Application Note

A Media Server Solution Recipe: Network Interactive Voice Response

Executive Summary

Dialogic® voice and telephony products can be configured in many unique ways. These configurations form smaller, functional categories referred to as building blocks. The Network Interactive Voice Response (NIVR) is the most common building block, and helps form the basis for various feature rich solutions. The NIVR building block is a basic media server solution configuration and a capability of many enterprise communication platforms.

Building blocks, such as the NIVR, can provide the system developer or integrator with a starting point that is closer to system deployment. This means that rather than having to individually characterize each component, the system designer can start with a comparatively more complete and known assembly of telephony and voice features, which in turn can greatly reduce project cost and time to market.

Table of Contents

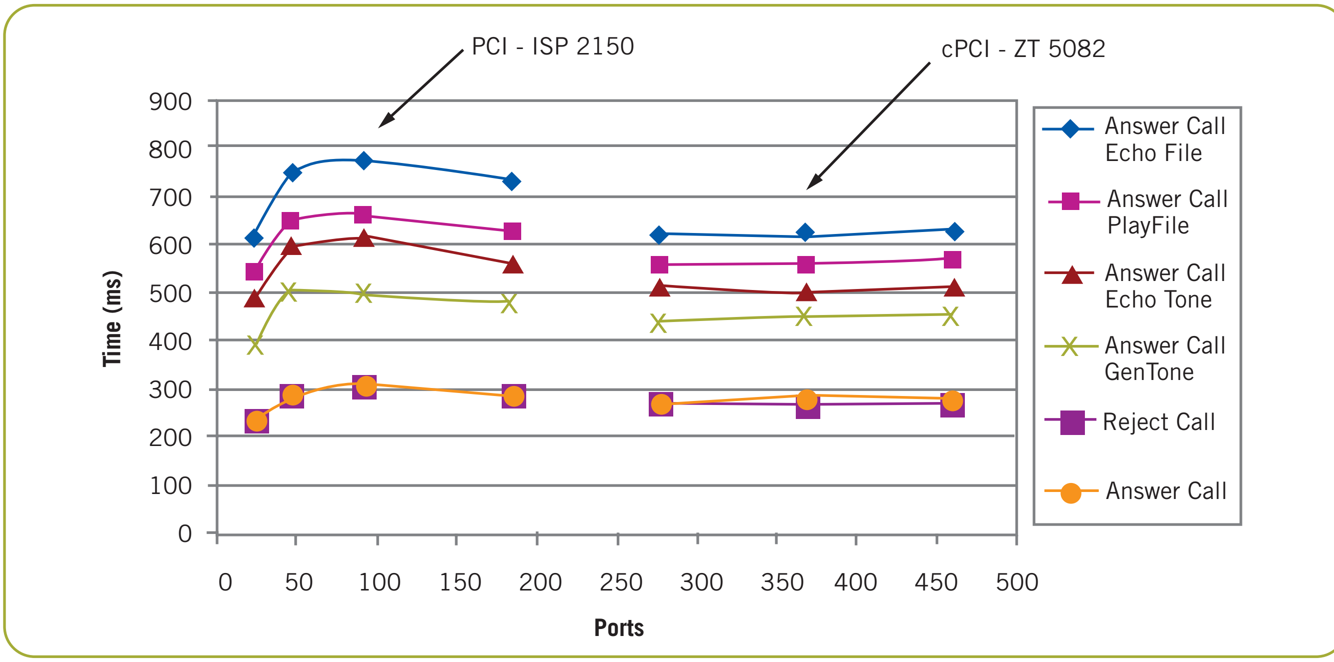


Figure 1. Performance On Dialogic® DM3 Media Boards With Dialogic® System Release 5.1 Software

Purpose

This application note describes a standard solution with capabilities similar to platforms used by service providers and carriers. This instance is a Network Interactive Voice Response (NIVR) solution, used to characterize IVR system performance under various load conditions using Dialogic® System Release 5.1 for Windows® software. The NIVR solution uses Dialogic® computer telephony cards, industry standard operating systems, and Dialogic® System Release Software packages. The applications used to characterize the NIVR simulate common telephony applications, such as Interactive Voice Response (IVR) and Voice Mail (VM) systems, and reflect a well thought out call model based on real system application experiences and information gathered from vendors and service providers. The available source code components and performance data provide a baseline implementation for users wishing to create a customized version of the NIVR environment.

The NIVR solution includes Target, Probe, and Load applications for Windows® and Linux platforms that interface with T-1, E-1, and SS7 Networks. The source code uses the Dialogic® Global Call Software on Dialogic® JCT Media Boards and DM3 Media Boards on the single-threaded asynchronous programming model in a multi-process environment. The Probe application also includes Simple Network Management Protocol (SNMP) management capabilities, which can be extended to define additional parameters that can be set remotely.

The Target application is the system under test that simulates IVR and VM call scenarios. The Probe application initiates instrumented calls to the Target application to calculate the NIVR response times. The Load application is used to stress the target system with additional stimulus that simulates realistic call scenarios to the point of call congestion. Call scenarios can be easily added or modified to the NIVR test environment to better simulate customized call flows.

Performance results are included in the Results section of this document and should be used by a system developer to reduce a project's time to market. The graph in Figure 1 illustrates a comparison of Dialogic® DM/V960-4T1 Voice Board (PCI) performance in an Intel ISP 2150 versus DM/V960-4T1 (cPCI) performance in an Intel ZT 5082 chassis, both using the Dialogic® System Release 5.1 software. This graph highlights the consistent performance of the Dialogic® DM3 Media Boards as they scale to densities up to 460 ISDN channels. The *Results* section of this document includes benchmark data for the following solution configurations:

- Dialogic® DM/V960-4T1 Voice Board (PCI) in an Intel ISP 2150
- Dialogic® DM/V960-4T1 Voice Board (cPCI) in an Intel ZT 5082

- Dialogic® D/480JCT-2T1 Media Board in an Intel ISP 2150
- Dialogic® D/600JCT-2E1 Media Board and Dialogic® DSC131 SIU in an Intel ISP 2150

Note: The DSC131 SIU has been retired.

Introduction

The NIVR Target, Probe, and Load modules are three separate applications that are interconnected to simulate real world call scenarios and measure system performances. The following sections discuss the roles of these applications in the test environment.

Target System

The target consists of an Intel ISP 2150 system and represents the system under test. The target application (Target.zip) is included in a Zip file called *Network Interactive Voice Response* (see the *For More Information* section for this downloadable Zip file). The target application simulates IVR and VM call scenarios based on DNIS digits. The DNIS digits of an incoming call are passed to the Test Manager component, which assigns an IVR, VM, or Probe call scenario to the channel. The channel follows the call scenario state machine until the conclusion of the call. When the target application is interfacing with an SS7 network, it establishes an IPC connection with its corresponding SIU for SS7 messages.

Probe System

The probe consists of an Intel ISP 2150 system and generates instrumented call traffic on one channel of the target application. The probe application (Probe.zip) is included in a Zip file called *Network Interactive Voice Response* (see the *For More Information* section for this downloadable Zip file). The probe application creates a "ProbeResults" log file that contains raw NIVR Target performance data on a per call basis. A scenario script file is used to specify the list of instrumented calls to initiate with the target application. The probe application can be compiled to sequentially or randomly select tests from the script file and use Dialed Number Identification Service (DNIS) digits to inform the target application of the call scenario it expects to execute.

The probe measurements include the time required to answer calls (with and without checking DNIS), the time to reject calls (with and without checking DNIS), the detection and generation of tones, and the opening and playing of a prerecorded message.

The probe application includes sample Simple Network Management Protocol (SNMP) capabilities. The Test Manager component stores scenario statistics in a shared memory location, making the statistics available to Simple Network Management Protocol (SNMP) agents. An SNMP Management Console provides remote access to the statistics of the probed device on a per scenario basis. The Management Console used in this solution is the ACE-SNMP Web Based Management System from Diversified Data Resources, Inc (no longer available) providing access to SNMP management capabilities from any web browser. Statistics that are tracked by the Test Manager and are accessible via SNMP Agents are listed in Table 1.

SNMP Statistics
Incoming Calls Offered to the Target Application
Incoming Calls Rejected by the Target Application
Probe Scenario Average Response Time
Probe Scenario Maximum Response Time
Probe Scenario Minimum Response Time

Table 1. SNMP Statistics

Load System

The load generates excessive call traffic on the target system to simulate realistic call scenarios. A scenario script file is used to specify the list of call scenarios that are initiated on the target application. The load application (Load.zip) is included in a Zip file called *Network Interactive Voice Response* (see the *For More Information* section for this downloadable Zip file). The load application can be compiled to sequentially or randomly select tests from the script file and uses DNIS digits to inform the target application of the call scenario it expects to execute.

The load system performs activities such as information retrieval from an IVR and retrieval or generation of voice messages to a VM system. The Load also makes requests for the target to deliver voice mail messages by initiating outbound calls into the load system. The load application creates a "Load Results" log file that contains information and call flows on a per call basis.

Prior to initiating a call to the target application, the load system determines how it reacts to particular stages in the

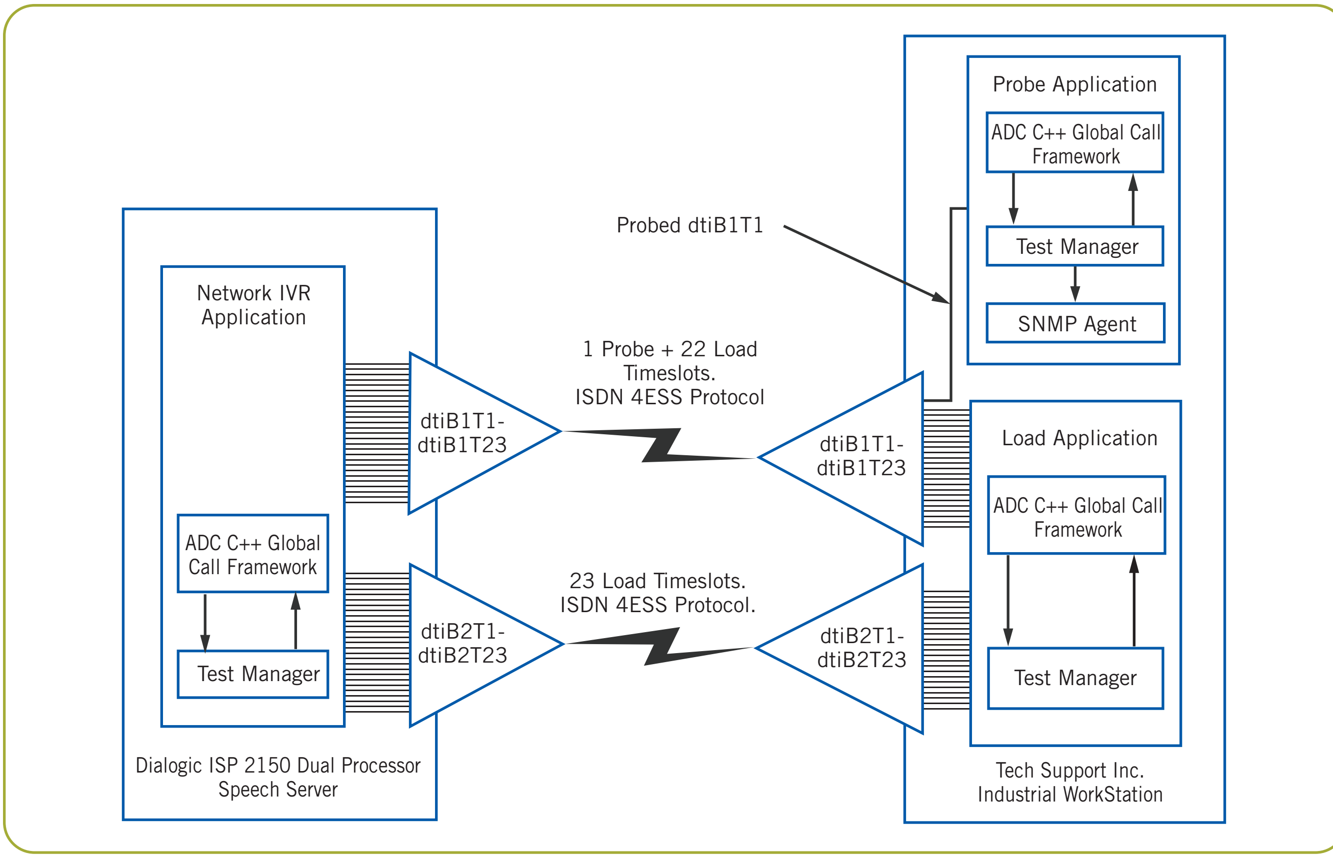


Figure 2. Default System Configuration

call scenario it is going to execute. The reactions to a call stage include:

- Waiting for a prompt to complete before responding with DTMF
- DTMF barge-in during a prompt
- Disconnecting from the call during a particular prompt

The load application can be compiled to make predetermined or random decisions for each call.

DSC131 Signaling Interface Unit

The DSC131 Signaling Interface Unit (SIU) is used to handle SS7 messages for the corresponding Probe and Target systems. The two SIUs are interconnected to form an SS7 network.

Solution Architecture

C++ Communication Services Framework (see the *For More Information section*) was used as the foundation for the NIVR Target, Load, and Probe applications. Test Manager functionality was added to handle the assignment of test scenarios to resource and interface devices. A test scenario is the base class for all call scenarios used in this solution. A scenario defines the state machine for each call flow that is executed in this solution. The probe application includes Simple Network Management Protocol (SNMP) Management Information Base (MIBs) to allow SNMP queries of probe statistics. The diagram in Figure 2 illustrates the overall system design.

SS7 capabilities were added with few overall source code changes. The diagram in Figure 3 shows the additional components required for SS7 network interaction. Inter Process Communication (IPC) links are required between the systems and the DSC131 SIU. Call Control messages are sent through the links for call setup, establishment, and teardown.

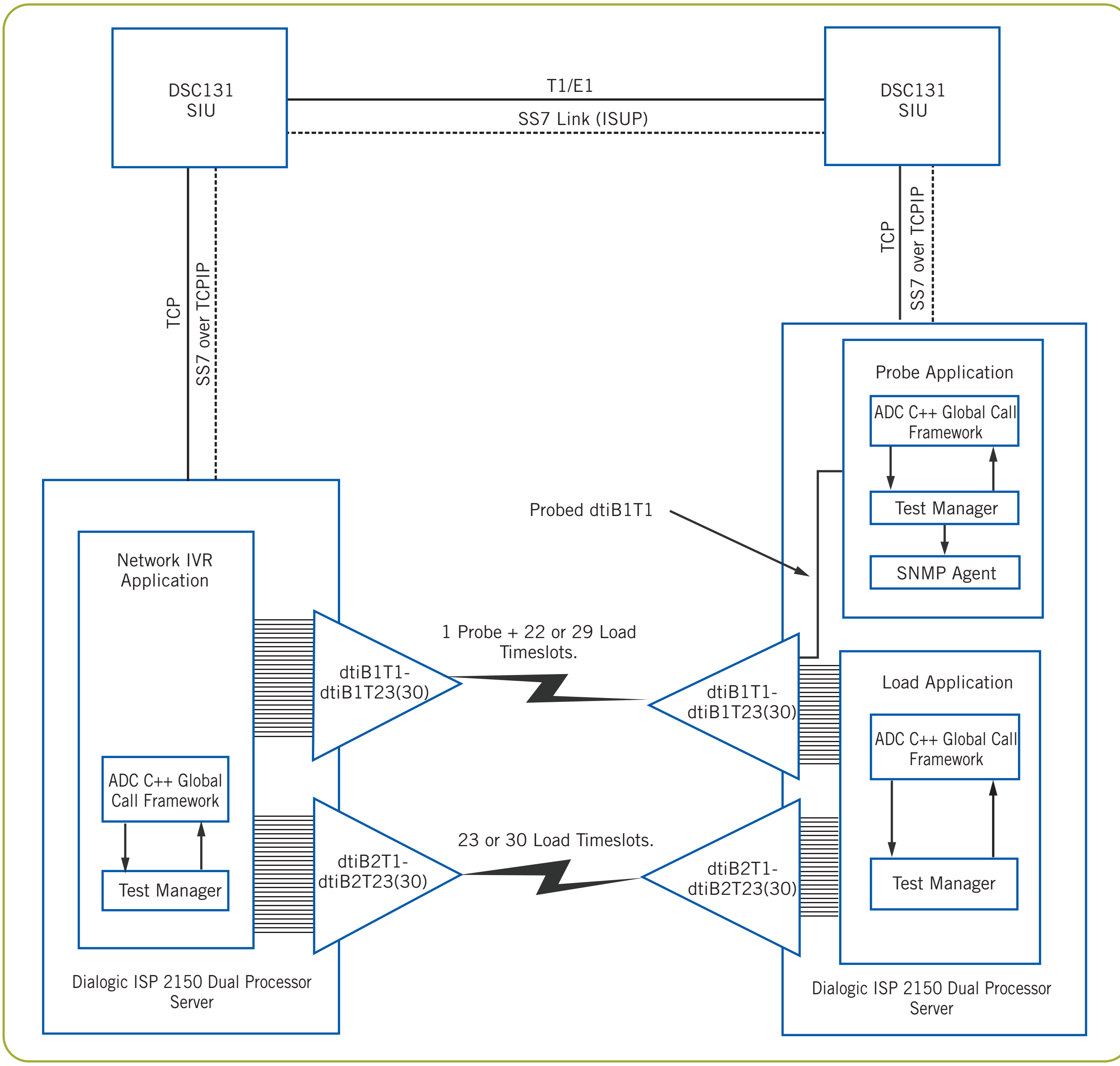


Figure 3. SS7 Configuration

Application Design

Components of the Communication Services Framework (CSF) were used to implement the source code baseline for this solution. For a more detailed description of the CSF, please refer to the *Introduction to Communication Services Framework (CSF)* (see the *For More Information* section). In addition to the baseline, Test Manager, Test Scenarios, and SNMP Agent components were added to the solution software. The application modules of this test environment use a scenario script file, called "scenario.txt" by default, to specify the call scenarios that a solution application will execute. During system initialization, the Test Manager parses the scenario script file and places each script entry into a test array. Each index of the test array represents a unique call scenario instance that can be assigned to a Dialogic® Global Call Software device. The execution probability of a particular call scenario is directly proportional to the number of instances that the call scenario is specified in the scenario script file. The Test Manager objects can be compiled to randomly or sequentially select call scenarios from the test array.

The call scenarios included in the NIVR test environment require the Load and Probe applications to initiate calls to the Target application. The Target script file is configured

Scenario Method	Function Mask Variable	Bit Mask Value
OnBeginScenario	mskOnBeginScenario	0x00000001
OnCallOffered	mskOnCallOffered	0x00000002
OnResetLineDevice	mskOnResetLineDevice	0x00000004
OnConnected	mskOnConnected	0x00000008
OnCallAnswered	mskOnCallAnswered	0x00000010
OnCallDisconnected	mskOnCallDisconnected	0x00000020
OnCallDropped	mskOnCallDropped	0x00000040
OnDigitsReceived	mskOnDigitsReceived	0x00000080
OnTimerEvent	mskOnTimerEvent	0x00000100
OnPlayComplete	mskOnPlayComplete	0x00000200
OnRecordComplete	mskOnRecordComplete	0x00000400
OnErrorCondition	mskOnErrorCondition	0x10000000

Table 2. Test Scenario Methods and Bitmasks

to specify how the target responds to incoming calls. The target script file can either specify a particular test to execute in response to an incoming call or the "DNIS_React" scenario can be used. The DNIS React scenario instructs the target to pass the DNIS digits of the incoming call to its Test Manager object. The Test Manager object then assigns the Global Call Software device to a state machine of the call scenario that is mapped to the incoming DNIS digits.

Call scenarios are state machines that determine the call flow that a device follows during a particular call. Each call scenario is identified by an enumerated scenario identifier, the scenarioID. If the Test Manager is using the sequential test selection, then the test-tracking array forces each device to sequentially traverse test array on each call to assign a new owner to the device. The Test Manager object assigns a state machine to each device at the conclusion of each call, or any other time the TestManager::AssignOwner() method is called. When sequential test execution is specified and the test-tracking array has reached the final index of the test array, it is reset to the first index of test array. If random test selection is used, the Test Manager randomly selects a number and performs a modulus operation by the number of indexes in the test array.

Test Scenario objects include a name string (scenName), a scenario number (scenID), and a trace variable (mskTrace). The trace variable (mskTrace) tracks the state machine methods that the device has entered during the life of the scenario. The member functions for each Test Scenario object are listed in Table 2. Test Scenario Methods and Bitmasks and are uniquely identified by a bit mask. When a Test Scenario member function is called, its trace bit is set to 1 in the Test Scenario trace flag. Adding a new Test Scenario to the test environment requires the following steps:

1. Create the new Test Scenario, which publicly inherits the Test Scenario base class.
2. Implement the Test Scenario methods that apply to the new Test Scenario. The Test Scenario base class provides virtual implementations for each method. The default implementation for each method is a call to OnErrorCondition, which disconnects or rejects the incoming call.
3. Uniquely identify the new Test Scenario by adding a new variable to the enumerated test list, located in the Test Manager header file.
4. Add the call to the new Test Scenario constructor in TestManager::GetTestScenario().
5. Add the #include statement for the new Test Scenario header file to the Test Manager header file.

Test Scenario methods are modified to abstract the type of event (Dialogic SRL, Timer, Exit, etc.) from the state machine. The following list discusses the individual Test Scenario methods that are currently included in the NIVR solution source code.

- **OnBeginScenario** — The method that is executed after a test scenario is successfully assigned to Load and Probe devices. This method could be used to perform tasks that signal the initial execution of a Test Scenario. The load application uses this method to select reactions to target prompts that occur during the execution of a load call scenario. The load and probe applications also use this method to set guard timers before the actual call is made to the target application.

- **OnCallOffered** — The method that is called in response to an offered call. The target application has compiled time definitions that determine if the target performs a quick answer or accept the call prior to answering. The DNIS_React scenario uses the DNIS digits from the offered call to assign a new call scenario to the device. The new call scenario then reacts to the offered call according to its state machine.
- **OnCallDisconnected** — The method that responds to a remote disconnect during a call scenario. This method stops all pending timers and verifies that the voice device is idle before dropping the disconnected call.
- **OnCallDropped** — The method that responds to a dropped call event. This method stops pending timers and verifies that the voice device is idle before releasing the dropped call. This method also logs the final results of the call to the result log file. The result contents include the time that the call completed, the network device handle that managed the call, the name and ID of the call scenario that completed, and the function trace bitmask of the call. The probe application logs all the previously mentioned data, plus the measured time result for the completed scenario. The Load application also logs the bitmasks that specify how the load device responded to the target prompts. Following the completion of this function, an application-specific event (DLGC_TEST_COMPLETE) is passed to the SRL Queue to signify the end of the scenario.

The following methods represent events that are handled by the Network IVR, the Probe, and the Load applications, but may not be used in most call scenarios that are included in the NIVR test environment.

- **OnResetLineDevice** — The method that is called after a device is reset by a call to gc_ResetLineDev. This method sets the network device to detect incoming calls, by calling the GCChan::WaitForCall method that is included in the *Introduction to Communication Services Framework (CSF)* (see the *For More Information section*).
- **OnConnected** — The method that is called following the connection of an outgoing call.
- **OnCallAccepted** — The method that is called following the acceptance of an offered call. This method is only called when the GCChan::AcceptCall method is used.
- **OnCallAnswered** — The method that is called following the connection of an incoming call.
- **OnDigitsReceived** — The method that is called following the detection of incoming DTMF digits or a user-defined tone. Digits are used by the Load and Target applications to signal completion of voice prompts. The Probe and Target applications use user-defined tones to measure performance of generating a tone or playing a file that contains a tone.
- **OnTimerEvent** — The method that responds to an expired timer. Cancelled timers do not trigger this method.
- **OnPlayComplete** — The method that is called following the completion of the DlgcChan::PlayFile or DlgcChan::PlayTone method has completed.
- **OnRecordComplete** — The method that is called following the completion of a DlgcChan::RecordCaller method.
- **OnErrorCondition** — The method that is called following an error condition or an unexpected event that was determined by the developer to be a fatal condition. This method marks the call scenario as a failure and then disconnects the connected call.

The Target, Load, and Probe applications utilize the Test Scenario class to create all of the call scenarios included in the test environment. The Test Scenario that is used in the probe application includes all of the methods previously discussed, but it also utilizes the following methods to measure time results. The methods used to calculate time results are described as follows:

- **StartClock**() — This method stores the number of clock ticks since the start of the process into a variable called timeStart.
- **StopClock**() — This method stores the number of clock ticks since the beginning of the process into a variable called timeStop.
- **CalculateResult**() — This method stores the difference between timeStart and timeStop into a variable called timeResult.
- **GetTimeResult**() — This method returns the value stored in timeResult.

Scenario Description	Probe Scenario	Target Scenario	Load Scenario
Make Accepted No DNIS	Probe Accept	Accept Call	N/A
Make Rejected No DNIS	Probe Reject	Reject Call	N/A
Make Accepted	Probe Accept	Accept Call	N/A
Make Rejected	Probe DNIS Reject	Reject Call	N/A
Detect Tone	Probe Accept Call, Detect Tone	Accept Play	N/A
Detect Tone File	Probe Accept Call, Detect Tone File	Accept Play File	N/A
Echo Tone	Probe Accept Call, Generate Tone	Accept Detect Gen.	N/A
Echo Tone File	Probe Accept Call, Generate Tone and Receive Tone	Accept Detect Play File	N/A
Inbound Info Call	N/A	Inbound Info Call	Make Info Call
Outbound Info Call	N/A	Outbound Info Call	Receive Info Call
Inbound Put Message	N/A	Inbound Put Message	Make Put Message
Inbound Get Message	N/A	Inbound Get Message	Make Get Message

Table 3. IVR Test Description and Call Scenario Combinations

The methodology used to create the NIVR solution test environment consists of a passive call scenario that is used in conjunction with a stimulus call scenario. The load and probe applications generally drive the target application by providing stimulus to the target application. The types of stimulus that the load and probe applications provide includes offered calls, playing files, dialing digits, responding to prompts, etc. Table 3 lists the high-level test descriptions and the probe, target, and load scenarios that are required to execute the respective test.

Probe Scenarios

The stimulus provided by the probe application is used to measure and gather timing statistics for a call scenario. As the call scenario executes, the trace mask (mskTrace) is updated to keep track of each scenario milestone. The meanings of the bitmask values are specified in the namespace of the particular scenario. When the call is dropped, the scenario summary is logged and the Statistics array is updated for the completed scenario. The following list summarizes scenarios that are used by the Probe System:

- **Probe Reject** — This scenario measures the time to get a GCEV_DISCONNECT event from the Target System. This test requires the Target System to run the Reject Call scenario.
- **Probe DNIS Reject** — This scenario measures the time for the Target System to check the call DNIS digits prior to rejecting the incoming call.
- **Probe Accept** — This scenario measures the time for the Target System to answer an incoming call. If this DNIS_React scenario is not used, then this scenario is used to calculate the time to answer the incoming call without checking DNIS digits. When the DNIS_React scenario is used, by the target application, this scenario measures the time to answer the incoming call after checking DNIS digits.
- **Probe Accept Call, Detect Tone** — This scenario measures the time for the Target System to retrieve DNIS digits, answer the call, and then generate a user-defined tone. This scenario is used to calculate the time for the target application to generate a tone.
- **Probe Accept Call, Detect Tone File** — This scenario is similar to the *Probe Accept Call Detect Tone* scenario, but the Target System plays a user-defined tone file instead of generating the tone. This scenario is used to calculate the time for the target application to play a file.
- **Probe Accept Call, Generate Tone** — This scenario measures the time for the Target System to answer an incoming call and detect a tone that the Probe System generates. After the tone is detected from the probe application, then the target application generates a back tone to signal the detection of the probe tone. This scenario is used to calculate the time for the target application to detect a tone.
- **Probe Accept Call, Generate Tone and Receive Tone** — This scenario measures the time for the target application to answer an incoming call, detect a user-defined tone that is generated by the probe, and respond with another user-defined tone that the

Target System generates. This scenario is used to calculate the time for the target application to play a file after the detection of a tone.

Target Scenarios

Three types of scenarios are used by the Target application. The three types of scenarios are used for scenario management (ex: DNIS_React), target-load scenarios, and target-probe scenarios. The target-load scenarios are used to simulate ”services” that are provided by a computer telephony service provider, such as Voice Mail. The target-probe call scenarios are used to perform the actions that are under test by the probe application, such as Tone Generation.

The target-probe scenarios are discussed in the following list. The target-probe call scenarios are listed in Appendix D: Probe / Target Scenario Communication, using GR1129 format.

- **Reject Call** — This scenario rejects all incoming calls. If this scenario is not directly specified in the scenario script, then the NIVR checks DNIS digits when the incoming call was offered.
- **Accept Call** — This scenario accepts and answers an incoming call. It remains in the connected state until the caller disconnects. If this scenario is not directly specified in the scenario script, then the NIVR checks DNIS digits when the incoming call is offered.
- **Accept Call, Play Tone** — This scenario accepts and answers an incoming call and generates a DTMF signal until the caller disconnects.
- **Accept Call, Play Tone File** — This scenario accepts and answers an incoming call and plays a file that contains a dual tone frequency until the caller disconnects.
- **Accept Call, Detect Tone, and Generate Tone** — This scenario accepts and answers an incoming call. When a dual tone frequency is detected, the target responds by generating a back tone.
- **Accept Call, Detect Tone, and Play Tone File** — This scenario accepts and answers an incoming call. When a dual tone frequency is detected, the target responds by opening a dual tone frequency file and playing the file until the caller disconnects.

The target application also handles calls from the Load application to simulate peak usage congestion for the density that is under test. The scenarios in the following list are target-load scenarios and include Telcordia GR 1129 call flows described in Appendix E: Load/Target Scenario Communication. While the load and target devices are in a call, the load application may interrupt messages with digits, hang up during message delivery, or wait for the completion of messages.

- **Inbound Information Call** — The target accepts and answers the call prior to simulating an information system. During this call, the target requests a User ID from the caller. If the caller enters a User ID, then the target delivers an information message to the caller. The information message that is played is a random file that is selected at runtime; this forces the target application to perform a disk hit prior to playing the information message.
- **Outbound Information Call** — The target initiates a call to a Load application. When the call is answered, the target requests a User ID from the call destination. If the User ID is entered, the target performs a disk hit and plays a randomly selected file to the called party.
- **Inbound Get Message** — The target simulates a voice mail system when this call scenario is executed. After the call is answered, the target plays a greeting file and requests a User ID and Password. After the caller has entered a User ID and Password, the mailbox status is delivered to the caller. If the caller enters a voice mail command to listen to messages, then a random file is selected from the hard disk and is played to the caller.
- **Inbound Put Message** — The target simulates a voice mail system when this call scenario is executed. This call scenario is used to simulate a caller leaving a message in a voice mailbox. After the call is answered, the target plays a greeting file and requests the caller to enter a DTMF digit to begin recording. After the caller enters a DTMF digit, the target records until another DTMF digit is detected to signal the end of the message.

Load Scenarios

Stimulus provided by the load application is used to simulate realistic call congestion while the target application is under test. The OnBeginScenario method of the load call scenarios is used to determine how the load application will respond to prompts by the target application. In the source code, these decisions are called ”predictions” and are marked with a prediction bitmask that is included in the namespace of the particular load scenario. The possible responses include waiting for a particular prompt to complete before entering DTMF digits, interrupting a particular prompt with DTMF digits, or disconnecting the call at a particular prompt. These are the possible responses to prompts that callers perform on a daily basis. After all the decisions are made, the load application initiates the call and executes according to the predictions that were made during OnBeginScenario. At the conclusion of the call, the call data and predictions are entered into the load result file. The following list describes the call scenarios that run in the load application.

- **Make Get Message** — The Load application initiates a call to the target application, passing the scenario ID for the Inbound Get Message scenario. This call scenario is comprised of four individual stages. The four stages are: prompts for a User ID, password, a mailbox status, and the delivery of a voice mail message.

  Load application prediction determines if the Load application waits for the prompt or message delivery prior to sending digits to the Target, interrupts the Target prompt or message delivery with digits, or disconnects the call. All Load responses have the same probability of occurring at each stage. This scenario has 31 possible permutations. Using the current voice files and prediction probabilities, this scenario has connect times ranging from 2 to 21 seconds in length.

- **Make Put Message** — The Load application initiates a call to the target application, passing the scenario ID for the Inbound Put Message scenario. This call scenario is comprised of three individual stages. The Target answers the call and informs the caller that the person is not available. The Load application can send a digit to leave a message. After the message is recorded, the Target asks the caller if the message should be forwarded. The Load application predictions determine if the caller waits for the prompts to complete prior to sending digits, interrupts the prompts with digits, or disconnects the call during the prompt. Each prediction has the same probability of occurring at each stage. This scenario has 7 possible permutations. Using the current voice files and prediction probabilities, this scenario has connect times ranging from 2 to 20 seconds in length.

- **Make Info Call** — The Load application initiates a call to the target application, passing the scenario ID for the Inbound Information Call. This scenario is comprised of two individual stages. The Target requests a User ID and delivers an information message to the caller. The Load application predictions determine if the caller waits for the prompt or message to complete prior to sending digits, interrupts the prompt or message with digits, or disconnects the call during the prompt or message. Each prediction has the same probability of occurring at each stage. This scenario has 7 possible permutations. Using the current voice files and prediction possibilities, this scenario has connect times ranging from 2 to 9 seconds in length.

- **Retrieve Information Call** — The Load application initiates a call to the target application, passing the scenario ID for the Outbound Information Call. The target rejects this call and initiates an outbound call to the Load application. This scenario is comprised of two individual stages. The Target requests a User Password, and delivers the information message. The Load application predictions determine if the caller waits for the prompt or message to complete prior to sending digits, interrupts the prompt or message with digits, or disconnects the call during the prompt or message. Each prediction has the same probability of occurring at each stage. This scenario has 7 possible permutations. Using the current voice files and prediction probabilities, this scenario has connect times ranging from 2 to 9 seconds in length.

Solution Administration Design

The administration of the NIVR Platform utilizes a script file by default, called scenario.txt, which is included in the Target, Load, and Probe Zip files (see the *For More Information* section for the downloadable Zip file called *Network Interactive Voice Response*). The Test Manager expects each line in the test script to specify a scenario identification number. If the scenario number is valid, the scenario is assigned to the first empty element in the test array. The Test Manager reads each line of the test script file, placing all valid scenario numbers into the test array. Compile-time options allow a user to specify whether the Test Manager should randomly or sequentially assign scenarios to Global Call Software devices. The Test Manager uses an array to track the index of the test array that each device handle is currently using. If sequential test selection is used, each device traverses the test array until it reaches the end of the array, where it loops to the beginning of the test array.

NIVR Application Setup

Setting up the NIVR applications can be divided into three individual categories, which are the Target, Probe, and Load applications. The steps to configure and execute the three applications are listed as follows, in their respective sections.

Target Application

The Target application should be executed prior to the Probe and Load applications. To customize a particular characteristic of the target application, compile-time definitions need to be modified in the nivr.h file. The following list defines and discusses each compile-time option that can be modified.

- **TIMER_POOL_SIZE** — The number of CTimer objects that are queued in the TimerQ. TimerQ is maintained by the Registration object and keeps track of the timer information for the DlgcDev objects that are evenly distributed across all CTimer objects. The maximum value is 16, due to a Windows® Multi-Media Timer limitation. The default size of the timer pool is 2.
- **ACCEPT_CALLS_BEFORE_ANSWERCALL** — This option allows the traversal of both Dialogic® state machine paths that go from the OFFER state to the CONNECT state. This option is enabled by default.
- **EXIT_ON_TASKFAIL_EVENT** — This compile time option causes the application to shut down in response to a GCEV_TASKFAIL event. This option is disabled by default.
- **TRACK_EVENT_HANDLING_TIME** — This option keeps track of the time required to handle each type of event that the Event Manager receives. When the application is shut down, the average, min, max, and event counts are logged to the log file. This option is enabled by default.
- **SAVE_ALL_RECORD_FILES** — This option creates a unique file name for each recorded file when this option is enabled. When this option is disabled, each device has at most one file for recordings. This option is disabled by default

The following steps are required to execute the Target application:

1. Ensure that the working directory has the Target executable file, the scenario.txt script file, bin directory, and audio directory.
2. From the bin directory, execute the following commands:

   >createfiles vox
   >createfiles pcm
3. After the audio files are created in the audio directory, open the scenario.txt file and verify that it contains only the number 0. Zero is the number of the DNIS_React scenario and results in the target application assigning Test Scenarios by the DNIS digits that are passed with the offered call.
4. Open an MS-DOS® window in the working directory for every two spans that execute the target application.
5. Start each target application, using the following command syntax:

>NivrTarget	<trunk> <start chan> <max chan>
<trunk>	Represents the trunk number of the first board that runs for the respective process
<start chan>	Represents the first channel on the specified trunk that runs for the respective process
<max chan>	Represents the number of devices that runs in the respective process
	Note: You don't have to perform the data channel subtraction when running ISDN

6. After the application has started, you can verify that it is ready to accept a call by using ISDIAG, which is the Dialogic® ISDN Utility (for more information see *ISDN Software Reference for Linux and Windows* in the *For More Information* section) and pass the scenario ID for the Answer Call scenario (15). The call is accepted when the target is ready.

Probe Application

The Probe application should be executed after the appropriate number of loads is applied to the target application. To customize a particular characteristic of the probe application, compile-time definitions need to be modified in the probe.h file. The following list defines and discusses each compile-time option that can be modified.

- **LIMIT_RETRIES** — This option places a limit on the number of times a failed test can be retried before moving to the next test. When this option is enabled, the MAX_RETRIES variable is used to specify the maximum number of times a failed test can fail. This prevents a failed test from running continuously. The default value of MAX_RETRIES is 5. IMIT_RETRIES is disabled by default.
- **SNMP** — This option enables or disables SNMP capabilities. This option is disabled by default.

The following steps are required to execute the probe application:

1. Ensure that the working directory has the executable file, the scenario.txt script file, and the audio directory.

2. Open the scenario.txt script file and verify that the appropriate test IDs are included in this file. The following list includes the default probe test IDs:

14: Probe a Rejected Call
15: Probe an Answered Call
16: Probe Target Answering Call and Generating a Tone
17: Probe Target Answering Call and Playing a File
18: Probe Target Answering Call, Detecting a Tone, Replying Tone
19: Probe Target Answering Call, Detecting a Tone, Playing File

3. Open an MS-DOS® window in the working directory and start the probe application by executing the following command syntax:

>NivrProbe	<trunk> <start chan> 1
<trunk>.	Represents the trunk number of the first board that runs for the respective process
<start chan>	Represents the first channel on the specified trunk that runs for the respective process

Load Application

The Load application should be executed after the target application is ready to answer calls. To customize a particular characteristic of the load application, compile-time definitions need to be modified in the load.h file. The following list discusses each compile-time option that can be modified.

- **SAFETY_TIMER** — This option executes a safety timer during states that wait for a particular type of response from the target application. When this option is enabled, a timer is set to a default of 25 seconds to wait for particular responses. If the responses are not received prior to the expiration of the safety timer, the call is dropped and the next scenario executes. If the safety timer executes, the call scenario is marked with a failure in the results log file. This option is disabled by default.
- **RESTART_LINE_DEVICE_ON_FAILURE** — This option restarts the line device each time a call scenario is marked as a failure. This option is disabled by default.
- **TIMER_POOL_SIZE** — The number of CTimer objects that are queued in the TimerQ. TimerQ is maintained by the Registration object and keeps track of the timer information for the DlgcDev objects that are evenly distributed across all CTimer objects.The maximum value is 16, due to a Windows® Multi-Media Timer limitation. The default size of the timer pool is 2.
- **RANDOM_TEST_SELECTION** — This option specified whether the Test Manager selects call scenarios randomly or sequentially from the test array. The default configuration is random test selection.
- **TEST_DECISION** — This option specifies how the load application responds to prompts made by the target application. This option has four values:

DECIDE_RANDOM	A random response is selected for each target prompt
DECIDE_WAIT	The load devices wait for each prompt to complete before entering DTMF digits that push the target to the next stage of the call
DECIDE_DIGITS	The load devices barge-in each prompt with DTMF digits that push the target to the next stage of the call
DECIDE_HANGUP	The load devices disconnect during the first prompt by the target application

The default setting for this option is DECIDE_ RANDOM.

The following steps are required to execute the load application:

1. Ensure that the working directory has the executable file, the scenario.txt script file, and the audio directory

2. Open the scenario.txt script file and verify that the appropriate test IDs are included in this file. The following list includes the default probe test IDs:

    7: Make an information call
    8: Make a call that will be rejected
    9: Receive an information call from the target
    10: Leave a message in a target mailbox
    11: Retrieve a voice message from a mailbox
    12: Make a call that is connected for a random amount of time
    20: Force target to answer a call and play a tone, disconnect after a random amount of time
    21: Force target to answer a call and play a file, disconnect after a random amount of time

3. Open an MS-DOS® window in the working directory and start the probe application by executing the following command syntax:

>NivrLoad	<trunk> <start chan><max chan>
<trunk>	Represents the trunk number of the first board that runs for the respective process
<start chan>	Represents the first channel on the specified trunk that runs for the respective process
<mx chan>	Represents the number of devices that runs in the respective process

SS7 Test Environment Setup

Setting up the SS7 Test Environment can be divided into two categories: (1) SIU setup and (2) Target, Load, and Probe system setup. The steps to configure each category are listed in the following section.

SIU Setup

1. Connect a dumb terminal to the SIU using the serial interface. Directly access the SIU using a terminal emulation program (for example, hyperlink).
2. At the DK prompt >, set the IP address and subnet mask for the SIU:
   >cnsys:ipaddr=xx.xx.xx.xx;
   >cnsys:subnet=xx.xx.xx.xx;
3. Use the following command to verify the settings and features available on the SIU:
   >cnsyp;
4. Restart the SIU to enable the IP address by issuing the following command:
   >mnrsi;
5. After the SIU has restarted, access it through a telnet session:
   telnet xx.xx.xx.xx 8100
6. Edit and change the configuration file. The configuration file can be downloaded to the SIU through a FTP session.
7. A sample configuration file can be found in Appendix B: SS7 Config.txt File Contents.

Target and Probe System

In the directory containing the gctload executable, edit the system.txt file. The key parameters to be set are the Host ID, the Service ID, and AppModuleID. A sample system.txt file can be found in Appendix C: SS7 System.txt File Contents (host id=0, Dialogic Service ID=0x4d, and four application module Ids=0x1d, 0x2d, 0x3d, 0x5d).

Note: The number of application module IDs to be defined depends on the number of applications required to be activated on the Target or Probe system that requires interaction with the SIU.

Using the Dialogic® Configuration Manager (DCM), the following steps are required to set up an interface to the SIU:

1. Select Add Device from the Action pull-down menu
2. The Add Hardware Wizard window opens
3. Select **SS7** from the Family box and **DK SIU** from the Model box.
4. Click the Next button.
5. In the next screen, enter a name for the device DK SIU – "xxxxx" (xxxxx can be any name you give this device).
6. Click the Next button.
7. In the next screen define the properties of the DK SIU – "xxxxx".
8. On the System tab, ensure that the Host ID, Service Module ID, and AppModuleID matches those defined in the system.txt file, as defined earlier.
9. On the SIU Server tab, ensure that the IP address for the SIU is correctly defined.
10. To avoid re-entering the settings after a system restart, the above settings can be entered into the \Program Files\Dialogic\INF\SS7.inf file.

Results

Configuration 1 — DM/V960-4T1 Voice Board (PCI) in Intel ISP 2150

Configuration 1 studies the performance of Dialogic® System Release 5.1 software with Dialogic® DM/V960-4T1 Voice Boards (PCI) running 4ESS ISDN protocol in an Intel ISP 2150 on the Windows NT® platform. The Intel ISP 2150 has two PCI slots, limiting this configuration density to 184 ISDN channels. The first channel was probed for all tests, as density increased at single 23 channel intervals. The guard time, or inter-call delay, for the probe and load stimulus was set to 2.5 seconds. During the testing of this configuration, average system CPU usage was between 5-10%. Busy Hour Call Completion was measured between12,200 and 49,200 calls per hour, as the call density was increased. Tables 4 through 7 present the performance statistics for 1, 2, 4, and 8 trunks.

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	240	190	280	20
Reject Call	240	190	290	20
Answer Call Generate Tone	390	340	490	30
Answer Call Play File	630	450	860	80
Answer Call Echo Tone	490	430	550	20
Answer Call Echo File	710	560	970	80

Table 4. Configuration 1 — 1 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	290	220	430	40
Reject Call	290	220	470	40
Answer Call Generate Tone	500	380	780	80
Answer Call Play File	650	470	970	100
Answer Call Echo Tone	590	480	840	70
Answer Call Echo File	750	600	1000	90

Table 5. Configuration 1 — 2 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	310	190	500	50
Reject Call	310	220	590	50
Answer Call Generate Tone	500	390	780	70
Answer Call Play File	660	490	940	90
Answer Call Echo Tone	610	490	870	70
Answer Call Echo File	770	590	1040	90

Table 6. Configuration 1 — 4 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	290	190	470	40
Reject Call	290	200	470	40
Answer Call Generate Tone	480	360	750	60
Answer Call Play File	630	480	890	80
Answer Call Echo Tone	560	450	750	50
Answer Call Echo File	730	580	920	80

Table 7. Configuration 1 — 8 Trunk Performance Statistics

Configuration 2 — Dialogic® DM/V960-4T1 Voice Board (cPCI) in an Intel ZT 5082

Configuration 2 studies the performance of Dialogic® System Release 5.1 software with Dialogic® DM/V960-4T1 Voice Boards (cPCI) running 4ESS ISDN protocol in an Intel ZT 5082 on the Windows® 2000 platform. The densities that are under test for this configuration range from 276 to 460 ISDN channels. The first channel was probed for all tests, as density increased at 184 channel intervals. The guard time for the probe and load stimulus was set to 2.5 seconds. During the testing of this configuration, average system CPU usage was between 7-28%. Busy Hour Call Completion was measured between 130,000 and 220,000 calls per hour, as the call density was increased. Tables 8 through 10 present the performance statistics for 12, 16, and 20 trunks.

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	270	200	360	30
Reject Call	270	220	380	20
Answer Call Generate Tone	440	360	560	40
Answer Call Play File	560	470	670	40
Answer Call Echo Tone	510	420	780	50
Answer Call Echo File	620	550	730	40

Table 8. Configuration 2 — 12 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	270	220	410	40
Reject Call	280	200	410	30
Answer Call Generate Tone	450	380	590	40
Answer Call Play File	560	480	690	40
Answer Call Echo Tone	500	480	630	40
Answer Call Echo File	620	520	810	50

Table 9. Configuration 2 — 16 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	270	220	440	40
Reject Call	280	220	360	30
Answer Call Generate Tone	450	380	580	40
Answer Call Play File	570	470	750	60
Answer Call Echo Tone	510	440	660	40
Answer Call Echo File	630	520	890	60

Table 10. Configuration 2 — 20 Trunk Performance Statistics

Configuration 3 — Dialogic® D/480JCT-2T1 Media Board in an Intel ISP 2150

Configuration 3 studies the performance of Dialogic® System Release 5.1 Production with Dialogic® D/480JCT-2T1 Media Boards running 4ESS ISDN protocol in an Intel ISP 2150 on the Windows NT® platform. This configuration is limited to 92 ISDN channels because the ISP 2150 has two PCI slots. The first channel was probed for all tests, as density increased at 23 channel intervals. The guard time for the probe and load stimulus was set to 2.5 seconds. During the testing of this configuration, average system CPU usage was between 2-5%. Busy Hour Call Completion was measured between 12,000 and 46,000 calls per hour as the call density was increased. Tables 11 through 14 present the performance statistics for 1, 2, and 4 trunks.

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	80	60	200	20
Reject Call	60	40	170	20
Answer Call Generate Tone	190	150	420	30
Answer Call Play File	240	200	410	30
Answer Call Echo Tone	310	270	470	30
Answer Call Echo File	350	310	530	30

Table 11. Configuration 3 — 1 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	110	60	340	40
Reject Call	60	40	280	40
Answer Call Generate Tone	250	160	600	70
Answer Call Play File	290	200	590	70
Answer Call Echo Tone	360	270	1,020	70
Answer Call Echo File	410	310	670	70

Table 12. Configuration 3 — 2 Trunk Performance

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	140	60	420	60
Reject Call	150	60	480	70
Answer Call Generate Tone	300	180	800	90
Answer Call Play File	360	230	840	90
Answer Call Echo Tone	410	280	690	80
Answer Call Echo File	460	320	820	80

Table 13. Configuration 3 — 4 Trunk Performance Statistics

Configuration 4 — Dialogic® D/600JCT-2E1 Media Board and Dialogic® DSC131 SIU in an Intel ISP 2150

Configuration 4 studies the performance of SR 5.01 Production with Dialogic® D/6000JCT-2E1 boards and a Dialogic DSC131 Signaling Interface Unit in an Intel ISP 2150 on the Windows NT® platform. This configuration is limited to 119 ISDN channels because the ISP 2150 has two PCI slots. The first channel was probed for all tests, as density increased at 30 channel intervals. The guard time for the probe and load stimulus was set to 2.5 seconds. During the testing of this configuration, average system CPU usage was between 1.5-2%. Busy Hour Call Completion was measured between 12,200 and 49,200 calls per hour as the call density was increased. Tables 14 and 15 present the performance statistics for 2 and 4 trunks.

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	100	60	410	40
Reject Call	110	30	440	40
Answer Call Generate Tone	220	160	660	60
Answer Call Play File	260	200	590	60
Answer Call Echo Tone	340	270	740	50
Answer Call Echo File	390	310	720	50

Table 14. Configuration 4 — 2 Trunk Performance Statistics

Scenario Name	Average Time	Min Time	Max Time	Std. Dev.
Answer Call	160	60	590	80
Reject Call	160	60	630	70
Answer Call Generate Tone	210	160	800	60
Answer Call Play File	270	200	740	60
Answer Call Echo Tone	380	270	950	80
Answer Call Echo File	440	310	890	80

Table 15. Configuration 4 — 4 Trunk Performance Statistics

Conclusions

The tables and graphs presented in the Results section present benchmark performance statistics for various NIVR solution configurations. The NIVR solution configurations benchmarked in this document use a single-threaded, multi-process environment that implements the Dialogic® Global Call API on Windows NT® and Windows® 2000 platforms. Preliminary result analysis resulted in limiting each individual process to two ISDN trunks. Limiting a process to two ISDN trunks introduces a common denominator for the comparison of benchmark test results for various Dialogic® hardware configurations at different system densities.

The resultant benchmark data included hundreds of measurements for each probed scenario. It was observed that there was a very short settling time at the start of testing. Within seconds, the system approached a steady state for both CPU loading and timed call measurements. Current benchmark test results highlight Dialogic® DM3 Media Boards scalability while retaining acceptable system resource usage measurements. At a port density of 460 ports, Performance Monitor reported 28% CPU Utilization. It is expected that significant capacity expansion is potentially available. It is intended that higher densities will be investigated and this Application Note updated with the resultant data. It is observed that at higher densities the measured scenario max values increase, but not to an unacceptable level. The benchmark statistics represent a snapshot of the performance statistics that are currently available.

Appendix A: Hardware Component Specifications

Intel ISP 2150 Internet Server Platform

Component	Component Description
CPU	Dual Pentium III 750 MHz
Operating System	Windows NT® 4.0 / Service Pack 5
Memory	512 MB
Memory Speed	100 MHz
L1 Cache Size	32 KB
L2 Cache Size	256 KB
Disk Size	9.1 GB SCSI
SCSI Controller	Adaptec AIC-7896 Dual Channel- one Ultra2 (LVD) (up to 80 MB/s)
Supplier Information	Intel Corporation

Intel ZT 5082 cPCI Platform

Component	Component Description
CPU	Pentium III 800 MHz
Operating System	Windows® 2000 Server
Memory	512 MB
Memory Speed	100 MHz
L1 Cache Size	32 KB
L2 Cache Size	256 KB
Disk Size	9.1 GB SCSI
SCSI Controller	Adaptec AIC-7896 Dual Channel - one Ultra2 (LVD) (up to 80 MB/s)
Supplier Information	Intel Corporation

Tech Support Inc, Industrial Workstation

Component	Component Description
CPU	Pentium III 500 MHz
Memory	128 MB
Memory Speed	PC 100 MHz
L1 Cache Size	322 KB
L2 Cache Size	512 KB
Disk Size	9.1 GB SCSI
SCSI Controller	Adaptec 1790
Supplier Information	Tech Support, Inc.

DataKinetics Signaling Interface Unit (SIU)

Component	Component Description
System	DSC 131
System Software Version	5.12
Protocol	ISUP
Supplier Information	Intel Corporation

Appendix B: SS7 Config.txt File Contents

```
*    DSC231 Protocol Configuration File (config.txt)
*    Refer to the DSC131/DSC231 User Manual.

*    SIU commands :
*    Set the SIU instance. Set to SIUA for standalone, SIUA or SIUB for dual
operation.
*    SIU_INSTANCE instance_token = SIUA | SIUB
*SIU_INSTANCE             SIUA
SIU_INSTANCE              SIUA
*
*    Define the network address of this SIU :
*    SIU_ADDR    network_address
SIU_ADDR         10.253.34.81
*
*    Define the network address of the partner SIU (dual operation only) :
*    SIU_REM_ADDR remote_address
*SIU_REM_ADDR    146.152.185.182
*
*    Define the number of hosts that this SIU will connect to :
*    SIU_HOSTS num_hosts
SIU_HOSTS        1
*
*    Set physical Interface Parameters :
*    PCCS6_BOARD port_id bpos num_pcm     flags
*    PCCS3_BOARD port_id bpos num_pcm     flags
PCCS6_BOARD     0        4       0       0x0002
*
*    MTP Parameters :
*    MTP_CONFIG local_spc ssf   options
*MTP_CONFIG      0x12c     0x8    0x0000
MTP_CONFIG       1         0x8    0x0000
* try to config as multiple local point code
*MTP_CONFIG      0         0x0    0x0000
*
*    Define linksets :
*    MTP_LINKSET linkset_id adjacent_spc num_links flags   local_spc ssf
*MTP_LINKSET     0                3     1          0x0000  0x12c       0x08
MTP_LINKSET      0                2     1          0x0000  1           0x8
*
*MTP_LINKSET     1                100     1        0x8000  200        0x8
*
*    Define signalling links :
*    MTP_LINK
*      link_id linkset_id link_ref slc bpos blink stream timeslot flags
MTP_LINK  0   0           0         0   4    0     0x10    0x10      0x0006
*MTP_LINK  1   1          0         0   4    0     0x11    0x10      0x0006
*MTP_LINK  0              0         0   4    1     0x11    0x10      0x0006

*    Define a route for each remote signalling point :
*    MTP_ROUTE dpc linkset_id user_part_mask flag   bkplinkset
*MTP_ROUTE     200  0          0x0028          0x0000 1
*MTP_ROUTE     3    0          0x0028
MTP_ROUTE      2    0          0x0020

*    ISUP Parameters :
*    ISUP_CONFIG local_pc ssf user_id options num_grps num_ccts
ISUP_CONFIG     0x1       0x8  0x4d     0x0474   4        128
*ISUP_CONFIG    200       0x8  0x1d     0x0434   1         32
*
```

```
*    Define ISUP circuit (groups) :
*    ISUP_CFG_CCTGRP
*                 gid dpc base_cic base_cid cic_mask    options host_id
*[ dtiB1]
ISUP_CFG_CCTGRP  0  2  1        1         0x7fff7fff  0x08000003  0x00
*[ dtiB2]
ISUP_CFG_CCTGRP  1  2  0x21        0x21         0x7fff7fff  0x08000003  0x00
*[ dtiB3]
ISUP_CFG_CCTGRP  2  2  0x41        0x41         0x7fff7fff  0x08000003  0x00
*[ dtiB4]
ISUP_CFG_CCTGRP  3  2  0x61        0x61         0x7fff7fff  0x08000003  0x00
*ISUP_TRACE 0x0000003f 0x0000007f 0x000003fff
*ISUP_CFG_CCTGRP 1  2   0x1    0x21    0x7fff7fff  0x0003  0x00  0x1d 3 0x08
*ISUP_CFG_CCTGRP 0  100 1      1       0x7fff7fff  0x0003  0x01
*ISUP_CFG_CCTGRP 1  200 0x21   0x21    0x7fff7fff  0x0003  0x00
*
*    TUP Parameters :
*    TUP_CONFIG local_pc ssf user_id options num_grps num_ccts
*
*TUP_CONFIG      2  0x8  0x1d  0x0000  8  96
*
*    Define TUP circuit (groups) :
*    TUP_CFG_CCTGRP gid dpc base_cic base_cid cic_mask options host_id
*
*TUP_CFG_CCTGRP  0  1  1  1  0x7fff7fff  0x0000 0x00
*
*    NUP Parameters :
*    NUP_CONFIG local_pc ssf user_id options [ num_grps num_ccts]
*
*NUP_CONFIG 2    0x8  0x1d  0x0000  8  32
*
*    Define NUP circuit (groups) :
*    NUP_CFG_CCTGRP gid dpc base_cic base_cid cic_mask options [ host_id]
*
*NUP_CFG_CCTGRP  0  1  1  1  0x7fff7fff  0x0003 * 0x00

*    SCCP Parameters :
*    SCCP_CONFIG local_pc ssf  options
*SCCP_CONFIG     0x12c      0x8  0x0004

*    Define SCCP Remote signalling points :
*    SCCP_RSP spc  rsp_flags
*SCCP_RSP     3  0x00

*    Define all local sub-systems :
*    SCCP_LSS ssn   module_id lss_flags
*SCCP_LSS        0xfc  0x0d      0x00
*
*    Define all remote sub-systems :
*    SCCP_RSS spc ssn   rss_flags
*SCCP_RSS    3   0xfc 0x0
*
*    Define all local sub-systems that require notification of
*    changes in state of other signalling points or sub-systems :
*    SCCP_CONC_LSS local_ssn RSP remote_spc
*SCCP_CONC_LSS local_ssn RSS remote_spc remote_ssn
*SCCP_CONC_LSS    0xfc      RSP  3
*SCCP_CONC_LSS    0xfc      RSS  3         0xfc

*    Define all remote signalling points that require notification
*    of change in state of local sub-systems :
```

```
*    SCCP_CONC_RSP remote_spc LSS local_ssn
*SCCP_CONC_RSP      3         LSS   0xfc

*    The following commands allow the REM_API_ID module to receive
*    notification of changes of state of other signalling points or
*    sub-systems.
*SCCP_LSS  0x12c   0x0d 0
*SCCP_CONC_LSS      0x12c  RSS 3  0xfc
*
*TCAP_TRACE 0x07   0x0f 0x00
*
*    Cross Connections :
*    These commands control the connection of voice channels through
*    the SIU. The default configuration cross-connects all timeslots
*    other than 0 and 16 between the two ports of each PCCS6 card.
*    MVIP_XCON bpos op_stream op_slot mode ip_stream ip_slot pattern
*
*STREAM_XCON 4 16 17 3 0xfffefffe 0x00
*STREAM_XCON 5 16 17 3 0xfffefffe 0x00
*
* End of file
```

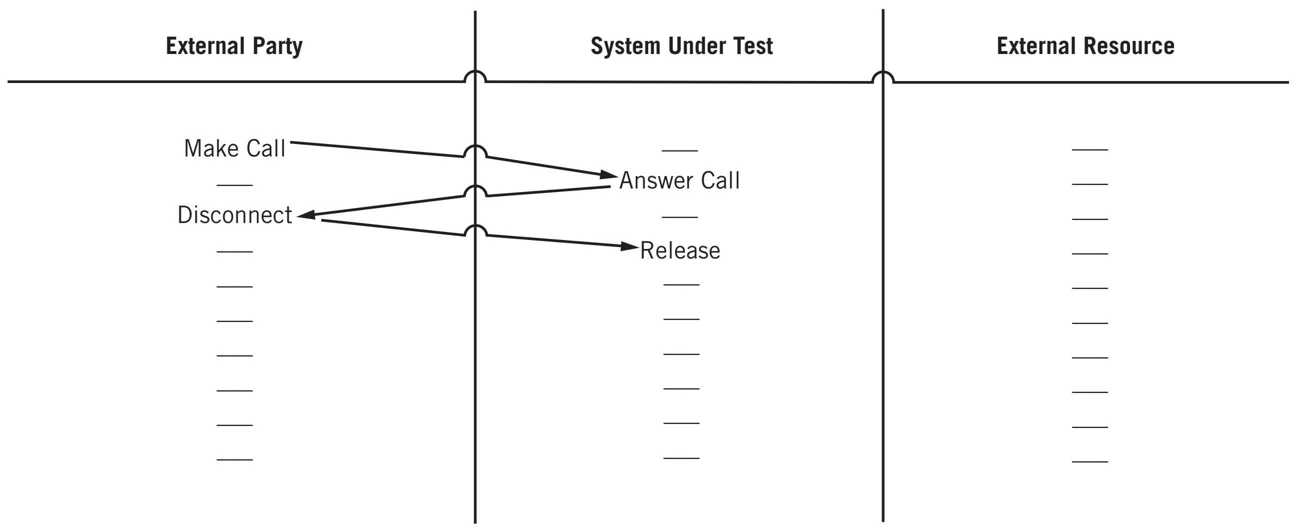
Appendix C: SS7 System.txt File Contents

```
*
* Module Id's running locally on the host machine:
*
LOCAL          0xb0           * rsi Module Id
LOCAL          0xef           * REM_API_ID Module Id (s7_log)
LOCAL          0xfd           * rsicmd Module Id
LOCAL          0x1d           * ctu Module Id
LOCAL          0x2d        * application Module Id
LOCAL          0x3d           * application Module Id
LOCAL          0x4d           * dialogic Module Id
LOCAL          0x5d           * application Module Id
LOCAL          0x6d           * application Module Id
*
* Redirect modules running on the SIU to RSI:
*
REDIRECT       0x20    0xb0    * SSD module Id
REDIRECT       0xdf    0xb0    * SIU_MGT module Id
REDIRECT       0x22    0xb0    * MTP3 module Id
REDIRECT       0x14    0xb0    * TCAP module Id
REDIRECT       0x33    0xb0    * SCCP module Id
REDIRECT       0x32    0xb0    * RMM module Id
REDIRECT       0x23    0xb0    * ISUP module Id
REDIRECT       0x4a    0xb0    * TUP/NUP module Id
*
* Now start-up the Host tasks ....
*
FORK_PROCESS    .\s7_log.exe
FORK_PROCESS    .\rsi.exe -r.\rsi_lnk.exe  -l1
*
* Start the Host-SIU link:
*
* FORK_PROCESS    .\rsicmd.exe 0 0xef 0 10.253.34.81 9000
*
* Example application programs:
*
* FORK_PROCESS  .\ctu.exe -m0x1d -o0x1fff
* FORK_PROCESS  .\ttu.exe -m0x0d -n0x66
```

Appendix D: Probe/Target Scenario Communication

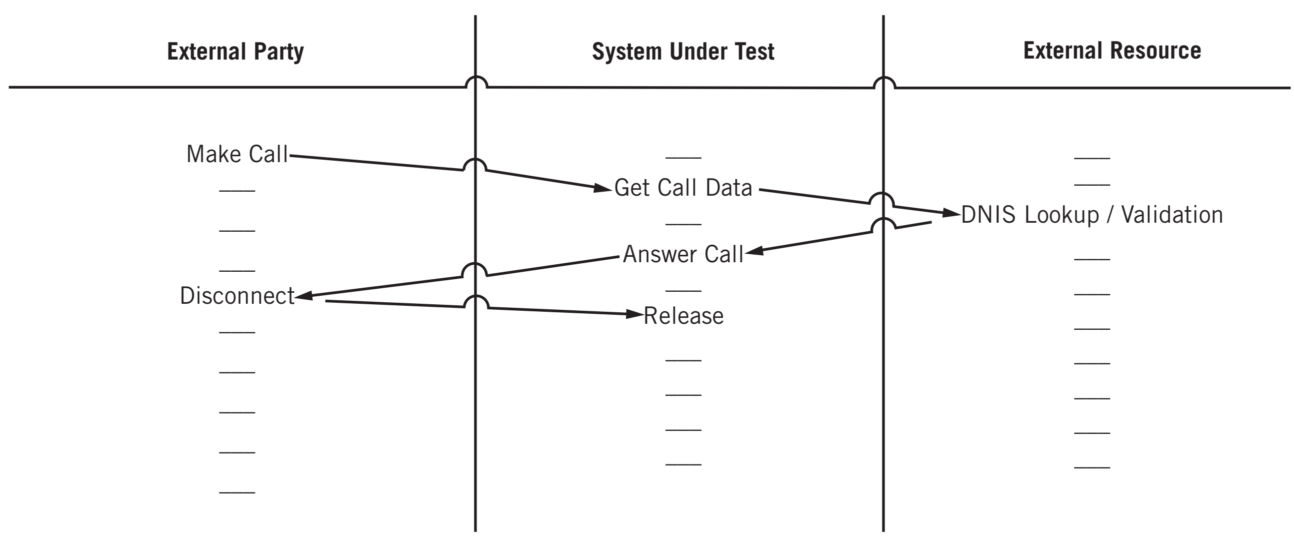
Call Type: Immediate ACCEPT

External Party: Probe Application
System Under Test: Target Application



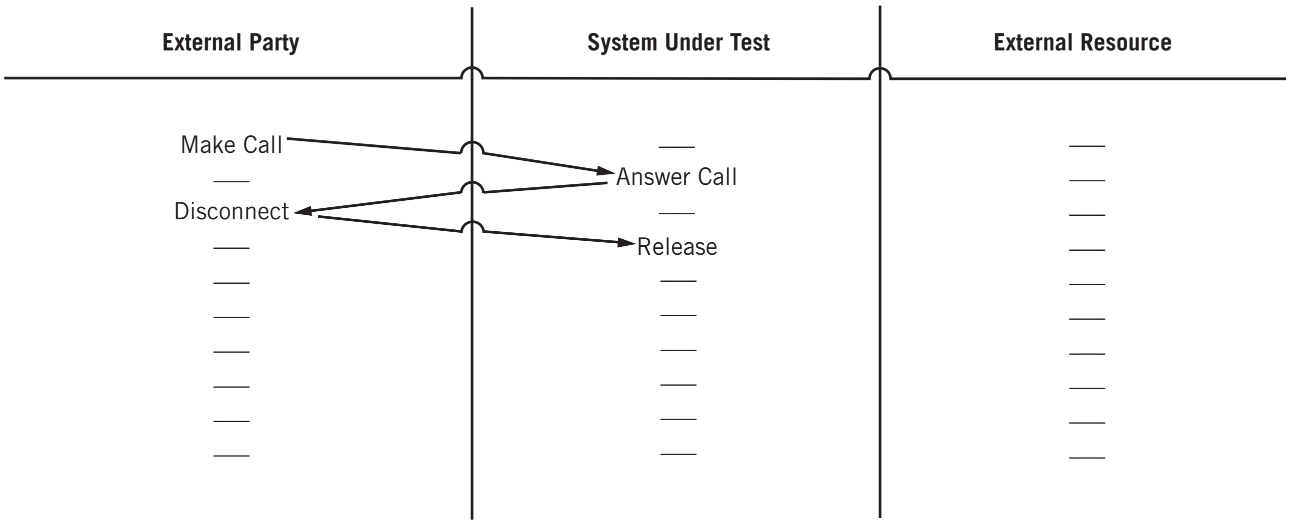
Call Type: DNIS ACCEPT

External Party: Probe Application
System Under Test: Target Application



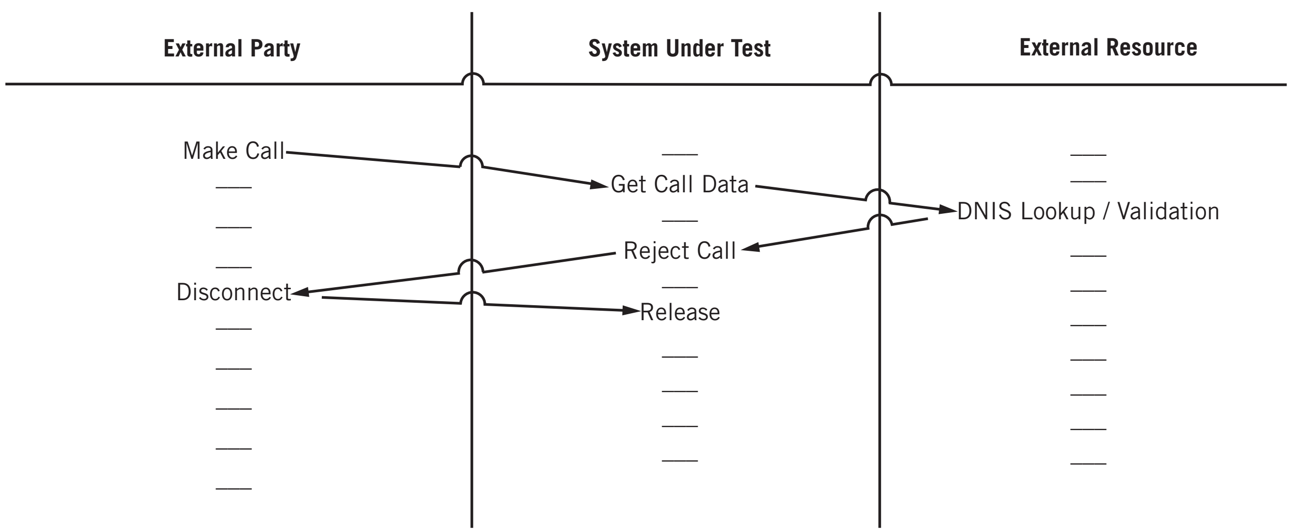
Call Type: Immediate REJ

External Party: Probe Application
System Under Test: Target Application



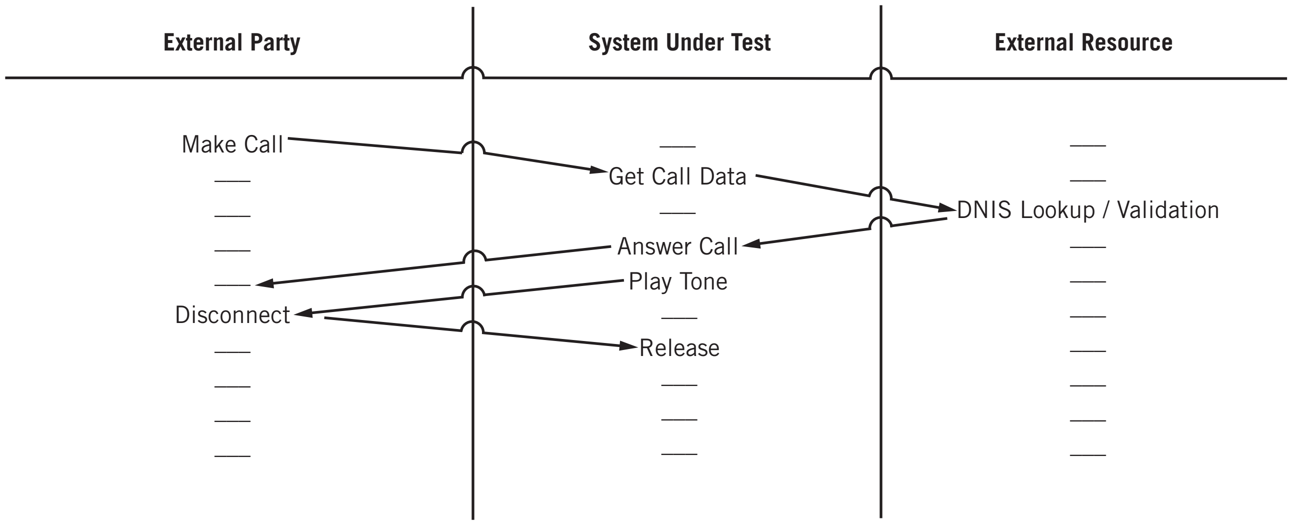
Call Type: DNIS REJ

External Party: Probe Application
System Under Test: Target Application



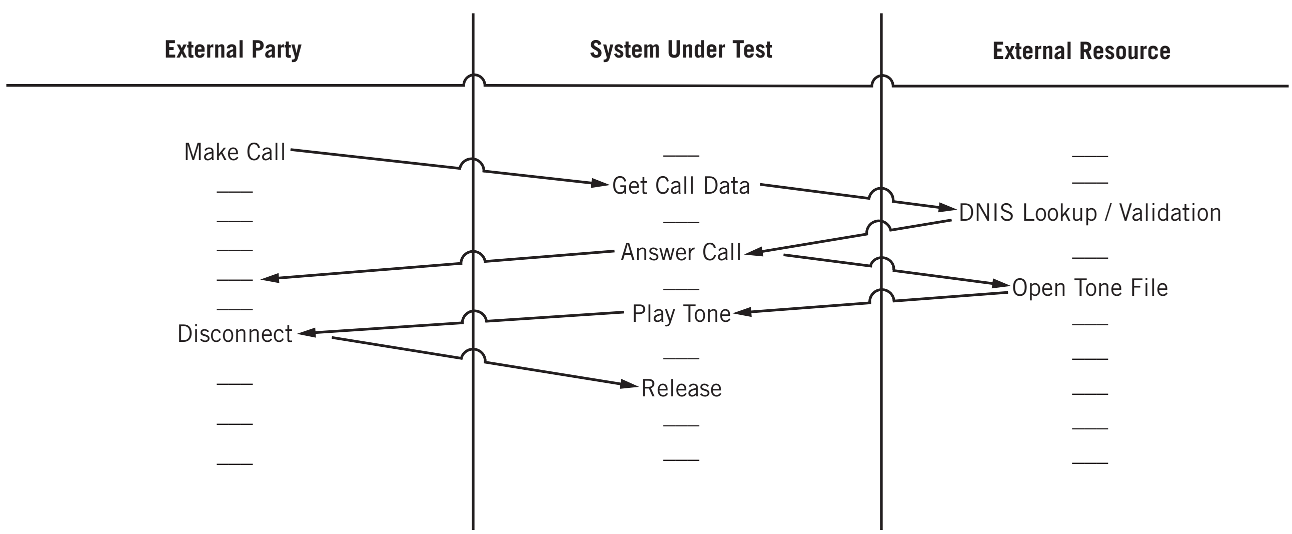
Call Type: Detect Tone

External Party: Probe Application
System Under Test: Target Application



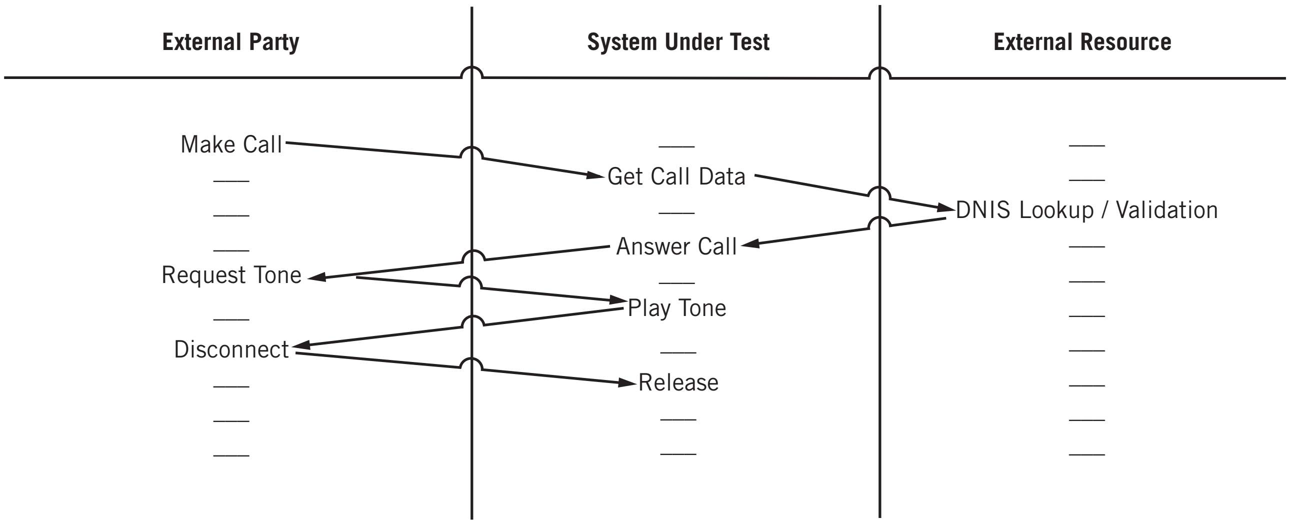
External Party: Probe Application
System Under Test: Target Application

Call Type: Detect Tone File



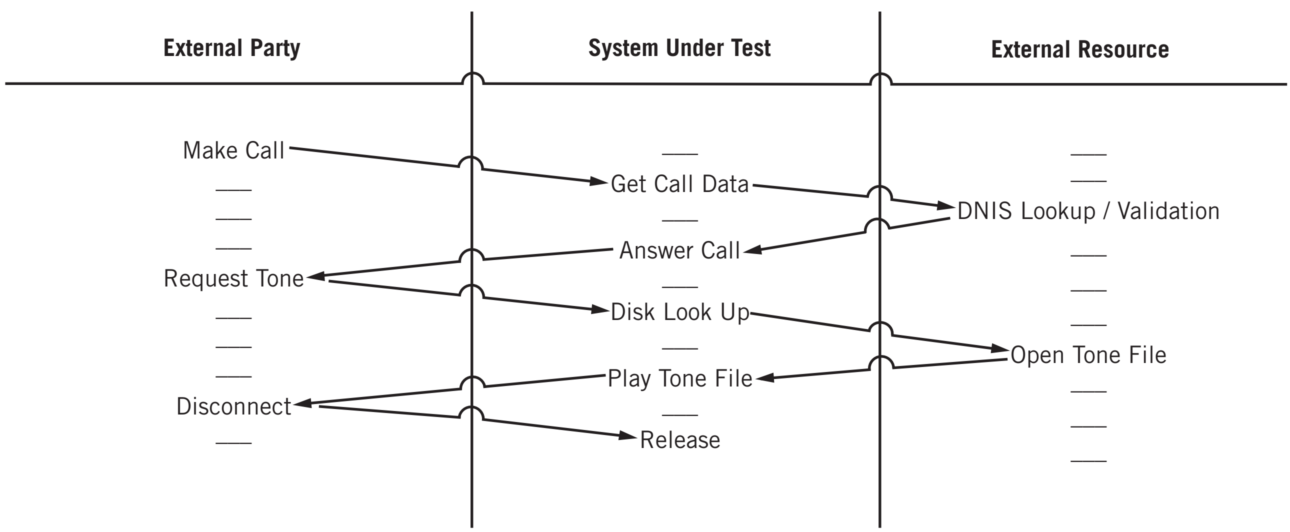
Call Type: Echo Tone

External Party: Probe Application
System Under Test: Target Application



Call Type: Echo ToneFile

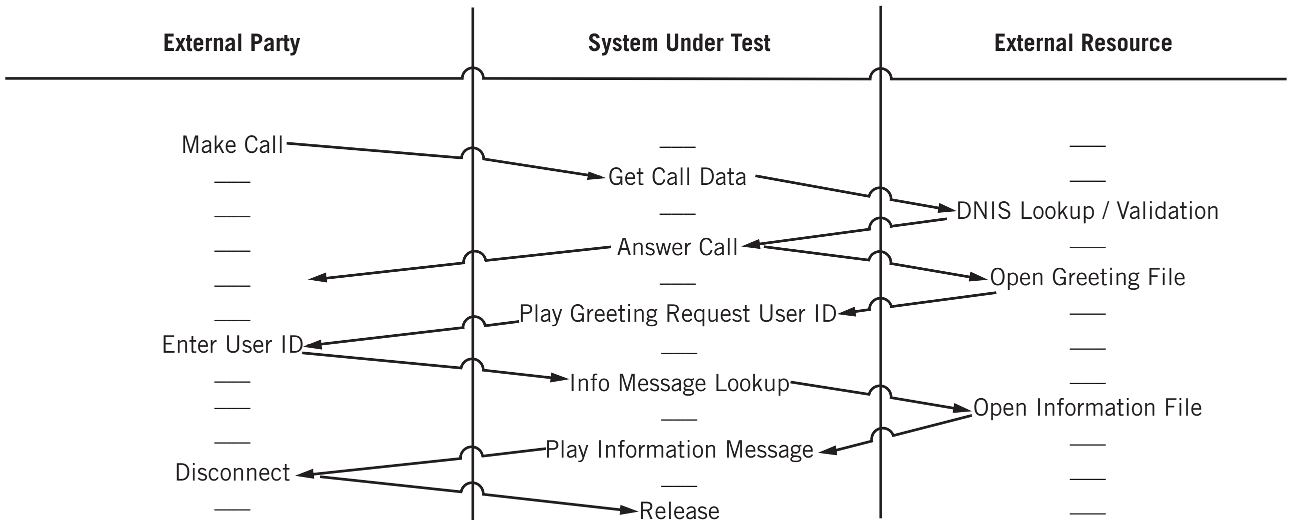
External Party: Probe Application
System Under Test: Target Application



Appendix E: Load / Target Scenario Communication

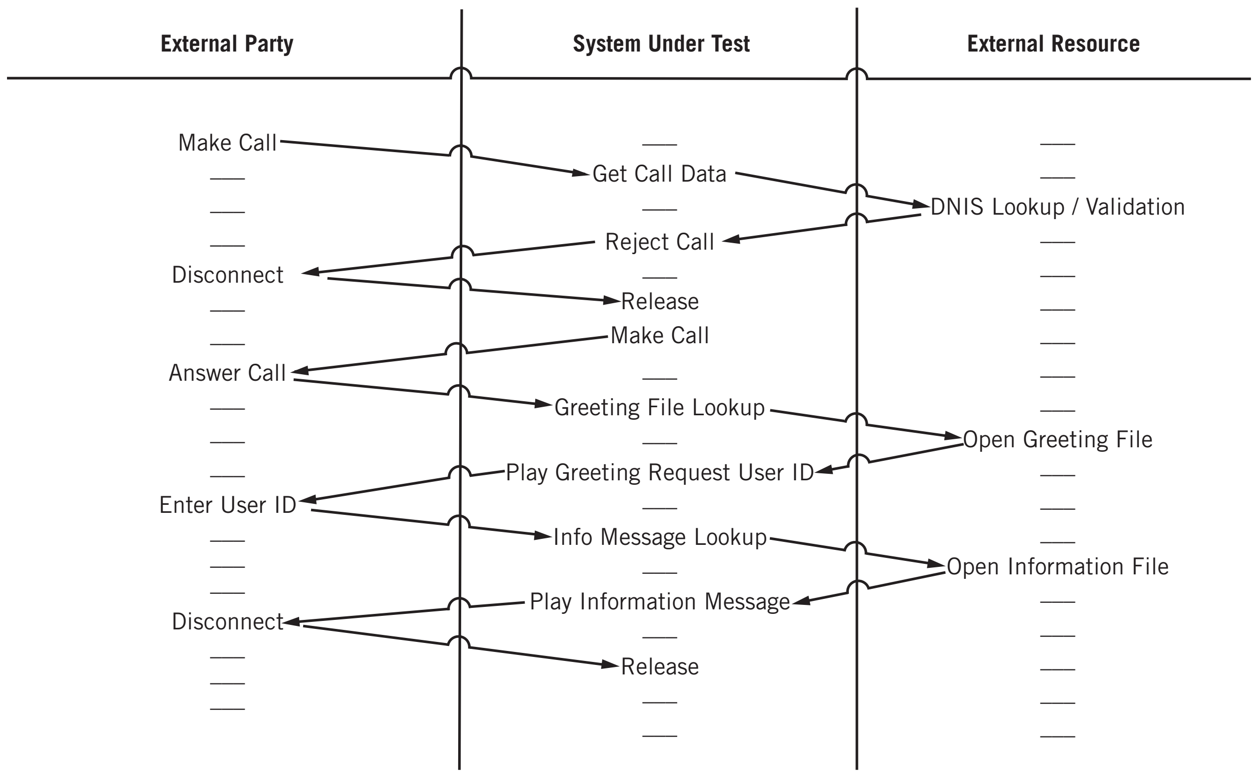
Call Type: Information Call

External Party: Load Application
System Under Test: Target Application



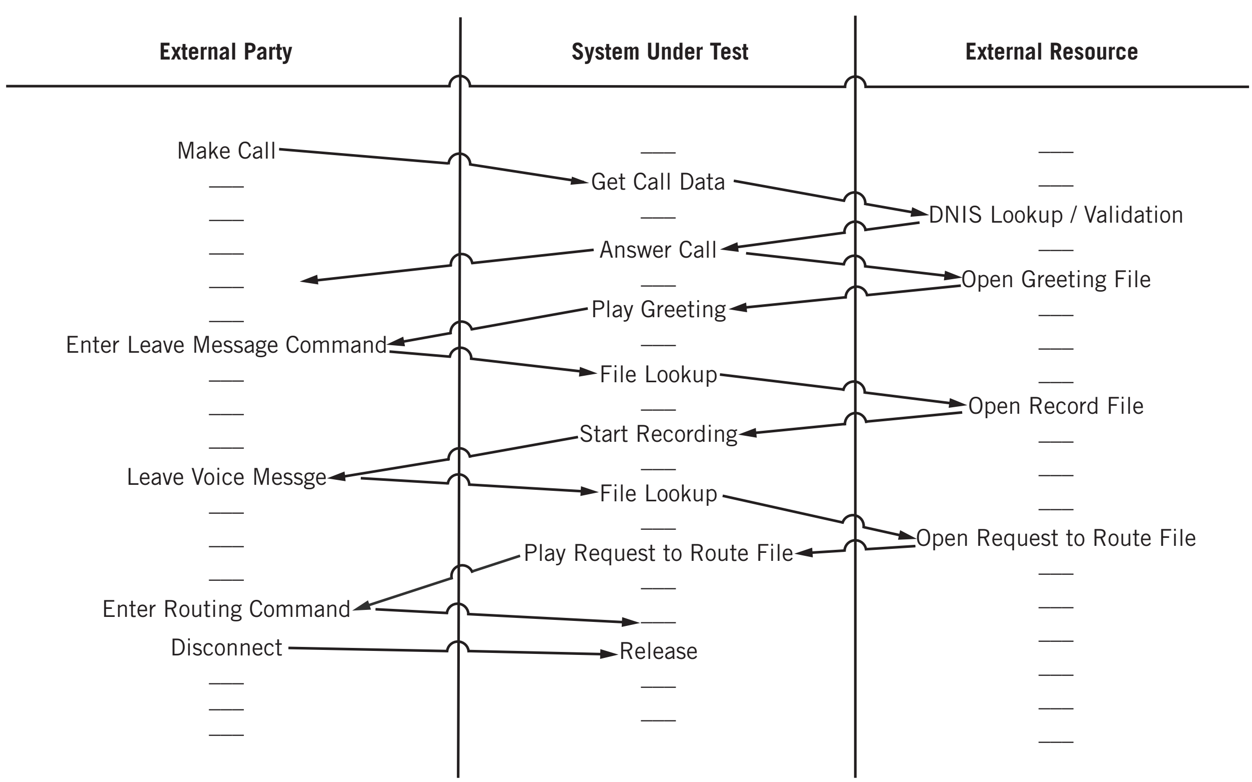
Call Type: Outbound Information Call

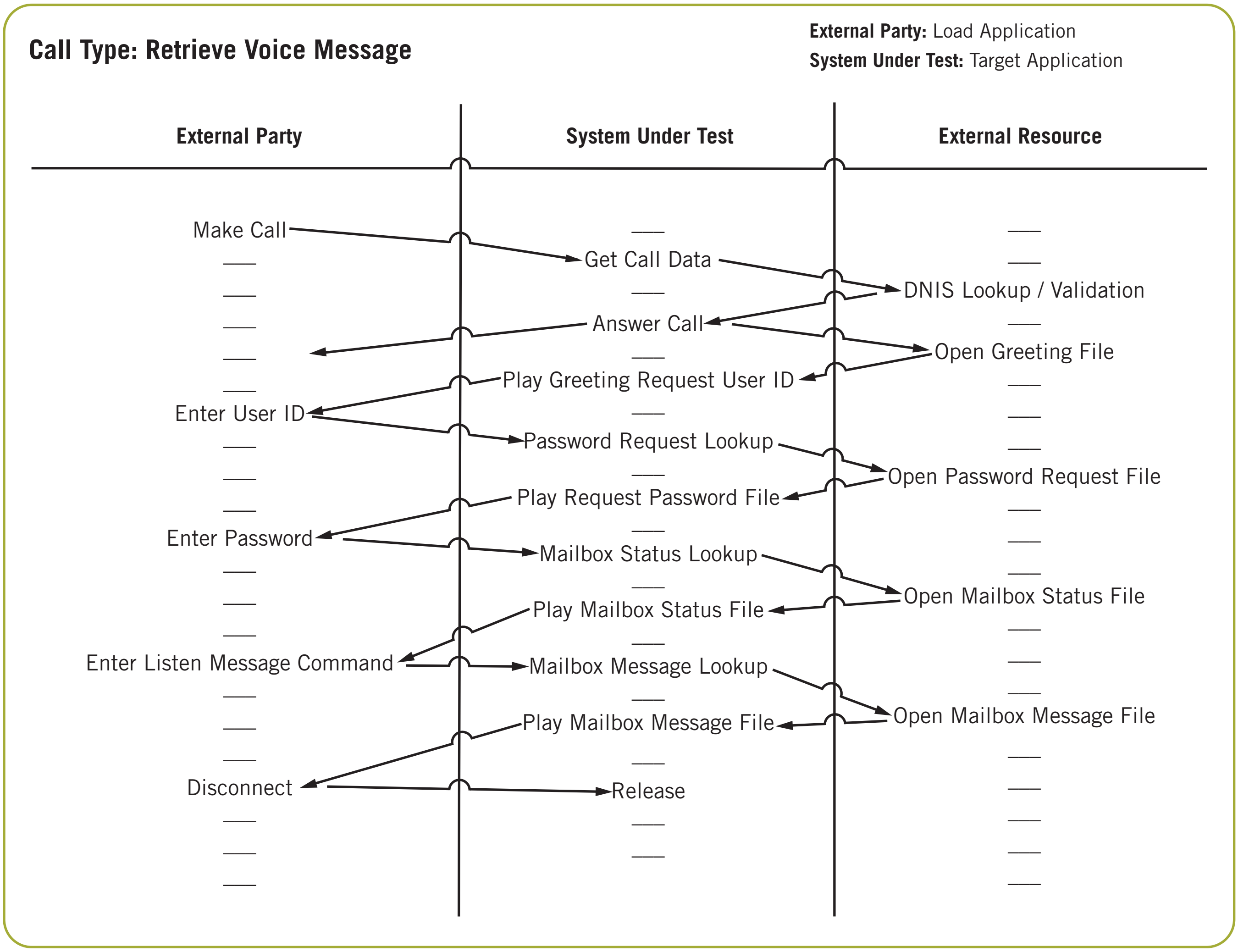
External Party: Load Application
System Under Test: Target Application



Call Type: Leave Voice Message

External Party: Load Application
System Under Test: Target Application




Call Type: Retrieve Voice Message
External Party: Load Application
System Under Test: Target Application
External Party
System Under Test
External Resource
Make Call
Get Call Data
DNIS Lookup / Validation
Answer Call
Open Greeting File
Play Greeting Request User ID
Enter User ID
Password Request Lookup
Open Password Request File
Play Request Password File
Enter Password
Mailbox Status Lookup
Open Mailbox Status File
Play Mailbox Status File
Enter Listen Message Command
Mailbox Message Lookup
Open Mailbox Message File
Play Mailbox Message File
Disconnect
Release

Acronyms

ACE SNMP	The SNMP Management Console provided by Diversified Data Resources, Inc.
ANI	Automatic Number Identification, the address of the caller, or call originator
BHCC	Busy Hour Call Completion, a measure of the capability limits of the specified configuration to handle the specific calls
DNIS	Dialed Number Identification String, the number or address dialed by the caller; the desired destination
DTMF	Dual Tone Multi-Frequency
IPC	Inter Process Communication
Global Call	The Dialogic name of the standard network interface access API that provides the application a single API for all interface technologies
IVR	Interactive Voice Response system
Load	The call generation portion of the test environment
MIB	Management Information Base, the file that describes the specific details of a particular resource within an SNMP environment
NIVR	Network IVR, an Interactive Voice Response system specifically designed to be one component, within a networked service delivery environment
PBX	Private Branch Exchange
Probe	The portion of the test environment that "probes", or queries a system under test and collects performance data
SNMP	Simple Network Management Protocol, the specification and standard for open networked system management architecture
SIU	Signaling Interface Unit
UM	Unified messaging
VM	Voice Mail or Voice Messaging

For More Information

C++ Communication Services Framework — http://sourceforge.net/projects/commsvcfw

Introduction to Communication Services Framework (CSF) — http://www.dialogic.com/ goto/?8989

A Zip file, *Network Interactive Voice Response,* containing the source code for the Target Application (Target.zip), Probe Application (Probe.zip), and Load Application (Load.zip) can be downloaded at http://www.dialogic.com/goto/?10642

ISDN Software Reference for Linux and Windows — http://www.dialogic.com/manuals/docs/isdn_api_v5.pdf

To learn more, visit our site on the World Wide Web at **http://www.dialogic.com/.**

Dialogic Corporation
9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9

 09/07 10464-01