

# **DM3 IPLink™ User's Guide**

## **for Windows NT**

**Copyright © 1998 Dialogic Corporation**



PRINTED ON RECYCLED PAPER

05-0917-001

## COPYRIGHT NOTICE

Copyright 1998 Dialogic Corporation. All Rights Reserved.

All contents of this document are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation. Every effort is made to ensure the accuracy of this information. However, due to ongoing product improvements and revisions, Dialogic Corporation cannot guarantee the accuracy of this material, nor can it accept responsibility for errors or omissions. No warranties of any nature are extended by the information contained in these copyrighted materials. Use or implementation of any one of the concepts, applications, or ideas described in this document or on Web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not condone or encourage such infringement. Dialogic makes no warranty with respect to such infringement, nor does Dialogic waive any of its own intellectual property rights which may cover systems implementing one or more of the ideas contained herein. Procurement of appropriate intellectual property rights and licenses is solely the responsibility of the system implementer. The software referred to in this document is provided under a Software License Agreement. Refer to the Software License Agreement for complete details governing the use of the software.

All names, products, and services mentioned herein are the trademarks or registered trademarks of their respective organizations and are the sole property of their respective owners. DIALOGIC (including the Dialogic logo), DTI/124, SpringBoard, and Signal Computing System Architecture (SCSA) are registered trademarks of Dialogic Corporation. A detailed trademark listing can be found at <http://www.dialogic.com/legal.htm>.

Publication Date: April, 1998

Part Number: 05-0917-001

Dialogic Corporation  
1515 Route 10  
Parsippany NJ 07054

**Technical Support**  
Phone: 973-993-1443  
Fax: 973-993-8387  
Email: [CustEng@dialogic.com](mailto:CustEng@dialogic.com)

For **Sales Offices** and other contact information, visit our website at <http://www.dialogic.com>

# Table of Contents

---

<b>1. Introduction.....</b>	<b>11</b>
1.1. About This Guide.....	11
1.2. IP Telephony Overview.....	12
1.3. What You Need to Know To Create An Application.....	13
<b>2. The DM3 IPLink Platform .....</b>	<b>15</b>
2.1. What is the DM3 IPLink Platform?.....	15
2.2. The DM3 Platform.....	16
2.2.1. The DM3 Platform.....	16
2.2.2. The DM3 Technology Resources.....	17
2.2.3. DM3 Component.....	17
2.2.4. Clusters.....	17
2.2.5. DM3 Component Communication.....	18
2.2.6. The SCbus.....	18
2.3. IPLink Platform Capabilities.....	18
2.4. Quality of Service.....	19
2.5. Hardware Configuration.....	19
2.6. The Net Telephony Service Provider (NetTSP) Resource.....	21
2.6.1. NetTSP Resource Overview.....	21
2.6.2. NetTSP Resource Features.....	21
2.6.3. NetTSP Resource Elements.....	22
2.6.4. NetTSC Component.....	24
2.6.5. NetTSC Component Activities.....	25
2.7. Line Administration Component.....	25
2.7.1. Line Administration Component Activities.....	25
2.8. Internal Components.....	26
2.8.1. Voice Stream Resource Component (VSR).....	26
2.8.2. H.323 Component.....	27
<b>3. Host Application.....</b>	<b>29</b>
3.1. Host Application Responsibilities.....	29
3.2. Host Application Communication with Dialogic Resources.....	30
3.2.1. Event Reporting.....	31
<b>4. Programming Model.....</b>	<b>33</b>
4.1. What are Call-States?.....	33
4.2. What are Call-State Changes?.....	35

**DM3 IPLink™ User's Guide for Windows NT**

4.3. Enabling Detection of Call-State Changes.....	36
4.4. State Diagrams .....	36
4.4.1. Outbound Internet Calls .....	37
4.4.2. Inbound Internet Calls.....	38
<b>5. Using the NetTSP Resource.....</b>	<b>41</b>
5.1. Messages .....	41
5.1.1. Standard Messages Used by the NetTSC Component.....	41
5.1.2. TSC Messages Used by the NetTSC Component.....	43
5.1.3. NetTSC Component Messages.....	44
5.2. Detecting Events.....	45
5.2.1. About Events.....	45
5.2.2. Event Messages.....	46
5.2.3. The Event-Reporting Process.....	46
5.2.4. Events Specific to the NetTSC Component.....	47
5.3. Allocating a NetTSP.....	48
5.3.1. Procedure: Allocating a Cluster.....	48
5.4. Assigning Timeslots .....	49
5.4.1. Procedure: Assigning a Receive Timeslot to the NetTSC Component .....	49
5.4.2. Querying About the SCbus Transmit Timeslot.....	50
5.5. Making an Outbound Call.....	52
5.5.1. Procedure: Making an Outbound Call.....	53
5.6. Receiving an Inbound Call.....	55
5.6.1. Procedure: Answering an Inbound Call .....	56
5.6.2. Procedure: Accepting a Call .....	59
5.7. Terminating Calls.....	63
5.7.1. Procedure: Terminating a Call by the Host .....	64
5.7.2. Procedure: Call Terminated by the IP Network.....	65
5.8. Getting Call Statistics .....	68
5.8.1. Procedure: Getting Call Statistics .....	69
5.9. Other Call Control Operations.....	70
5.9.1. Rejecting an Inbound Call .....	70
5.9.2. Failed Outbound Call .....	71
<b>6. Setting IPLink Parameters.....</b>	<b>77</b>
6.1. FCD File.....	77
6.1.1. SP Parameters .....	78
6.1.2. H.323 Parameters .....	81
6.1.3. Miscellaneous Parameters.....	83

## 1. Introduction

6.2. PCD File .....	87
6.3. Config.val .....	88
6.3.1. Describe Coder Capabilities .....	89
6.3.2. Enable a Gatekeeper .....	90
<b>7. Debugging.....</b>	<b>95</b>
7.1. Download Failed .....	95
7.1.1. Check the Hardware .....	97
7.1.2. Check the Drivers .....	99
7.1.3. Check the PCD/FCD files .....	100
7.2. Download Succeeded.....	101
7.2.1. H.323 Service .....	102
7.2.2. H.323 Protocol Messages .....	104
7.2.3. CP and SP Errors.....	104
7.3. Information for Customer Support.....	105
<b>Appendix A - Introduction to Internet Telephony .....</b>	<b>109</b>
Standards.....	110
ITU Recommendation H.323 .....	111
<b>Appendix B - config.val File.....</b>	<b>115</b>
<b>Appendix C - ipt.fcd File .....</b>	<b>119</b>
<b>Glossary.....</b>	<b>127</b>
<b>Index .....</b>	<b>135</b>



## List of Tables

---

Table 1. Call-States.....	34
Table 2. State Transitions .....	35
Table 3. DM3 Standard Messages Used by the NetTSC Component.....	42
Table 4. TSC Component Messages Used by the NetTSC Component.....	43
Table 5. NetTSC Component Messages.....	45
Table 6. NetTSC Component Event Types .....	47
Table 7. Procedure: Allocating a Cluster .....	48
Table 8. Procedure: Assigning a Timeslot .....	50
Table 9. Procedure: Querying the PSTN.....	51
Table 10. Procedure: Querying the DM3 .....	52
Table 11. Procedure: Outbound Calls .....	54
Table 12. Procedure: Answering Inbound Calls.....	58
Table 13. Procedure: Accepting Inbound Calls.....	61
Table 14. Terminating a Call by the Host.....	65
Table 15. Call Terminated by the IP Network.....	66
Table 16. Getting Call Statistics.....	69
Table 17. Rejecting an Inbound Call .....	71
Table 18. Failed Outbound Call .....	73





## List of Figures

---

Figure 1. DM3 IPLink Platform Configuration Tree.....	16
Figure 2. PSTN-IP Gateway Configuration (separate Network Interface Board) .....	20
Figure 3. PSTN-IP Gateway Configuration (Network Interface on IPLink).....	21
Figure 4. NetTSP Architecture - DM/IPLink-T1(E1) Board .....	23
Figure 5. NetTSP Architecture - DM/IPLink-T1(E1)_NIC Board.....	24
Figure 6. Typical Gateway .....	30
Figure 7. State Diagram Explanation .....	37
Figure 8. Outbound Call-State Diagram .....	38
Figure 9. Inbound Call-State Diagram.....	39
Figure 10. Outbound Call - Simplified State Machine .....	53
Figure 11. Answer Inbound Call - Simplified State Diagram.....	57
Figure 12. Call Establishment Messages - Simple Model .....	59
Figure 13. Accept Inbound Call - Simplified State Diagram .....	60
Figure 14. Call Establishment Messages - Accept Call .....	63
Figure 15. Call Termination - Simplified State Diagram .....	64
Figure 16. Call Termination Messages .....	68
Figure 17. Rejecting an Inbound Call - Simplified State Diagram .....	70
Figure 18. Failed Outbound Call - Simplified State Diagram.....	72
Figure 19. Debug Flow - Download Failed .....	96
Figure 20. Boot Kernel Versions .....	97
Figure 21. SCbusClockMaster Parameter Setting .....	99
Figure 22. Debug Flow - Download Succeeded.....	102
Figure 23. Typical IP Telephony Configuration.....	110



# 1. Introduction

---

The *DM3 IPLink User's Guide* provides the information necessary to build an Internet Telephony application using Dialogic Corporation's DM3 IPLink platform.

Topics discussed in this introduction include:

- About This Guide
- IP Telephony Overview
- What You Need to Know To Create An Application
- What Resources are Available?

The *DM3 IPLink User's Guide* assumes that the hardware is installed and the firmware has been download, and therefore will not describe hardware installation or firmware download. Refer to the *DM3 IPLink Series Quick Install Card* included with each board for a description of hardware installation and *Installing and Configuring The DM3 IPLink SDK for Windows NT* for firmware download.

## 1.1. About This Guide

This document is divided into two sections:

- **Section 1** provides an introduction to the DM3 IPLink platform. It includes:
  - Chapter 2. The DM3 IPLink Platform
  - Chapter 3. Host Application
- **Section 2** provides programmer resource material. It includes:
  - Chapter 4. Programming Model
  - Chapter 5. Using the NetTSP Resource
  - Chapter 6. Setting the DM3 IPLink Parameters
  - Chapter 7. Debugging

## ***DM3 IPLink™ User's Guide for Windows NT***

Background information about Internet Telephony can be found in the Appendix.

**NOTE:** Dialogic Corporation recommends that first-time users read through this entire User's Guide before beginning to write a host application. Experienced users can refer directly to the chapters in Section 2 when writing an application.

### **1.2. IP Telephony Overview**

Transporting voice over the Internet Protocol (IP) requires computationally intensive signal processing functions, including low bandwidth coders, advanced echo cancellation, multimedia call control and high performance protocol stacks.

The DM3 IPLink platform allows a “voice over IP” call to be connected via the DM3 platform to the SCbus. By adding additional SCbus boards, one can build a variety of applications, such as an IP—PSTN gateway.

The DM3 IPLink board and firmware, together with standard Dialogic Corporation SCbus PSTN cards, provide all the call control functions necessary for IP telephony:

- Answer incoming calls from either the IP network or the PSTN network
- Collect relevant information from the incoming call
- Make a corresponding call on the complementary network
- Bridge the two calls

This capability can be used to build a gateway, bridging the traditional circuit-switched telephony world with the Internet.

The IP Telephony configuration enables current Dialogic Corporation telephony resources to be mixed and matched with an IP connection and Dialogic's DM3 IPLink platform to build a wide variety of solutions.

### 1.3. What You Need to Know To Create An Application

This section describes the knowledge you will need to write an Internet telephony application. Before beginning to write an application, you should already be familiar with:

- C or C++ programming language
- Windows NT Programming

In order to write an Internet telephony application, you should be familiar with:

- DM3 messages, clustering, timeslot assignment
  - See *DM3 Architecture Overview* for a description of clustering concepts and timeslot assignment
  - See *IPLink Reference Guide* for a complete description of TSC messages
    - For a list of TSC messages - see *Section 5.1.2. TSC Messages Used by the NetTSC Component*
  - See *DM3 Standard Component Interface Messages* for a complete description of DM3 standard messages
    - For a list of standard messages - see *Section 5.1.1. Standard Messages Used by the NetTSC Component*
- Dialogic System Software for Windows NT (when using Dialogic Corporation SCbus PSTN cards)
  - The Dialogic System Software on-line bookshelf is included with this IPLink SDK.
- DM3 Direct Interface
  - See *DM3 Direct Interface Function Reference* and *Using the DM3 Direct Interface for Windows NT*
- Win32 Synchronization Methods
  - See *Using NSM for Event Notification with DM3, R4, and Win32 Devices*



## 2. The DM3 IPLink Platform

---

This chapter presents a high-level description of the DM3 IPLink platform, describing both the hardware platform and the software.

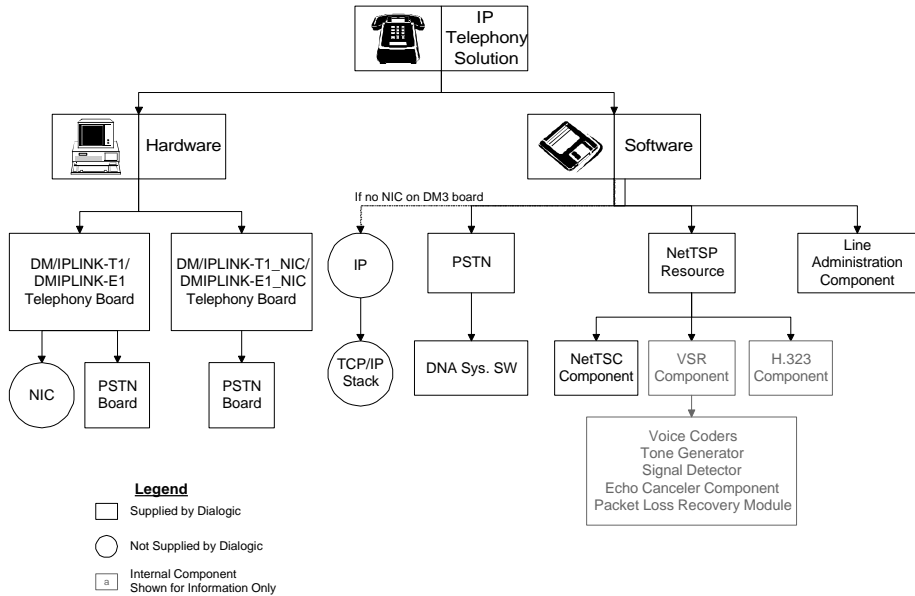
Topics discussed include:

- The DM3 IPLink Platform
- DM3 IPLink Capabilities
- Quality of Service
- Hardware Platform
- The NetTSP Resource
- Timeslot Management
- Line Administration Component
- Internal Components

### 2.1. What is the DM3 IPLink Platform?

The DM3 IPLink platform includes both hardware and software. Figure 1 displays the various parts of the IPLink platform. The grayed areas are internal components not accessible to the application writer and are shown in order to present a complete picture of the IPLink platform. The components shown in circles are not supplied by Dialogic Corporation.

**DM3 IPLink™ User's Guide for Windows NT**



**Figure 1. DM3 IPLink Platform Configuration Tree**

**2.2. The DM3 Platform**

This section introduces basic concepts and terminology relating to the DM3 Mediastream Architecture. **DM3** is an architecture on which a whole set of Dialogic products are built. The DM3 architecture is open, layered, and flexible, encompassing hardware as well as software components.

**2.2.1. The DM3 Platform**

The software on the DM3 platform consists of the following types of software:

- **Core platform software**, which includes the platform-specific DM3 kernel and device driver. All base platforms of the same type will have the same core platform software.
- **DM3 resource software**, which provides the features of a resulting product.



## 2. The DM3 IPLink Platform

### 2.2.2. The DM3 Technology Resources

A **DM3 technology resource** is a set of embedded resource software that provides a specific set of features, such as voice over Internet Protocol, for a DM3 product. A resource represents a set of features to be performed and the functions of one resource are logically independent of any other resource. A DM3 resource closely matches the SCSA concept of a resource (such as a player, recorder, fax transmitter, etc.).

A DM3 signal-computing resource is made up of a host software component that resides on the host platform containing the DM3 board(s), and one or more DM3 firmware components that reside on the various processors on the boards themselves.

### 2.2.3. DM3 Component

DM3 **components** are logical entities that **implement specific features** within a resource. A resource may be separated into components to distribute functions among processors (for example, control functions versus signal processing algorithms), or to provide alternative or optional services within a resource (such as multiple coders in a player resource).

A **component instance** is an **addressable unit** within the DM3 software architecture; it represents a single thread of control. The DM3 system resource management and messaging services operate at the instance level. A set of component instances communicate with one another using the DM3 system messaging services. A set of component instances is usually associated with a channel of call processing.

### 2.2.4. Clusters

A **set of component instances** that share SCbus timeslots and collectively deliver a set of functions for the host controller are grouped into a **cluster**. Clusters control how TDM data is transferred from one component to another, and how TDM data is routed to and from the SCbus. When a DM3 component instance is allocated, it is allocated into a specific cluster. Once a cluster has been created, the DM3 kernel can control the ports to allow the resources within that cluster to share ports in an intelligent manner.

## ***DM3 IPLink™ User's Guide for Windows NT***

While the cluster contents can be dynamically configured at run-time, default setups are usually enough for most applications and offer a higher level granularity of access to the DM3 platform.

### **2.2.5. DM3 Component Communication**

DM3 components have two communication mechanisms:

- Sending messages
- Sending blocks of data

A message-based protocol is used for communicating between DM3 components or component instances in the system.

Bulk data in a DM3 system is manipulated using global streams to transfer the data between processors, between the host and a component or instance, and between the SCbus and a component or instance.

### **2.2.6. The SCbus**

The SCbus allows different SCSA compatible boards to communicate with each other. The DM3 IPLink platform allows a “voice over IP” call to be connected via the DM3 platform to the SCbus.

## **2.3. IPLink Platform Capabilities**

The DM3 IPLink platform supplies the following capabilities:

- **Call Management** is provided by the NetTSC component which, in turn, uses an H.323 stack.
- The **RTP/RTCP** media stream is a sequence of packets using UDP (User Datagram Protocol) as transport. Packetization is handled by the resources on the IPLink board.
- The following **Voice Coders** are supported by the DM3 IPLink platform (at the time of writing this manual):
  - G.723.1

## 2. The DM3 IPLink Platform

- G.711
- GSM

**NOTE:** Call your Dialogic Corporation sales engineer for the complete list of supported voice coders.

- DTMF Detection/Generation  
The IPLink platform provides both inband and out-of-band tone detection and generation.
- Packet Loss Recovery Mechanism  
The IPLink platform includes a sophisticated algorithm for lost packet recovery.

### 2.4. Quality of Service

Unlike the PSTN, in which quality of service is defined as a particular level of service (“toll-like”), quality of service for voice over the Internet Protocol is defined as a continuum of levels, affected by packet delay or loss, line congestion, and hardware quality such as microphone quality.

The DM3 IPLink platform is designed to operate along the entire range of quality of service, enabling the application to retrieve information necessary for correct billing.

### 2.5. Hardware Configuration

The IPLink platform is available in two hardware configurations:

- DM/IPLink-T1 and DM/IPLink-E1
  - Requires a host Network Interface Card (NIC) to connect to the IP network
- DM/IPLink-T1\_NIC and DM/IPLink-E1\_NIC
  - Network interface is included on the IPLink board

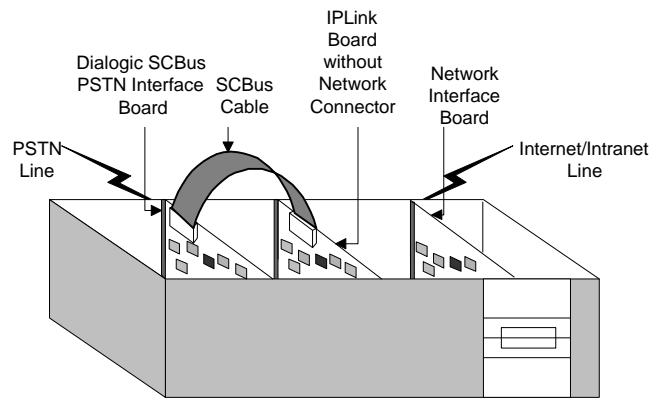
**DM3 IPLink™ User's Guide for Windows NT**

Both models of the IPLink board connect to a separate PSTN board via the SCbus.

A typical PSTN-IP Gateway configuration consists of three parts:

- Dialogic SCBus PSTN Interface Board (PSTN Connection)
- DM3 IPLink Board (IP Network Connection)
- Network Interface Board (IP Interface) or DM3 IPLink Board with NIC

Figure 2 illustrates a possible hardware configuration using an IPLink board with a separate network connector:



**Figure 2. PSTN-IP Gateway Configuration (separate Network Interface Board)**

Figure 3 illustrates a possible hardware configuration using an IPLink board with a network connector:

## 2. The DM3 IPLink Platform

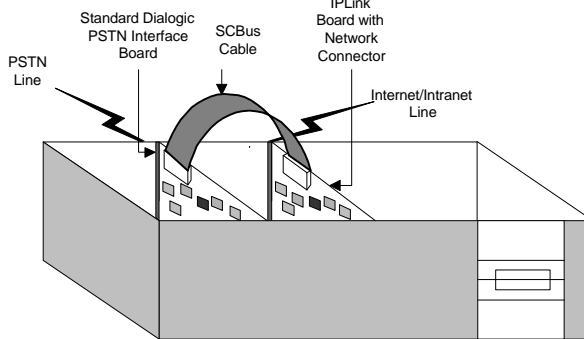


Figure 3. PSTN-IP Gateway Configuration (Network Interface on IPLink)

## 2.6. The Net Telephony Service Provider (NetTSP) Resource

### 2.6.1. NetTSP Resource Overview

The NetTSP resource provides complete embedded IP call control from within the DM3 platform. The NetTSP resource allows a host application to:

- Make a call
- Answer a call
- Manage established calls
- Gather call statistics
- Connect the IP network to the SCbus

### 2.6.2. NetTSP Resource Features

The NetTSP resource provides all the features necessary to rapidly build an Internet phone gateway:

- High-level API

### ***DM3 IPLink™ User's Guide for Windows NT***

- Call control capabilities (H.323)
- RTP/RTCP protocol
- Connectivity to IP Stack
- Family of voice coders
- Timeslot management

#### **2.6.3. NetTSP Resource Elements**

The NetTSP Resource contains the following component:

- NetTSC Component

The NetTSC component acts as a liaison between the host and all the sub-components in the NetTSP resource. It is the manager for all NetTSP services on the platform.

In addition, the NetTSP resource contains these additional components whose function is transparent to the host application:

- H.323 Component

Supplies the services required for sending voice over the Internet, e.g., call management.

- VSR (Voice Stream Resource) Component

Performs the transcoding between PCM audio to and from the PSTN and a digitally coded data stream to and from the Internet. The VSR contains the following main features:

- Voice Coders
- Tone Generator Component
- Signal Detector Component
- Echo Canceler Component
- Packet Loss Recovery Module

## 2. The DM3 IPLink Platform

Figure 4 shows the relationship among the IP Telephony components.

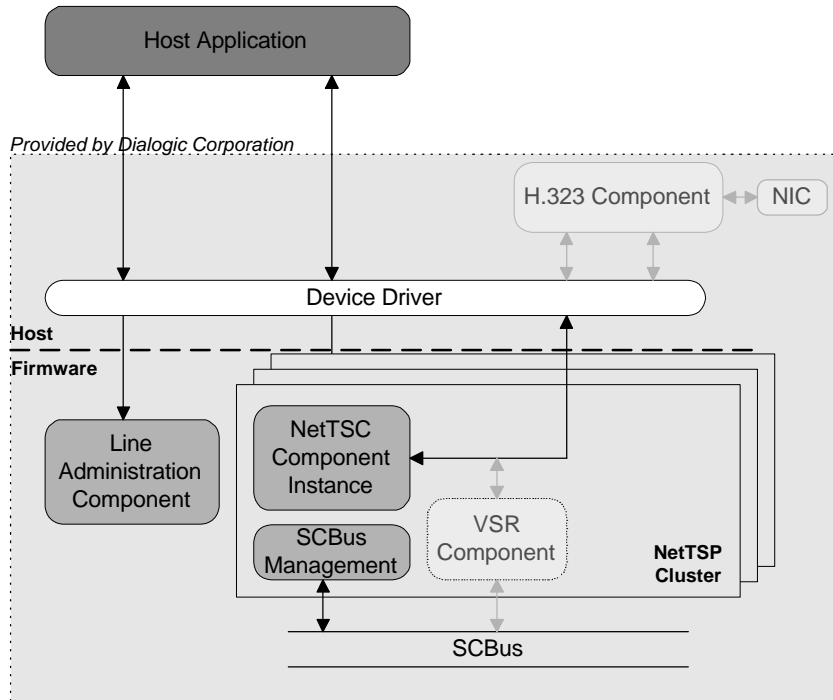


Figure 4. NetTSP Architecture - DM/IPLink-T1(E1) Board

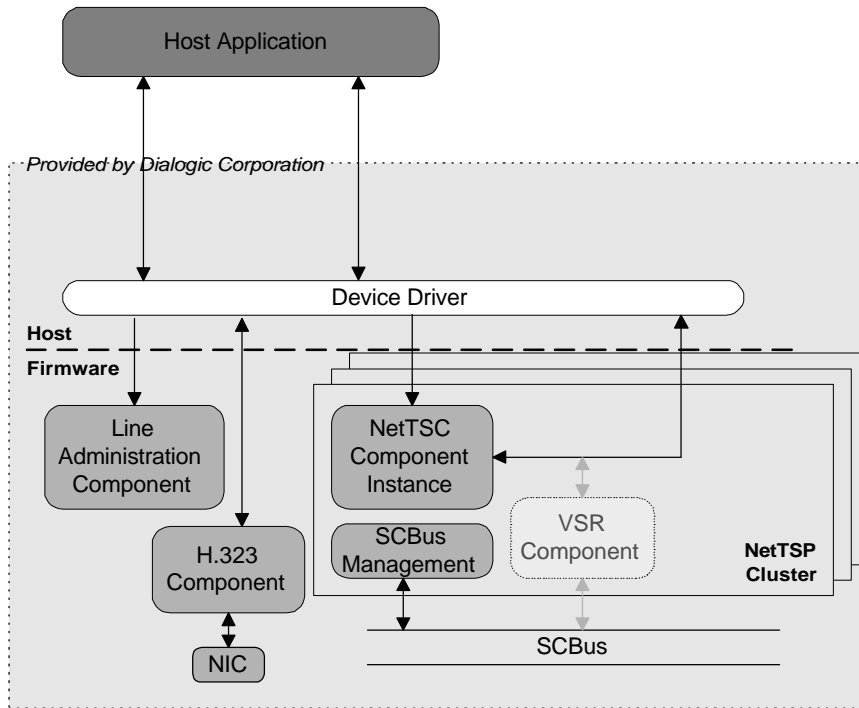


Figure 5. NetTSP Architecture - DM/IPLink-T1(E1)\_NIC Board

#### 2.6.4. NetTSC Component

The NetTSC component is the primary component of the NetTSP resource. The IPLink platform has a single NetTSC component and many instances. Each instance of the NetTSC component represents a single channel of call control.

Applications using the runtime features of the NetTSP resource communicate only with NetTSC component instances.

**Runtime features** are features enabled after instance initialization, such as call connection, statistic gathering, etc.



## 2. The DM3 IPLink Platform

Each NetTSC instance provides complete call control for inbound and/or outbound calls on its logical channel.

The NetTSC component acts as the liaison between the host and all the components of the NetTSP resource. The NetTSC component is the manager for all the TSP services on the platform.

The host application accesses the features of the NetTSP resource by communicating with an instance of the NetTSC component. *Chapter 6. Setting IPLink Parameters.*

### 2.6.5. NetTSC Component Activities

The NetTSC component performs the following activities:

- Interfaces to the NetTSP resource
- Creates clusters and allocates instances of the VSR component to each cluster (at initialization)
- Communicates with the H.323 component
- Provides information on Quality of Service (QoS) per call
- Manages Alarm Events

## 2.7. Line Administration Component

**NOTE:** The Line Administration Component is not implemented in this version of the IPLink Development Kit.

The Line Administration component collects statistics and provides information on board-level activities. The host application can communicate directly with this component to collect the required information.

### 2.7.1. Line Administration Component Activities

The Line Administration component performs the following activities:

- Statistic Collection (board level)

- Number of concurrent calls
- Maximum number of concurrent calls
- Average number of lost packets
- Network Alarms
  - Network disconnect

## **2.8. Internal Components**

**NOTE:** This section is informational only. It describes components that operate internally and are not accessible by the application.

### **2.8.1. Voice Stream Resource Component (VSR)**

The VSR component performs the transcoding between PCM audio to and from the PSTN and a digitally coded data stream to and from the Internet. VSR activities include:

- Encoding/Decoding (Voice Coder)
- Creating RTP packets
- Tone (DTMF) detection
- Tone (DTMF) generation
- Echo cancellation
- Lost packet recovery

The operation of the VSR resource is transparent to the user, except for optionally setting several parameters. See *Chapter 6. Setting IPLink Parameters* for a description of parameters that may be set by the application.

#### **Voice Coder**

The voice coder implements the bi-directional translation between a compressed speech format (on the IP network) and an 8 KHz linear stream of PCM audio samples (on the SCbus).

## 2. The DM3 IPLink Platform

Key features include:

- Low bandwidth
- High voice quality (3.5 MOS<sup>\*</sup> rating or better)
- Low latency

The IPLink platform supports multiple coders, allowing proprietary coders as well as new standard coders to be added to the system.

### **Echo Canceler**

The echo canceler removes the echo from the far signal channel.

Key features include:

- “Echo-tail” length of up to 64 ms.
- Programmable length

### **DTMF Tone Detector/Generator**

The IPLink platform provides a DTMF tone detection and suppression algorithm, and supports out-of-band as well as in-band DTMF handling.

### **Packet Loss Recovery Module (PLR)**

The IP Telephony solution includes a sophisticated packet loss recovery mechanism, enabling the system to maintain the highest quality of service.

### **2.8.2. H.323 Component**

The H.323 Protocol component provides the call signaling and management services as defined in the ITU-T recommendation H.323, *Visual Telephone*

---

\* Mean Opinion Score

***DM3 IPLink™ User's Guide for Windows NT***

*Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service.*

The H.323 component receives the IP media streams from the IP Network Interface, exchanging the stream contents with resources on the DM3 IPLink board.

Depending on the hardware configuration, the H.323 component can sit either on the host or on the DM3 board. In both cases it is transparent to the host application and communicates via the NetTSC component.

## 3. Host Application

---

Dialogic's IPLink platform provides a standards-based platform for developing Internet telephony applications. It is the user's responsibility to develop the host application according to the specific needs the application is addressing. Several guidelines are provided in this document, although they are not meant to be specific solutions to specific design problems. Before writing a host application Dialogic Corporation recommends that you thoroughly read this document, the *IPTGate Demo Guide* and run the IPTGate demo. Many suggestions for application design can be found in the demo.

Topics discussed in this chapter include:

- Host Application Responsibilities
- Host Application Communication with Dialogic Resources

### 3.1. Host Application Responsibilities

The host application is responsible for all of the connection control, directory services and session control processing, and all of the interaction between the NetTSP and the data network traffic as part of the data transfer. The IPLink platform can be used to build a wide variety of applications, meeting a multitude of user needs.

Following are some typical responsibilities of the host application.

- Overall system management
- PSTN and NetTSP control applications
- Routing

The host application communicates with the IPLink platform is via a high-level API, enabling the application writer to concentrate on application specific issues without having to master the complexities of the IPLink internal design.

### 3.2. Host Application Communication with Dialogic Resources

The host application communicates with the different Dialogic boards via the Dialogic APIs and other interfaces. Figure 6 presents a sample gateway configuration:

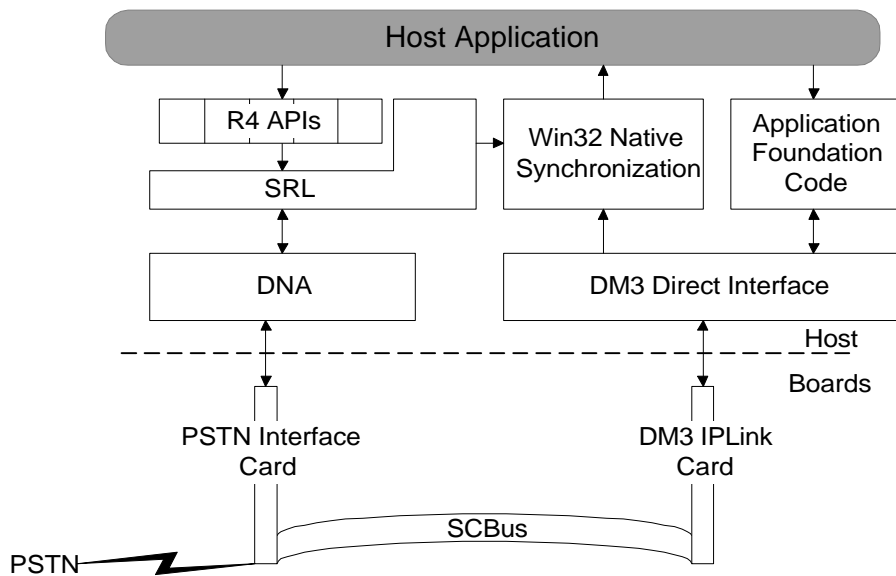


Figure 6. Typical Gateway

Host application access to DM3 devices on Windows NT is via the DM3 Direct Interface host library functions. The Direct Interface is used in conjunction with the Win32 API to produce highly native applications. The Direct Interface library provides a method for sending and receiving messages, as well as providing a full array of convenience functions for cluster and component management. The DM3 Mediastream Message Block (MMB) is the primary data structure used by an MTNI application to send and receive messages to and from the DM3 platform. See *Using the DM3 Direct Interface for Windows NT* and

### 3. Host Application

*DM3 Direct Interface Function Reference for Windows NT* for more information about the Direct Interface.

Host application access to the PSTN Interface Card is via familiar R4 APIs. Refer to the *Standard Runtime Library for Windows NT Programmer's Guide* for information about the standard Dialogic Corporation PSTN Card APIs.

#### 3.2.1. Event Reporting

Retrieving events is a key element in designing a WinNT application. In the Windows NT Native R4 environment, the application is notified about an event via the Standard Runtime Library (SRL). In the DM3 environment an application can use any Win32 native synchronization, such as I/O Completion Ports, to wait for event notification.

The Native Synchronization Methods (NSM) solution enables the eventing solution to have one synchronization point in the application for DM3, R4, and other Win32 devices. The NSM solution uses Windows NT native synchronization methods, such as I/O Completion Ports, together with SRL enhancements. Using this approach, DM3 is completely native and R4 devices report their events to this I/O Completion Port using SRL support for NSM.

See *Using NSM for Event Notification with DM3, R4, and Win32 Devices* for a detailed description of using I/O Completion Ports.





## 4. Programming Model

---

This chapter describes the programming model used by the NetTSP resource. It discusses the general philosophy and method of tracking events. It explains call-states and call-state transitions and how the NetTSC component uses these state-transitions to manage the call.

Topics discussed include:

- What are call-states?
- What are call-state transitions?
- Enabling detection of call-state changes

### 4.1. What are Call-States?

The NetTSP resource presents a call-control programming model based on the concept of **call-state**.

A **call** is a point-to-point multimedia communication between two Internet endpoints. The call begins with the call setup procedure and ends with the call termination procedure. The call consists of the collection of virtual channels between the endpoints.

A **call-state** is a clearly defined stage of a call's progression, for example, Initiated, Connected, Disconnected, Idle, etc.

A call is created in one of two ways:

- The host application issues a *TSC\_MsgMakeCall* command to initiate an outbound call.
- The H.323 stack notifies the NetTSC component instance that an incoming call is present. This results in a call being created. A call-state transition is generated from the Null state to the Offered state. The host application is notified of the new call via a *Std\_MsgEvtDetected* message about this transition to the Offered state and receives the call identifier for this call.

Each call moves from state to state as the call connection is established, connected, and concluded. The particular state a call is in determines what commands and actions may be performed. Such states are called **valid states** for a particular command. If a command is issued when the call is not in a valid state for that command an error occurs. The following table lists the call-states:

**Table 1. Call-States**

<b>Call State</b>	<b>Description</b>
<b>Accepted</b>	The call that was offered has been accepted by the host application. The host application has accepted responsibility in answering the call.
<b>Alerting</b>	The caller is notified that the call was accepted.
<b>Connected</b>	The calling and called parties are connected and the call is active on the related call channel. Information may be exchanged. In the case of an outbound call, this state indicates that the remote party has answered. In the case of an inbound call, this state indicates that the local party has answered the call. <b>This is the only state in which the audio path is established and open.</b>
<b>Disconnected</b>	The remote party has disconnected from the call.
<b>Failed</b>	The outbound call attempt was unsuccessful. The call attempt will transition to this state if it is determined to be unsuccessful due to rejection by the other side.
<b>Idle</b>	The local party to the call has disconnected or has been disconnected.
<b>Initiated</b>	The outbound call attempt has been initiated ( <i>TSC_MsgMakeCall</i> ).
<b>Null</b>	This indicates that the call has been released and all call information related to the call has been destroyed.

#### 4. Programming Model

Call State	Description
<b>Offered</b>	The inbound call is newly arrived and is being offered to the host application. Call information is available to the host application at this time in order for the host application to determine the appropriate action to take with regards to the call.
<b>Proceeding</b>	Part of the call setup. When the inbound call state transitions to Offered, the outbound state transitions to Proceeding.

#### 4.2. What are Call-State Changes?

Each call maintains a current state, transitioning from its current state to a new state for one of two reasons:

**Table 2. State Transitions**

Reason	Example
The host application requests a transition.	A call that is in the Offered state (an incoming call) will transition to the Connected state if the host application issues a <i>TSC_MsgAnswerCall</i> .
Changes occur on the network channel associated with the call via the H.323 connection.	A call that is currently in the Connected state will transition to the Disconnected state if the remote party disconnects.

Transition to a new state occurs when a message or event relating to a call is received by the NetTSC component instance. It may be triggered by either the host application or by the network on which the call exists. For example, if the host application sends a *TSC\_MsgDropCall* message, the state machine transitions from the Connected state to the Idle state. The host is notified via the *Std\_MsgEvtDetected* message. See Section 5.2. *Detecting Events* for more information about events.

### **4.3. Enabling Detection of Call-State Changes**

Call-state changes are reported to the application using the standard DM3 Run Time Control (RTC) mechanism.

Call-state changes (events) must be enabled before they can be reported to the application. The following procedure must be followed for each call-state to be enabled:

1. Send the following message to an appropriate NetTSC instance (*see Section 5.3. Allocating a NetTSP* for a description of allocating NetTSP instances) for each event to be registered:

- ***Std\_MsgDetectEvt***

The NetTSC component instance will send a ***Std\_MsgEvtDetected*** message each time its call state changes to one of those enabled by the application.

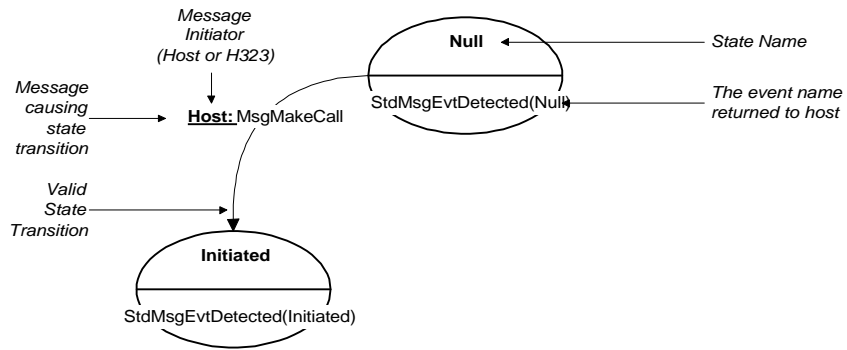
2. Repeat step 1 for each event to be enabled.
3. Repeat steps 1 and 2 for each additional NetTSC instance.

**NOTE:** An alternative to sending ***Std\_MsgDetectEvt*** for each event is to send ***Std\_MsgDetectxEvts*** listing all relevant events.

### **4.4. State Diagrams**

The following state diagrams describe the call-states for outbound (to the IP network) and inbound (from the IP network) calls. Each state is represented by an ellipse containing the state name and the ***Std\_MsgEvtDetected*** message sent to the host upon that state transition. The states are connected with arrows indicating the valid state-changes. Each state change includes the message used to cause that state change.

## 4. Programming Model



**Figure 7. State Diagram Explanation**

**NOTE:** These state diagrams are complex. Simplified versions will be shown in Chapter 6 for each call operation.

### 4.4.1. Outbound Internet Calls

The following state diagram describes the call-states and the state transitions when making an outbound call.

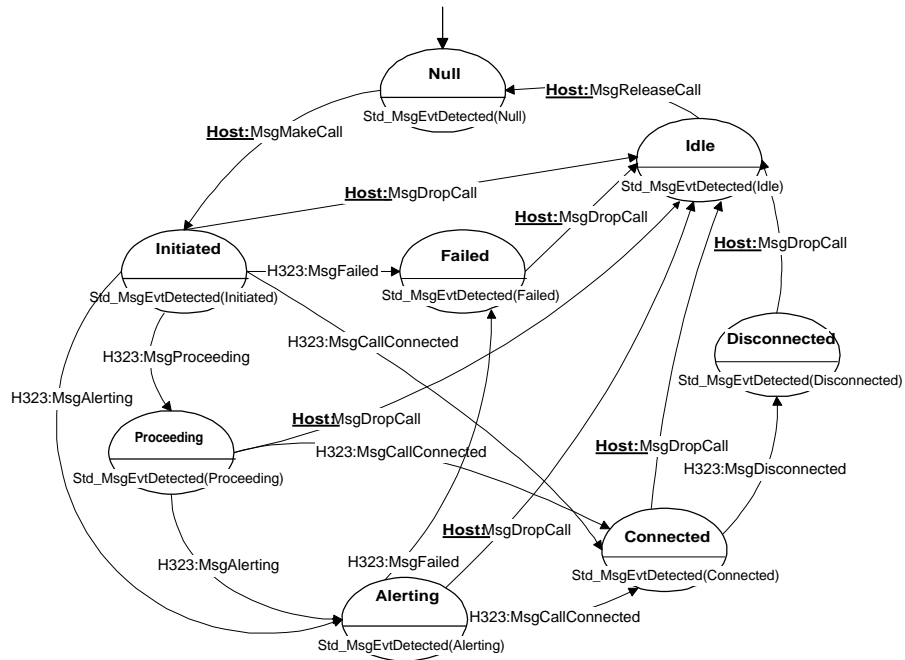


Figure 8. Outbound Call-State Diagram

#### 4.4.2. Inbound Internet Calls

The following state diagram describes the call-states and the state transitions when making an inbound call.

#### 4. Programming Model

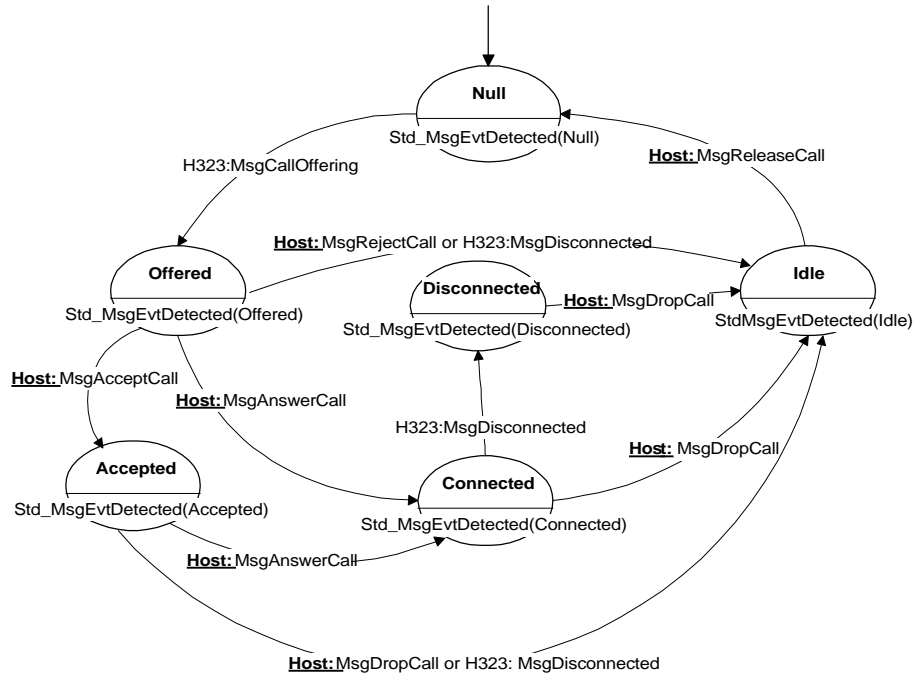


Figure 9. Inbound Call-State Diagram





## 5. Using the NetTSP Resource

---

This chapter explains how the DM3 messages (standard messages, TSC component messages, and NetTSC component messages) are used to implement the various features visible to the host application software. Typical call progressions are presented.

Topics discussed include:

- Messages
- Allocating a NetTSP
- Detecting Events
- Making an Outbound Call
- Receiving an Inbound Call
- Terminating Calls
- Getting Call Statistics
- Other Call Control Operations

### 5.1. Messages

Communication between the host application and the NetTSC component instances is via messages. This section briefly describes all the messages used by the NetTSC component.

#### 5.1.1. Standard Messages Used by the NetTSC Component

The following DM3 Standard Messages are used by the NetTSC component. Refer to *DM3 Standard Component Interface Messages* for a complete description of all the DM3 Standard Messages.

**Table 3. DM3 Standard Messages Used by the NetTSC Component**

<b>Command Message</b>	<b>Description</b>	<b>Reply Message (indicates success)</b>
<i>Std_MsgCancelAllEvts</i>	Cancels all RTC requests	<i>Std_MsgCancelAllEvtsCmplt</i>
<i>Std_MsgCancelEvt</i>	Cancel RTC request	<i>Std_MsgCancelEvtCmplt</i> <i>MsgCancelEvtCmplt</i> <i>MsgCancelEvtCmplt</i>
<i>Std_MsgCancelxEvts</i>	Cancels multiple RTC requests	<i>Std_MsgCancelxEvtsCmplt</i>
<i>Std_MsgComtest</i>	Ping message	<i>Std_MsgComtestCmplt</i>
<i>Std_MsgDetectEvt</i>	Detect RTC event	<i>Std_MsgDetectEvtCmplt</i>
<i>Std_MsgDetectxEvts</i>	Detect multiple RTC events	<i>Std_MsgDetectxEvtsCmplt</i>
<i>Std_MsgError</i>	Error detected	Not applicable
<i>Std_MsgEvtDetected</i>	Event detected	Not applicable
<i>Std_MsgExit</i>	Shuts down an instance in a proprietary way.	<i>Std_MsgExitCmplt</i>
<i>Std_MsgGetParm</i>	Read parameter value	<i>Std_MsgGetParmCmplt</i>
<i>Std_MsgInit</i>	Initialize resource	<i>Std_MsgInitCmplt</i>

## 5. Using the NetTSP Resource

<b>Command Message</b>	<b>Description</b>	<b>Reply Message (indicates success)</b>
<i>Std_MsgSetAllParamsDef</i>	Sets parameter values to default values	<i>Std_MsgSetAllParamsDefCmplt</i>
<i>Std_MsgSetParm</i>	Set parameter	<i>Std_MsgSetParmCmplt</i>
<i>Std_MsgSetParmDef</i>	Changes the value of a parameter	<i>Std_MsgSetParmDefCmplt</i>

### 5.1.2. TSC Messages Used by the NetTSC Component

The following TSC Messages are used by the NetTSC component. Refer to *the IPT Reference Guide* for a complete description of all the TSC Messages.

**Table 4. TSC Component Messages Used by the NetTSC Component**

<b>Command Message</b>	<b>Description</b>	<b>Reply Message (indicates success)</b>
<i>TSC_MsgAcceptCall</i>	Accepts the incoming call	<i>Std_MsgEvtDetected(Accepted)</i>
<i>TSC_MsgAnswerCall</i>	Answers the incoming call	<i>Std_MsgEvtDetected(Connected)</i>
<i>TSC_MsgDropCall</i>	Drops or disconnects a call	<i>Std_MsgEvtDetected(Idle)</i>

### DM3 IPLink™ User's Guide for Windows NT

<b>Command Message</b>	<b>Description</b>	<b>Reply Message (indicates success)</b>
<i>TSC_MsgGetCallInfo</i>	Requests stored information related to a call	<i>TSC_MsgGetCallInfoCmplt</i>
<i>TSC_MsgGetCallState</i>	Requests the current state of a call	<i>TSC_MsgGetCallStateCmplt</i>
<i>TSC_MsgMakeCall</i>	Initiates a call to a specified destination address  DestAddr is the IP address, rather than the phone number.	<i>TSC_MsgMakeCallCmplt</i>
<i>TSC_MsgReleaseCall</i>	Releases a call identifier	<i>Std_MsgEvtDetected(Null)</i>
<i>TSC_MsgRejectCall</i>	Rejects an incoming call	<i>Std_MsgEvtDetected(Idle)</i>

#### 5.1.3. NetTSC Component Messages

The following messages are specific to the NetTSC component.

## 5. Using the NetTSP Resource

**Table 5. NetTSC Component Messages**

<b>Command Message</b>	<b>Description</b>	<b>Reply Message (indicates success)</b>
<i>NetTSC_MsgSendUserInputIndication</i>	Puts DTMF input on the H.245 channel	<i>NA</i>
<i>NetTSC_MsgSendNonStandardCmd</i>	Sends non-standard data on the H.245 channel, between two Dialogic gateways	<i>NA</i>

### 5.2. Detecting Events

This section describes detecting events in detail.

#### 5.2.1. About Events

An **event** is an asynchronous message sent by a resource component to inform the host application of a change in state or other information.

### 5.2.2. Event Messages

The components and component instances of the NetTSC component use two standard messages to communicate information about events:

Standard Message	Description
<i>Std_MsgDetectEvt</i>	Performs two functions: <ul style="list-style-type: none"><li>• enables a component or component instance to detect the event specified by the message</li><li>• registers the host application to receive notification whenever the specified event occurs.</li></ul>
<i>Std_MsgEventDetected</i>	Alerts the host application registered for notification about this particular event.

The *Std\_MsgEventDetected* message is the only message that conveys event information. To accommodate a variety of events, the *Std\_MsgEventDetected* message has a variable body; the type of event that has occurred determines the type and number of fields inside the message.

Each DM3 component has its own set of valid event types. NetTSC component event types are discussed in *Section 5.2.4. Events Specific to the NetTSC Component.*

### 5.2.3. The Event-Reporting Process

In order for event reporting to occur, all these steps must be performed in the specified order:

## 5. Using the NetTSP Resource

1. The host application interested in a particular event must register for notification with the component or component instance that will encounter the event. Send a *Std\_MsgDetectEvt* message to the appropriate NetTSC component instance providing: the event type, e.g., *TSC\_EvtCallState\_Offered*; an appropriate Event Label (generated by the application); and the address to send the event to (i.e., the application's address) as the body of the message.
2. The specified event occurs.
3. The component instance that encountered the event sends a *Std\_MsgEvtDetected* event message to the host application. The type of event determines the body and fields of this message

### 5.2.4. Events Specific to the NetTSC Component

Table 6 shows the events specific to the NetTSC component. Depending on the event that has occurred, the body of the *Std\_MsgEvtDetected* message will contain specific fields related to that event. The shaded rows indicate that the event is not supported in the current revision of the IPLink development kit.

**Table 6. NetTSC Component Event Types**

<b>Event</b>	<b>Description</b>
<i>TSC_EvtCallInfo</i>	Delivers call-related information to the application as the information becomes available.
<i>TSC_EvtCallState</i>	Indicates a change in call state.
<i>NetTSC_EvtH245Data_NonStdCmd</i>	Notifies the application that non-standard data has been received on the H.245 channel.

Event	Description
<i>NetTSC_EvtH245Data_UsrInputIndication</i>	Notifies the application that the User-Input-Indication data has been received on the H.245 channel.
<i>NetTSC_EvtThresholdAlarm</i>	Notifies that an alarm occurs.
<i>NetTSC_EvtSystemFailed</i>	H.323 failure, e.g., NIC disconnect

### 5.3. Allocating a NetTSP

Before a call can be opened, a cluster must be allocated. This identifies, for the host application, which cluster is to manage the call.

#### 5.3.1. Procedure: Allocating a Cluster

The host application allocates a cluster by issuing a *mntClusterAllocate* message followed by an *mntClusterComplyAttribute* message.

The steps involved are as follows:

**Table 7. Procedure: Allocating a Cluster**

Step	Operation	Message	Description
1	Host allocates a NetTSP cluster	<i>mntClusterAllocate</i> (sent from application to Direct Interface)	Selects an H.323 cluster



## 5. Using the NetTSP Resource

Step	Operation	Message	Description
2	Host identifies the NetTSC component	<i>mntClusterCompyAttribute</i> (sent from application to Direct Interface)	Identifies the NetTSC component within the cluster.

### 5.4. Assigning Timeslots

In order for data to be transmitted to and received from the SCbus, timeslots must be assigned to the cluster's components. A transmit timeslot is automatically assigned to the NetTSC component during the download procedure. The receive timeslot must be assigned by the application. The procedure for assigning a receive timeslot to the NetTSC component is described here.

#### 5.4.1. Procedure: Assigning a Receive Timeslot to the NetTSC Component

The host application assigns a receive timeslot by issuing a *mntClusterTSAssign* message followed by a *mntClusterActivate* message.

The steps involved are as follows:

**Table 8. Procedure: Assigning a Timeslot**

<b>Step</b>	<b>Operation</b>	<b>Message</b>	<b>Description</b>
1	Host assigns a receive timeslot to a NetTSC instance	<i>xx_mntClusterTSAssign</i> (sent from application to Direct Interface)	Assigns a receive timeslot to a NetTSC component instance.
2	Host activates the connection	<i>mntClusterActivate</i> (sent from application to Direct Interface)	Activates the IN-port in the NetTSC component instance. This allows TDM data to flow from the SCbus into the NetTSC component instance.

#### **5.4.2. Querying About the SCbus Transmit Timeslot**

Components can receive data only when they are aware of the transmitting component's Tx timeslot. Each side must query the other to determine the transmit timeslot.

#### **Listening to the PSTN Card**

The NetTSC component must query the PSTN card to determine the transmit slot. The procedure is as follows:

## 5. Using the NetTSP Resource

**Table 9. Procedure: Querying the PSTN**

Step	Operation	Message	Description
1	Get the SCbus transmit timeslot assigned to the PSTN card	<i>xx_getXmitSlo (where xx indicates the type of PSTN card)</i>	The legacy card Tx slot is identified
2	Assign the timeslot to the NetTSP cluster's Rx port	<i>mntClusterTSAssign</i>	Assigns a receive timeslot to the NetTSC component instance.
3	Activate the connection	<i>mntClusterActivate</i>	Activates the IN-port in the NetTSP cluster. This allows TDM data to flow from the SCbus into the NetTSP cluster.

### Listening to the DM3 Card

The legacy card must query the NetTSP cluster to determine the Transmit slot. The procedure is as follows:

**Table 10. Procedure: Querying the DM3**

<b>Step</b>	<b>Operation</b>	<b>Message</b>	<b>Description</b>
1	Get the SCbus transmit timeslot assigned to the NetTSP cluster	<i>mntClusterSlotInfo</i>	The NetTSP cluster Tx slot is identified
2	Assign the timeslot to the legacy card Rx port	<i>xx_listen (where xx indicates the type of PSTN card)</i>	Assigns a receive timeslot to the legacy card.

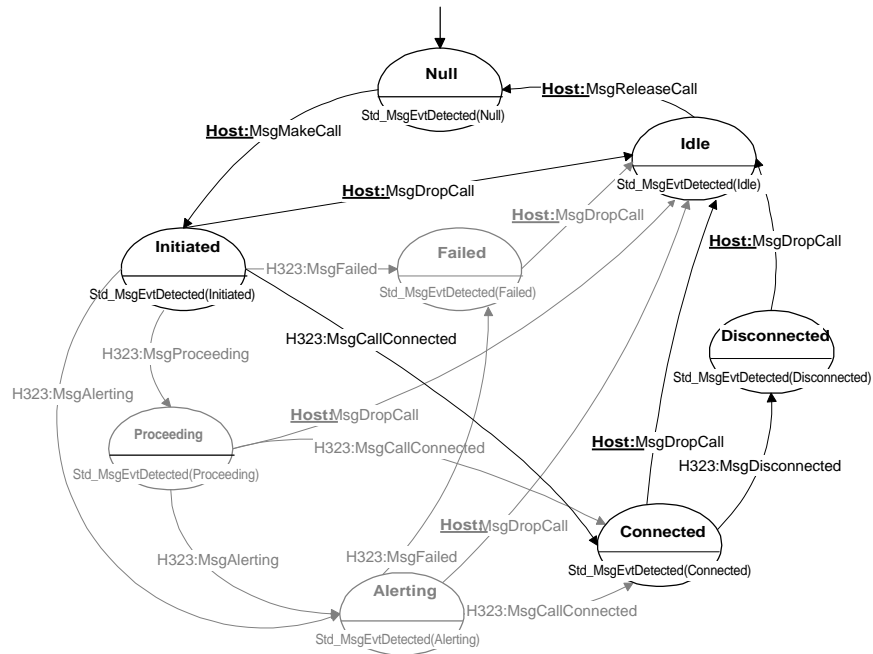
## 5.5. Making an Outbound Call

This section describes a typical procedure for making an outbound call to the IP Network. A simplified call session model is presented here, illustrating the call-state transitions: Null → Initiated → Connected → Idle → Null. Other states will be added in later sections.

**NOTE:** This section, and the sections that follow, present only one possible way of implementing these call control scenarios. It is the responsibility of the application developer to select the implementation most suitable for the particular application needs.

The state diagram below illustrates the state transitions for making an outbound call. See *Section 5.7. Terminating Calls* for an explanation of the call termination options.

## 5. Using the NetTSP Resource



**Figure 10. Outbound Call - Simplified State Machine**

### 5.5.1. Procedure: Making an Outbound Call

The host application places an outgoing call to the IP Network by issuing a *TSC\_MsgMakeCall* message. The Call State RTC mechanism is used to monitor the outcome of the call.

The steps involved are as follows:

Table 11. Procedure: Outbound Calls

Step	Operation	Message	Description	State
1	Host registers events	<i>Std_MsgDetectxEvts (event_list)</i>  (sent from application to NetTSC component instance)	All relevant events are registered with the kernel.	Null
2	Host initiates call	<i>TSC_MsgMakeCall</i>  (sent from application to NetTSC component instance)	The NetTSC component instance returns <i>TSC_MsgMakeCallCmplt</i> with the call ID. Logical channels are opened and data streams are allocated.  Note that a <i>Std_MsgError</i> message could be received instead, indicating that an error occurred trying to initiate the outgoing call.	Initiated
3	Host is notified of connection	<i>Std_MsgEvtDetected (Connected)</i>  (sent from NetTSC component instance to application)	Call-state transitions to Connected.	Connected

## 5. Using the NetTSP Resource

**NOTE:** The host application does not have to register all events. For example, in this call session the application might not want to be notified of a transition to the Initiated state. The application should then register only the Null, Connected, Disconnected, and Idle states. The state-transition to Initiated will occur without notifying the application. When the call-state transitions to Connected, a *Std\_MsgEvtDetected(Connected)* message is sent to the host application.

### 5.6. Receiving an Inbound Call

This section describes the procedure for receiving an inbound call from the IP Network. A simplified call session model is presented first, illustrating the call-state transitions: Null → Offered → Connected → Idle → Null.

When the host application receives the event message *Std\_MsgEvtDetected(Offered)*, indicating that an incoming call is being offered, it has a choice of operations on the call:

- **answering** the call by issuing a *TSC\_MsgAnswerCall* message  
This command instructs the NetTSC component instance to open the audio streams and allow conversation to begin. The call-state transitions to Connected.
- **accepting** a call by issuing a *TSC\_MsgAcceptCall* message  
This command instructs the NetTSC component instance to respond to the calling party while waiting to open the audio streams. The call-state transitions to an interim state – Accepted . This keeps the calling party from “timing-out” while the gateway performs some other action. A *TSC\_MsgAnswerCall* message must be sent to open the audio channels and enable conversation.

- **rejecting** the call by issuing a *TSC\_MsgRejectCall* message

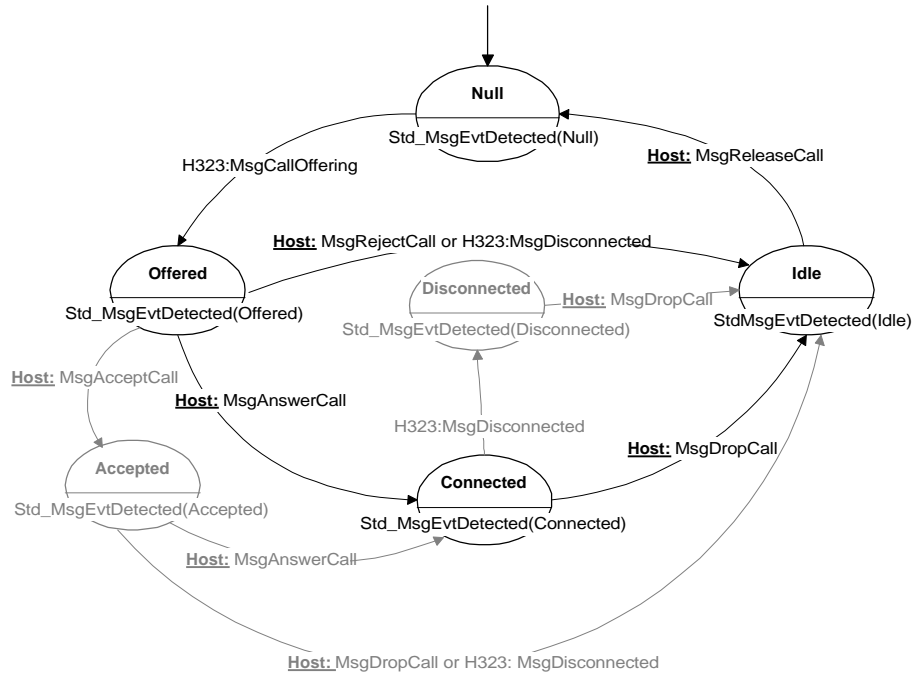
This command instructs the NetTSC component instance to reject the call. The call-state transitions to Idle. Call information is still available (e.g., calling party) until the *TSC\_MsgReleaseCall* message is sent. See *Section 5.9.1. Rejecting an Inbound Call* for a description of rejecting a call.

#### **5.6.1. Procedure: Answering an Inbound Call**

The state diagram below illustrates the state transitions for answering an inbound call. See *Section 5.7. Terminating Calls* for an explanation of the call termination options.



5. Using the NetTSP Resource



**Figure 11. Answer Inbound Call - Simplified State Diagram**

The host application receives incoming call notification via the Standard DM3 Event mechanism.

The steps involved are as follows:

**Table 12. Procedure: Answering Inbound Calls**

<b>Step</b>	<b>Operation</b>	<b>Message</b>	<b>Description</b>	<b>State</b>
1	Host registers events	<i>Std_MsgDetectxEvts (event_list)</i>  (sent from application to NetTSC component instance)	All relevant events are registered with the kernel.	Null
2	Host notified of incoming call	<i>Std_MsgEvtDetected (Offered, CallID)</i>  (sent from NetTSC component to application)	Call-state changed from Idle to Offered	Offered
3	Host answers call	<i>TSC_MsgAnswerCall (CallID, coder)</i>  (sent from application to NetTSC component instance)	Logical channels are opened and data streams are allocated.	Connected
4	Host is notified of connection	<i>Std_MsgEvtDetected (Connected)</i>  (sent from NetTSC component to application)	Call-state transitions to Connected	Connected

## 5. Using the NetTSP Resource

### Message Summary

The following figure illustrates the messages and call-states on either side of an Internet gateway for the above model.

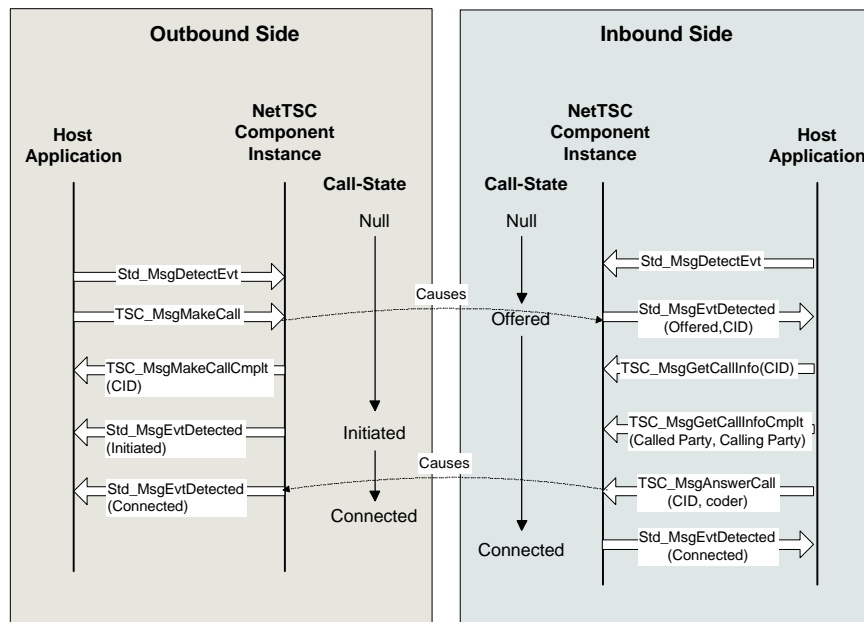


Figure 12. Call Establishment Messages - Simple Model

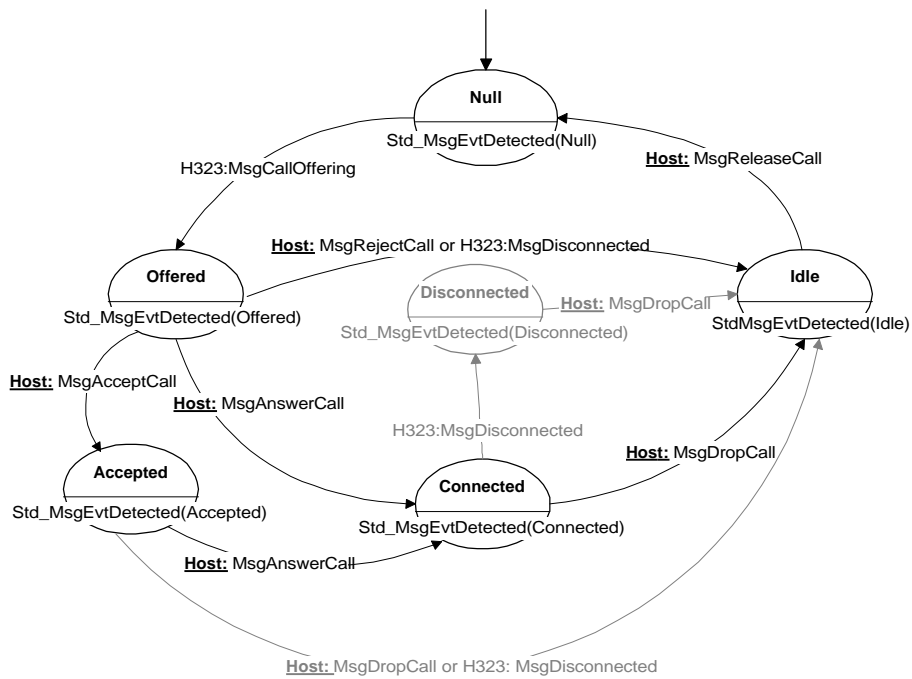
### 5.6.2. Procedure: Accepting a Call

A gateway may accept a call without opening the audio path. This allows the gateway to respond to the call offering, attempt to contact the destination address, and only after the destination answers the call, to open the audio path.

**DM3 IPLink™ User's Guide for Windows NT**

This feature allows the application developer to begin the billing procedure only when a call is actually connected.

The state diagram below illustrates the state transitions for the above inbound call. See Section 5.7. *Terminating Calls* for an explanation of the call termination options.



**Figure 13. Accept Inbound Call - Simplified State Diagram**

## 5. Using the NetTSP Resource

The host application receives incoming call notification via the Standard DM3 Event mechanism.

The steps involved are as follows:

**Table 13. Procedure: Accepting Inbound Calls**

Step	Operation	Message	Description	State
1	Host registers events	<i>Std_MsgDetectxEvts (event_list)</i>  (sent from application to NetTSC component instance)	All relevant events are registered with the kernel.	Null
2	Host notified of incoming call	<i>Std_MsgEvtDetected (Offered, CallID)</i>  (sent from NetTSC component to application)	Call-state changed from Idle to Offered	Offered
3	Host accepts call	<i>TSC_MsgAcceptCall (CallID)</i>  (sent from application to NetTSC component instance)	The inbound call is acknowledged, but not yet answered.  This interim state allows the gateway to perform other actions before connecting the call.	Accepted

**DM3 IPLink™ User's Guide for Windows NT**

<b>Step</b>	<b>Operation</b>	<b>Message</b>	<b>Description</b>	<b>State</b>
4	Host answers call	<i>TSC_MsgAnswerCall (CallID, coder)</i> (sent from application to NetTSC component instance)	Logical channels are opened and data streams are allocated.  <b>NOTE:</b> The audio path is opened only when <i>TSC_MsgAnswerCall</i> is sent.	Connected
5	Host is notified of connection	<i>Std_MsgEvtDetected (Connected)</i> (sent from NetTSC component to application)	Call-state transitions to Connected	Connected

**Message Summary**

The following figure illustrates the messages and call-states on either side of an Internet gateway for the above model.

## 5. Using the NetTSP Resource

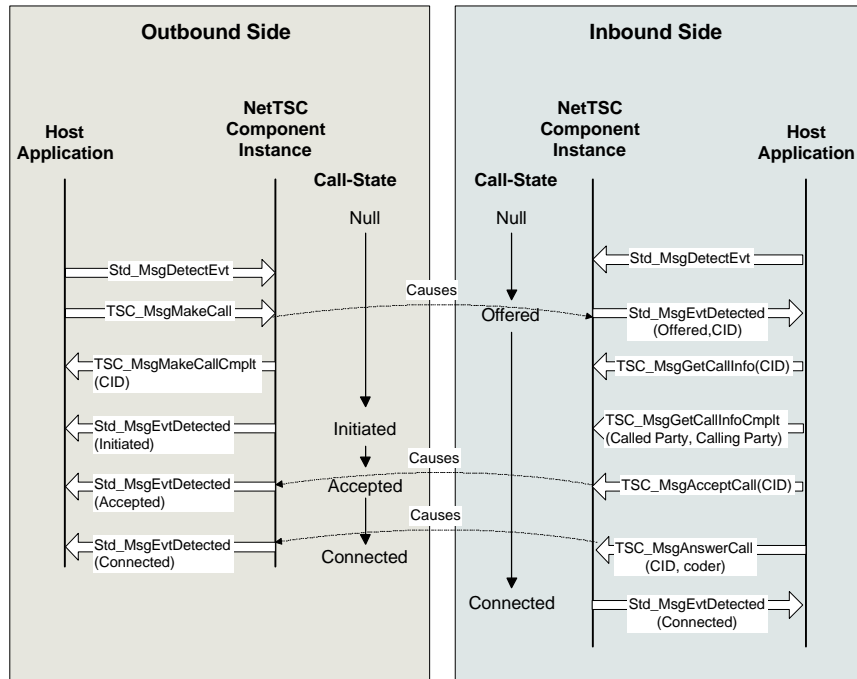


Figure 14. Call Establishment Messages - Accept Call

### 5.7. Terminating Calls

Once a call has been established (call-state = Connected), it can be terminated by either the host application (using call control operations) or the network (using H.323 Protocol termination).

The state diagram below illustrates the state transitions for terminating a call:

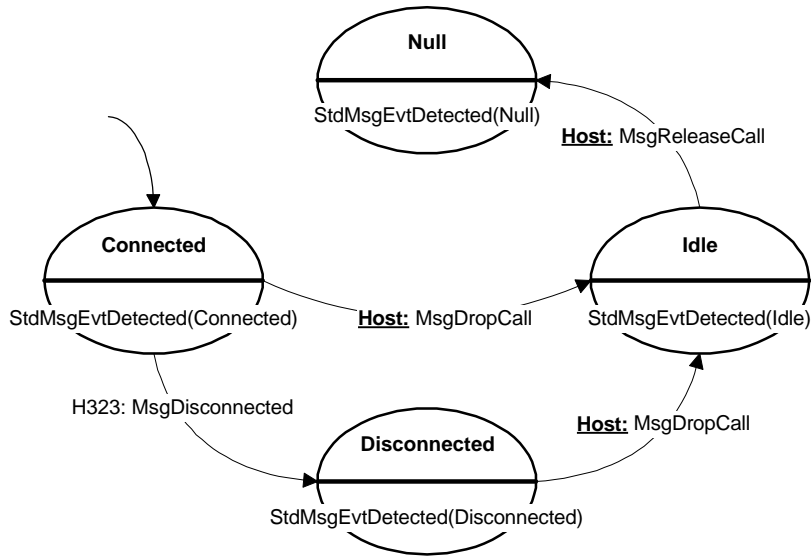


Figure 15. Call Termination - Simplified State Diagram

### 5.7.1. Procedure: Terminating a Call by the Host

The host application can terminate a call using the command *TSC\_MsgDropCall*. This will result in two actions:

- a call state transition to Idle
- *Std\_MsgEvtDetected(Idle)* is sent to the host application, if it is registered for this type of event notification.

**NOTE:** Call statistics are stored until the call state transitions to Null. See *Section 5.8. Getting Call Statistics* for the procedure for accessing the call statistics.



## 5. Using the NetTSP Resource

The steps involved are as follows:

**Table 14. Terminating a Call by the Host**

Step	Operation	Message	Description	State
1	Host application disconnects	<b><i>TSC_MsgDropCall</i></b> (sent from application to NetTSC component instance)	Tells the NetTSC instance to begin the sequence to disconnect the call.  The NetTSC instance returns <b><i>Std_MsgEvtDetected (Idle)</i></b> to the application upon successful completion of the disconnect.	Idle
2	Host clears the call	<b><i>TSC_MsgReleaseCall</i></b> (sent from application to NetTSC component instance)	CallID is cleared and the call state transitions to Idle.  The NetTSC instance returns <b><i>Std_MsgEvtDetected (Null)</i></b>	Null

### 5.7.2. Procedure: Call Terminated by the IP Network

The network can terminate the call while the call is in the Connected state. This will result in two actions:

- a call state transition to Disconnected
- ***Std\_MsgEvtDetected(Disconnected)*** is issued to the application to report the transition, if the application is registered for this type of event notification.

The steps involved are as follows:

**Table 15. Call Terminated by the IP Network**

Step	Operation	Message	Description	State
1	Network disconnects	<i>TSC_MsgDisconnected</i> (sent from H.323 component to NetTSC component instance)	Tells the NetTSC instance that the call has been disconnected.  The NetTSC instance returns <i>Std_MsgEvtDetected (Disconnected)</i> to the application.	Disconnected
2	Host application disconnects	<i>TSC_MsgDropCall</i> (sent from application to NetTSC component instance)	Tells the NetTSC instance to begin the sequence to disconnect the call.  The NetTSC instance returns <i>Std_MsgEvtDetected (Idle)</i> to the application upon successful completion of the disconnect.	Idle

### 5. Using the NetTSP Resource

Step	Operation	Message	Description	State
3	Host clears the call	<b><i>TSC_MsgReleaseCall</i></b> (sent from application to NetTSC component instance)	CallID is cleared and the call state transitions to Idle.  The NetTSC instance returns <b><i>Std_MsgEvtDetected (Null)</i></b>	Null

#### Message Summary

The following figure illustrates the messages and call-states on either side of an Internet gateway for the above model.

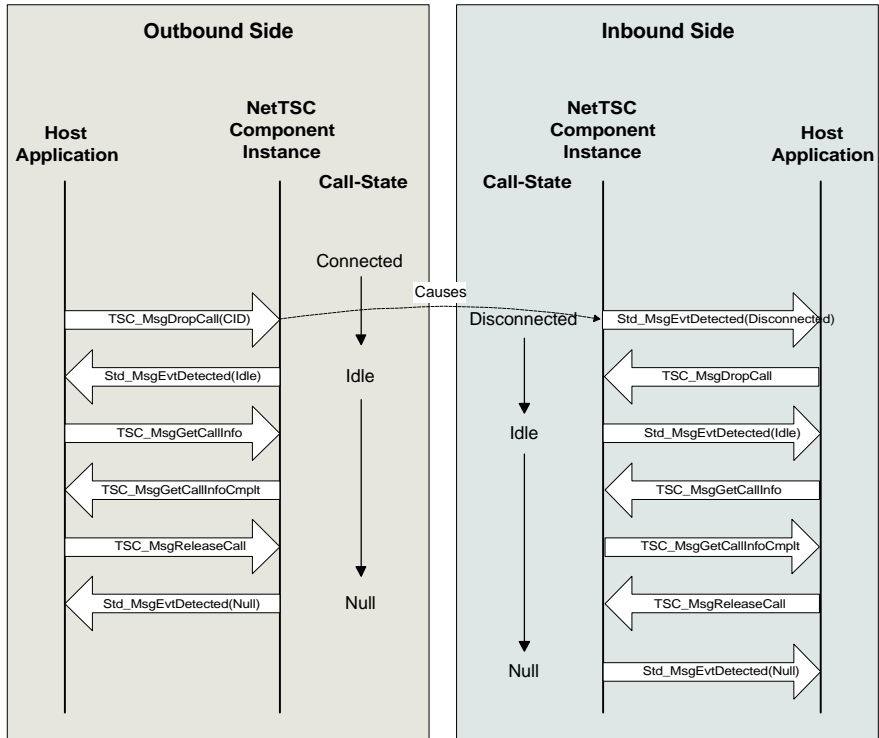


Figure 16. Call Termination Messages

## 5.8. Getting Call Statistics

The host application can get call statistics from the NetTSC component instance by issuing either a `TSC_MsgGetCallInfo` or a `TSC_MsgGetCallInfoExt` message. Information can be gathered at any point during a call, until the call state transitions to Null.

## 5. Using the NetTSP Resource

### 5.8.1. Procedure: Getting Call Statistics

The following procedure illustrates how to get call statistics at the end of a call.

**Table 16. Getting Call Statistics**

Step	Operation	Message	Description	State
1	Host application disconnects	<i>TSC_MsgDropCall</i> (sent from application to NetTSC component instance)	Tells the NetTSC instance to begin the sequence to disconnect the call.  The NetTSC instance returns <i>Std_MsgEvtDetected (Idle)</i> to the application upon successful completion of the disconnect.	Idle
2	Host requests call information	<i>TSC_MsgGetCallInfo</i> (sent from application to NetTSC component instance)	The host application request call information from the NetTSC instance.  The NetTSC instance returns <i>TSC_MsgGetCallInfoCmplt</i> with the requested information.	NA
3	Host clears the call	<i>TSC_MsgReleaseCall</i> (sent from application to NetTSC component instance)	CallID is cleared and the call state transitions to Idlez.  The NetTSC instance returns <i>Std_MsgEvtDetected (Null)</i>	Null

## 5.9. Other Call Control Operations

### 5.9.1. Rejecting an Inbound Call

This section describes the procedure for rejecting a call.

The state diagram below illustrates the state transitions for rejecting a call:

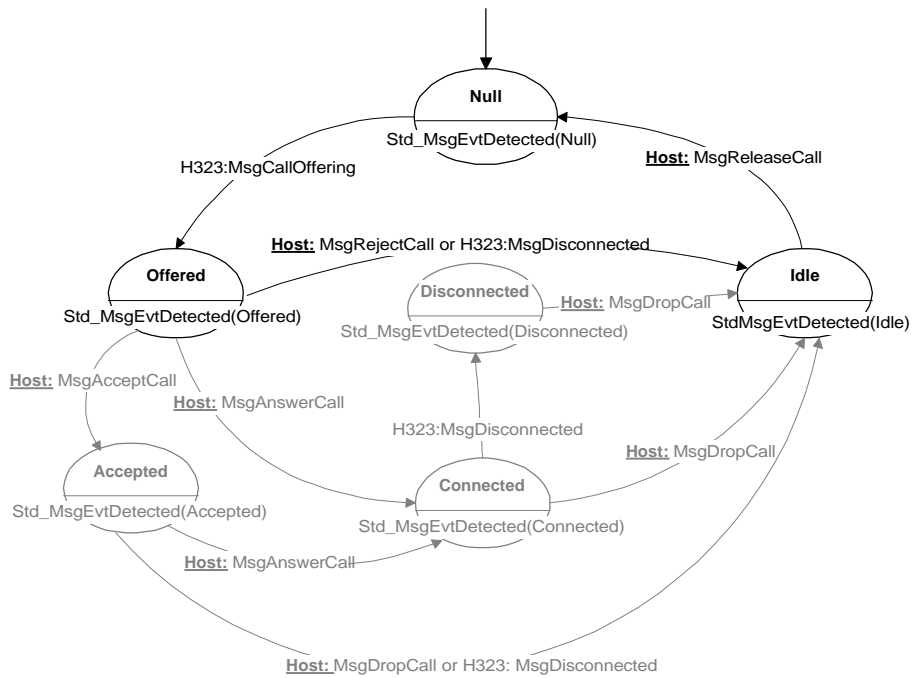


Figure 17. Rejecting an Inbound Call - Simplified State Diagram

## 5. Using the NetTSP Resource

The following procedure illustrates how to reject an inbound call:

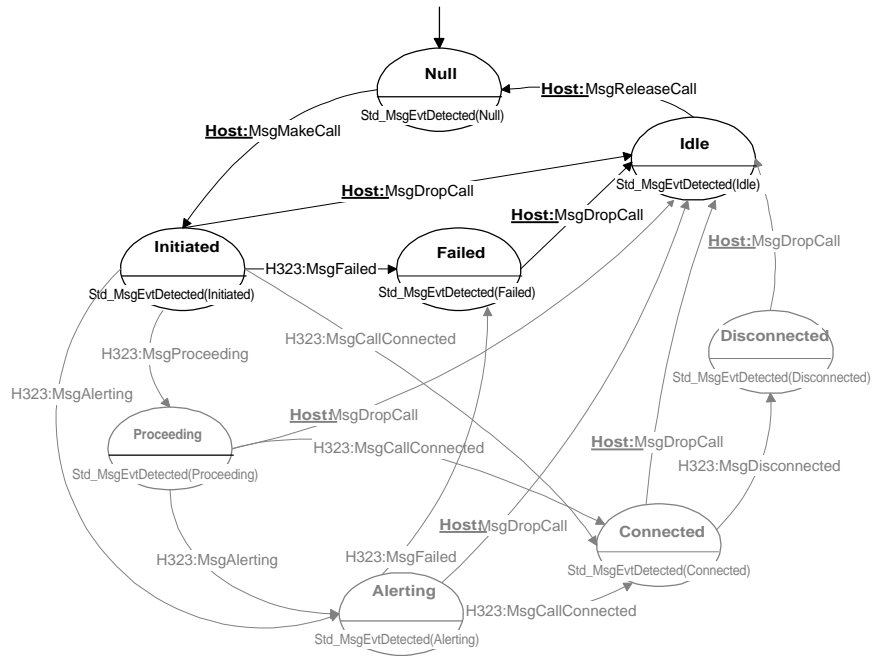
**Table 17. Rejecting an Inbound Call**

Step	Operation	Message	Description	State
1	Host notified of incoming call	<i>Std_MsgEvtDetected (Offered, CallID)</i> (sent from NetTSC component to application)	Call-state changed from Idle to Offered	Offered
2	Host rejects call	<i>TSC_MsgRejectCall</i> (sent from application to NetTSC component instance)	The inbound call is rejected.	Idle
3	Host releases call	<i>TSC_MsgReleaseCall</i> (sent from application to NetTSC component instance)	Call ID is cleared and the call state transitions to Idle.  The NetTSC instance returns <i>Std_MsgEvtDetected (Null)</i>	Null

### 5.9.2. Failed Outbound Call

This section describes what happens when an outbound call fails.

The state diagram below illustrates the state transitions when a call is rejected by the IP Network:



**Figure 18. Failed Outbound Call - Simplified State Diagram**

The following procedure illustrates how an outbound call fails, i.e., is rejected by the IP network:



## 5. Using the NetTSP Resource

**Table 18. Failed Outbound Call**

Step	Operation	Message	Description	State
1	Host registers events	<i>Std_MsgDetectxEvts</i> ( <i>event_list</i> )  (sent from application to NetTSC component instance)	All relevant events are registered with the kernel.	Null
2	Host initiates call	<i>TSC_MsgMakeCall</i>  (sent from application to NetTSC component instance)	The NetTSC component instance returns <i>TSC_MsgMakeCallCmplt</i> with the call ID. Logical channels are opened and data streams are allocated.  Note that a <i>Std_MsgError</i> message could be received instead, indicating that an error occurred trying to initiate the outgoing call.	Initiated

**DM3 IPLink™ User's Guide for Windows NT**

<b>Step</b>	<b>Operation</b>	<b>Message</b>	<b>Description</b>	<b>State</b>
3	Call is rejected by the IP network	<b><i>MsgFailed</i></b> (sent from the H.323 component to the NetTSC component instance)	The H.323 component receives a rejection from the IP Network and informs the NetTSC component instance.  The NetTSC component instance returns <b><i>Std_MsgEvtDetected (Failed)</i></b> to the host application.	Failed
4	Host application disconnects	<b><i>TSC_MsgDropCall</i></b> (sent from the host application to the NetTSC component instance)	Tells the NetTSC instance to begin the sequence to disconnect the call.  The NetTSC instance returns <b><i>Std_MsgEvtDetected (Idle)</i></b> to the application upon successful completion of the disconnect.	Idle
5	Host releases call	<b><i>TSC_MsgReleaseCall</i></b> (sent from application to NetTSC component instance)	Call ID is cleared and the call state transitions to Idle.  The NetTSC instance returns <b><i>Std_MsgEvtDetected (Null)</i></b>	Null

**5. Using the NetTSP Resource**



## 6. Setting IPLink Parameters

---

Various parameters may be set to configure the IPLink platform for an application's particular needs. This chapter discusses the content of each parameter and includes examples of how they are set in the parameter configuration files supplied by Dialogic Corporation.

Topics discussed include:

- SP parameters
- H.323 parameters
- Miscellaneous parameters
- Setting number of instances
- H.323 Stack configuration

### 6.1. FCD File

The Feature Configuration Description (FCD) file defines certain components that need to be configured with a unique set of parameters. The parameters are defined in the DM3 messages and sent to the components.

A default FCD file, *ipt.fcd*, is supplied with the IPLink platform software. The following sections describe the IPLink parameters and how to set them. There are three types of IPLink parameters set in the FCD file:

- SP parameters
- H.323 parameters
- Miscellaneous parameters

The relevant sections of the *ipt.pcd* file can be found in Appendix C.

### 6.1.1. SP Parameters

#### Echo Canceler

Two parameters can be set for the echo canceler:

- Enable/Disable

Parameter Name	Parameter Number
prmECActive	0x1b12

Possible values are:

0	Enable (Default)
1	Disable

- Number of taps (1 tap = 1/8 msec.)

The echo canceler can be set to sample up to 256 taps (32 msec).

**NOTE:** While the echo canceler is designed to support an “echo-tail” of up to 64 msec, due to testing limitations this parameter has been limited to a maximum value of 32 msec.

Parameter Name	Parameter Number
prmECOrder	0x1b13

Possible values are:

Any integer between:	32 – 256 (Default = 128)
----------------------	--------------------------

## 6. Setting IPLink Parameters

### Power Level

The power level parameter attenuates the power level of the encoded frames. It is computed by multiplying the encoded frame by the value of the PwrLvlCtrl parameter:

$$\text{PowerLevel} = \text{EncodedFrame} * \text{PwrLvlCtrl parameter}$$

Each sample of decoded PCM data is multiplied by this factor before being transmitted to the bus.

The default is 0.5.

Parameter Name	Parameter Number
prmPwrLvlCtrl	0x1b14

Possible values are:

Any number between:	0 – 0.999999 (0x0 – 0x7ffff)
Default value:	0.5 (0x400000)

**NOTE:** The parameter must assume a hexadecimal value corresponding to the DSP's 24 bit fractional representation.

To compute the value:

1. Choose the desired attenuation factor in the range 0.0 – 0.999999.
2. Multiply that value by  $2^{23}$ .

**DM3 IPLink™ User's Guide for Windows NT**

3. Convert that value to its hexadecimal representation and set as the VolumeControl parameter.

**Automatic Gain Control (AGC)**

The AGC maintains a uniform signal power level as the data is retrieved from the bus.

<b>Parameter Name</b>	<b>Parameter Number</b>
prmAGCActive	0x1b1c

Possible values are:

0	Enable (Default)
1	Disable

**High Pass Filter (HPF)**

The HPF removes DC and very low frequency corruption of the data.

<b>Parameter Name</b>	<b>Parameter Number</b>
prmHPFActive	0x1b1d

Possible values are:

0	Enable (Default)
---	------------------



## 6. Setting IPLink Parameters

1	Disable
---	---------

### 6.1.2. H.323 Parameters

#### Dialogic Enable

If set, enables Dialogic specific features. This parameter must be set to use the *TSC\_MsgNonStandardCmd* message

Parameter Name	Parameter Number
PrmDialogicEnable	0x1e19

**DM3 IPLink™ User's Guide for Windows NT**

Possible values are:

0	Standard Gateway(Default)
1	Dialogic Gateway

**Debug Print Levels**

Sets the debug print level for the:

- Stack module (PrmDebugLevelStack)
- Message module (PrmDebugLevelMsg)
- Stream module (PrmDebugLevelStream)
- State Machine module (PrmDebugLevelStates)
- Timer module (PrmDebugLevelTimer)
- Utilities module (PrmDebugLevelUtil)
- MNTI module (PrmDebugLevelMNTI)
- RV module (PrmDebugLevelRVSTACK)

<b>Parameter Name</b>	<b>Parameter Number</b>	<b>Default</b>
PrmDebugLevelStack	0x1e0e	2
PrmDebugLevelMsg	0x1e0f	3
PrmDebugLevelStream	0x1e10	0
PrmDebugLevelStates	0x1e11	3
PrmDebugLevelTimer	0x1e12	0
PrmDebugLevelUtil	0x1e13	3

## 6. Setting IPLink Parameters

Parameter Name	Parameter Number	Default
PrmDebugLevelMNTI	0x1e14	0
PrmDebugLevelRVSTACK	0x1e1e	0

Possible values are:

0	no printouts
1	fatal errors only
2	adds non-fatal error printouts
3	adds warning printouts
4	adds trace printouts

### 6.1.3. Miscellaneous Parameters

#### Compute Call Duration

This parameter specifies the method used to compute the call duration:

- When the NefTSC transitions to Connected state
- When the NefTSC transitions to Initiated state

Parameter Name	Parameter Number
PrmCallDurationComput	0x1e1a

**DM3 IPLink™ User's Guide for Windows NT**

Possible values are:

0 (default)	Compute call duration from the Connected state
1	Compute call duration from the Initiated state

**Set Number of Read Stream Buffers**

This parameter sets the number of packets buffered between the VSR component and the H.323 component. If there is excessive load on the host, or many interruptions on the IP network, you may need to set this parameter to a higher value. Otherwise, Dialogic recommends leaving it at its default value.

**NOTE:** This parameter is used only with DM/IPLink-T1 or DM/IPLink-E1 boards.

<b>Parameter Name</b>	<b>Parameter Number</b>
PrmNumOfReadStreamBuf	0x1e15

## 6. Setting IPLink Parameters

Possible values are:

Any integer between:	1 - 10 (Default = 1)
----------------------	----------------------

### Set Number of Write Stream Buffers

This parameter sets the number of packets buffered between the H.323 component and the VSR component. If there is excessive load on the host, or many interruptions on the IP network, you may need to set this parameter to a higher value. Otherwise, Dialogic recommends leaving it at its default value.

Parameter Name	Parameter Number
PrmNumOfWriteStreamBuf	0x1e16

**DM3 IPLink™ User's Guide for Windows NT**

Possible values are:

Any integer between:	1 - 40 (Default = 1)
----------------------	----------------------

**Setting the Number of Write DTMF Buffers**

This parameter sets the number of out-of-band DTMFpackets buffered between the VSR component and the H.323 component.

<b>Parameter Name</b>	<b>Parameter Number</b>
PrmNumOfWriteDTMFBuf	0x1e22

## 6. Setting IPLink Parameters

Possible values are:

Any integer between:	1 - 40 (Default = 10)
----------------------	-----------------------

### 6.2. PCD File

The Product Configuration Description (PCD) file describes the configurable parameters for one or more Processor Load Modules (PLMs) that are defined for a single platform or board. You can change the number of instances, depending on your IP telephony application configuration.

#### CAUTION

Do not change any settings in the *PCD* file, except for those explained here. The firmware will not function properly if other changes are made.

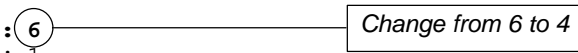
Use the following procedure to change the PCD file:

1. Locate the [CP] section
2. Change the **NumInstances** field for the following sections:
  - [COMP VSRP]
  - [COMP H323I]
  - [COMP NetTSC]
  - [COMP HOST]
  - [SP ONYX <1-6>] (change to <1-*n*>, where *n* = number of instances)

## DM3 IPLink™ User's Guide for Windows NT

For example, locate the [COMP VSRP] section. Change the **NumInstances** field from 6 channels to 4 channels.

```
[COMP VSRP]
{
  Attribute           : Std_ComponentType:0x1a
  NumInstances      : 6
  StartInstanceNum   : 1
  ConfigOption       : YES
  InitOption         : YES
  DependentComp      : CP TSC
}
```



A callout box with a black border and white background contains the text "Change from 6 to 4". A horizontal line extends from the left side of the box to a circle that encloses the number "6" in the configuration text above.

### 6.3. Config.val

You can configure the H.323 Stack Configuration file (*config.val*) to:

- Describe coder capabilities
- Enable a gatekeeper

These capabilities are broadcast during the capability exchange at call setup.

The entire *config.val* file can be found in Appendix B.

#### CAUTION

Do not change any settings in the *config.val* file, except for those explained here. The H.323 stack will not function properly if other changes are made.



## 6. Setting IPLink Parameters

### 6.3.1. Describe Coder Capabilities

Each supported coder's capabilities are described in the *config.val* file. G.711 and G.723.1 are currently designated as the default coders. You may adjust certain receive parameters for each coder.

#### G.711

You may set the maximum frame size that this coder can receive. It can be set for either 30, 20, or 10 msec. The larger frame size also allows the smaller frame sizes, i.e., the default 30 msec frame size also allows the 20 msec and 10 msec frame sizes. The line to be changed is highlighted in the code fragment below:

```
4 capabilityTable = 0
5 * = 0 #Sequence
6   capabilityTableEntryNumber = 1 # INTEGER [1..65535]
6   capability = 0
7   receiveAudioCapability = 0
8   g711Ulaw64k = 30 # INTEGER [1..256]
```

30 = 30,20,10 msec frames  
20 = 20,10 msec frames  
10 = 10 msec frame

#### G.723.1

You may designate the number of frames per packet that the coder can receive, and enable or disable silence suppression (VAD). As of the time of the writing of this guide, the IPLink platform can receive up to 8 frames per packet. See the Release Notes for any changes. The lines to be changed are highlighted in the code fragment below:

```
5 * = 0 #Sequence
6   capabilityTableEntryNumber = 2 # INTEGER [1..65535]
6   capability = 0
7   receiveAudioCapability = 0
8   g7231 = 0
```

### DM3 IPLink™ User's Guide for Windows NT

```
9      maxA1-sduAudioFrames = 1 # INTEGER [1..256]
9      silenceSuppression = 0 # INTEGER [1..256]
```

Enter an integer  
from 1 to 8

0 = Off  
1 = On

**NOTE:** The **capabilityTableEntryNumber** must be sequential starting from 1. If you make any changes, make sure that the numbers are correct. In addition, the **capabilityDescriptors** section must contain the same number of capabilities. Comment out non-applicable lines:

```
4  capabilityDescriptors = 0
5  * = 0 #Sequence
6  capabilityDescriptorNumber = 0 #INTEGER [1..255]
6  simultaneousCapabilities = 0
7  * = 0
8  * = 1 # INTEGER [1..65535]
8  * = 2 # INTEGER [1..65535]
#8  * = 3 # INTEGER [1..65535]
#8  * = 4 # INTEGER [1..65535]
```

#### 6.3.2. Enable a Gatekeeper

A gatekeeper serves as a directory of Internet addresses and telephone numbers. The IPLink platform can be configured to register with a gatekeeper. By default, the IPLink platform is configured for no gatekeeper as shown in the following fragment from the *config.val* file:

```
1  RAS = 0
2  responseTimeOut = 1
#2  gatekeeper = 1
2  manualRAS = 1
#2  manual Discovery = 0
#3  defaultGatekeeper = 0
#4  ipAddress = 0
#5  ip = <200.202.200.200>
#5  port = 1719
```

## 6. Setting IPLink Parameters

To register with a gatekeeper, comment line 4 (of this section of code). It should appear as follows:

```
1 RAS = 0
2 responseTimeOut = 1
#2 gatekeeper = 1
#2 manualRAS = 1
#2 manual Discovery = 0
#3 defaultGatekeeper = 0
#4 ipAdres = 0
#5 ip = <200.202.200.200>
#5 port = 1719
```

In addition, you must either insert a specific gatekeeper IP address, or enable discovery (broadcasting a request for a gatekeeper).

If you know the IP address of the gatekeeper you want to register with, uncomment lines 5 – 9 and insert the correct IP address in line 8. It should appear as follows:

```
1 RAS = 0
2 responseTimeOut = 1
#2 gatekeeper = 1
#2 manualRAS = 1
2 manual Discovery = 0
3 defaultGatekeeper = 0
4 ipAdres = 0
5 ip = <insert IP address of gatekeeper here>
5 port = 1719
```

If you want to broadcast a general request for a gatekeeper, you can choose to either:

- broadcast to those gatekeepers registered for Multicasting or,
- broadcast a general request to all gatekeepers

### **DM3 IPLink™ User's Guide for Windows NT**

To broadcast to a Multicast address, there is no need to change the default settings:

```
#Not applicable when manualDiscovery is enabled !
2  rasMulticastAddress = 0
3  ipAddress = 0
4  ip = <224.0.1.41>
4  port = 1718
#Broadcast address
#4  ip = <255.255.255.255>
#4  port = 1718
#station ras port (reply from gatekeeper)
2  rasPort = 1719
```

To broadcast to all gatekeepers, comment lines 4 and 5 and uncomment lines 7 and 8 as shown below:

```
#Not applicable when manualDiscovery is enabled !
2  rasMulticastAddress = 0
3  ipAddress = 0
#4  ip = <224.0.1.41>
#4  port = 1718
#Broadcast address
4  ip = <255.255.255.255>
4  port = 1718
#station ras port (reply from gatekeeper)
2  rasPort = 1719
```

The gateway may identify itself by name and/or by phone number. If you wish to identify your gateway by only one characteristic, comment out the pair of lines that are not used (lines 7/8 or lines 9/10):

```
2  registrationInfo = 0
3  terminalType = 0
4  mc = 0
4  undefinedNode = 0
4  terminal = 0
```

## 6. Setting IPLink Parameters

```
3 terminalAlias = 0
4 * = 0
5 h323-ID = "Insert Gateway Name Here"
4 * = 0
5 e164 = 'Insert Gateway Phone Number Here'!
```

**NOTE:** The *config.val* syntax requires that the gateway name string be contained within double quotes and the gateway phone number string be contained within single quotes, followed by an exclamation point.



## 7. Debugging

---

Several debugging utilities are supplied with the IPLink Development Kit. They are designed to supply information that can help you debug your application.

Debugging can be divided into two basic flows, depending on whether or not the download was successful. Refer to Figure 19. Debug Flow - Download Failed and Figure 22. Debug Flow - Download Succeeded for the basic debug procedure.

Each diagram contains a series of decision and process boxes. Each step of the procedure deals with one possible diagnostic operation that should be performed before continuing to the next step. Several of the steps include extra information contained in “call-outs” as a reminder what should be checked in that particular step.

**NOTE:** These flow diagrams do not show how to perform the diagnostic procedures in detail. Refer to the description in the following sections for a complete explanation of each diagnostic step.

The following sections describe the diagnostic flow in greater detail using the various utilities supplied with the IPLink platform software.

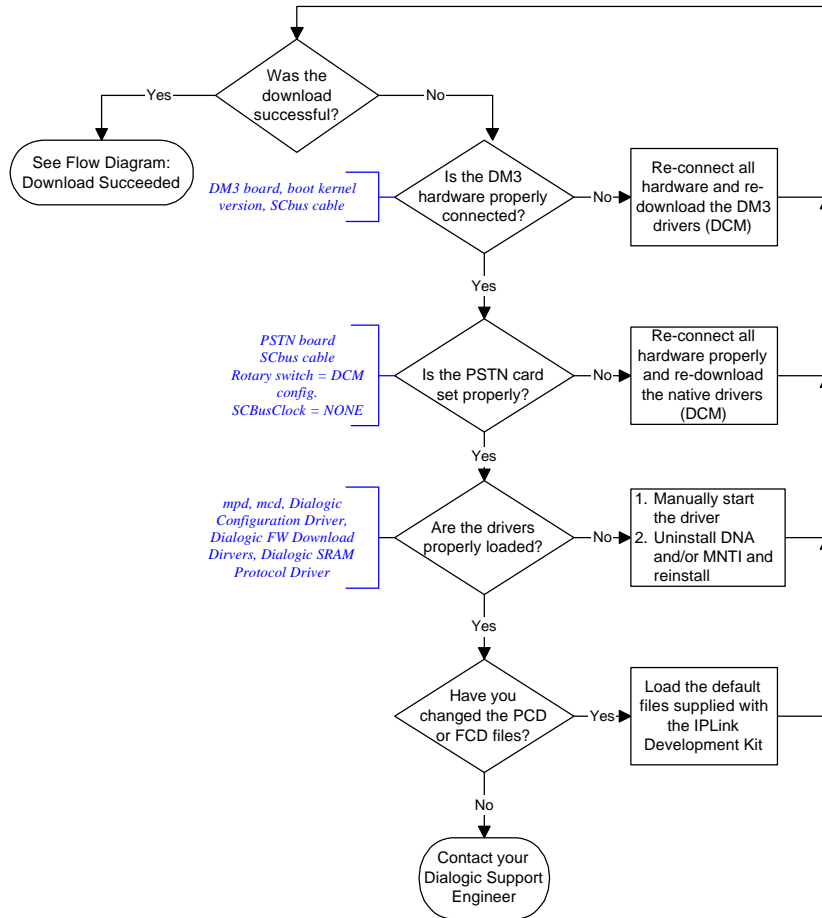
### 7.1. Download Failed

If the download failed, use the following debugging procedure:

1. Check the hardware
2. Check the drivers
3. Check the PCD/FCD files

Refer to Figure 19 as a guide to debugging:

**DM3 IPLink™ User's Guide for Windows NT**



**Figure 19. Debug Flow - Download Failed**



### 7.1.1. Check the Hardware

Check that all the hardware is properly connected:

1. Verify that the:
  - DM3 IPLink board is seated properly
  - DM3 boot kernel versions are correct: See the Release Catalog for the correct versions.

Open Dialogic Configuration Manager, double click on the IPT board and select the “Version Info” tab. The boot versions are listed.

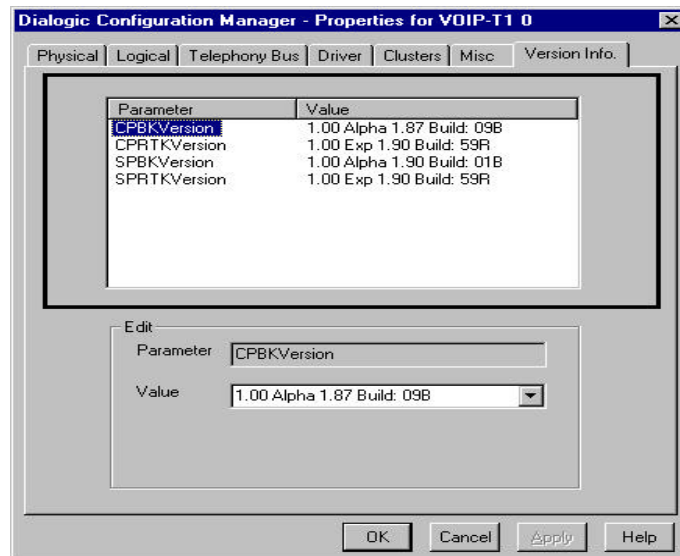


Figure 20. Boot Kernel Versions

***DM3 IPLink™ User's Guide for Windows NT***

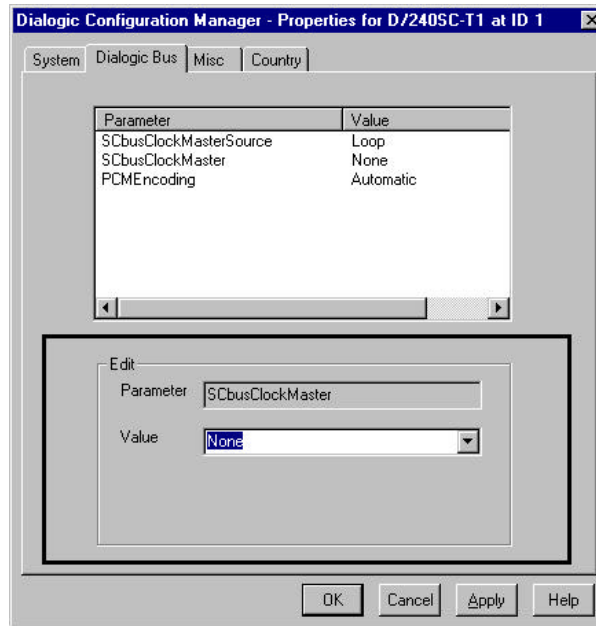
**NOTE:** If you cannot verify the boot kernel versions, you may not be able to debug your application. Contact your Dialogic Support Engineer.

2. Verify that the:

- PSTN board is seated properly
- Rotary switch on the PSTN board (if so configured) is set to the same value as in the DCM Configuration Manager.
- PSTN board is set as SCbusClockMaster = NONE:

Open the Dialogic Configuration Manager, double click on the PSTN card you are using and select the "Dialogic Bus" tab. If the SCbusClockMaster is set incorrectly, change the Value to "None".

## 7. Debugging



**Figure 21. SCbusClockMaster Parameter Setting**

3. Verify that the SCbus Cable is connected to both the DM3 IPLink board and the PSTN board.

### 7.1.2. Check the Drivers

1. Open the Windows Control Panel from the Start Icon.
2. Open the Devices folder. Check that the status of the following drivers is "Running":
  - MPD
  - MCD

### ***DM3 IPLink™ User's Guide for Windows NT***

- Dialogic Configuration Drivers
- Dialogic FW Download Drivers
- Dialogic SRAM Protocol Drivers

Manually start any drivers that are not running. If this fails, uninstall the firmware and then re-install it.

#### **7.1.3. Check the PCD/FCD files**

1. Verify that the PCD and FCD files are in your file structure and are pointed to correctly.
2. If you have changed any settings in either of the files, make sure that the changes are correct. Load the default files supplied with the IPLink Development Kit and check for proper operation of the application.

## **7.2. Download Succeeded**

If the download succeeded, use the following debug flow. You will check the output of various utilities. Each step consists of:

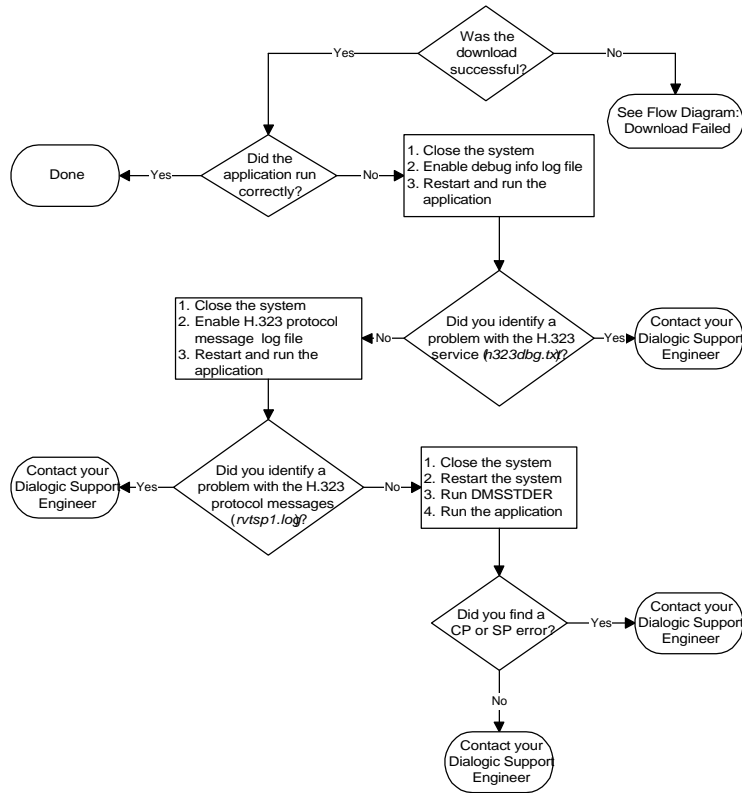
- closing the system
- enabling the log file
- re-running the application
- checking the log file

Use the following procedure for debugging your application:

1. Check for H.323 service errors
2. Check for H.323 protocol message errors
3. Check for CP and SP errors

Refer to Figure 22 as a guide to debugging:

**DM3 IPLink™ User's Guide for Windows NT**



**Figure 22. Debug Flow - Download Succeeded**

**7.2.1. H.323 Service**

By default, the log file *h323dbg.txt* is enabled in the FCD file to print both fatal and non-fatal errors.

## 7. Debugging

1. Close the system.
2. Re-configure the following parameters to print only fatal errors as described in Chapter 6. Setting IPLink Parameters:

- **PrmDebugLevelStack**
- **PrmDebugLevelMsg**
- **PrmDebugLevelStream**
- **PrmDebugLevelStates**
- **PrmDebugLevelTimer**
- **PrmDebugLevelUtil**
- **PrmDebugLevelMNTI**

This reduces the overhead on the CPU and reduces the size of the log file.

For example, change the PrmDebugLevelStack setting from 2 to 1::

```
! Setting NetTSC_PrmDebugLevelStack parameter to value 2 (Default)
! This parameter sets the debug print level for stack module of H323
! Values range: 0-4
!   0 sets no printouts
!   1 sets fatal errors only printouts
!   2 adds non-fatal errors printouts
!   3 adds warning printouts
!   4 adds trace printouts
!
MsgType      : 0x8
UInt32       : 0x1e0e
UInt32       : 0x2
}
```

Change from 0x2  
to 0x1

3. Restart the system and run the application.
4. If the problem doesn't show up in the log file, return to step 1 and reset the parameters to the next level.
5. Continue until the problem is found. If the problem is not found, continue to the next section.

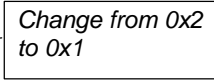
### **7.2.2. H.323 Protocol Messages**

By default, the log file *rvtsp1.log* is enabled in the FCD file to print fatal and non-fatal errors.

1. Close the system.
2. Re-configure the **PrmDebugLevelRVStack** parameter to print only fatal errors as described in Chapter 6. Setting IPLink Parameters. This reduces the overhead on the CPU and reduces the size of the log file.

For example, change the PrmDebugLevelRVStack setting from 2 to 1::

```
! Setting NetTSC_PrmDebugLevelRVStack parameter to value 2 (Default)
! This parameter sets the debug print level for RV stack module of H323
! Values range: 0-4
! 0 sets no printouts
! 1 sets fatal errors only printouts
! 2 adds non-fatal errors printouts
! 3 adds warning printouts
! 4 adds trace printouts
!
MsgType      : 0x8
UInt32       :
UInt32       : 0x2
}
```



3. Restart the system and run the application.
4. If the problem doesn't show up in the log file, return to step 1 and reset the parameter to the next level.
5. Continue until the problem is found. If the problem is not found, continue to the next section.

### **7.2.3. CP and SP Errors**

The DM3STDER utility is used to display CP and SP errors.



## 7. Debugging

**NOTE:** This utility must be invoked after a successful download, and before running the application.

After successfully downloading the firmware:

1. Open a DOS window.
2. Change to the \PATCH directory.
3. Invoke the DM3STDER utility by typing:

```
DM3STDER -b <board_number> -f <file_name>
```

where:

<board\_number> is the number of the DM3 board

<file\_name> is a file name for redirecting the data

For example, if there is only a single DM3 board, type:

```
DM3STDER -b0 -f dm3err.txt
```

4. Run the application.
5. If the problem is not found, contact your Dialogic Support Engineer.

### 7.3. Information for Customer Support

Have the following information ready when you contact you customer support engineer:

- Log files
- IPT SDK version
- MNTI version
- DNA version
- CP Boot Kernel version

***DM3 IPLink™ User's Guide for Windows NT***

- SP Boot Kernel version
- Platform Description
  - CPU type and speed
  - RAM
  - WindowsNT version
- Problem description

## 7. Debugging

The following table should be completed before contacting customer engineering:

<b>Description</b>	<b>Version Number/Other Information</b>
IPT SDK	
MNTI	
DNA	
CP Boot Kernel	
SP Boot Kernel	
CPU type and speed	
RAM	
WindowsNT version	
Problem description	



# Appendix A

---

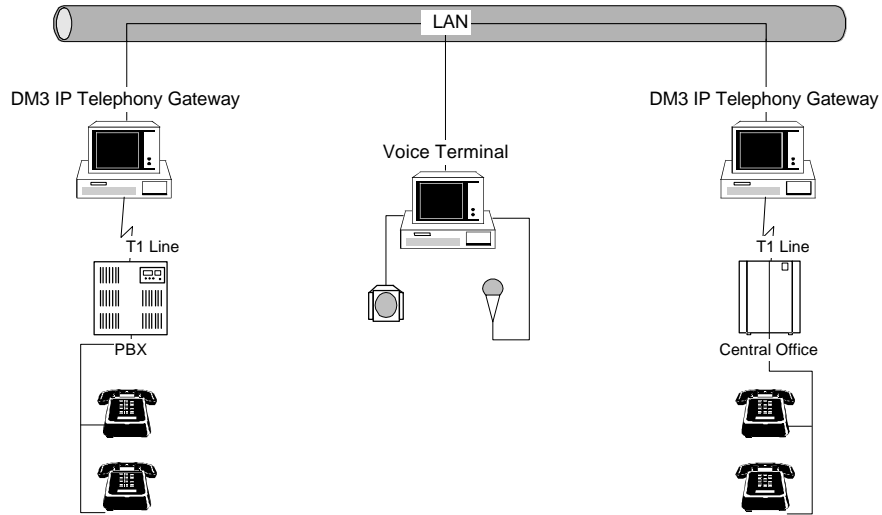
## Introduction to Internet Telephony

Internet Telephony is driving the convergence of the telephone network (PSTN) and the data network (Internet Protocol) into a single communication network that offers powerful, economical, new communications options. Conceptually, Internet telephone gateways provide the following services:

- **On one side, the gateway connects to the telephone world.** It can communicate with any phone in the world. A phone line plugs into the gateway on this end.
- **On the other side, the gateway connects to the Internet world.** It can communicate with any computer in the world that is connected to the Internet.
- **The gateway takes the standard telephone signal,** digitizes it (if it is not already digital), significantly compresses it, packetizes it for the IP, and routes it to a destination over the Internet.
- **The gateway reverses the operation** for packets coming in from the network and going out to the phone.
- **The gateway** also transfers and translates all the call control functions necessary for maintaining the call.
- **Both operations** (coming from and going to the phone network) take place at the same time, allowing a full duplex (two-way) conversation.

The figure below shows a typical configuration, utilizing a single gateway for Phone-to-PC operation and two gateways for Phone-to-Phone operation. Dialogic's IP Telephony solution provides the ability to configure a system for

both Phone-to-PC and Phone-to-Phone operation, with call control and call information retrieval built-in to the system.



**Figure 23. Typical IP Telephony Configuration**

A number of configurations can be built from this basic operation. Phone-to-PC or PC-to-phone operation can take place with one gateway. Phone-to-phone operation can occur with two gateways.

## **Standards**

Support for voice over the Internet, for both PCs and workstations, is rapidly being introduced into the marketplace. The VOIP Service Interoperability Implementation Agreement specifies baseline requirements so that products

complying with its specification can interoperate. The basic standards for Voice over the Internet are as follows:

- All connections are made using the H.323/H.245/H.225/Q.931 session protocol suite.

Dialogic's DM3 IPLink platform meets these standards for call establishment and call control. This feature is built-in "under-the-hood" and is invisible to the user. A set of high-level user interface functions simplify the task of creating a host application.

- Connections are made over TCP/UDP/IP protocol layers.

Dialogic's IPLink platform makes efficient use of the TCP/UDP/IP protocol layers. A sophisticated packet loss algorithm ensures a high quality of service.

- RTP is used to encapsulate real-time traffic.

Real-Time Transport Protocol (and its accompanying Real-Time Transport Control Protocol) are built-in to the IPLink platform. Packetization is completely handled by the internal components.

### **ITU Recommendation H.323**

Recommendation H.323 describes terminals, equipment, and services for multimedia communication over Local Area Networks (LAN) which do not provide a guaranteed Quality of Service. H.323 terminals and equipment may carry real-time voice, data, and video, or any combination, including videotelephony.

- An **H.323 terminal** is an endpoint on the LAN, capable of providing real-time, two-way communication with another H.323 terminal on another part of the LAN. A terminal, can therefore be either a multi-media PC connected to the LAN, or a gateway such as that described in this document.

### ***DM3 IPLink™ User's Guide for Windows NT***

The LAN over which H.323 terminals communicate may be a single segment or ring, or it may be multiple segments with complex topologies. The operation of H.323 terminals over the multiple LAN segments (including the Internet) may result in poor performance. The possible means by which quality of service might be assured on such types of LANs/internetworks is beyond the scope of this recommendation.

H.323 terminals may be integrated into personal computers or implemented in stand-alone devices such as videotelephones. Support for voice is mandatory, while data and video are optional, but if supported, the ability to use a specified common mode of operation is required, so that all terminals supporting that media type can interwork. H.323 allows more than one channel of each type to be in use. Other Recommendations in the H.323 series include H.225.0 packet and synchronization, H.245 control, H.261 and H.263 video codecs, G.711, G.722, G.728, G.729, and G.723.1 audio codecs, and the T.120 series of multimedia communications protocols.

H.323 makes use of the logical channel signaling procedures of Recommendation H.245, in which the content of each logical channel is described when the channel is opened. Procedures are provided for expression of receiver and transmitter capabilities, so transmissions are limited to what receivers can decode, and so that receivers may request a particular desired mode from transmitters.

The following elements are mandatory within the scope of H.323:

- The **Audio Codec** encodes the audio signal from the microphone for transmission and decodes the received audio code which is output to the loudspeaker.

H.323 specifies a number of audio codecs (G.711, G.722, G.723.1, MPEG1, G.728, and G.729). The standard allows the parties to negotiate the codec. Several of the specified codecs are especially relevant for Internet use. Other



codecs can be added, allowing proprietary codecs to be integrated into an Internet Gateway. Dialogic's IPLink platform supports multiple codecs, both standard and proprietary.

- The **System Control Unit** (H.245, H.225.0) provides signaling for proper operation of the H.323 terminal. It provides for call control, capability exchange, signaling of commands and indications, and messages to open and fully describe the content of logical channels.

This capability is built-in to the Dialogic IPLink platform. The application developer can easily access these features via high-level functions.

- **H.225.0 Layer** (H.225.0) formats the transmitted audio data and control streams into messages for output to the network interface, and retrieves the received audio and control streams from messages which have been input from the network interface. In addition, it performs logical framing, sequence numbering, error detection, and error correction.

H.225.0 makes use of **RTP/RTCP** (Real-time Transport Protocol/Real-Time Transport Control Protocol) for media stream packetization and synchronization for all underlying LANs.

This capability is built-in to the Dialogic IPLink platform. The application developer can easily access these features via high-level functions.

- The **LAN Interface** is implementation specific and is outside the scope of Recommendation H.323. However, the LAN interface shall provide the services described in Recommendation H.225.0. This includes:
  - Reliable (e.g., TCP, SPX) end-to-end service is mandatory for the H.245 Control Channel and the Call Signaling Channel.
  - Unreliable (e.g., UDP, IPX) end-to-end service is mandatory for the Audio Channels.

***DM3 IPLink™ User's Guide for Windows NT***

These services may be duplex or simplex, unicast or multicast depending on the application, the capabilities of the terminals, and the configuration of the LAN.

Dialogic's IPLink platform supports the reliable channels and ensures a high quality of service on the unreliable channel (UDP).

## Appendix B

### *config.val* File

---

# H.323 Stack Configuration File

```
1 system = 0
#2 pdlName = 'radvh323.raw'
2 allocations = 0
3 vtPoolSize = 100000
3 vtNodeCount = 20000
3 channels = 500
3 chanDescs = 5
3 messages = 200
3 nameChans = 10
3 tpktChans = 80
3 udpChans = 5
3 protocols = 400
3 maxProcs = 400
3 maxDepth = 200
```

```
1 RAS = 0
2 responseTimeout = 20
#2 gatekeeper = 1
2 manualRAS = 1
#2 manualDiscovery = 0
#3 defaultGatekeeper = 0
#4 ipAddress = 0
#5 ip = <200.200.200.200>
#5 port = 1719
```

```
2 registrationInfo = 0
3 terminalType = 0
4 mc = 0
4 undefinedNode = 0
4 terminal = 0
3 terminalAlias = 0
4 * = 0
5 h323-ID = "Name"
4 * = 0
5 e164 = '1111'!
```

#Not applicabile when manualDiscovery is enabled !

## DM3 IPLink™ User's Guide for Windows NT

```
2 rasMulticastAddress = 0
3   ipAddress = 0
4   ip = <224.0.1.41>
4   port = 1718
#BroadCast address
#4   ip = <255.255.255.255>
#4   port = 1718
#station ras port(reply from gatekeeper)
2 rasPort = 1719

1 Q931 = 0
2 responseTimeOut = 50
2 connectTimeOut = 500
2 callSignalingPort = 1720
2 maxCalls = 10
#2 notEstablishControl = 0
2 manualAccept = 1

1 h245 = 0
2 masterSlave = 0 # Sequence <Optional>
3   terminalType = 70 # INTEGER [0..255]
#3   manualOperation = 0 # NULL
3   timeout = 40

2 capabilities = 0 # Sequence <Optional>
3   timeout = 40
#3   manualOperation = 0
3   terminalCapabilitySet = 0 # Sequence
4   sequenceNumber = 0 # INTEGER [0..255]
4   protocolIdentifier = [00] # Object Identifier [0..0]

4   multiplexCapability = 0
5   h2250Capability = 0 # Sequence
6     maximumAudioDelayJitter = 60 # INTEGER [0..1023]
6     receiveMultipointCapability = 0 # Sequence
7     multicastCapability = 0 # BOOLEAN [0..0]
7     multiUniCastConference = 0 # BOOLEAN [0..0]
7     mediaDistributionCapability = 0
8     * = 0 # Sequence
9     centralizedControl = 0 # BOOLEAN [0..0]
9     distributedControl = 0 # BOOLEAN [0..0]
9     centralizedAudio = 0 # BOOLEAN [0..0]
9     distributedAudio = 0 # BOOLEAN [0..0]
9     centralizedVideo = 0 # BOOLEAN [0..0]
9     distributedVideo = 0 # BOOLEAN [0..0]
6     transmitMultipointCapability = 0 # Sequence
```

## Appendix B

```
7      multicastCapability = 0 # BOOLEAN [0..0]
7      multiUniCastConference = 0 # BOOLEAN [0..0]
7      mediaDistributionCapability = 0
8      * = 0 # Sequence
9      centralizedControl = 0 # BOOLEAN [0..0]
9      distributedControl = 0 # BOOLEAN [0..0]
9      centralizedAudio = 0 # BOOLEAN [0..0]
9      distributedAudio = 0 # BOOLEAN [0..0]
9      centralizedVideo = 0 # BOOLEAN [0..0]
9      distributedVideo = 0 # BOOLEAN [0..0]
6      receiveAndTransmitMultipointCapability = 0 # Sequence
7      multicastCapability = 0 # BOOLEAN [0..0]
7      multiUniCastConference = 0 # BOOLEAN [0..0]
7      mediaDistributionCapability = 0
8      * = 0 # Sequence
9      centralizedControl = 0 # BOOLEAN [0..0]
9      distributedControl = 0 # BOOLEAN [0..0]
9      centralizedAudio = 0 # BOOLEAN [0..0]
9      distributedAudio = 0 # BOOLEAN [0..0]
9      centralizedVideo = 0 # BOOLEAN [0..0]
9      distributedVideo = 0 # BOOLEAN [0..0]
6      mcCapability = 0 # Sequence
7      centralizedConferenceMC = 0 # BOOLEAN [0..0]
7      decentralizedConferenceMC = 0 # BOOLEAN [0..0]
6      rtcpVideoControlCapability = 0 # BOOLEAN [0..0]
6      mediaPacketizationCapability = 0 # Sequence
7      h261aVideoPacketization = 0 # BOOLEAN [0..0]

4      capabilityTable = 0
5      * = 0 # Sequence
6      capabilityTableEntryNumber = 1 # INTEGER [1..65535]
6      capability = 0
7      receiveAudioCapability = 0
8      g711Ulaw64k = 30 # INTEGER [1..256]

5      * = 0 # Sequence
6      capabilityTableEntryNumber = 2 # INTEGER [1..65535]
6      capability = 0
7      receiveAudioCapability = 0
8      g7231 = 0
9      maxAl-sduAudioFrames = 1 # INTEGER [1..256]
9      silenceSuppression = 1 # INTEGER [1..256]

4      capabilityDescriptors = 0
5      * = 0 # Sequence
6      capabilityDescriptorNumber = 0 # INTEGER [0..255]
```

### **DM3 IPLink™ User's Guide for Windows NT**

```
6     simultaneousCapabilities = 0
7     * = 0
8     * = 1 # INTEGER [1..65535]
8     * = 2 # INTEGER [1..65535]

2 channels = 0

3 * = 0 # Sequence
4 name = 'g711Ulaw10' # IA5String [1..128]
4 dataType = 0
5 audioData = 0
6 g711Ulaw64k = 10 # INTEGER [1..256]

3 * = 0 # Sequence
4 name = 'g711Ulaw20' # IA5String [1..128]
4 dataType = 0
5 audioData = 0
6 g711Ulaw64k = 20 # INTEGER [1..256]

3 * = 0 # Sequence
4 name = 'g711Ulaw30' # IA5String [1..128]
4 dataType = 0
5 audioData = 0
6 g711Ulaw64k = 30 # INTEGER [1..256]

3 * = 0 # Sequence
4 name = 'g7231' # IA5String [1..128]
4 dataType = 0
5 audioData = 0
6 g7231 = 0
7 maxAl-sduAudioFrames = 1 # INTEGER [1..256]
7 silenceSuppression = 1 # INTEGER [1..256]
```

## Appendix C

### *ipt.fcd* File

---

```
;;; FCD File generated from : qvs_t1.config
;;; And appended to for the IPT NetTSC
;;;
;;; Search down for [NetTSC] to see beginning of additions
;;; which were typed in (not generated)
;;;
;;; Note that for each ["component"]
;;; [SendMsg] sends the parameter
;;; and then [RcvMsg] verifies that it was received

*****
*
[NOTE: This file illustrates only the IP Telephony
parameters.]
*****
*

[NetTSC]
{
    Attribute          : 0:0x1e

! NetTSC_PrmDebugLevelStack
(0=OFF,1=FATAL,2=ERROR,3=Warning,4=Info)
;; SetParam (7694,2) on NetTSC
    [SendMsg]
    {
! Setting parameter for NetTSC.0
        MsgType       : 0x8    ;8
        UInt32        : 0x1e0e  ;7694
        UInt32        : 0x2    ;2
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType       : 0x9    ;9
        TimeOut       : 0x1388  ;5000
    }

! NetTSC_PrmDebugLevelMsg
(0=OFF,1=FATAL,2=ERROR,3=Warning,4=Info)
;; SetParam (7695,2) on NetTSC
```

### DM3 IPLink™ User's Guide for Windows NT

```
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8    ;8
  UInt32       : 0x1e0f ;7695
  UInt32       : 0x2    ;2
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9    ;9
  TimeOut      : 0x1388 ;5000
}

! NetTSC_PrmDebugLevelStream
(0=OFF,1=FATAL,2=ERROR,3=Warning,4=Info)
;; SetParam (7696,0) on NetTSC
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8    ;8
  UInt32       : 0x1e10 ;7696
  UInt32       : 0x0    ;0
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9    ;9
  TimeOut      : 0x1388 ;5000
}

! NetTSC_PrmDebugLevelStates
(0=OFF,1=FATAL,2=ERROR,3=Warning,4=Info)
;; SetParam (7697,4) on NetTSC
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8    ;8
  UInt32       : 0x1e11 ;7697
  UInt32       : 0x4    ;4
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9    ;9
}
```



## Appendix C

```
        TimeOut      : 0x1388    ;5000
    }

! NetTSC_PrmDebugLevelTimer
(0=OFF,1=FATAL,2=ERROR,3=Warning,4=Info)
;; SetParam (7698,0) on NetTSC
  [SendMsg]
  {
! Setting parameter for NetTSC.0
    MsgType      : 0x8      ;8
    UInt32       : 0x1e12   ;7698
    UInt32       : 0x0      ;0
  }

  [RcvMsg]
  {
! Expecting parm Complete from NetTSC.0
    MsgType      : 0x9      ;9
    TimeOut      : 0x1388   ;5000
  }

! NetTSC_PrmDebugLevelUtil
(0=OFF,1=FATAL,2=ERROR,3=Warning,4=Info)
;; SetParam (7699,2) on NetTSC
  [SendMsg]
  {
! Setting parameter for NetTSC.0
    MsgType      : 0x8      ;8
    UInt32       : 0x1e13   ;7699
    UInt32       : 0x2      ;2
  }

  [RcvMsg]
  {
! Expecting parm Complete from NetTSC.0
    MsgType      : 0x9      ;9
    TimeOut      : 0x1388   ;5000
  }

! NetTSC_PrmDebugLevelMNTI (0-3=OFF,4=Info)
;; SetParam (7700,0) on NetTSC
  [SendMsg]
  {
! Setting parameter for NetTSC.0
    MsgType      : 0x8      ;8
```

**DM3 IPLink™ User's Guide for Windows NT**

```
        UInt32      : 0x1e14 ;7700
        UInt32      : 0x0    ;0
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType     : 0x9    ;9
        TimeOut     : 0x1388 ;5000
    }

! NetTSC_PrmDebugLevelRVSTACK (0-3=OFF,4=Info)
;; SetParam (7710,0) on NetTSC
    [SendMsg]
    {
! Setting parameter for NetTSC.0
        MsgType     : 0x8    ;8
        UInt32      : 0x1e1e ;7710
        UInt32      : 0x0    ;0
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType     : 0x9    ;9
        TimeOut     : 0x1388 ;5000
    }

! NetTSC_PrmDialogicEnable (0=Standard Gateway,1=Dialogic
Gateway)
;; SetParam (7705,1) on NetTSC
    [SendMsg]
    {
! Setting parameter for NetTSC.0
        MsgType     : 0x8    ;8
        UInt32      : 0x1e19 ;7705
        UInt32      : 0x1    ;1
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType     : 0x9    ;9
        TimeOut     : 0x1388 ;5000
    }
}
```

## Appendix C

```
! prmEActive (0=Enable,1=Disable)
;; SetParam (6930,0) on NetTSC
  [SendMsg]
  {
! Setting parameter for NetTSC.0
    MsgType      : 0x8      ;8
    UInt32       : 0x1b12   ;6930
    UInt32       : 0x1      ;0
  }

  [RcvMsg]
  {
! Expecting parm Complete from NetTSC.0
    MsgType      : 0x9      ;9
    TimeOut      : 0x1388   ;5000
  }

! prmEOrder (128/256)
;; SetParam (6931,128) on NetTSC
  [SendMsg]
  {
! Setting parameter for NetTSC.0
    MsgType      : 0x8      ;8
    UInt32       : 0x1b13   ;6931
    UInt32       : 0x80     ;128
  }

  [RcvMsg]
  {
! Expecting parm Complete from NetTSC.0
    MsgType      : 0x9      ;9
    TimeOut      : 0x1388   ;5000
  }

! prmVolumeControl (1-8)
;; SetParam (6932,2) on NetTSC
  [SendMsg]
  {
! Setting parameter for NetTSC.0
    MsgType      : 0x8      ;8
    UInt32       : 0x7fffff ;.999999
    UInt32       : 0x0      ;0
  }
}
```

**DM3 IPLink™ User's Guide for Windows NT**

```
[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9    ;9
  TimeOut      : 0x1388 ;5000
}

! prmSQM_1
;; SetParam (6919,2) on NetTSC
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8    ;8
  UInt32       : 0x1b07 ;6919
  UInt32       : 0x6    ;6
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9    ;9
  TimeOut      : 0x1388 ;5000
}

! prmSQM_2
;; SetParam (6918,2) on NetTSC
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8    ;8
  UInt32       : 0x1b08 ;6918
  UInt32       : 0xf    ;15
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9    ;9
  TimeOut      : 0x1388 ;5000
}

! prmSQM_3
;; SetParam (6927,2) on NetTSC
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8    ;8
}
```

## Appendix C

```
        UInt32      : 0x1b0f    ;6927
        UInt32      : 0x78      ;120
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType     : 0x9       ;9
        TimeOut     : 0x1388    ;5000
    }

! prmsQM_4
;; SetParam (6928,100) on NetTSC
    [SendMsg]
    {
! Setting parameter for NetTSC.0
        MsgType     : 0x8       ;8
        UInt32      : 0x1b10    ;6928
        UInt32      : 0x3c      ;60
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType     : 0x9       ;9
        TimeOut     : 0x1388    ;5000
    }

! prmsQM_5
;; SetParam (6918,0) on NetTSC
    [SendMsg]
    {
! Setting parameter for NetTSC.0
        MsgType     : 0x8       ;8
        UInt32      : 0x1b06    ;6918
        UInt32      : 0x2       ;2
    }

    [RcvMsg]
    {
! Expecting parm Complete from NetTSC.0
        MsgType     : 0x9       ;9
        TimeOut     : 0x1388    ;5000
    }

! prmsQM_6
;; SetParam (6917,2) on NetTSC
```

**DM3 IPLink™ User's Guide for Windows NT**

```
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8      ;8
  UInt32       : 0x1b05   ;6917
  UInt32       : 0x4      ;4
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9      ;9
  TimeOut      : 0x1388   ;5000
}

! prmSQM_7
;; SetParam (6933,1) on NetTSC
[SendMsg]
{
! Setting parameter for NetTSC.0
  MsgType      : 0x8      ;8
  UInt32       : 0x1b15   ;6933
  UInt32       : 0x1      ;1
}

[RcvMsg]
{
! Expecting parm Complete from NetTSC.0
  MsgType      : 0x9      ;9
  TimeOut      : 0x1388   ;5000
}
}
```

## Glossary

---

**Asynchronous mode** An operating mode of certain DM3 kernel or host library function calls. Asynchronous mode is a non-blocking mode that is typically used when a function involves operation on a remote processor (e.g. a host library function that initiates a kernel operation on a DM3 platform's Control Processor) or when data movement via the MMA and global memory is required. In asynchronous mode a value signifying "pending" status is immediately returned to the calling function, and the actual completion or failure of the operation is reported to the caller via a DM3 result message.

**Cluster** A collection of DM3 components that share a set of TDM timeslots on the network interface or the SCbus. Components are bound to a particular cluster and its assigned timeslots in an allocation operation. The component that contains the ports to the TDM timeslots (generally the first component allocated into the cluster) is called the central component.

**Codec** see COder/DECoder

**COder/DECoder** A circuit used on Dialogic boards to convert analog voice data to digital and digital voice data to analog audio

**Component** An executable entity that resides on a particular processor on the DM3 platform and provides a single type of service to a DM3 resource. For example, a decoder component may be part of the Player resource, residing on a signal processor (SP) and providing specific decoding services for mediastream playback. Currently, the object code for a DM3 component is linked with the DM3 and RTOS kernels and downloaded to the target processor as a single Processor Load Module (PLM). The DM3 architecture supports multiple instances of components to allow independent simultaneous handling of multiple media streams.

**Component instance** A logical entity that represents a single thread of control for the operations associated with a DM3 component. Each DM3 component supports multiple instances up to some defined

## **DM3 IPLink™ User's Guide for Windows NT**

maximum number (which may be greater than the number of instances that may actually be implemented with existing platform hardware). Instances are addressable units, and DM3 messages may be sent to individual instances of a component. An instance generally corresponds to an individual media stream, and when an instance is active it is usually bound to a particular set of SCbus data timeslots and to a SCSA Group.

**Computer Telephony (CT)** Adding computer intelligence to the making, receiving, and managing of telephone calls.

**Control Processor (CP)** A processor that is present on the motherboard of every DM3 platform for the management of the SCbus, for host communication configuration, and to implement some of the features of DM3 Resources. All current DM3 platforms use an Intel i960 as the Control Processor.

**CP** See Control Processor

**CT** See Computer Telephony

**DM3 address** A bitmapped data structure that identifies a specific DM3 software entity and the processor, board, and/or node where the entity resides. DM3 addresses are used to identify components, component instances, clusters, and tasks, and partially specified addresses may be used to identify the board or processor. Each DM3 message contains source and destination component addresses, and many function calls require one or more component addresses as arguments.

**DM3 architecture** A platform architecture for a family of high density, high performance, call processing products. The architecture specifies both the hardware platform and the firmware modules in such a way that resources are highly portable across multiple platform implementations, including VME, PCI, and CompactPCI variants.

**DM3 kernel** Firmware that is present on each processor on a DM3 platform to support configuration management, host communication, inter processor communication and control, and SCSA firmware services among others. There may be more than one version of the DM3 kernel firmware depending on the type of processor (e.g. CP vs. DSP or RISC signal processor) and the intended application of each processor.



**DM3 load module** A loadable block of executable firmware for a particular processor on a DM3 platform. A DM3 Resource may be packaged as one or more DM3 load modules.

**DM3 message** A formatted block of data that is passed between the host and a DM3 component or between two DM3 components via the global memory and MMA ASIC. All DM3 messages have a packed, fixed-format header that contains transport information (source and destination addresses, priority, amount of data in the message body, etc.) and an arbitrary transaction ID number as well as the message type, which is the primary content of the message. The DM3 software environment provides macros to put values into and get values from the packed message header independent of the processor word length and endianness. In addition to the fixed-format header, messages may optionally have a body that contains additional data in typed fields that are accessed using macros or low-level function calls.

**DM3 platform** A specific DM3 hardware implementation (a board) for a particular system bus, which includes a control processor (CP), optional signal processors (SPs), and optional peripherals (e.g. communication daughterboard for development or DNI daughterboard for additional network interfaces).

**DM3 signal computing resource** Also “DM3 technology resource” or simply “DM3 resource”. A software component which may be deployed on compatible platforms based upon the DM3 architecture. A resource is a logical entity that supports a number of instances of a fixed set of features. The DM3 resource maps closely, though not exactly, with the SCSA Resource paradigm. For example, a DM3 Player resource may be a DM3 resource providing playback capability; it may carry out some or all of the functionality defined in the SCSA Player API. A DM3 Resource is typically made up of a DM3 host-side library and one or more DM3 firmware resource components (or simply “DM3 components”).

**DM3 task** An executable entity which is managed and scheduled by the DM3 kernel and underlying real-time operating system that is present on all of the processors on the DM3 platform. Each task maintains its own environment and has its own stack. Each DM3 task implements a message queue that services one or more components or component instances.

**DTMF** See Dual-Tone Multi-Frequency

**Dual-Tone Multi-Frequency** A way of signaling consisting of a push-button or touch-tone dial that sends out a sound consisting of two discrete tones that are picked up and interpreted by telephone switches (either PBXs or central offices).

**Endpoint** An H.323 Gateway, CMA client, LDAP server, or CMA server. An endpoint can call and be called. It generates and/or terminates information streams.

**FCD file** An ASCII file that lists any non-default parameter settings that are necessary to configure a DM3 hardware/firmware product for a particular feature set. The downloader utility reads this file, and for each parameter listed generates and sends the DM3 message necessary to set that parameter value.

**Feature Configuration File** See FCD file

**Frame** 1) A set of SCbus timeslots which are grouped together for synchronization purposes. The period of a frame is fixed (at 125  $\mu$ sec) so that the number of time slots per frame depends on the SCbus data rate.  
2) In the context of DSP programming (e.g. DM3 component development), the period defined by the sample rate of the signal data.

**Gatekeeper** An H.323 entity on the Internet that provides address translation and control access to the network for H.323 Terminals and Gateways. The Gatekeeper may also provide other services to the H.323 terminals and Gateways, such as bandwidth management and locating Gateways.

**Gateway** An H.323 Gateway (GW) is an endpoint on the Internet which provides for real-time, two-way communications between H.323 Terminals on the Network and other ITU Terminals on a wide area network, or to IP Telephony clients.

**H.323** The specification that defines packet standards for terminals, equipment, and services for multimedia communications over LANs. Adopted by the Internet telephony community as the standard for communicating over any packet network, including the Internet.

**International Telecommunications Union (ITU)** An organization established by the United Nations to set telecommunications standards, allocate frequencies to various uses, and hold trade shows every four years.

**Internet** An inter-network of networks interconnected by bridges or routers. LANs described in H.323 may be considered part of such inter-networks.

**Internet Protocol (IP)** The TCP/IP standard protocol that defines the IP datagram as the unit of information passed across an internet and provides the basis for connectionless, best-effort packet delivery service.

**Internet Service Provider (ISP)** A vendor who provides direct access to the Internet.

**Internet Telephony** Technology that lets you make voice phone calls over the Internet or other packet networks using your PC, via gateways and standard telephones.

**IP** See Internet Protocol

**ISP** See Internet Service Provider

**ITU** See International Telecommunications Union.

**Parameter** A type of datum that is passed to or from a component via a DM3 message. Parameters generally have two elements, a type and a specific value. Parameters passed to a component affects the operational characteristics of the component, and parameters passed from a component can be used to report the operating state of the component. As an example, the standard Player component accepts a parameter called ParmDuration, which specifies the length of the file that is being played back, and can report its current state (e.g. playing, paused, etc.) via a parameter called ParmState.

**PCD file** An ASCII text file that contains product or platform configuration description information that is used by the DM3 downloader utility program. Each of these files identifies the hardware configuration and firmware modules that make up a specific hardware/firmware product. Each type of DM3-based product used in a system requires a product-specific PCD file.

## **DM3 IPLink™ User's Guide for Windows NT**

**Reliable Channel** A transport connection used for reliable transmission of an information stream from its source to one or more destinations.

**Reliable Transmission** Transmission of messages from a sender to a receiver using connection-mode data transmission. The transmission service guarantees sequenced, error-free, flow-controlled transmission of messages to the receiver for the duration of the transport connection.

**Resource** A conceptual entity defining a feature set, which can be implemented by writing firmware components that run on one or more processors on a DM3 platform. A simple example of a resource is a player, which is composed of a player component, which runs on the Control Processor of a DM3 platform and handles control and communications functions, plus a decoder component which typically runs on a Signal Processor and processes the encoded data stream that is being played back.

**RTCP** Real Time Control Protocol

**RTP** Real Time Protocol

**SCbus** The standard bus for communication within a SCSA node. The architecture of SCbus includes a 16-wire TDM data bus that operates at 2, 4 or 8 Mbps and a serial message bus for control and signaling. DM3 platforms provide an SCbus interface for interconnection of multiple DM3 platforms, or connection to other SCSA-compatible hardware. The DM3 platform supports timeslot bundling for high bandwidth, and can access up to 256 of the 2048 SCbus timeslots via two SC4000 ASICs.

**Signal Processor (SP)** An embedded processor on a DM3 platform that is used to execute signal processing algorithms. The DM3 architecture supports multiple types of SPs, including RISC processors (e.g. PowerPC) as well as general purpose DSPs (e.g. Motorola 5630x), and platforms may be configured with a mixed complement of SP types.

**Standard message set** A predefined set of DM3 messages which handle functionality that is common to all DM3 component, such as setting and getting parameters and managing RTC events. Each DM3 component should support all of the standard messages that are relevant to the component's operation. Each component will generally

supplement the standard message set with its own set of proprietary or component-specific messages.

**Synchronous mode** An operating mode of certain DM3 kernel or host library function calls. Synchronous mode is a blocking mode that is typically used only when a function executes on the local processor using data that is also local to the processor (i.e. that does not require any data movement via the MMA).

**TCP** see Transmission Control Protocol

**Terminal** An H.323 Terminal is an endpoint on the local area network which provides for real-time, two-way communications with another H.323 terminal, Gateway, or Multipoint Control Unit. This communication consists of control, indications, audio, moving color video pictures, and/or data between the two terminals. A terminal may provide speech only, speech and data, speech and video, or speech, data, and video.

**Transmission Control Protocol** The TCP/IP standard transport level protocol that provides the reliable, full duplex, stream service on which many application protocols depend. TCP allows a process on one machine to send a stream of data to a process on another. It is connection-oriented in the sense that before transmitting data, participants must establish a connection.

**UDP** see User Datagram Protocol

**User Datagram Protocol** The TCP/IP standard protocol that allows an application program on one machine to send a datagram to an application program on another machine. Conceptually, the important difference between UDP datagrams and IP datagrams is that UDP includes a protocol port number, allowing the sender to distinguish among multiple destinations on the remote machine.



# Index

---

## A

Accepted (state), 34, 55, 60  
Accepting a call, 34, 43, 55, 58  
Alerting (state), 34  
Allocating a cluster, 48  
Answering a call, 43, 55, 56  
Assigning timeslots, 49

## B

Billing, 59  
Boot kernel, 95

## C

Call control, 21, 52, 109  
Call failure, 34  
Call identifier, 33, 44  
Call offering, 35  
Call setup, 86  
Call states, 33, 68  
    Accepted, 34, 55, 60  
    Alerting, 34  
    Connected, 34, 61, 62, 64, 81  
    Disconnected, 34, 64  
    Failed, 34  
    Idle, 34, 65, 68, 70  
    Initiated, 34, 52, 54, 72, 81  
    Null, 33, 34, 52, 55, 57, 60, 63, 67  
    Offered, 33, 35, 55, 57, 60, 70  
    Proceeding, 35  
Call statistics, 63, 67, 68

Cancel RTC requests, 42  
Change a parameter value, 43  
Cluster, 13, 17, 25, 129  
    Management, 30, 48  
Coder, 26  
Coder capabilities, 86  
Coders, 18, 57, 61, 87  
    G.711, 18, 87  
    G.723.1, 18, 87  
Compute call duration parameter, 81  
Computer Telephony, 130  
*config.val*, 86  
Configuring the IPLink Platform, 75  
Connected (state), 34, 61, 62, 64, 81

## D

Data network, 107  
Data structure, 30  
DCM, 96  
Debug print level parameter, 80  
Debugging procedure  
    Download failed, 93  
    Download succeeded, 99  
Debugging utilities, 93  
Default coders, 87  
Destination address, 44  
Detecting events, 42, 45  
Dialogic Configuration Drivers, 98

## **DM3 IPLink™ User's Guide for Windows NT**

Dialogic Configuration Manager, 96

Dialogic enable parameter, 79

Dialogic gateway, 80

Disconnected (state), 34, 64

Disconnecting a call, 43

DM3 components, 17

DM3 component, 131

DM3 Mediastream Architecture, 16

DM3 messages, 13, 41, 75, 131

DM3 technology resource, 17

DM3STDER utility, 102

DTMF tone detection, 27

### **E**

Echo canceler, 27

- Parameters, 76

Error detected, 42

Event detected, 42

Event label, 47

Event type, 47

Events, 31, 46, 47, 54, 60, 134

- NetTSC Events
  - NetTSC\_EvtH245Data\_NonStdCmd, 47
  - NetTSC\_EvtH245Data\_UsrInputIndication, 48
  - NetTSC\_EvtSystemFailed, 48
  - NetTSC\_EvtThresholdAlarm, 48
- Retrieving, 31
- TSC Events
  - TSC\_EvtCallInfo, 47
  - TSC\_EvtCallState, 47

### **F**

Failed (state), 34

Failed outbound call

- Procedure, 71
- State diagram, 70

FCD file, 75, 98, 100, 102

Feature Configuration Description file.

- See FCD

FW download drivers, 98

### **G**

G.711 coder, 18, 87, 110

G.723.1 coder, 18, 87, 110

Gatekeeper, 86, 88, 132

- Enabling, 86, 88

Gateway, 107, 132

Getting call statistics, 67

- Procedure, 68

**getXmitSlot**, 50

Global streams, 18

### **H**

H.323 Parameters, 79

- PrmDebugLevelMNTI, 80, 101
- PrmDebugLevelMsg, 80, 101
- PrmDebugLevelRVSTACK, 81, 102
- PrmDebugLevelStack, 80, 101
- PrmDebugLevelStates, 80, 101
- PrmDebugLevelStream, 80, 101
- PrmDebugLevelTimer, 80, 101
- PrmDebugLevelUtil, 80, 101
- PrmDialogicEnable, 79

H.323 Recommendation, 109, 132



H.323 Stack, 22  
    Configuration file, 86  
*h323dbg.txt*, 100  
Host application, 29, 30  
    API, 30  
    Communication, 30  
    Guidelines, 29  
    Responsibilities, 29  
Host controller, 17  
**I**  
I/O completion ports, 31  
Idle (state), 34, 65, 68, 70  
Inbound call  
    Procedure, 55, 57  
    State diagram, 38, 56  
Initialize resource, 42  
Initiated (state), 34, 52, 54, 72, 81  
Initiating a call, 34, 44  
Interim states, 55  
Internet gateway, 58  
Internet Protocol, 12, 17, 107, 133  
Internet Telephony, 11, 29, 107, 133  
IP address, 44, 89  
IPLink platform, 15  
IP-PSTN gateway, 12  
*ipt.fcd*, 75  
**L**  
Line Administration component, 25  
Log file, 100, 102  
    *h323dbg.txt*, 100

*rvtspl.log*, 102

**M**

Making a call, 52

MCD, 97

Mediastream Message Block, 30

Message Summary  
    Accept Call, 61  
    Answer Call, 58  
    Terminate Call, 66

Messages

    Standard Messages

        Std\_MsgCancelAllEvts, 42  
        Std\_MsgCancelAllEvtsCmplt,  
            42  
        Std\_MsgCancelEvt, 42  
        Std\_MsgCancelEvtCmplt, 42  
        Std\_MsgCancelxEvts, 42  
        Std\_MsgCancelxEvtsCmplt, 42  
        Std\_MsgComtest, 42  
        Std\_MsgComtestCmplt, 42  
        Std\_MsgDetectEvt, 36, 42  
        Std\_MsgDetectEvtCmplt, 42  
        Std\_MsgDetectxEvts, 42, 57  
        Std\_MsgDetectxEvtsCmplt, 42  
        Std\_MsgError, 42  
        Std\_MsgEvtDetected, 35, 42,  
            57  
        Std\_MsgExit, 42  
        Std\_MsgExitCmplt, 42  
        Std\_MsgGetParm, 42  
        Std\_MsgGetParmCmplt, 42  
        Std\_MsgInit, 42  
        Std\_MsgInitCmplt, 42  
        Std\_MsgSetAllParmsDef, 42  
        Std\_MsgSetAllParmsDefCmplt,  
            42  
        Std\_MsgSetParm, 43  
        Std\_MsgSetParmCmplt, 43  
        Std\_MsgSetParmDef, 43

## **DM3 IPLink™ User's Guide for Windows NT**

- Std\_MsgSetParmDefCmplt, 43
- TSC Messages
  - TSC\_MsgAcceptCall, 43, 55, 60
  - TSC\_MsgAnswerCall, 43, 55, 57
  - TSC\_MsgDisconnected, 65
  - TSC\_MsgDropCall, 35, 43, 63, 73
  - TSC\_MsgGetCallInfo, 43, 67
  - TSC\_MsgGetCallInfoExt, 67
  - TSC\_MsgGetCallState, 43
  - TSC\_MsgMakeCall, 34, 44, 53, 72
  - TSC\_MsgMakeCallCmplt, 72
  - TSC\_MsgRejectCall, 44, 55, 70
  - TSC\_MsgReleaseCall, 44, 55, 64, 73
- Miscellaneous Parameters, 81
  - PrmCallDurationComput, 81, 82, 83, 84
- MMB, 30
- mntClusterActivate, 49, 51
- mntClusterAllocate, 48
- mntClusterCompbyAttribute, 48
- mntClusterSlotInfo, 51
- mntClusterTSAssign, 49, 51
- MNTI, 30
- MPD, 97
- Multicast address, 90
- N**
  - Native Synchronization Methods, 31
  - NetTSC component, 22, 35, 54
  - NetTSC events, 47
  - NetTSC\_EvtH245Data\_NonStdCmd, 47
  - NetTSC\_EvtH245Data\_UsrInputIndication, 48
  - NetTSC\_EvtSystemFailed, 48
  - NetTSC\_EvtThresholdAlarm, 48
- NetTSC\_EvtH245Data\_NonStdCmd, 47
- NetTSC\_EvtH245Data\_UsrInputIndication, 48
- NetTSC\_EvtSystemFailed, 48
- NetTSC\_EvtThresholdAlarm, 48
- NetTSP resource, 21, 33
- NSM, 31
- Null (state), 33, 34, 52, 55, 57, 60, 63, 67
- O**
  - Offered (state), 33, 35, 55, 60, 70
  - Offered state, 57
  - Outbound call
    - Procedure, 52
    - State diagram, 37
- P**
  - Packet loss recovery, 27
  - Parameter configuration files, 75
  - PCD file, 85, 98
  - Ping message, 42
  - PLM, 85
  - Power level parameter, 77
  - PrmCallDurationComput, 81, 82, 83, 84

- PrmDebugLevelMNTI, 80, 101
- PrmDebugLevelMsg, 80, 101
- PrmDebugLevelRVSTACK, 81, 102
- PrmDebugLevelStack, 80, 101
- PrmDebugLevelStates, 80, 101
- PrmDebugLevelStream, 80, 101
- PrmDebugLevelTimer, 80, 101
- PrmDebugLevelUtil, 80, 101
- PrmDialogicEnable, 79
- PrmECActive, 76
- PrmECOrder, 76
- PrmPwrLvlCtrl, 77, 78
- Proceeding (state), 35
- Processor Load Modules, 85
- Product Configuration Description file, 85
- Programming model, 33
- PSTN, 107
- Q**
- Quality of Service, 19, 109
- Querying the NetTSP cluster, 51
- Querying the PSTN card, 50
- R**
- R4 events, 31
- Read parameter value, 42
- Real-Time Transport Control Protocol, 109
- Real-Time Transport Protocol, 109
- Receive timeslot, 49
- Receiving a call, 55
- Registering for event notification, 47
- Reject incoming call, 44
- Rejecting a call, 55
  - Procedure, 70
  - State Diagram, 69
- Releasing a call, 34, 44
- Request current state, 43
- Request information, 43
- Requesting a gatekeeper, 89
- Routing, 29
- RTCP**, 18, 22, 111, 134
- RTP**, 18, 22, 26, 109, 111, 134
- rvtsp1.log, 102
- S**
- SCbus, 12, 13, 20, 26, 49, 50, 96, 97, 129, 132, 134
- SCbus timeslots, 17
- SCbusClockMaster, 96
- Set parameter value, 43
- Set parameter values to default, 42
- Setting Parameters
  - H.323 Parameters
    - PrmDebugLevelMNTI, 80, 101
    - PrmDebugLevelMsg, 80, 101
    - PrmDebugLevelRVSTACK, 81, 102
    - PrmDebugLevelStack, 80, 101
    - PrmDebugLevelStates, 80, 101
    - PrmDebugLevelStream, 80, 101
    - PrmDebugLevelTimer, 80, 101

## **DM3 IPLink™ User's Guide for Windows NT**

- PrmDebugLevelUtil, 80, 101
- PrmDialogicEnable, 79
- Miscellaneous Parameters
  - PrmCallDurationComput, 81, 82, 83, 84
- SP Parameters
  - PrmEActive, 76
  - PrmEOrder, 76
  - PrmPwrLvlCtrl, 77, 78
- Shut down an instance, 42
- Silence suppression, 87
- SP Parameters, 76
  - PrmEActive, 76
  - PrmEOrder, 76
  - PrmPwrLvlCtrl, 77, 78
- SRAM protocol drivers, 98
- SRL, 31
- Standard Messages
  - Std\_MsgCancelAllEvts, 42
  - Std\_MsgCancelAllEvtsCmplt, 42
  - Std\_MsgCancelEvt, 42
  - Std\_MsgCancelEvtCmplt, 42
  - Std\_MsgComtest, 42
  - Std\_MsgComtestCmplt, 42
  - Std\_MsgDetectEvt, 36, 42
  - Std\_MsgDetectEvtCmplt, 42
  - Std\_MsgDetectxEvts, 42, 57
  - Std\_MsgDetectxEvtsCmplt, 42
  - Std\_MsgError, 42
  - Std\_MsgEvtDetected, 35, 42, 57
  - Std\_MsgExitCmplt, 42
  - Std\_MsgGetParm, 42
  - Std\_MsgGetParmCmplt, 42
  - Std\_MsgInit, 42
  - Std\_MsgInitCmplt, 42
  - Std\_MsgSetAllParmsDef, 42
  - Std\_MsgSetAllParmsDefCmplt, 42
  - Std\_MsgSetParm, 43
  - Std\_MsgSetParmCmplt, 43
  - Std\_MsgSetParmDef, 43
  - Std\_MsgSetParmDefCmplt, 43
- Standard Runtime Library, 31
- State diagram, 36
  - Inbound call, 38
  - Outbound call, 37, 52
- State transitions, 35
- Std\_MsgCancelAllEvts, 42
- Std\_MsgCancelAllEvtsCmplt, 42
- Std\_MsgCancelEvt, 42
- Std\_MsgCancelEvtCmplt, 42
- Std\_MsgCancelxEvts, 42
- Std\_MsgCancelxEvtsCmplt, 42
- Std\_MsgComtest, 42
- Std\_MsgComtestCmplt, 42
- Std\_MsgDetectEvt, 36, 42
- Std\_MsgDetectEvtCmplt, 42
- Std\_MsgDetectxEvts, 42, 57
- Std\_MsgDetectxEvtsCmplt, 42
- Std\_MsgError, 42
- Std\_MsgEvtDetected, 35, 42, 57
- Std\_MsgExitCmplt, 42
- Std\_MsgGetParm, 42
- Std\_MsgGetParmCmplt, 42
- Std\_MsgInit, 42
- Std\_MsgInitCmplt, 42
- Std\_MsgSetAllParmsDef, 42
- Std\_MsgSetAllParmsDefCmplt, 42
- Std\_MsgSetParm, 43

Std\_MsgSetParmCmplt, 43  
Std\_MsgSetParmDef, 43  
Std\_MsgSetParmDefCmplt, 43

**T**

TDM data, 17  
Telephone network, 107  
Terminating a call, 34  
    Procedure, 63  
    State diagram, 62  
Timeslots, 49  
Transmit timeslot, 49  
TSC Events  
    TSC\_EvtCallInfo, 47  
    TSC\_EvtCallState, 47  
TSC Messages, 43  
    TSC\_MsgAcceptCall, 43, 55, 60  
    TSC\_MsgAnswerCall, 43, 55, 57  
    TSC\_MsgDisconnected, 65  
    TSC\_MsgDropCall, 35, 43, 63, 73  
    TSC\_MsgGetCallInfo, 43, 67  
    TSC\_MsgGetCallInfoExt, 67  
    TSC\_MsgGetCallState, 43  
    TSC\_MsgMakeCall, 34, 44, 53, 72  
    TSC\_MsgMakeCallCmplt, 72  
    TSC\_MsgRejectCall, 44, 55, 70  
    TSC\_MsgReleaseCall, 44, 55, 64,  
        73  
TSC\_EvtCallInfo, 47  
TSC\_EvtCallState, 47  
TSC\_MsgAcceptCall, 43, 55, 60  
TSC\_MsgAnswerCall, 43, 55, 57  
TSC\_MsgDisconnected, 65  
TSC\_MsgDropCall, 35, 43, 63, 73

TSC\_MsgGetCallInfo, 43, 67  
TSC\_MsgGetCallInfoExt, 67  
TSC\_MsgGetCallState, 43  
TSC\_MsgMakeCall, 34, 44, 53, 72  
TSC\_MsgMakeCallCmplt, 72  
TSC\_MsgRejectCall, 44, 55, 70  
TSC\_MsgReleaseCall, 44, 55, 64, 73  
TSP services, 25

**U**

UDP, 18, 109, 111, 135

**V**

VAD, 87  
Valid state, 34  
Voice coder, 26  
Voice coders  
    G.711, 18  
    G.723.1, 18  
VSR component, 22, 26

**W**

Windows NT, 30, 31  
WinNT, 31

**X**

xx\_listen, 51

## NOTES

---

## NOTES

---

## NOTES

---