INTRODUCTION to MAPS™
MAPS™

MA - Message Automation
+ 
PS - Protocol Simulation
About MAPS™

- **MAPS** stands for **Message Automation** and **Protocol Simulation**
- It is a generic framework for the generation of telecommunications protocol messages and transmission of bearer traffic.
- **MAPS™** is built on a proprietary scripting language developed by GL Communications.
- All **MAPS™** products come with out-of-the-box scripts that act as fully functional state machines for the relevant protocol.
About MAPS™

- **Scripts**: Scripts act as the state-machine, or engine for a given call. The logic of what messages to send when is all contained in a script.

- **Messages**: MAPS has an inventory of generic Message Templates (ex: Invite.txt) which it loads from the hard drive when transmitting an actual message. Messages are completely customizable.

- **Profiles**: Scripts and Messages are kept as generic as possible. Specific information (ex: Contact = 12345@sip.carrier.com) about a call is sourced from .xml profiles.
Basic Requirements for Simulation

- **Message Templates**
  The message templates are nothing but structure of message stored in particular file format. e.g.: SS7 Protocol suite message template will have “. HDL” format.

- **A ‘Script’**
  To send and receive these messages between two nodes and take appropriate actions for a particular message.

- **An ‘Import’ mechanism**
  A mechanism for reading the contents of the message template, and replacing the Key Identifier with the value given by the user (or some other means) at the run time. This process of inserting the user values into the message template before sending is called “Import”.
Basic Requirements for Simulation...

- **An ‘Export’ mechanism**
  A mechanism to extracting Key Identifier values from the received response and store for the future use (in the same call scenario) is called “Export” (This exported value can also be imported to message template in future).

- **A ‘Profile’ file**
  Once the Key Identifiers are identified for all the message templates in a call scenario, required values are configured for these Key Identifiers in a file called Profile.
Supported Protocols / Interfaces

**LTE (X2, S1, eGTP, SGs, SLs)**
- IMS, Diameter

**IMS**

**4G Access (E-RAN)**

**3G Access (UTRAN)**
- UMTS (IuH, IuCS, IuPS, GnGp)
- GPRS

**Signalling and Traffic Generation and Analysis**
- MAPS™ 1G – Software-based, or Hardware-based
- MAPS™ HD 40G, 10G, 1G – Rackmount Platforms

**SIP, Megaco, MGCP, SIP-I, SIGTRAN (ISUP, ISDN), SKINNY**

**VoIP (Packet Network)**

**SIP, RTR, H.323, T.38**

**2G Access (GERAN)**
- GSM (A, Abis)

**http://www.gl.com/maps.html**
Common Protocol Emulation Framework

LTE Simulation

SS7 Simulation

SIP Simulation
MAPS™ Working Principle

Message Automation and Protocol Simulation (MAPS)

- **Message Templates**
  - Create Call Sequence
  - Load Script
  - Assign values to variables
  - Load Profile
  - Load Event Profile

- **Call Control**
  - CALL GENERATION
  - (Load Scripts, Profiles)
  - CALL RECEPTION

- **IP/TDM Network**

- **DUT**
Customize Test Scenarios using Scripts

• Unlimited access in creating test scenarios.

• Build valid or invalid & conformance test cases.

• A simple, easy to learn but very powerful scripting language.

• Can be an Extremely simple scripts to test a particular scenario. But Flexible enough to simulate a complete protocol state machine.

• A GUI based 'Script Editor' helps to build scripts even before syntax and semantics of the scripting language is familiar.
Sample Script

Send "Initial Address" "InitialAddressImport";

Recv "Address Complete" "AddressCompleteExport";

Recv "Answer" "AnswerImport";

**TxRx:tx _TDM file: filename = "Vijay.pcm"**;

Send "Release" "ReleaseImport";

Recv "Release Complete" "ReleaseCompleteExport";
Customize Protocol Messages

Message Editor

[Diagram showing the Message Editor interface with various options and parameters for protocol messages]
Call Generation

Call Generation

Active Calls
Call Status
Call Events

Loading Scripts and Profiles

Message Sequence

Decode Message
MAPS™ Features

Fine Control over Call Behavior

Control moves to "Retrieve" section, after selecting the "Retrieve" User Event
User Events

Control moves to "Hold" section, after selecting the "Hold" User Event

ActiveUserEvent: Add: "Retrieve"; "Retrieve" User Event is added

Control moves to "Retrieve" section, after selecting the "Retrieve" User Event
Script Contents

"PlaceCall"(opc, dpc, cic):
    StartChildScript (ISUPScriptId, "ISUP", SLS, LoadedProfileName);
    ActiveUserEvent:Add: "Terminate Call", "Initiate Reset", "Clear Call";
    Status = "ISUP Call Initiated";
    ISUPState = "ISUP CALL INITIATED";
    (ISUPScriptId) goto "ISUPMakeCall": cic, opc, dpc, SLS, NetInd, ConnectionId, StreamId,
    return;

"OnISUPCallInitiated"(opc, dpc, cic):
    ReportEvent (InitialAddress = "Initial Address");
    resume;

"OnISUPCallProgressReceived":
    ReportEvent (AddressComplete = "Address Complete");
    resume;

"OnISUPCallConnected":
    Result = "Pass";
    ReportEvent (Answer = "Answer");
    Status = "ISUP Call Connected";
    ISUPState = "ISUP CALL CONNECTED";
    if (StopAll == 1) 
        goto "Terminate Call": Cause;
    endif
    ActiveUserEvent: Add: "Hold", "Suspend";
    ActiveUserEvent: Remove: "Accept Call", "Reject Call";
    if (CallDuration != 0)
        starttimer CallDurationTimer CallDuration msec;
    else 
        goto "Terminate Call": Cause;

<

17
### Script Flow

<table>
<thead>
<tr>
<th>Script ID</th>
<th>Script Name</th>
<th>Subscript Name</th>
<th>Line No</th>
<th>Script Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>63</td>
<td>goto &quot;PlaceCall&quot;: opc , dpc , cic ;</td>
</tr>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>94</td>
<td>&quot;PlaceCall&quot;: opc , dpc , cic ;</td>
</tr>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>95</td>
<td>StartChildScript ( ISUPScriptId , &quot;ISUP&quot; , &quot;ISUP.gls&quot; , LoadedProfileName ) ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>6</td>
<td>&quot;Init&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>7</td>
<td>ISUPState = &quot;IDLE&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>8</td>
<td>ISUPResult = &quot;Unknown&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>9</td>
<td>SetScriptVariable ( ParentScriptId , ISUPResult , ISUPResult ) ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>10</td>
<td>ParentScriptId = &quot;&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>11</td>
<td>Cause = 16 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>12</td>
<td>COTExpected = 0 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>13</td>
<td>AddressCompleteSent = 0 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>14</td>
<td>KeyIdentifier : opc , dpc , cic ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>15</td>
<td>ReleaseInitiated = 0 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>16</td>
<td>ReleaseReceived = 0 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>17</td>
<td>CallActive = 0 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>18</td>
<td>MagHandler : &quot;ISUPMagHandler&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>19</td>
<td>ReleaseGuardTimerStarted = 0 ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>21</td>
<td>wait ;</td>
</tr>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>95</td>
<td>ActiveUserEventsAdd ( &quot;Terminate Call&quot; , &quot;Initiate Reset&quot; , &quot;Clear Call&quot; ) ;</td>
</tr>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>97</td>
<td>Status = &quot;ISUP Call Initiated&quot; ;</td>
</tr>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>98</td>
<td>ISUPState = &quot;ISUP CALL INITIATED&quot; ;</td>
</tr>
<tr>
<td>*</td>
<td>Isup_Call.gls</td>
<td></td>
<td>99</td>
<td>( ISUPScriptId ) goto &quot;ISUPMakeCall&quot;: cic , opc , dpc , SLS , NetEnd , ConnectionId , StreamId , ...</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>32</td>
<td>&quot;ISUPMakeCall&quot; ( cic , opc , dpc , SLS , NetEnd , ConnectionId , StreamId , CallingNumber , CalledNR...</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>33</td>
<td>send ( &quot;InitialAddress&quot; , &quot;InitialAddressReply&quot; , &quot;StreamId&quot; = StreamId , &quot;ConnectionId&quot;...</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>34</td>
<td>if ( ContinuityCheckIndicator = 0 )</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>35</td>
<td>endif</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>37</td>
<td>ISUPStates = &quot;CALL INITIATED&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>38</td>
<td>Status = &quot;Call Initiated&quot; ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>39</td>
<td>EventLog ( &quot;Call Initiated&quot; ) ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>40</td>
<td>starttimer T7_T7Timeout ;</td>
</tr>
<tr>
<td>ISUP</td>
<td>ISUP.gls</td>
<td></td>
<td>41</td>
<td>( ParentScriptId ) goto &quot;UnISUPCallInitiated&quot;: opc , dpc , cic ;</td>
</tr>
</tbody>
</table>
Incoming Call Handler

MAPSTM Features

Incoming Call Handlers Configuration - default

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Script Name</th>
<th>Scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signalling Link Test Message</td>
<td>SLM.gl</td>
<td>Isup_Call.gl</td>
</tr>
<tr>
<td>Initial Address</td>
<td>Isup_Call.gl...</td>
<td></td>
</tr>
<tr>
<td>Release</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Reset Circuit</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Continuity Check Request</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Blocking</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Unblocking</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Circuit Group Reset</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Circuit Group Blocking</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Circuit Group Unblocking</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
<tr>
<td>Release Complete</td>
<td>Rx_CIC_Management.gl</td>
<td></td>
</tr>
</tbody>
</table>

Sequence

Add Delete Clear

Up Down

19
Incoming Call Handler

<table>
<thead>
<tr>
<th>IAM</th>
<th>AnswerCall.gls</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td></td>
</tr>
<tr>
<td>ANM</td>
<td></td>
</tr>
</tbody>
</table>

MAPS™ searching for appropriate script to load against the received msg

IAM CIC=1

Answer call Script is bind to the recv msg with CIC=1

IAM CIC=1

Another IAM msg is received with CIC=1

IAM CIC=2

Another IAM msg is received with CIC=

IAM CIC=2

Script Loaded

Answer Call Script ID 1 CIC=1

Answer Call Script ID 2 CIC=2

Recv msg is bind to the same script

Binds to the new script with ID=2

MAPS™

Courtesy of GL Communications
Call Reception...

### MAPS Features

**Active Calls**

**Completed Calls**

<table>
<thead>
<tr>
<th>Sr No</th>
<th>Script Name</th>
<th>Calls</th>
<th>Script Execution</th>
<th>Status</th>
<th>Events</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check_SCTP Status</td>
<td></td>
<td>Stop</td>
<td></td>
<td>None</td>
<td>Unknown</td>
</tr>
<tr>
<td>2</td>
<td>Message</td>
<td>1001</td>
<td>Stop</td>
<td></td>
<td>None</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>Transfer_Calls</td>
<td></td>
<td>Stop</td>
<td></td>
<td>None</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>IUCS Calls</td>
<td>IMSI (001) 0123456789 IMSI (0000010000)</td>
<td>Completed</td>
<td>SCTP Connection Released</td>
<td>None</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>IUCS Calls</td>
<td>IMSI (001) 0123456789 IMSI (0000010000)</td>
<td>Completed</td>
<td>SCTP Connection Released</td>
<td>None</td>
<td>Pass</td>
</tr>
</tbody>
</table>

**Message Sequence**

**Decoded Message Details**

![Screenshot of the MAPS software interface showing Active Calls and Completed Calls with detailed message sequences and decoded message details.](Image Link)
MAPS™ Features

Load Generation

- Stability/Stress and Performance testing using Load Generation
- Different types of Load patterns to distribute load
- User can load multiple patterns for selected script
- User configurable Test Duration, CPS, Maximum and Minimum Call Rate etc.
Success Call Ratio Statistics

Call Graph

Call Stats
MAPS™ Features

Message Statistics

- **Message Stats** provides a running tabular log of all messages transmitted, retransmitted and received during the session.
- Provides an easy way to monitor the reception of error responses during load generation.
User Defined Graphs and Statistics
Traffic Simulation

- Voice Files (wav, pcm, GL's Proprietary GLW)
- Video Trace Files (hdl - GL's Proprietary format)
- DTMF and MF Digits
- Tones (single, dual and user-defined)
- Industry Standard Codecs (a-law, u-law, gsm and more)
- FAX (T.30 RTP pass-through and T.38 UDPTL)
- Impairments (Latency, Packet Loss, Packet Effects)
- Mobile Data, Packet, GTP Gateway Traffic

(for details see http://www.gl.com/traffic-simulation.html)
Supported RTP Codecs

- **PCM/PCMA**: 64kbps, 8000Hz, VAD
- **G.722/G.722.1**: 24/32/64kbps, 16000Hz, No VAD
- **G.729/G.729B**: 8kbps, 8000Hz, VAD
- **GSM 6.10 FR**: 13.2kbps, 8000Hz, No VAD
- **GSM EFR**: 12.2kbps, 8000Hz Yes VAD
- **GSM**: 5.6kbps, 8000Hz, Yes VAD
- **G.726**: 16/24/32/40kbps, 8000Hz, Yes VAD
- **AMR**: 4.75/5.15/5.9/6.7/7.4/7.95/10.2/12.2kbps, 8000Hz, Yes VAD *(OPTIONAL LICENSE)*
- **AMR WB**: 4.75/5.15/5.9/6.7/7.4/7.95/10.2/12.2kbps, 16000Hz, Yes VAD *(OPTIONAL LICENSE)*
- **EVRC**: 1/8, 1/2, 1 rate, 8000Hz, No VAD *(OPTIONAL LICENSE)*
- **EVRC_B**: 1/8, 1/2, 1 rate, 8000Hz, Yes VAD *(OPTIONAL LICENSE)*
- **EVRC_C**: 1/8, 1/2, 1 rate, 16000Hz, Yes VAD *(OPTIONAL LICENSE)*
- **SMV**: Modes 0,1,2 and 3, 8000Hz, No VAD *(OPTIONAL LICENSE)*
- **ILBC**: 15.2/13.33kbps, 8000Hz, No VAD
- **SPEEX**: 8kbps, 8000Hz, Yes VAD
- **SPEEX WB**: 11.2kbps, 16000Hz, Yes VAD

*(see [http://www.gl.com/voice-codecs.html](http://www.gl.com/voice-codecs.html) for details)
TDM Traffic Simulation

TDM Network

MAPSTM
SS7

Voice Channels

PSTN

TDM Network

Tx
- Pre recorded PCM files
- DTMF, MF, MFR2B and MFR2F Digits
- User Defined Tones
- FAX
- AAL2

Rx
- PCM files
- DTMF, MF, MFR2B and MFR2F Digits
- User Defined Tones
- FAX
- AAL2
RTP Traffic Simulation

IP Network

MAPSTM

Tx
- Pre recorded GLW files
- DTMF, MF Digits
- User Defined Tones
- Insert Voice
- FAX T.30

Voice Channels

123
456

PSTN/IP Phones

Rx
- GLW files
- DTMF, MF Digits
- User Defined Tones
- FAX T.30
Single Interface Simulation
MAPS™ Features

Multiple Transport Support

OC-3/STM-1 OC-12/STM-4 Card
Quad/ Octal T1 E1 PCIe Card
Dual T1 E1 Express (PCIe) T1/E1 Card

tProbe™
T1/E1/VF/Datacom/ FXO/FXS Analyzer
USB Dual T1 E1 Analyzer
USB based Dual T3 E3 Analyzer Unit
IP variants of MAPS™ can be run on any modern Windows server.

A typical i7 platform will be able to handle ~2000 concurrent RTP sessions through a conventional server-grade NIC.

We also offer an HD (High Density) appliance which can deliver up to 20,000 concurrent RTP sessions per U of rack space.
High Density (HD) RTP Traffic Simulation

- Rackmount network appliance with 4x1GigE NIC.
- Transport over UDP and TCP, IPv4 and IPv6, and TLS for secure transport.
- Easily achieve up to 20,000 endpoints per appliance (5000 per port).
- Up to 350 calls per second (with RTP traffic).
- Scales to around 100,000 to 200,000 endpoints with use of Master Controller for single point of control.
- Manage 10+ MAPS™ systems with single point of control from Master Controller.
Introduction to MAPS™ Configurations

- Testbed Setup
- Global Configuration
- Profiles
Testbed Configuration
Sample Global Configuration
Local and Global Variables
MAPS™ Scripting
MAPS™ Scripts can be written in different ways as we have flexible commands such as Go to, IF Else IF, Timers, Actions, User Events etc.

Two Types of Scripting
- Simple, Non Event driven
- Event Driven

Non Event driven: Defines flow sequentially without monitoring any events. These can be small and simple scripts using send and receive actions

Event Driven: Defines flow on basis of user selected events. Using Event Driven scripting one can achieve Protocol State Machines as per protocol specifications
//Script Description ..................

//Initialization Section
---------
---------

//Action section
Send "MessageName" "ImportFile Name";
Recv "Message" "ExportFile Name";
Result = "Pass";
State = ".............";
Status = ".............";
Exit;
Script Events

**Message Handler:** On Receipt of any Message Event control move to defined section in script

**Traffic Event:** On detection of any traffic actions, Control move to detected Traffic Event Section like “Digits Detected”, Tone Detected”, etc.

**Timer:** On Expiry of Timer, control moves to respective Timer Expiry section

**User Events:**
- Within scripts: Goto “Label”
- User Intervention: User Event
- Intervention from another Script: Apply Event to another script
# Structure of Event Driven Script

<table>
<thead>
<tr>
<th>//Script Description ..........................</th>
</tr>
</thead>
<tbody>
<tr>
<td>//Initialization Section</td>
</tr>
<tr>
<td>Initialize Variables</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Initialize Message Handler</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Message Handler Section</td>
</tr>
<tr>
<td>&quot;Message 1&quot;:</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>&quot;Message 2&quot;:</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>&quot;Timer Expiry Section&quot;:</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>&quot;Traffic Handler Section&quot;:</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>&quot;User Events Section&quot;:</td>
</tr>
</tbody>
</table>
Below call flow scenario using MAPS Script.

Send “Initial Address” “InitialAddressImport”;  
Recv “Address Complete” “AddressCompleteExport”;  
Recv “Answer” “AnswerImport”;  
TxRx:tx _TDM file: filename = “Vijay.pcm”;  
Send “Release” “ReleaseImport”;  
Recv “Release Complete” “ReleaseCompleteExport”;
Sample Script
Understanding Send and Receive Messages
Basic Send Command

Decoded Message

Protocol Engine

Decoder

Import

Encoder

Message Sent without Impairment
Basic Receive/Decode Command

Decode Message

- List of Protocol Fields in a Message
  - Protocol
  - Message Type
- Mandatory Fixed Parameters
- Mandatory Var Parameters
  - Calling Party Number
    - Numbering Plan Indicator
    - Internal Network Number Indic
    - Called Address Signal= 5551234
  - Called Party Number
    - Numbering Plan Indicator
    - Internal Network Number Indic
    - Called Address Signal= 5551111
- Optional Var Parameters

Extracted Variables

Called Address Signal= 5551234
Called Address Signal= 5551111

Protocol Engine

DECODER

Incoming Message

54 68 47 66 54
OA 00 02 32 43
06 05 03 83 15
10 10 A0 68 10

EXPORT

Called Address Signal
Calling Address Signal
Scanning Messages....

```
"Init":
  MsgHandler : "ISUPMsgHandler";
  wait;

"ISUPMsgHandler":
  goto MessageType:"DefaultMsgHandler";
  resume;

"Initial Address":
  "Initial Address":
    decode "InitialAddressExport";

"Answer":
"Release":
"Release Complete":
"DefaultMsgHandler":
```

DATA BUFFER

```
CIC =
CalledNumber =
CallingNumber =

"Init":
MsgHandler : "ISUPMsgHandler";
wait;

"ISUPMsgHandler":
  goto MessageType:"DefaultMsgHandler";
  resume;

"Initial Address":
  "Initial Address":
    decode "InitialAddressExport";

"Answer":
"Release":
"Release Complete":
"DefaultMsgHandler":
```

Message Templates

```
0000 Service Indicator         = ....0011 SCCP
0001 DPC                           = 2.2.2(00010010 ..010000)
0002 OPC                           = 1.1.1(01...... 00000010 ....0010)

00000101 ....00 (5)
9440094400
```

```
CIC = 5
CalledNumber = 9880098800
CallingNumber = 9440094400
```

Send Action

Recv Action

```
CIC = 5
CalledNumber = 9880098800
CallingNumber = 9440094400
```

```
0005 Circuit Identification Code = 00000101 ....00 (5)
0007 Message Type = 00000001 Initial address
Mandatory Variable Parameters = .......... 000000010 ....0010

0012 Called Address Signal = 9880098800
Optional Variable Parameters = ..........
0014 Calling Party Number = optional parameter

0012 Calling Address Signal = 9440094400
001E End of optional parameters octet = x00
```

```
"Initial Address":
Initial Address:
AddressComplete:
Release:
Release Complete:
```

```
"Suspend":
Suspend
```

```
Init Address
```

```
Loadprofile(CalledNumber);
```

```
ISUPState = CALL RECEIVED;
```

```
resume;
```
Send Command With Impairment

Called Ackness Signal = 8078675400
Calling Address Signal = 9987095827

Message Type Field

In Message Editor “8th byte” with Offset ‘7’ = represents the Message Type Field.

Create MsgTypeImpair.txt file
IMPAIR REP 1 OFFS 7 AND 00

Message Sent with Impairment

DECODER

IMPORT

ENCODER

Message Template

0 5 1 2 5 0 0 2 3 2 0 1 0 0
0 1 0 2 2 0 0 1 4 0 0 0 2
0 9 0 6 8 3 1 0 5 1 5 1 1
0 1 0 0 6 6 9 1 1 1 5 1 5

Message Field

0 5 1 2 5 0 0 2 3 2 0 1 0 0
0 1 0 2 2 0 0 1 4 0 0 0 2
0 9 0 6 8 3 1 0 5 1 5 1 1
0 1 0 0 6 6 9 1 1 1 5 1 5

Protocol Engine

Called Ackness Signal = 8078675400
Calling Address Signal = 9987095827

Message Type Field

In Message Editor “8th byte” with Offset ‘7’ = represents the Message Type Field.

Create MsgTypeImpair.txt file
IMPAIR REP 1 OFFS 7 AND 00

Message Sent with Impairment

DECODER

IMPORT

ENCODER

Message Template

0 5 1 2 5 0 0 2 3 2 0 1 0 0
0 1 0 2 2 0 0 1 4 0 0 0 2
0 9 0 6 8 3 1 0 5 1 5 1 1
0 1 0 0 6 6 9 1 1 1 5 1 5

Protocol Engine

Called Ackness Signal = 8078675400
Calling Address Signal = 9987095827

Message Type Field

In Message Editor “8th byte” with Offset ‘7’ = represents the Message Type Field.

Create MsgTypeImpair.txt file
IMPAIR REP 1 OFFS 7 AND 00

Message Sent with Impairment

DECODER

IMPORT

ENCODER

Message Template

0 5 1 2 5 0 0 2 3 2 0 1 0 0
0 1 0 2 2 0 0 1 4 0 0 0 2
0 9 0 6 8 3 1 0 5 1 5 1 1
0 1 0 0 6 6 9 1 1 1 5 1 5

Protocol Engine

Called Ackness Signal = 8078675400
Calling Address Signal = 9987095827

Message Type Field

In Message Editor “8th byte” with Offset ‘7’ = represents the Message Type Field.

Create MsgTypeImpair.txt file
IMPAIR REP 1 OFFS 7 AND 00

Message Sent with Impairment
CLI/APIs for Remote Control and Test Automation
CLI Working Principle

**Client User** (TCL, Python, VBScript, Java, .Net)
- Start Script1
- User Events
- Wait for Response
- Stop Script1

**MAPS Client IFC**
- Storage Space for Script ID

**MAPS CLI Server**
- Command Processor
- Response Processor

**Server Command**
- Server Response
Sample TCL Script (Place_Call.tcl) for MAPS™ ISDN
Sample SIP Registration VB Script
Java Client
• Multi-node and multi-interface simulation from a single GUI
• Suitable for testing any core network, access network, and inter-operability functions
• Single Licensing Server controlling server and client licenses (no. of users)
• Unlimited number of remote client user can be defined at the server
• Admin privileges to control Testbed and access to configuration files for each remote client user
• Remote Client users has privileges to perform all other functions - call simulation, edit scripts/profiles, and view statistics
• Option to license multiple clients either at Remote client systems (MAPS™ Remote Client to control one or more MAPS™ Server - PKS111) or at the MAPS™ Server systems (MAPS™ Server with Multi-user capability - PKS113).
• Simultaneous traffic generation/reception at 100% on all servers
Send Reports to Database

- MAPS™ generated reports can be sent to Database using built in commands
- This helps to monitor and Analyze test Remotely