SmartControl™

dsTest® External API Specification

Version 10

April 5, 2016

Version	Date	Author	Comments
10	Apr 5, 2016	Developing Solutions	10 connections supported
9	Mar 22, 2016	Developing Solutions	Added Error Message Contents
8	Feb 26. 2016	Developing Solutions	Minor Corrections
7	Feb 24, 2016	Developing Solutions	Corrected definition of Received Error value element; Added event generation example
6	Feb 19, 2016	Developing Solutions	Latest Enhancements
5	Jan 25, 2016	Developing Solutions	Updated for Event Notification Enhancements
4	Jan 1, 2016	Developing Solutions	Update Copyright, remove message types 2 and 4
3	Dec 12, 2015	Developing Solutions	Minor Cosmetic Corrections
2	Dec 24, 2014	Developing Solutions	Additional Features
1	Sep 26, 2014	Developing Solutions	Initial Draft

Table of Contents

1 Introduction

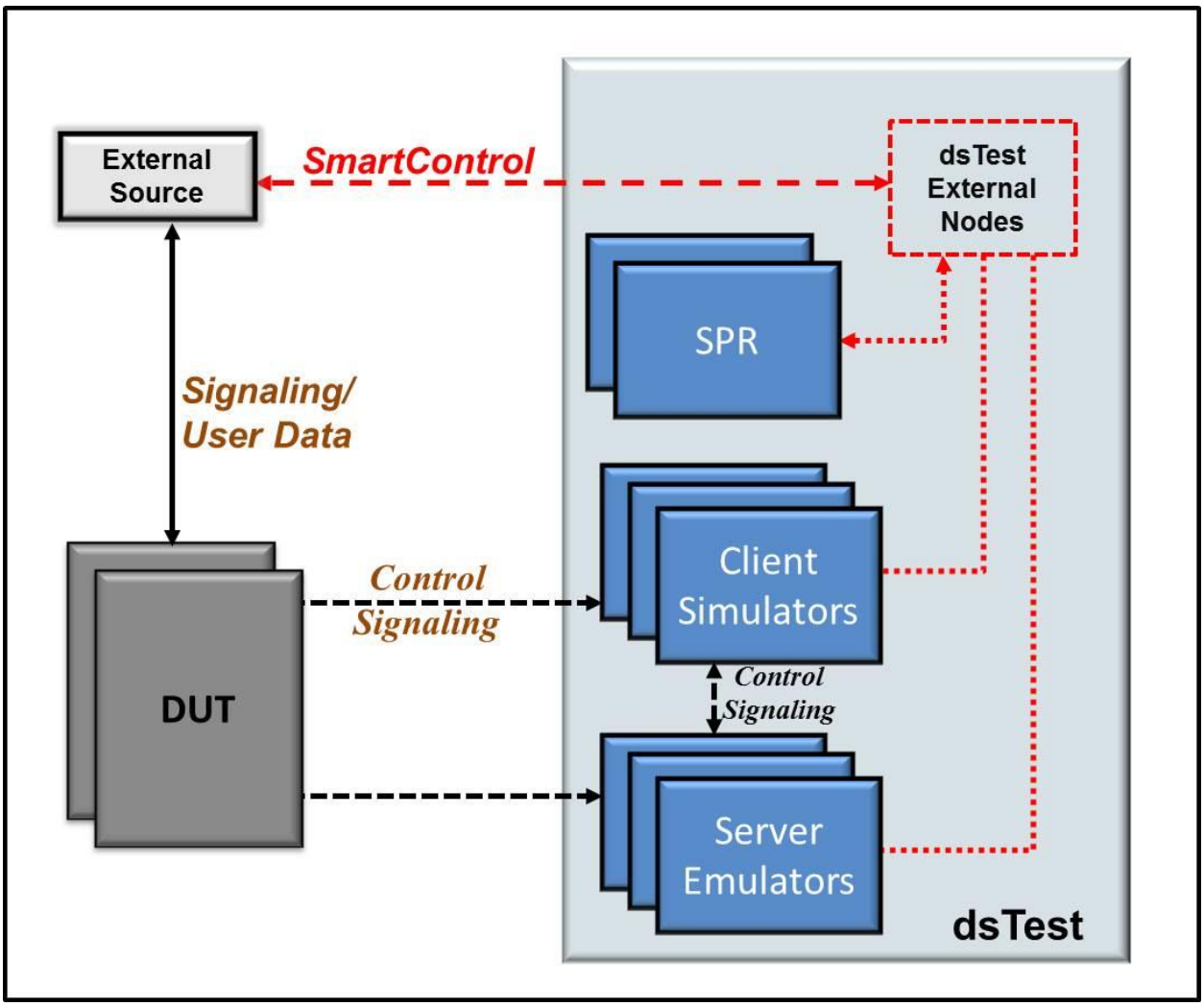

Figure 1 dsTest SmartControl API

dsTest's SmartControl API allows an External Client to register for, send, and receive notification of **dsTest** events, allowing the External Client to coordinate actions with device(s) under test (DUT). **dsTest** simulators and emulators, such as a Policy and Charging Enforcement Function (PCEF) or Online Charging Sever (OCS), can also interact with the DUTs as part of the **SmartControl** coordinated events with the External Client. Subscriber control can be maintained via the use of a **dsTest** Subscriber Profile Repository (SPR) node.

NOTE: Not all of the functionality described in this document has been fully implemented.

2 Architecture

2.1 Connection

dsTest maintains a TCP socket (server and passive) on port 10000. Upon successful registration by an external client, that client becomes associated with **dsTest** node with the name supplied by the client in the Sender Identification element contained in the registration message. Up to 10 **SmartControl** connections are supported by **dsTest**.

TCP is used as transport because:

- Under load, streaming will allow for joining of messages into fewer packets;
- No need for acknowledgment of ANY messages (there are result messages, but no acks);
- Underlying stack will deal with streaming and retransmission.

Within 5 seconds of opening a TCP connection the external client shall initiate a Registration Message which includes identification information (as well as potential future ranging information and/or capabilities).

dsTest will respond with a Registration Message with **dsTest** specific information.

At this point, connection is open for activity.

If any message is received on a connection prior to a Registration Message, or no Registration Message is received with 5 seconds, the connection will be closed by **dsTest**.

2.2 Structures

The message structure is intended for high performances construction and parsing. All structures are Type-Length-Value (TLV) format, 4-byte (32 bit) sized and aligned.

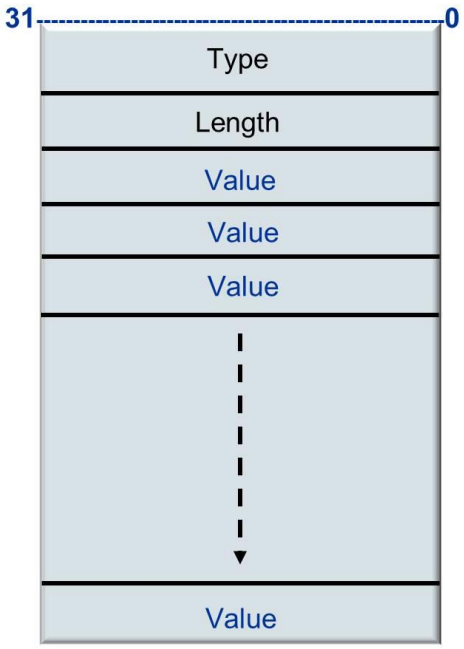

Figure 2 TLV Structure

Type: Message type or Value element type. The type field is globally unique across this interface with no context specifics within messages or value elements. If the Most Significant Bit (MSB) of the Type field is set, the values within the structure are in TLV format.

Types are:

1. Constructed - value contains TLV structures
2. UintX – if length is 4, value is a 4 byte network ordered unsigned value, else if length is 8, value is an 8 byte network ordered unsigned value;
3. String – Value is a sequence of bytes contained a printable string (not null terminated);
4. Octet String – Value is a sequence of bytes;
5. IpAddress – If length is 4, value is an IPv4 address represented as 4 byte network ordered value. If length is 16, value is an IPV6 address represented as 16 byte network ordered value.

Length: Length of value, not including size of Type and Length, and only actual value data, potentially not padded length to 4-byte boundary.

Value Elements: A string of octets, unless MSB of Type set, which designates TLV structure(s).

2.3 Message Structure

MSB set indicates that message contains values that are TLV structures.

2.3.1 Message Value Elements Structure

MSB set indicates the value is a TLV structure.

3 Message Types

All are 31-bit values, however most or all would have the MSB set due to their value being a series of TLV structures.

Message Type	Message Value
Registration	0x00000001
Error Indication	0x00000002
Event Trigger	0x00000003

4 Value Element Types

All are 31-bit values; however some may have the MSB set due to their value being a series of TLV structures.

Value Element	Type	Value
Sender Identification (registration message)	String	0x00001001
dsTest Info (internal use only, should be ignored)		0x00001002
Error Message		0x00001003
Event Registration	Constructed	0x80001004
Requested Event Content	UintX	0x80001005
Event Destination Node	String	0x00002001
Event Source Node	String	0x00002002
Event Application	String	0x00002003
Event Name (within scope of application)	string	0x00002004
Event ID (globally unique, internal use only)	UintX	0x00002005
Event Map Entry (must contain Event Application, Event Name, Event ID)	Constructed	0x80002006
Subscriber Index within Database	UintX	0x00003001
Subscriber IMSI	Octet String	0x00003002
Subscriber MSISDN	Octet String	0x00003003
Subscriber IMEI	Octet String	0x00003004
Subscriber Private Identify	String	0x00003005
Subscriber Public Identify	String	0x00003006
Subscriber IP Address	IpAddress	0x00003007
APN	String	0x00005001
Requested Data	String	0x00005002
Service Name/Indication	String	0x00005003
Destination Host	String	0x00005004
Destination Realm	String	0x00005005
Cancellation Type	UintX	0x00005006
Peer ISDN Number	Octet String	0x00005007
Notify Type	String	0x00005008
Termination Cause	UintX	0x00005009
Trigger Type	String	0x0000500a
Filter Name	String	0x0000500b
RAI Change Parameter	String	0x0000500c
Monitoring Key	String	0x0000500d
Message Name	String	0x0000500e
Accounting Session Name	String	0x0000500f
Session Release Cause	UintX	0x00005010

Bearer ID	UintX	0x00005011
Rule Name	String	0x00005012
Client IP Address	IpAddress	0x00006001
GTP Tunnel Source IP	IpAddress	0x00006002
GTP Tunnel Destination IP	IpAddress	0x00006003
GTP Tunnel Source TEID	UintX	0x00006004
GTP Tunnel Destination TEID	UintX	0x00006005
Packets Sent	UintX	0x00006006
Packets Rcvd	UintX	0x00006007
Payload Bytes Sent	UintX	0x00006008
Payload Bytes Rcvd	UintX	0x00006009
Total Bytes Sent	UintX	0x0000600a
Total Bytes Rcvd	UintX	0x0000600b
Received Error	UintX	0x0000600c

5 Message Descriptions

5.1 Registration Message

The Registration Message is used to indicate to the receiver that the sender wishes to receive all events of the type “Event Application-Event Name” that occur within the destination node's database.

This message may optionally contain one or more additional Requested Event Contents elements specifying additional elements to be in the resulting Event Trigger Messages.

Type: 0x00000001

Required Elements:

- Sender Identification
- Event Source Node
- Event Destination Node

Optional Elements:

- Event Registration
- Event Map Entry

5.2 Error Indication Message

This message is sent to the originator of an Event Trigger message if the message cannot be successfully delivered. The Error Indication is only sent as an indication that an Event Trigger message failed to be successfully processed. The Error message will contain some explanation of the issue, if known.

Type = 0x00000002

Required Elements:

- Error Message

Optional Elements:
Any of the following available when generating the error:

- Event Application
- Event Name
- Subscriber Index within Database
- Subscriber IMSI
- Subscriber MSISDN
- Subscriber IMEI
- Subscriber Private Identify
- Subscriber IP Address

5.3 Event Trigger Message

An event is triggered on a peer node by sending the Event Trigger message. This message must contain an indication of the event (App/Name or ID) and the identity of the subscriber. It should also contain any event specific data needed to specifically target, direct and/or qualify the event. There is no response to this message if successfully delivered. On failure, an Error Indication message will be sent.

Type Value: 0x00000003

Required Elements:

- Event Application
- Event Name
- One or more of the following:
    - Subscriber Index within Database
    - Subscriber IMSI
    - Subscriber MSISDN
    - Subscriber IMEI
    - Subscriber Private Identify
    - Subscriber IP Address

Optional Elements specific to message type: (tbd)

6 Constructed TLV Element Descriptions

Event Registration Element:

Type = 0x80001004;
Required Elements:

- Event Application
- Event Name
- Requested Event Contents (containing a 3xxx identifier for requested subscriber ID)

Optional Elements:

- Requested Event Contents (containing a 3xxx, 5xxx, or 6xxx identifier elements)

One or more of these elements may be included in the Registration Message to indicate to the receiver that the sender wishes to receive all events of this type “Event Application-Event Name” that occur within the destination node's database. It must also include at least one Requested Event Contents element containing the requested subscriber ID type to be included in the resulting Event Trigger Messages.

The message may optionally contain one or more additional Requested Event Contents elements specifying additional elements to be in the resulting Event Trigger Messages.

Event Map Entry: (future)
Type = 0x80002006;

Required Elements:

- Event Application
- Event Name
- Event ID

Optional Elements:

- none

One or more of these elements may be included in the Registration Message to provide a mapping of the sender’s string based event identifiers to uint32 based event identifiers. When these values are present in the registration message, the sender WILL send and the receiver MAY send the Event ID rather than the Event Application and Event Name in Event Trigger notifications.

7 SmartControl Example External Client App

The **dsTest** installation includes an example application that can be used to demonstrate the use of **SmartControl**.

The example app smartControl can be found on the dsTest server in /usr/local/devsol/bin.

Command line Options:

```
-h<addr>                hostname/ip address of server
-n<name>                node name of dsTest node
-a<name>                application name on dsTest
-v<name>                event name to send or receive notification on
-i                      subscriber index instead of IMSI
-r                      register for notification instead of sending event
-s<num>                 start subscriber id (N/A for notifications)
-e<num>                 end subscriber id (N/A for notifications)
-p<num>                 events per second to trigger (N/A for notifications)
-d                      enable verbose output
-x<num:value,...>       add custom attribute/values TLVs to event (N/A for
                        notifications)
```

The source for smartControl can be found at /usr/local/devsol/dsTest/lib. SmartControl must be executed with root privileges.

7.1 Registering for Notifications

An example use of smartControl app to register for notifications is as follows:

```
>/usr/local/devsol/bin/smartControl -h test44 -n hss -a s6 -v update -r -i -d
```

In this example, server test44 is running the S6 HSS and MME example configurations found on the **Developing Solutions** Online Help website.

The smartControl app registers for notifications of all (subscribers) S6 update events on server test44 with **dsTest** node “hss”, specifying the use of subscriber index as the subscriber id, with verbose output of the event notifications.

7.2 Sending Events to a dsTest Node

An example use of smartControl app to send an event to a dsTest Node is as follows:

```
>/usr/local/devsol/bin/smartControl -h test44 -n mme_1 –i –s 1 –e 2 –p 1 -a
s6 -v update –d
```

In this example, server test44 is running the S6 HSS and MME example configurations found on the **Developing Solutions** Online Help website.

The smartControl app sends an S6 update for subscribers 1 through 2 at a rate of one per second the MME node on server test44, specifying the use of subscriber index as the subscriber id, with verbose output of the responses.