Event Support

Table 17 DTR Reports Events to UPA

Event	Description
DTR_PEER_CONN_ESTABLISHED	A Diameter connection has been established with a peer.

Table 17 DTR Reports Events to UPA

Event	Description
DTR_PEER_CONN_STOPPED	The STOP-PEER MML command was entered.
DTR_PEER_CONN_STARTED	The Peer connection was started either as part of DTR startup or is started based on the START-PEER MML command.
DTR_DIAMETER_LISTEN_STARTED	Diameter connections are being accepted.

Index