MAPS™ APIs for Complete Automation
API Overview

- API wraps our proprietary scripting language in standard languages familiar to the user:
  - Python
  - Java
  - VB Scripts
  - TCL

- Clients and Servers support a “Many-to-Many” relationship, making it very easy for users to develop **complex test cases involving multiple signaling protocols**.
Working Principle of MAPS™ CLI

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

MAPS Client IFC
- Storage Space for Script ID
- Command Processor
- Response Processor

MAPS CLI Server
- Server Command
- Server Response
The same Client Application used to control MAPS™ can be, and very often is, used to control other elements of the System Under Test.
System Integration

- Client Application can be as simple as executing a script from an IDE or it can be integrated into a full-fledged automation test suite like QualiSystems TestShell or HP UFT.
def answer_call(self):
    if uas_call.bind_incoming_call(call_vars[header],
        uas_call.place_call()
    for x in range(0, 10),
        uas_call.get_call_status()
    uas_call.get_call_status()
    if (uas_call.status == 'Connected') and
        break
    else:
        if uas_call.status == 'Progress':
            uas_call.answer_call()
        wait(t)

SIP_AcceptCall:
    if (_RtpCreaseSession == 0):
        EnableSsrc = 0;
        RtpCoreSsrcAlgorithm = "";
        RtpCoreSsrcKey = "";
    endif
    startTimer, CallAnswerTimer:
    //EventLog("RtpCoreSsrcKey", RtpCoreSsrcKey);
    //EventLog("RtpCoreSsrcAlgorithm", RtpCoreSsrcAlgorithm);
    if (_RtpCreateSession == 0):
        VideoMediaPort = 1014;
    endif
    (SipScriptId) goto "SipAcceptCall":RtpIpAddress,AudioMedia
    resume;
Each protocol comes with a pre-built set of functions for basic calling features, ie `place_call()`, `answer_call()`, `reject_call()`, `terminate_call()` etc can be found in (almost) all protocols.

Many protocols also have specialized functions unique to them ie `register()` and `deregister()` for SIP.
The API also delivers a library of Traffic functions of the generation of RTP or TDM traffic.

- Digit, Tone, Voice File, Video File and Fax File transmission and reception are all supported.

- Same default values supplied for all functions to make it easy for users who don’t require fine grained control.
We can extract complete message sequences from calls into objects in our API languages. These objects will hold:

- Message type (ie INVITE)
- Message direction (Tx / Rx)
- Message timestamp (w/ ms accuracy)
- Full message decode

Use this for custom pass/fail verification, message/response delay calculation, etc.

Messages can even be extracted in real time for custom parsing in the API language.
MAPS™ offers two integrated forms of Voice Quality Measurement: Packet Analysis and Waveform analysis.

Packet-based protocols which carry RTP traffic can be analyzed for MOS, loss, discard, sequence errors, duplication errors and jitter.

TDM and packet protocols can employ our VQT product to deliver PESQ and/or POLQA (essentially this involves the transmission of a known voice file through an SUT, and then a post-processed comparison of the degraded file to the pre-transmission reference file).
The API is broken into High and Low level function calls / scripts.

For High Level scripts, all the fine-grained protocol control happen in the script running on the MAPS™ server, hidden from the API user.

Low Level scripts put the API user in complete control of the protocol stack. This makes Low Level scripts more flexible and powerful, but also correspondingly more complex.
High Level Example: SipBasicCall.py

- Initialize variables, this is the only part of the script the user needs to modify to place a basic call.
- Those variables get passed to script before call is started.
- Call is generated with a single line, all fine-grained details of the protocol are hidden from the user. The same `place_call()` function works in all protocols supported by MAPS™.
- RTP is transmitted with a single line, arguments permitted but not required.
- Call is terminated with a single line, all fine-grained details of the protocol are hidden from the user. The same `terminate_call()` function works in all protocols supported by MAPS™.
Low Level Example: SipLowBasicCall.py

> The same set of variables exist in the Low Level Scripts and are passed with the same function
> Where the High Level user just issues `place_call()`, the low level user must:
  - `create_session()` to open an RTP socket
  - `send_message("Invite")` to start the call
  - Manually process responses
CAS/FXO/FXS API

• Channel Associated Signaling
  ➢ Method of signaling where a channel carrying speech also carries the signaling and addressing to set up and tear down calls
  ➢ Supervisory signaling carried as “onhook” and “offhook”, addressing signaling carried as DTMF or MF tones
• All functions are “low level”
  ➢ Signaling bits manipulation, call progress tone/signal detection, TDM traffic transmission/reception

```java
CasClient client = new CasClient("192.168.30.235", 10024);
System.out.println("Connecting to server...");
if (!client.connect())
    return;

CasCall line1 = client.openLine(1);
CasCall line2 = client.openLine(2);
if (line1.getCallHandle() == 0 || line2.getCallHandle() == 0)
    return;

line1.offhook();
line1.detectDialTone(20000);
line1.dial("102");
line2.detectRingingSignal(1, 20000);
line1.detectRingbackTone(20000);
line2.offhook();
Thread.sleep(3000);
line1.tdmSendTestTone(3000);
line2.detectTestTone(20000);
line1.onhook();
line2.onhook();
client.closeLine(line1);
client.closeLine(line2);
client.disconnect();
```
Regression from .csv

- Use the API language to easily access and read large regression configurations from local .csv files.
- Similarly, the API language can pull regression configurations from a database instead.
Multiplex Regression

- Use the advanced features of API languages to quickly and simply build complex regressions.

- This example shows a Python script that will iterate over every possible combination of values in the variable `regression_table`.

```python
regression_header = ['Contact', 'AddressOfRecord', 'To', 'CodecOption', 'PacketizationTime']
type_table = [('a',) 5]
regression_table = [['0001@192.168.30.213',
                     '0001@192.168.30.194', '1000@192.168.30.159', '2000@192.168.30.159',
                     'Profile0001', 'Profile0003', 'Profile0005', 'Profile0006',
                     '10', '20', '30']
]

uas = init_client(uas_ip, maps_port)
uac = init_client(uac_ip, maps_port)
if (uas.status == 'STARTED') and (uac.status == 'STARTED'):
    print "SERVERS INITIALIZED"
uac_call_list = []
uas_call_list = []

for args in itertools.product(*regression_table):
    print args
    uac_call, uas_call = two_way_call(uac, uas, regression_header, type_table, args)
uac_call_list.append(uac_call)
uas_call_list.append(uas_call)
print "Status:" + uac_call.status
print "CMOS:" + str(uac_call.rtp_stats_cmos)
print ""
Typical Test Systems
Test Setup for Gateway Testing

MAPSTM
SIP | MGCP | Megaco
RTP Core

CLI

NIC

IP

Gateway

FXS

FXO

Analog Hardware

PSTN
Connections

TDM

T1 card

MAPSTM
CAS | ISDN | SS7

CLI

IP - TDM
A Typical MAPS™ CAS Test System

- TCL user communicating over TCP/IP
- MAPS™ Client IFC, MAPS™ CAS CLI Server, T1 Software (including Windows® Client Server software) and a Dual T1 Card
- Analog Hardware with FXO Cards
- A patch panel for RJ-11 connections to the outside world
A Typical MAPS™ SIP Test System

- TCL user communicating over TCP/IP
- MAPS™ Client IFC, and MAPS™ SIP CLI Server

![Diagram of a typical MAPS™ SIP Test System]

- TCL User
- MAPS Client Interface
- MAPS IP CLI Server
- User Event
- Report Event
- SIP IP Network
- RTP
The MAPS API is also now fully integrated with GL’s VQT software which delivers PESQ/POLQA scores (i.e. waveform analysis, rather than packet analysis).
Thank You