

EPJS

API Design Document
of
Call Control SDK Service for
Windows & MAC

Table of Contents

Revision History	7
Introduction	8
Purpose of this document	8
Overview	8
Audience	8
Abbreviations	8
Architecture.....	9
Installation and Distribution.....	10
End-user Installation	10
SDK Agent.....	10
Startup	10
Installation from System Context (Windows Only)	10
Integrating with the SDK.....	12
SDK on Windows	12
Standard installation path.....	12
Files installed.....	12
PDS file	13
ABL config file.....	14
SDK on MacOS	14
Standard installation path.....	14
Files installed.....	14
PDS file	14
ABL config file.....	14
Interfacing with the SDK using TCPIP socket	15
Initial Connection	15
Connect to the socket	16
Connect to SDK	17
Connection is being closed.....	18
Closing the connection.....	19
Login and logout.....	20
Softphone login.....	20
Softphone logout	21
Incoming calls	22
Making an incoming call	22
.....	22

EPDS

.....	22
Accept and incoming call from softphone	23
Incoming call accepted from headset	24
Reject an incoming call from softphone	25
Incoming call has been rejected from headset	26
Active calls	27
Ending a call from softphone	27
Call has been ended from headset	28
Putting a call on hold from softphone	29
Call has been put on hold from headset	30
Resume a call from softphone	31
Call has been resumed from headset	32
Outgoing calls	33
Making an outgoing call	33
Outgoing call has been accepted by remote party	34
Outgoing call has been rejected by remote party	35
Outgoing call has been ended by caller	36
Conference calls	37
Start a conference call	37
End a conference call	38
Put a conference call on hold	39
Resume a conference call	40
Current headset	41
Headset connected	41
Headset disconnected	42
Other SDK functions	44
Redial last user from headset	44
Off-hook from headset	45
On-hook from headset	46
Send Presence status from softphone	47
Enable Logging (Deprecated)	49
Disable Logging (Deprecated)	51
Mute from softphone	52
Mute from headset	53
Unmute from softphone	54
Unmute from headset	55
Get softphone mute status	56

EPoS

Fail a call.....	57
System Information.....	58
Interfacing with the SDK using web socket.....	59
Initial Connection	59
Connection to the socket.....	60
Connection to SDK	61
Connection is being closed.....	62
Closing the connection.....	63
Login and logout.....	64
Softphone login.....	64
Softphone logout	65
Incoming calls.....	66
Making an incoming call	66
Accept and incoming call from softphone	67
Incoming call accepted from headset	68
Reject an incoming call from softphone	69
Incoming call has been rejected from headset	70
Active calls.....	71
Ending a call from softphone	71
Call has been ended from headset	72
Putting a call on hold from softphone	73
Call has been put on hold from headset.....	74
Resume a call from softphone	75
Call has been resumed from headset	76
Outgoing calls.....	77
Making an outgoing call	77
Outgoing call has been accepted by remote party	78
Outgoing call has been rejected by remote party	79
Outgoing call has been ended by caller	80
Conference calls	81
Start a conference call	81
End a conference call	82
Put a conference call on hold.....	83
Resume a conference call	84
Current headset.....	85
Headset Connected.....	85
Headset Disconnected	85

EPoS

Headset Available	86
Headset Removed	87
Active Device Change.....	88
Other SDK functions	89
Redial last user from headset	89
Off-hook from headset	90
On-hook from headset.....	91
Send Presence status from softphone	92
Enable Logging (Deprecated)	93
Disable Logging (Deprecated)	96
Mute from softphone	97
Mute from headset	98
Unmute from softphone	99
Unmute from headset.....	100
Get softphone mute status	101
Fail a call.....	102
System Information.....	103
Interfacing with the SDK using REST API.....	104
Initial Connection	105
Connection is being closed.....	105
.....	105
Login and logout.....	106
Softphone login.....	106
Softphone logout	107
Incoming call	108
Making an incoming call	108
Accept and incoming call from softphone	109
Incoming call accepted from headset	110
Reject an incoming call from softphone	111
Incoming call has been rejected from headset	112
Active calls.....	113
Ending a call from softphone	113
Call has been ended from headset	114
Putting a call on hold from softphone	115
Call has been put on hold from headset.....	116
Resume a call from softphone	117
Call has been resumed from headset	118

EPoS

Outgoing calls	119
Making an outgoing call	119
Outgoing call has been accepted by remote party	120
Outgoing call has been rejected by remote party	121
Outgoing call has been ended by caller	122
Conference calls	123
Start a conference call	123
End a conference call	124
Put a conference call on hold.....	125
Resume a conference call	126
Current headset.....	127
Headset connected	127
Headset disconnected.....	128
Other SDK functions	Error! Bookmark not defined.
Redial last user from headset	130
Off-hook from headset	131
On-hook from headset.....	132
Send Presence status from softphone	133
Enable Logging (Deprecated)	134
Disable Logging (Deprecated)	136
Mute from softphone	137
Mute from headset.....	138
Unmute from softphone	139
Unmute from headset.....	140
Get softphone mute status	141
Fail a call.....	142
System Information.....	143
HTTP status Codes	144
Application specific error codes	145

EPDS

Revision History

Date	Version	Change
December 12 2020	0.1	First Draft
January 25 2021	0.5	Pre release to partners
July 20 2021	0.6	Release to partners
January 08 2022	0.7	Marked logging request as deprecated
April 6, 2022	0.8	Updating section on Installation and Distribution
October 31, 2022	0.9	Added WebSocket V2 protocol APIs
November 18, 2022	1.0	Busy light Custom presence, Presenting event and Set active device changed
April 28, 2026	1.1	Update section SDK Agent and Startup

EPOS

Introduction

Purpose of this document

This document provides a high-level description of the design and implementation of the SDK and aim to provide software developers the information needed to integrate the features of EPOS headsets into softphones.

Overview

The SDK is a software interface between the softphone and headsets. This interface will aid the softphone users to control various call events on the softphone like accepting the incoming call, ending the active call and many more directly from the headsets.

Audience

This document has been written for software developers working for softphone vendors who intend to enable call control for headsets in their softphones.

The reader of this document is expected to know the basic concepts of Softphone events, XML, JSON, TCP, HTTP, HTTPS etc.

Abbreviations

Abbreviation	Description
XML	Extensible Markup Language
HS	Headset
SF/SP	Softphone
USB	Universal Serial Bus
SDK	Software Development Kit
TWC	Three way calling supported headset
WS	WebSocket
WSS	WebSocket Secure

EPoS

Architecture

The SDK is a set of executables and libraries. To enable call control integration between the headset and the softphone, the SDK Service needs to be executed as a background process on the local PC. On execution, it will make several different types of channels available to the softphone.

The SDK offers simple TCP/IP sockets, where both the SDK and the softphone can send requests to the other part and expect to receive acknowledgement back. As an example, the softphone can send a request to the SDK saying that it has an incoming call and wants the headset to start ringing. Once the SDK reads the request it will attempt to make this happen and send back acknowledgement to the softphone containing information on whether it succeeded or not. It is basic asynchronous socket communication in where data is encoded in XML format.

The SDK also offers to establish both standard and secure web sockets that works in much the same way as the normal TCP/IP sockets. If the softphone is web based it can be a good choice to use web sockets. Web sockets can also be used from a desktop application.

Finally, the SDK offers a standard RESTful interface. This differs a bit from the sockets as whether the request succeeded or not is known upon reply, which makes the interface a bit simpler. However, the SDK cannot send events to the softphone directly, so instead it buffers up these events and rely on the softphone to request these events on a regular basis. To keep the interface of the softphone up to date, in sync with the headset, these requests to read events must be very frequent, perhaps once per sec.

EPOS

Installation and Distribution

The SDK Service is a process that runs in the background on either Windows or MacOS. It will expose several interfaces as described above that a softphone can use for easy integration of call control with EPOS devices. The SDK Service must be installed on the end-user's machine.

End-user Installation

The installation of the SDK Service to the end-user can be done in 2 ways:

- The SDK installer can be integrated into the softphone installer and executed as a silent installation. Alternatively, the softphone can run the SDK installation the first time an EPOS device is detected.
- The SDK Service is included in EPOS Connect installation. The end-user can download and install EPOS Connect directly from the EPOS webpage. This will install the SDK Service as well.

Please note that the SDK installer is shared under NDA and may not be distributed directly to end-users or customers.

SDK Agent

The installation provides two executables: an SDK Agent (application name: DSEAAgent) and an SDK Service (application name: DSEASDK). The SDK Agent is responsible for launching the SDK Service and continuously monitoring it to keep it healthy and active. For optimal performance, start only the SDK Agent (do not start the SDK Service manually).

Startup

The SDK Agent is configured to start automatically at user sign-in through the system startup items. However, because startup items can be disabled by the user, the softphone must also attempt to start the Agent at sign-in to ensure availability. The Agent supports only one running instance; any subsequent launch attempt detects the existing instance and terminates immediately.

The recommended method to launch the SDK Agent is to launch as an independent process. The SDK Agent should not launch as child process of the softphone. On windows, softphone can use the `CREATE_BREAKAWAY_FROM_JOB` flag to create the job object before launching the SDK Agent.

Installation from System Context (Windows Only)

By default, auto-launch is enabled after installation, however, auto-launch can only be executed on user profile context and will throw an error when attempted from system context.

If installing the SDK via system context, e.g., if an IT Admin distribute it to multiple users, auto-launch must be disabled. This can be done by adding custom parameter 'AUTOLAUNCH_ENABLED=0' when executing dseasdk.msi.

Disable auto-launch: Install DSEASDK.msi with AUTOLAUNCH_ENABLED=0 to disable auto-launch of DSEAAgent.exe after installation (DSEAAgent.exe will launch DSEASDK.exe)

- Command: `msiexec /i DSEASDK.msi AUTOLAUNCH_ENABLED=0`

Enable auto-launch (default behavior): Install DSEASDK.msi with AUTOLAUNCH_ENABLED=1 to enable auto-launch of DSEAAgent.exe after installation (DSEAAgent.exe will launch DSEASDK.exe)

- Command: `msiexec /i DSEASDK.msi AUTOLAUNCH_ENABLED=1`

Above command can be placed in installation script.

EPOS

Additionally, if EPOS Connect is installed from System account, the EPOS Connect MSI must be installed first, followed by the DSEASDK MSI.

EPOS

Integrating with the SDK

SDK on Windows

Standard installation path

%programfiles(x86)%\DSEA\DSEASDK

Which in most case is like

C:\Program Files (x86)\DSEA\DSEASDK

Files installed.

Filename	Description
DSEASDK.exe	SDK executable file
DSEAAgent.exe	DSEA SDK Agent. Responsible for launching the DSEASDK.exe
ABLHandler.dll	Library to manage communication with ABL devices
AppManager.dll	Library to manage all events from SP to HS and vice-versa
AvayaEquinoxAdapter.dll	Library to manage communication with Avaya Equinox SP
BroadSoftAdapter.dll	Library to manage communication with BroadSoft SP
CommonInterface.dll	Library to manage common utility functions used by Application
HSCommunicationAPI.dll	Library to manage Headset Interface
libcrypto-1_1.dll	
libeay32.dll	
libssl-1_1.dll	
LogUtility.dll	Library to manage logs creation activity
MediaPlayerManager.dll	Library to manage Media Player events
PocoCrypto.dll	
PocoFoundation.dll	
PocoJSON.dll	
PocoNet.dll	
PocoNetSSL.dll	
PocoUtil.dll	
PocoXML.dll	
SDKSPAdapter.dll	Library to manage SDK SP Adapter to communicate with Legacy SDK
SFCommunicationAPI.dll	

EPOS

ssleay32.dll	
USBInterfaceAPI.dll	Library to send receive Headset events
VSFAdapter.dll	Library to manage communication with socket-based SP
WBCCListener.dll	Library to manage communication with WebSocket and REST based SP
WBSFAdapter.dll	Library to manage communication between SDK core and WBCCListener.dll
CertUtil/certutil.exe	
CertUtil/freebl3.dll	
CertUtil/gtest132.dll	
CertUtil/nspr4.dll	
CertUtil/nss3.dll	
CertUtil/nssckbi.dll	
CertUtil/nssdbm3.dll	
CertUtil/nssutil3.dll	
CertUtil/plc4.dll	
CertUtil/plds4.dll	
CertUtil/smime3.dll	
CertUtil/softokn3.dll	
CertUtil/sqlite3.dll	
CertUtil/ssl3.dll	
Media Interface/ RemotingWithWMP.exe	

PDS file

The PDS file is a text file containing important settings for the SDK. When the SDK executes for the first time, it is automatically created at the following location.

%programdata%\DSEASDK\SDKCore\config.dat
Which in most case is like
C:\ProgramData\DSEASDK\SDKCore\config.dat

This file is especially important for softphones using standard TCPIP sockets to communicate to the SDK, as the PDS file will hold an entry for the current port used for this. It is important, to only read this setting after the SDK executable is up and running. The SDK executable will attempt to use same port as last, but if this port is not available, it will use another. Once it has a port, it will update the PDS file to reflect this.

EPOS

ABL config file

The ABLConfig.json file is a json file containing settings for the ABL device. When the SDK executes for the first time, it is automatically created at the following location.

%programdata%\DSEASDK\SDKCore\ABLConfig.json
Which in most case is like
C:\ProgramData\DSEASDK\SDKCore\ABLConfig.json

SDK on MacOS

Standard installation path

/Library/Application Support/DSEA/DSEASDK

Files installed.

Filename	Description
LaunchSDK	This file is used by socket based softphone(plugin) to launch SDK
DSEASDK	SDK application file
Openssl.cnf	To generate SSL certificate
uninstall.sh	Script use to remove all install files
SoftphoneInfo.dat	Contains List of softphone information

PDS file

The PDS file is a text file containing important settings for the SDK. When the SDK executes for the first time, it is automatically created at the following location.

/Library/Application Support/DSEA/DSEASDK/config.dat

This file is especially important for softphones using standard TCPIP sockets to communicate to the SDK, as the PDS file will hold an entry for the current port used for this. It is important, to only read this setting after the SDK executable is up and running. The SDK executable will attempt to use same port as last, but if this port is not available, it will use another. Once it has a port, it will update the PDS file to reflect this.

ABL config file

The ABLConfig.json file is a json file containing settings for the ABL device. When the SDK executes for the first time, it is automatically created at the following location.

/Library/Application Support/DSEA/DSEASDK/ABLConfig.json

EPOS

Interfacing with the SDK using TCPIP socket

Initial Connection

It is quite straight forward to use the TCPIP sockets. Run the SDK executable and read the port number used by the SDK in the PDS file. The relevant entry looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SeComSFSDK>
  <PortNumber>8001</PortNumber>
  <LogToFile>Yes</LogToFile>
  <SupportedVID Count="2">
    <VID>0x1395</VID>
    <VID>0x1A17</VID>
  </SupportedVID>
  .....
</SeComSFSDK>
```

With the port number available, a socket can be created like this in Qt:

```
m_pSocket = new QTcpSocket();
if (m_pSocket){
  connect(m_pSocket, &QTcpSocket::readyRead,this, &tcpSDK::readyRead);
  m_pSocket->connectToHost("127.0.0.1", m_port);
  return true;
}
else
  return false;
```

Where m_port is the port read from the PDS file. Make sure never to assume this port is always the same. Once the socket is connected and starts listening to the port, the event EvtSocketConnected is received shortly. See next section for details.

The following pages will describe important events related to the SDK. It should be enough to implement call control from the softphone to SDK and headset.

EPoS

Connect to the socket

When the softphone initiates the connection to the SDK port, it will receive the following event that will indicate that the socket has connected:

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtSocketConnected>
<ReturnCode>0</ReturnCode>
</EvtSocketConnected>
```

Event parameters

ReturnCode: 0 on pass or the error code.

EPoS

Connect to SDK

Sending this request to the SDK will just register the application as a potential user. The user does not have to be logged in the softphone.

The client socket should send the following request to establish connection.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqEstablishConnection>
<ReqID>1</ReqID>
<SFName>Vendor Softphone 1</SFName>
<SFBrandName>Softphone Brand Name</SFBrandName>
<SFVersion>Softphone Version</SFVersion>
<SFIconImage>SFImage.ico</SFIconImage>
<RedialSupport>Yes</RedialSupport>
<OffHookSupport>Yes</OffHookSupport>
<MuteSupport>Yes</MuteSupport>
</ReqEstablishConnection>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFName: Name of the softphone

SFBrandName: Brandname of the softphone

SFVersion: Version of the softphone

SFIconImage: Fully qualified path of the icon image file of the softphone. The file must adhere to the following conditions:

The file format is '.ico'. Files in any other format or renamed as '.ico' may not give the same effect.

The size of the image is 16*16 pixels.

RedialSupport: If the softphone provides support for placing an outgoing call to the last called user when the event is received from the headset, set the value as "Yes" else use "No". By default, SDK assumes the value to be "Yes".

OffHookSupport: Off hook event is meant to provide a dial tone on the headset. If the softphone can provide this functionality set the value as "Yes" else set it as "No". By default, SDK assumes the value to be "Yes".

MuteSupport: If the softphone provides support for setting mute/unmute from headset and sends requests for setting mute/unmute to SDK, set this value to "Yes". By default, SDK assumes the value to be "Yes".

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckEstablishConnection>
<SFID>2</SFID>
<ReqID>1</ReqID>
<ReturnCode>0</ReturnCode>
</AckEstablishConnection>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge.

SFID: The unique id for the softphone. This id should be used in all subsequent requests.

ReturnCode: 0 on pass or the error code.

EPoS

Connection is being closed

If the SDK is being shut down, the SDK will send the following message:

Event

```
<?xml version="1.0" encoding="utf-8"?>  
<EvtListenerClosed>  
<Reason>Reason</Reason>  
</EvtListenerClosed>
```

Event parameters

Reason: The reason for the listener being shut down.

EPoS

Closing the connection

The client socket should send the following request to terminate the connection.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqTerminateConnection>
<ReqID>2</ReqID>
<SFID>2</SFID>
</ReqTerminateConnection>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckTerminateConnection>
<ReqID>2</ReqID>
<SFID>2</SFID>
<ReturnCode>0</ReturnCode>
</AckTerminateConnection>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

ReturnCode: 0 on pass or the error code.

After the SDK receives this request it will assume that the softphone will not be sending any more messages. All messages sent after this message will be failed.

EPoS

Login and logout

Softphone login

When the user logs in to the SF, the following request message must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqSFLoggedIn>
<SFID>2</SFID>
<ReqID>3</ReqID>
</ReqSFLoggedIn>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckSFLoggedIn>
<SFID>2</SFID>
<ReqID>3</ReqID>
<ReturnCode>0</ReturnCode>
</AckSFLoggedIn>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

ReturnCode: 0 on pass or the error code.

EPDS

Softphone logout

When the user logs out from the SF, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqSFLoggedOut>
  <SFID>2</SFID>
  <ReqID>4</ReqID>
</ReqSFLoggedOut>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckSFLoggedOut>
  <SFID>2</SFID>
  <ReqID>4</ReqID>
  <ReturnCode>0</ReturnCode>
</AckSFLoggedOut>
```

Acknowledgement parameters:

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ReturnCode: 0 on pass or the error code.

EPDS

Incoming calls

Making an incoming call

When an incoming call is received, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqIncomingCall>
  <SFID>2</SFID>
  <ReqID>5</ReqID>
  <CallID>1A2Z</CallID>
  <CallerName>User2</CallerName>
</ReqIncomingCall>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

CallID: The unique id for the call is provided by the softphone. Softphone must have an intelligence to generate call id uniquely for all calls present on the softphone. SDK will use this ID for further communication.

CallerName: The name of the caller.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckIncomingCall>
  <SFID>2</SFID>
  <ReqID>5</ReqID>
  <CallID>1A2Z</CallID>
  <ReturnCode>0</ReturnCode>
</AckIncomingCall>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPDS

Accept and incoming call from softphone

If the incoming call is accepted from the softphone, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqInCallAccepted>
  <SFID>2</SFID>
  <ReqID>6</ReqID>
  <CallID>1</CallID>
</ReqInCallAccepted>
```

Request Parameters

ReqID: The id of the request. This must be different for each request.

SFID: The id for the softphone assigned by the SDK.

CallID: The ID of the call to accept.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckInCallAccepted>
  <SFID>2</SFID>
  <ReqID>6</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckInCallAccepted>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Incoming call accepted from headset

If the incoming call is accepted by using the headset, the SDK will send the following event information to the softphone

```
<?xml version="1.0" encoding="utf-8"?>
<EvtInCallAccepted>
  <SFID>2</SFID>
  <CallID>1</CallID>
  <HeadsetPath>PID_0043</HeadsetPath>
</EvtInCallAccepted>
```

Parameters:

SFID: The unique id for the softphone assigned by the SDK.

CallID: The ID of the call accepted.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

EPoS

Reject an incoming call from softphone

If the incoming call is rejected from the softphone UI, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqInCallRejected>
  <SFID>2</SFID>
  <ReqID>10</ReqID>
  <CallID>1</CallID>
</ReqInCallRejected>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
CallID: The ID of the call to be rejected.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckInCallRejected>
  <SFID>2</SFID>
  <ReqID>10</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckInCallRejected>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPoS

Incoming call has been rejected from headset

If the incoming call is rejected by using the headset, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtInCallRejected>
  <SFID>2</SFID>
  <CallID>1</CallID>
  <HeadsetPath>PID_0043</HeadsetPath>
</EvtInCallRejected>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

CallID: The id for the call rejected.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

EPoS

Active calls

Ending a call from softphone

If the active call is ended from the softphone UI, the following request must be sent to the SDK and the acknowledgement will follow.

```
<?xml version="1.0" encoding="utf-8"?>
<ReqCallEnded>
<SFID>2</SFID>
<ReqID>8</ReqID>
<CallID>1</CallID>
</ReqCallEnded>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

CallID: The ID of the call to ended.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckCallEnded>
<SFID>2</SFID>
<ReqID>8</ReqID>
<CallID>1</CallID>
<ReturnCode>0</ReturnCode>
</AckCallEnded>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Call has been ended from headset

If the active call is ended by using the headset, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtCallEnded>
<SFID>2</SFID>
<CallID>1</CallID>
<HeadsetPath>PID_0043</HeadsetPath>
</EvtCallEnded>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

CallID: The ID of the call ended.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

Note: If the ended call was a conference call, the CallID will contain the ConfID.

EPDS

Putting a call on hold from softphone

If the active call is put on hold from the softphone UI, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqCallHeld>
  <SFID>2</SFID>
  <ReqID>12</ReqID>
  <CallID>1</CallID>
</ReqCallHeld>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
CallID: The ID of the call to put on hold.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckCallHeld>
  <SFID>2</SFID>
  <ReqID>12</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckCallHeld>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPoS

Call has been put on hold from headset

If the active call is put on hold from the Headset, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtCallOnHold>
<SFID>2</SFID>
<CallID>1</CallID>
<HeadsetPath>PID_0043</HeadsetPath>
</ EvtCallOnHold >
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

CallID: The ID of the call put on hold

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

EPDS

Resume a call from softphone

When the call that is put on hold is resumed from the softphone UI, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqHeldCallResumed>
  <SFID>2</SFID>
  <ReqID>13</ReqID>
  <CallID>1</CallID>
</ReqHeldCallResumed>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
CallID: The ID of the call to resume.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckHeldCallResumed>
  <SFID>2</SFID>
  <ReqID>13</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckHeldCallResumed>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPoS

Call has been resumed from headset

When the call that is put on hold is resumed from the headset, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtHeldCallResumed>
  <SFID>2</SFID>
  <CallID>1</CallID>
  <HeadsetPath>PID_0043</HeadsetPath>
</EvtHeldCallResumed>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

CallID: The ID of the call resumed

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

Note: if the call resumed was a conference call, the CallID will hold the ConfID.

EPDS

Outgoing calls

Making an outgoing call

If an outgoing call is made from the softphone UI the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqOutgoingCall>
  <SFID>2</SFID>
  <ReqID>14</ReqID>
  <CallID>1</CallID>
  <CallerName>Caller1</CallerName>
</ReqOutgoingCall>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

CallID: The unique id for the call is provided by the softphone. Softphone must have an intelligence to generate call id uniquely for all calls present on the softphone. SDK will use this ID for further communication.

CallerName: The name of the user to whom the call has been placed.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckOutgoingCall>
  <SFID>2</SFID>
  <ReqID>14</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckOutgoingCall>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPDS

Outgoing call has been accepted by remote party

If the outgoing call is accepted by the remote party, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqOutCallAccepted>
  <SFID>2</SFID>
  <ReqID>15</ReqID>
  <CallID>1</CallID>
</ReqOutCallAccepted>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
CallID: The ID of the call to accept.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckOutCallAccepted>
  <SFID>2</SFID>
  <ReqID>15</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckOutCallAccepted>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPDS

Outgoing call has been rejected by remote party

If the remote party rejects the outgoing call, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqOutCallRejected>
  <SFID>2</SFID>
  <ReqID>16</ReqID>
  <CallID>1</CallID>
</ReqOutCallRejected>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
CallID: The ID of the call to reject.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckOutCallRejected>
  <SFID>2</SFID>
  <ReqID>16</ReqID>
  <CallID>1</CallID>
  <ReturnCode>0</ReturnCode>
</AckOutCallRejected>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPoS

Outgoing call has been ended by caller

If the caller ends the call before remote party accepts it, the following request must be sent to the SDK and the acknowledgement will follow.

Request

```
<?xml version="1.0"?>
<ReqCallEnded>
<SFID>0</SFID>
<CallID>1</CallID>
<ReqID>3</ReqID>
</ReqCallEnded>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
CallID: The ID of the call ended by the caller.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8" ?>
<AckCallEnded>
<ReqID>3</ReqID>
<SFID>0</SFID>
<CallID>1</CallID>
<ReturnCode>0</ReturnCode>
</AckCallEnded>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPoS

Conference calls

Start a conference call

If the user starts a conference call, the following request must be sent:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqStartConfCall>
  <SFID>2</SFID>
  <ReqID>17</ReqID>
  <ConfiD>1</ConfiD>
</ReqStartConfCall>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
ConfiD: The unique id for the conference call. SF must ensure the uniqueness.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckStartConfCall>
  <SFID>2</SFID>
  <ReqID>17</ReqID>
  <ConfiD>1</ConfiD>
  <ReturnCode>0</ReturnCode>
</AckStartConfCall>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ConfiD: The unique id for the conference call provided by the softphone.
ReturnCode: 0 on pass or the error code.

Irrespective of the number of participants in the conference, the SDK treats the conference as a single call. If participants are added to the conference after the conference has been started, do not send any additional messages to the SDK.

To start a conference, you need to end all the calls you wish to combine into a conference and then start the conference.

EPoS

End a conference call

When the conference call is ended, the following request must be sent:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqEndConfCall>
  <SFID>2</SFID>
  <ReqID>18</ReqID>
  <ConfID>1</ConfID>
</ReqEndConfCall>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
ConfID: The ID of the conference to end.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckEndConfCall>
  <SFID>2</SFID>
  <ReqID>18</ReqID>
  <ConfID>1</ConfID>
  <ReturnCode>0</ReturnCode>
</AckEndConfCall>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ConfID: The unique id for the conference call provided by the softphone.
ReturnCode: 0 on pass or the error code.

Assume there are 3 callers in a conference viz. Caller1, Caller2, and Caller3, where Caller1 is the host (initiator) of the conference. If Caller3 ends the call from the remote end, do not end the conference.

This is because irrespective of the number of participants in the conference, the conference is treated as a single call. Only when the last participant in the conference ends the call you should end the conference.

EPoS

Put a conference call on hold

If the conference call is put on hold, the following request must be sent:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqConfCallOnHold>
  <SFID>2</SFID>
  <ReqID>19</ReqID>
  <ConfID>1</ConfID>
</ReqConfCallOnHold>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
ConfID: The ID of the conference to put on hold.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckConfCallOnHold>
  <SFID>2</SFID>
  <ReqID>19</ReqID>
  <ConfID>1</ConfID>
  <ReturnCode>0</ReturnCode>
</AckConfCallOnHold>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ConfID: The unique id for the conference call provided by the softphone.
ReturnCode: 0 on pass or the error code.

EPoS

Resume a conference call

If the conference call that was put on is resumed from the softphone UI, the following request must be sent:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqHeldConfCallResumed>
  <SFID>2</SFID>
  <ReqID>20</ReqID>
  <ConfID>1</ConfID>
</ReqHeldConfCallResumed>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
ConfID: The ID of the conference to resume.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckHeldConfCallResumed>
  <SFID>2</SFID>
  <ConfID>1</ConfID>
  <ReqID>20</ReqID>
  <ReturnCode>0</ReturnCode>
</AckHeldConfCallResumed>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ConfID: The unique id for the conference call provided by the softphone.
ReturnCode: 0 on pass or the error code.

EPoS

Current headset

Headset connected

If the user connects the Headset the following event will be sent by SDK

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtHeadsetConnected>
  <HeadsetName>Name of the Headset</HeadsetName>
  <HeadsetType>Type of the Headset</HeadsetType>
  <HeadsetPath>PID_740</HeadsetPath>
</EvtHeadsetConnected>
```

Event parameters

HeadsetName: Name of the connected headset
HeadsetType: Type of connected headset
HeadsetPath: Full device path of the connected device

EPoS

Headset disconnected

If the user disconnects the Headset the following event will be sent by SDK

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtHeadsetDisconnected>
<HeadsetName>Name of headset</HeadsetName>
<HeadsetType>Type of headset</HeadsetType>
<HeadsetPathPID_740</HeadsetPath>
</ EvtHeadsetDisconnected >
```

Event parameters

HeadsetName: Name of the disconnected headset
HeadsetType: Type of the disconnected headset
HeadsetPath: Full device path of the disconnected device

EPOS

Set Active Device Change

Send Active Device change event to SDK. This is used to select the active headset by the softphone.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<ReqActiveDeviceChanged>
  <ReqID>7</ReqID>
  <SFID>0</SFID>
  <HSSTATUS>1</HSSTATUS>
  <HeadsetPath>VID_1395&PID_00c8</HeadsetPath>
</ReqActiveDeviceChanged>
```

Event parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

HSSTATUS: Set this to true if the device is EPOS/DSEA, false otherwise

HeadsetPath: Full device path

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
< AckActiveDeviceChanged>
  <ReqID>7</ReqID>
  <SFID>0</SFID>
  <ReturnCode>0</ReturnCode>
  <HeadsetPath>VID_1395&PID_00c8</HeadsetPath>
  <HSSTATUS>1</HSSTATUS>
</ AckActiveDeviceChanged>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

HeadsetPath: Full device path.

HSSTATUS: Set this to true if the device is EPOS/DSEA, false otherwise

ReturnCode: 0 on pass or the error code.

EPoS

Other SDK functions

Redial last user from headset

If the user redials the last user by using the headset, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtRedialLastUser>
  <SFID>2</SFID>
  <IDForResponse>1</IDForResponse>
  <HeadsetPath>PID_0043</HeadsetPath>
</EvtRedialLastUser>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: When the SDK expects a response to the event it has sent it will provide this ID with the event. This ID must be used in the response that is sent for this event.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

Response

```
<?xml version="1.0" encoding="utf-8"?>
<RespRedialLastUser>
  <SFID>2</SFID>
  <IDForResponse>1</IDForResponse>
  <CallPlacedTo>Caller 1</CallPlacedTo>
  <ReturnCode>0</ReturnCode>
</RespRedialLastUser>
```

Response parameters

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: The ID for the response

CallPlacedTo: The name of the caller to whom the call is placed.

ReturnCode: 0 on pass or the error code.

Note: If the softphone mentions during registration that the "Redial" functionality is not supported, this message will not be sent to the softphone.

EPoS

Off-hook from headset

If the user does an Off-Hook event on the headset, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtOffHook>
<SFID>2</SFID>
<IDForResponse>2</IDForResponse>
<HeadsetPath>PID_0043</HeadsetPath>
</EvtOffHook>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: The id for the response

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

Response

```
<?xml version="1.0" encoding="utf-8"?>
<RespOffHook>
<SFID>2</SFID>
<IDForResponse>2</IDForResponse>
<ReturnCode>0</ReturnCode>
</RespOffHook>
```

Response parameters

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: The ID for the response

ReturnCode: 0 on pass or the error code.

When the user does an “Off-hook” event, it is expected that the user should be able to hear the dial-tone on the headset. This can be achieved by placing a call with no caller information. However, this is not possible in certain softphones (especially commercial phones like Skype, etc).

EPDS

On-hook from headset

If the previous sent Off-Hook event fails on the headset or cannot be executed, the SDK will send the following event information.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtOnHook>
<SFID>2</SFID>
<Reason>Headset event failed</Reason>
</EvtOnHook>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.
Reason: The reason the on-hook event is receive

EPOS

Send Presence status from softphone

If there is a presence status change in softphone, the following request must be sent to the SDK and the acknowledgement will follow. This is used for EPOS USB Busylight devices.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqSetPresence>
<SFID>2</SFID>
<ReqID>10</ReqID>
<State>Available</State>
</ReqSetPresence >
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

State: The string containing the Presence status. The list of possible values are Available, Away, Busy, DoNotDisturb and IMNotification.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckSetPresence>
<SFID>2</SFID>
<ReturnCode>0</ReturnCode>
</AckSetPresence>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

ReturnCode: 0 on pass or the error code.

EPOS

Send User Defined Custom Presence status from softphone

If there is a presence status change in softphone, the following request must be sent to the SDK and the acknowledgement will follow. This is for the EPOS USB Busylight devices.

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqSetPresence>
<SFID>2</SFID>
<ReqID>10</ReqID>
<State>" presenceName, presenceColor , presencePattern"</State>
</ReqSetPresence >
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
State: The string containing the Presence status is the combination of presenceName, presenceColor, presencePattern.Sets a User Defined Custom Presence State on the Busylight device.
presenceName: Expected value is any string. This parameter has no validation; it can also be left empty.
presenceColor: Expected value is of the format "#RRGGBB".
presencePattern: Expected value is {0,1,2,3,4}:
0 -- Busylight will not Glow
1 -- Busylight will continuously Glow
2 -- Busylight will Flash
3 -- Busylight will Pulse (Breathing Light effect)
4 -- Busylight will glow with a Long Flash Pattern

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckSetPresence>
<SFID>2</SFID>
<ReturnCode>0</ReturnCode>
</AckSetPresence>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ReturnCode: 0 on pass or the error code.

EPoS

Presenting Status

Sets the Busylight device to Presenting state in an Active Call state if set to true.

Event

```
<?xml version="1.0" encoding="utf-8"?>
<ReqPresentingStatus>
<ReqID>8</ReqID>
<SFID>0</SFID>
<PresentingStatus>1</PresentingStatus>
</ReqPresentingStatus>
```

Event parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.
PresentingStatus: Presenting (true/false)

Response

```
<?xml version="1.0" encoding="utf-8"?>
<ReqID>8</ReqID>
<SFID>0</SFID>
<ReturnCode>0</ReturnCode>
</AckPresenting>
```

Response parameters

SFID: The unique id for the softphone assigned by the SDK.
ReturnCode: 0 on pass or the error code.

EPoS

Enable Logging (Deprecated)

This request is deprecated and should not be used.

It is possible to generate logs of the messages and events that happen in the SDK. The following request needs to be sent to enable logging:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqEnableLogging>
<SFID>2</SFID>
<ReqID>9</ReqID>
</ReqEnableLogging>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckEnableLogging>
<SFID>2</SFID>
<ReqID>9</ReqID>
<ReturnCode>0</ReturnCode>
</AckEnableLogging>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

ReturnCode: 0 on pass or the error code.

EPICS

Disable Logging (Deprecated)

This request is deprecated and should not be used.

The following request must be sent when the user wants to stop logging:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqDisableLogging>
<SFID>2</SFID>
<ReqID>9</ReqID>
</ReqDisableLogging>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckDisableLogging>
<SFID>2</SFID>
<ReqID>9</ReqID>
<ReturnCode>0</ReturnCode>
</AckDisableLogging>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

ReturnCode: 0 on pass or the error code.

EPoS

Mute from softphone

If the call on the softphone is muted, the following request must be sent to the SDK, so that the headset can also be put in mute state:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqMuteHeadset>
<SFID>2</SFID>
<ReqID>26</ReqID>
</ReqMuteHeadset>
```

Request parameters

ReqID: The id of the request. This must be different for each request.
SFID: The unique id for the softphone assigned by the SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckMuteHeadset>
<SFID>2</SFID>
<ReqID>26</ReqID>
<ReturnCode>0</ReturnCode>
</AckMuteHeadset>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge
SFID: The unique id for the softphone assigned by the SDK.
ReturnCode: 0 on pass or the error code.

Note: If the softphone mentions during registration that “Mute” functionality is not supported, the SDK will ignore this message.

EPoS

Mute from headset

If the user presses the mute button on the headset, the following event is sent by the SDK:

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtMuteSoftphone>
  <SFID>2</SFID>
  <HeadsetPath>PID_0043</HeadsetPath>
</EvtMuteSoftphone>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

Note: If the softphone mentions during registration that “Mute” functionality is not supported, the SDK will ignore this message.

EPoS

Unmute from softphone

If the user un-mutes the call from the softphone UI, the following request must be sent:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqUnmuteHeadset>
<SFID>2</SFID>
<ReqID>28</ReqID>
</ReqUnmuteHeadset>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

Acknowledgement

```
<?xml version="1.0" encoding="utf-8"?>
<AckUnmuteHeadset>
<SFID>2</SFID>
<ReqID>28</ReqID>
<ReturnCode>0</ReturnCode>
</AckUnmuteHeadset>
```

Acknowledgement parameters

ReqID: ID of the request that prompted the acknowledge

SFID: The unique id for the softphone assigned by the SDK.

ReturnCode: 0 on pass or the error code.

Note: If the softphone mentions during registration that “Mute” functionality is not supported, the SDK will ignore this message.

EPoS

Unmute from headset

If the user un-mutes the headset, the event will be received as the following event:

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtUnmuteSoftphone>
  <SFID>2</SFID>
  <HeadsetPath>PID_0043</HeadsetPath>
</EvtUnmuteSoftphone>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

EPDS

Get softphone mute status

If the SDK wants to know if the softphone is muted or not, it will send the following event:

Event

```
<?xml version="1.0" encoding="utf-8"?>
<EvtGetSFMuteStatus>
<SFID>2</SFID>
<IDForResponse>3</IDForResponse>
<HeadsetPath>PID_0043</HeadsetPath>
</EvtGetSFMuteStatus>
```

Event parameters

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: When the SDK expects a response to the event it has sent it will provide this ID with the event. This ID must be used in the response that is sent for this event.

HeadSetPath: It contains the PID of the headset that is retrieved from device path.

Response

```
<?xml version="1.0" encoding="utf-8"?>
<RespSFMuteStatus>
<SFID>2</SFID>
<IDForResponse>3</IDForResponse>
<Mute>Yes</Mute>
<ReturnCode>0</ReturnCode>
</RespSFMuteStatus>
```

Response parameters

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: The unique id for the response.

Mute: 'Yes' if the softphone is Muted else 'No'. If the softphone does not support mute, please state 'No'.

ReturnCode: 0 on pass or the error code.

EPoS

Fail a call

If a call fails on the softphone due to known or unknown reason, the following request must be sent:

Request

```
<?xml version="1.0" encoding="utf-8"?>
<ReqCallFailed>
<SFID>2</SFID>
<ReqID>21</ReqID>
<CallID>1</CallID>
<Reason>Network connection lost</Reason>
</ReqCallFailed>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

CallID: The ID of the call failing.

NOTE: If the call failing is a conference call, please supply the conference ID in the CallID parameter.

EPDS

System Information

This API is to get the information about the SDK like product version, product name and product build version. To get that details the following request must be sent:

Request

```
<?xml version="1.0"?>
<ReqSystemInformation>
  <SFID>0</SFID>
  <ReqID>2</ReqID>
</ReqSystemInformation>
<?xml version="1.0" encoding="utf-8" ?>
```

Request parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

Response

```
<AckSystemInformation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <ReqID>2</ReqID>
  <SFID>0</SFID>
  <ReturnCode>0</ReturnCode>
  <ProductName>DSEASDK</ProductName>
  <ProductVersion>10.4.0.7291</ProductVersion>
  <FileVersion>10.4.0.7291</FileVersion>
</AckSystemInformation>
```

Response parameters

ReqID: The id of the request. This must be different for each request.

SFID: The unique id for the softphone assigned by the SDK.

IDForResponse: The unique id for the response.

ProductName : The name of the SDK

ProductVersion: The version of the SDK.

EPoS

Interfacing with the SDK using web socket

Once the SDK is started the web socket server will be started and it will be listening for connections. The WebSocket server is hosted on localhost (127.0.0.1) on port 41087 (WS) and 41088(WSS).

Initial Connection

The SDK offers 2 options, you can either connect secure or unsecure to the SDK through web sockets. For unsecure use ws://127.0.0.1:41087 and for secure use wss://127.0.0.1:41088.

To open the web socket in Qt you would do something like this;

```
m_pSocket = new QWebSocket();
if (m_pSocket){
    connect(m_pSocket, &QWebSocket::textMessageReceived,this, &webSDK::readyRead);
    m_pSocket->open(QUrl(QString("ws://127.0.0.1:41087")));
    return true;
}
else
    return false;
```

Once you start listening to the port, you will soon, if all is well, receive the SocketConnected event described on the next page.

The following pages will describe the next important events to relate to, and soon you will be placing your first incoming call to the SDK.

Note: When reading events on the socket from the SDK, it is important to take into consideration, that there might be more than one event. Proper parsing of the socket is essential for a good integration.

EPDS

Connection to the socket

When the softphone initiates the connection to “ws://127.0.0.1:41087” or “wss://127.0.0.1:41088”, it will receive the following event that will indicate that the socket has connected:

Event

```
{  
  "Event": "SocketConnected"  
  "EventType": "Notification"  
  "ReturnCode": 0  
}
```

Event parameters

Event: Name of the event, here “SocketConnected”.

EventType: Type of the event, here “Notification”.

ReturnCode: 0 on pass or the error code.

EPoS

Connection to SDK

Sending this request to the SDK will register the application as a potential user. The user does not have to be logged in to the softphone.

The softphone should send the following request to establish connection.

Request

```
{
  "Event": "EstablishConnection",
  "EventType": "Request",
  "SPName": "Softphone::Vendor Softphone 1",
  "SPBrandName": "Softphone Vendor Brand",
  "SPVersion": "1.0.0",
  "SPIconImage": "SPImage.ico",
  "RedialSupport": "Yes",
  "OffHookSupport": "Yes",
  "MuteSupport": "Yes",
  "AudioDeviceChangesSupport": "Yes",
  "DNDOption": "Yes",
  "ProtocolVersion": "1.0"
}
```

Request parameters

EventType: Type of the event, here "Request".

SPName: Name of the softphone. Name of the softphone should contain "Softphone::" followed by softphone name. E.g. "Softphone::Vendor Softphone 1". Here softphone name is Vendor Softphone 1.

SPBrandName: Brand Name of the softphone. Ex: "Softphone Vendor Brand"

SPVersion :Version of the softphone. Ex: "1.0.0".

SPIconImage: Fully qualified path of the icon image file of the softphone. The file must adhere to the following conditions:

The file format is '.ico'. Files in any other format or renamed as '.ico' may not give the same effect.

The size of the image is 16*16 pixels.

Note: softphone will have to provide the complete path of the icon in machine. If this icon is part of installer, provide only the file name. If above condition is failed, then a default image will be used for this softphone.

RedialSupport: If the softphone provides support for placing an outgoing call to the last called user when the event is received from the headset, set the value as "Yes" else use "No". By default, SDK assumes the value to be 'Yes'.

OffHookSupport: Off hook event is meant to provide a dial tone on the headset. If the softphone can provide this functionality set the value as "Yes" else set it as "No". By default, SDK assumes the value to be 'Yes'.

MuteSupport: If the softphone provides support for setting mute/unmute from headset and sends requests for setting mute/unmute to SDK, set this value to "Yes". By default, SDK assumes the value to be 'Yes'.

ProtocolVersion: The WebSocket protocol version. Currently supported protocol versions are 1.0 & 2.0. Default protocol version is "1.0".

Acknowledgement

```
{
  "Event": "EstablishConnection",
  "EventType": "Acknowledgement",
}
```

EPoS

```
"ReturnCode":0  
}
```

Acknowledgement parameters

Event: Name of the event, here "EstablishConnection".
EventType: Type of the event, here "Acknowledgement".
ReturnCode: 0 on pass or the error code.

Connection is being closed

If the SDK is being shut down, the SDK will send the following message:

Event

```
{  
  "Event": "ListenerClosed",  
  "EventType": "Notification",  
  "Reason": "REASON FOR SHUTDOWN"  
}
```

Event parameters

Event: Name of the event, here "ListenerClosed."
EventType: Type of the event, here "Notification".
Reason: The reason for the listener being shut down.

EPoS

Closing the connection

The client socket should send the following request to terminate the connection.

Request

```
{  
  "Event": "TerminateConnection",  
  "EventType": "Request"  
}
```

Request parameters

Event: Name of the event, here "TerminateConnection".

EventType: Type of the event, here "Request".

Acknowledgement

```
{  
  "Event": "TerminateConnection",  
  "EventType": "Acknowledgement",  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "TerminateConnection".

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

After the SDK receives this request it will assume that the application will not be sending any more messages. All messages sent after this message will be failed.

If the SP must send any more messages it should first make a request to establish the connection i.e., send an EstablishConnection Request message.

EPoS

Login and logout

Softphone login

When the user logs in to the SP, the following message must be sent by softphone:

Request

```
{  
  "Event": "SPLoggedIn",  
  "EventType": "Request"  
}
```

Request parameters

Event: Name of the event, here "SPLoggedIn".

EventType: Type of the event, here "Request".

Acknowledgement

```
{  
  "Event": "SPLoggedIn",  
  "EventType": "Acknowledgement",  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "SPLoggedIn".

EventType: Type of event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

EPoS

Softphone logout

When the user logs out from the SP, the following message must be sent:

Request

```
{
  "Event": "SPLoggedOut",
  "EventType": "Request"
}
```

Request parameters

Event: Name of the event, here "SPLoggedOut".

EventType: Type of the event, here "Request".

Acknowledgement

```
{
  "Event": "SPLoggedOut"
  "EventType": "Acknowledgement"
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "SPLoggedOut"

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

EPDS

Incoming calls

Making an incoming call

When an incoming call is received, the following request must be sent to the SDK:

Request

```
{
  "Event": "IncomingCall",
  "EventType": "Request",
  "CallID": "1A2Z"
}
```

Request parameters

Event: Name of the event, here "IncomingCall".

EventType: Type of the event, here "Request".

CallID: The unique id for the call. The uniqueness of the ID is the responsibility of the SF.

Acknowledgement

```
{
  "Event": "IncomingCall",
  "EventType": "Acknowledgement",
  "CallID": "1A2Z",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "IncomingCall".

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Accept and incoming call from softphone

If the incoming call is accepted from the softphone, the following request must be sent to the SDK:

Request

```
{  
  "Event": "InCallAccepted",  
  "EventType": "Request",  
  "CallID": 1  
}
```

Request parameters

Event: Name of the event, here "InCallAccepted"

EventType: Type of the event, here "Request".

CallID: The id for the call to accept.

Acknowledgement

```
{  
  "Event": "InCallAccepted",  
  "EventType": "Acknowledgement",  
  "CallID": 1,  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "InCallAccepted"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Incoming call accepted from headset

If the incoming call is accepted by using the headset, the SDK will send the following event information.

Event

```
{
  "Event": "InCallAccepted"
  "EventType": "Notification"
  "CallID": "1"
}
```

Event parameters

Event: Name of the event, here "InCallAccepted"
EventType: Type of the event, here "Notification".
CallID: ID of the call this event is about.

EPoS

Reject an incoming call from softphone

If the incoming call is rejected from the softphone UI, the following request must be sent:

Request

```
{
  "Event": "InCallRejected",
  "EventType": "Request",
  "CallID": 1
}
```

Request parameters

Event: Name of the event, here "InCallRejected"

EventType: Type of the event, here "Request".

CallID: The id for the call to reject.

Acknowledgement

```
{
  "Event": "InCallRejected",
  "EventType": "Acknowledgement",
  "CallID": 1,
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "InCallRejected"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Incoming call has been rejected from headset

If the incoming call is rejected by using the headset, the SDK will send the following event information.

Event

```
{  
  "Event": "InCallRejected",  
  "EventType": "Notification",  
  "CallID": "1"  
}
```

Event parameters

Event: Name of the event, here "InCallRejected"
EventType: Name of the event, here "Notification".
CallID: The id of the call rejected.

EPoS

Active calls

Ending a call from softphone

If the active call is ended from the softphone UI, the following request must be sent:

Request

```
{
  "Event": "CallEnded"
  "EventType": "Request"
  "CallID": 1,
}
```

Request parameters

Event: Name of the event, here "CallEnded"
EventType: Type of the event, here "Request".
CallID: The id for the call to end.

Acknowledgement

```
{
  "Event": "CallEnded"
  "EventType": "Acknowledgement"
  "CallID": "1"
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "CallEnded"
EventType: Type of the event, here "Acknowledgement".
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPDS

Call has been ended from headset

If the active call is ended by using the headset, the SDK will send the following event information.

Event

```
{  
  "Event": "CallEnded"  
  "EventType": "Notification",  
  "CallID": "1"  
}
```

Event parameters

Event: Name of the event, here "CallEnded".

EventType: Type of the event, here "Notification".

CallID: The id for the call ended.

Note, if the call ended, the CallID will contain the ConfID.

EPoS

Putting a call on hold from softphone

If the active call is put on hold from the softphone UI, the following request must be sent:

Request

```
{
  "Event": "CallHold"
  "EventType": "Request"
  "CallID": "1"
}
```

Request parameters

Event: Name of the event, here "CallHold"

EventType: Type of the event, here "Request".

CallID: The id for the call to hold.

Acknowledgement

```
{
  "Event": "CallHold",
  "EventType": "Acknowledgement",
  "CallID": "1",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "CallHold"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPDS

Call has been put on hold from headset

If the active call is put on hold from the headset, the following request must be sent:

Event

```
{
  "Event": "CallHold",
  "EventType": "Notification",
  "CallID": "1",
}
```

Event parameters

Event: Name of the event, here "CallHold"

EventType: Type of the event, here "Notification".

CallID: The id for the call put on hold.

EPoS

Resume a call from softphone

When the call that is put on hold is resumed from the softphone UI, the following request must be sent:

Request

```
{
  "Event": "HeldCallResumed",
  "EventType": "Request",
  "CallID": 1
}
```

Request parameters

Event: Name of the event, here "HeldCallResumed"

EventType: Type of the event, here "Request".

CallID: The id for the call resumed.

Acknowledgement

```
{
  "Event": "HeldCallResumed",
  "EventType": "Acknowledgement",
  "CallID": 1,
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "HeldCallResumed"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Call has been resumed from headset

If the active call is resumed from the headset, the following request must be sent:

Event

```
{  
  "Event": "HeldCallResumed",  
  "EventType": "Notification"  
  "CallID": "1",  
}
```

Event parameters

Event: Name of the event, here "HeldCallResumed"

EventType: Type of the event, here "Notification".

CallID: The id for the call resumed.

EPDS

Outgoing calls

Making an outgoing call

If an outgoing call is made from the softphone UI, the following request must be sent:

Request

```
{
  "Event": "OutgoingCall",
  "EventType": "Request",
  "CallID": 1
}
```

Request parameters

Event: Name of the event, here "OutgoingCall".

EventType: Type of the event, here "Request".

CallID: The unique id for the call. The uniqueness must be ensured by the SF.

CallerName: The name of the user to whom the call has been placed.

Acknowledgement

```
{
  "Event": "OutgoingCall",
  "EventType": "Acknowledgement",
  "CallID": "1",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "OutgoingCall"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPDS

Outgoing call has been accepted by remote party

If the outgoing call is accepted by the remote party, the following request must be sent:

Request

```
{
  "Event": "OutCallAccepted",
  "EventType": "Request",
  "CallID": 1
}
```

Request parameters

Event: Name of the event, here "OutCallAccepted".

EventType: Type of the event, here "Request".

CallID: The id of the call accepted.

Acknowledgement

```
{
  "Event": "OutCallAccepted",
  "EventType": "Acknowledgement",
  "CallID": 1,
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "OutCallAccepted"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPDS

Outgoing call has been rejected by remote party

If the remote party rejects the outgoing call, the following request must be sent:

Request

```
{
  "Event": "OutCallRejected",
  "EventType": "Request",
  "CallID": "1"
}
```

Request parameters

Event: Name of the event, here "OutCallRejected"
EventType: Type of the event, here "Acknowledgement".
CallID: The id of the call rejected.

Acknowledgement

```
{
  "Event": "OutCallRejected",
  "EventType": "Acknowledgement",
  "CallID": "1",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "OutCallRejected".
EventType: Type of the event, here "Acknowledgement".
CallID: ID of the call this event is about.
ReturnCode: 0 on pass or the error code.

EPDS

Outgoing call has been ended by caller

If the caller ends the call before remote party accepts, the following request must be sent:

Request

```
{  
  "Event": "OutCallEnded",  
  "EventType": "Request",  
  "CallID": "1"  
}
```

Request parameters

Event: Name of the event, here "OutCallEnded"

EventType: Type of the event, here "Acknowledgement".

CallID: The id of the call ended.

Acknowledgement

```
{  
  "Event": "OutCallEnded",  
  "EventType": "Acknowledgement",  
  "CallID": "1",  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "OutCallEnded".

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Conference calls

Start a conference call

If the user starts a conference call, the following request must be sent:

Request

```
{
  "Event": "StartConfCall",
  "EventType": "Request",
  "ConfID": 1
}
```

Request parameters

Event: Name of the event, here "StartConfCall".

EventType: Type of the event, here "Request".

ConfID: The unique id for the conference call. The uniqueness is the responsibility of the SF.

Acknowledgement

```
{
  "Event": "StartConfCall",
  "EventType": "Acknowledgement",
  "ConfID": "1",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "StartConfCall".

EventType: Type of the event, here "Acknowledgement".

ConfID: ID of the conference this event is about.

ReturnCode: 0 on pass or the error code.

Irrespective of the number of participants in the conference, the SDK treats the conference as a single call. If participants are added to the conference after the conference has been started, do not send any additional messages to the SDK.

To start a conference, you need to end all the calls you wish to combine into a conference and then start the conference.

EPoS

End a conference call

When the conference call is ended, the following request must be sent:

Request

```
{
  "Event": "EndConfCall",
  "EventType": "Request",
  "ConfID": 1
}
```

Request parameters

Event: Name of the event, here "EndConfCall".
EventType: Type of the event, here "Request".
ConfID: The id for the conference to end.

Acknowledgement

```
{
  "Event": "EndConfCall",
  "EventType": "Acknowledgement",
  "CallID": "1",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "EndConfCall".
EventType: Type of the event, here "Acknowledgement".
ConfID: ID of the conference this event is about.
ReturnCode: 0 on pass or the error code.

Assume there are 3 callers in a conference viz. Caller1, Caller2, and Caller3, where Caller1 is the host (initiator) of the conference. If Caller3 ends the call from the remote end, do not end the conference.

This is because irrespective of the number of participants in the conference, the conference is treated as a single call. Only when the last participant in the conference ends the call you should end the conference.

EPoS

Put a conference call on hold

If the conference call is put on hold, the following request must be sent:

Request

```
{
  "Event": "ConfCallOnHold",
  "EventType": "Request",
  "ConfID": 1
}
```

Request parameters

Event: Name of the event, here "ConfCallOnHold".

EventType: Type of the event, here "Request".

ConfID: The id of the conference to put on hold.

Acknowledgement

```
{
  "Event": "ConfCallOnHold",
  "EventType": "Acknowledgement",
  "ConfID": 1,
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "ConfCallOnHold".

EventType: Type of the event, here "Acknowledgement".

ConfID: ID of the conference this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Resume a conference call

If the conference call that was put on is resumed from the softphone UI, the following request must be sent:

Request

```
{
  "Event": "HeldConfCallResumed",
  "EventType": "Request",
  "ConfID": 1
}
```

Request parameters

Event: Name of the event, here "HeldConfCallResumed".

EventType: Type of the event, here "Request".

ConfID: The id of the conference to resume.

Acknowledgement

```
{
  "Event": "HeldConfCallResumed",
  "EventType": "Acknowledgement",
  "ConfID": 1,
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: name of the event, here "HeldConfCallResumed".

EventType: Type of the event, here "Acknowledgement".

ConfID: ID of the conference this event is about.

ReturnCode: 0 on pass or the error code.

EPoS

Current headset

Headset Connected

The SDK would send HeadsetConnected notification when an connected headset is made active/default device.

Event

```
{
  "Event": "HeadsetConnected",
  "EventType": "Notification",
  "HeadsetName": "headset name",
  "HeadsetType": "headset type",
  "HeadsetPath": "PID_740"
}
```

Event parameters

Event: Name of the event, here "HeadsetConnected".
EventType: Type of the event, here "Notification".
HeadsetName: Name of the connected Headset.
HeadsetType: Type of the connected Headset.
HeadsetPath: Full device path of the connected device.

Headset Disconnected

The SDK would send HeadsetDisconnected notification when a connected headset is removed as active/default device.

Event

```
{
  "Event": "HeadsetDisconnected",
  "EventType": "Notification",
  "HeadsetName": "headset name",
  "HeadsetType": "headset type",
  "HeadsetPath": "PID_740"
}
```

Event parameters

Event: Name of the event, here "HeadsetDisconnected".
EventType: Type of the event, here "Notification".
HeadsetName: Name of the disconnected headset.
HeadsetType: Type of the disconnected headset.
HeadsetPath: Full device path of the disconnected device.

Commented [A(1)]: @Rashmi Vyshnavi (RAVY) (Consultant)
, this should be HeadsetDisconnected

EPDS

Headset Available

The SDK would send HeadsetAvailable notification when a new headset is connected. This API is available only with WebSocket V2 protocol.

Event

```
{
  "Event": "HeadsetAvailable",
  "EventType": "Notification",
  "HeadsetName": "headset name",
  "HeadsetType": "headset type",
  "HeadsetPath": "PID_740"
}
```

Event parameters

Event: Name of the event, here "HeadsetAvailable".
EventType: Type of the event, here "Notification".
HeadsetName: Name of the disconnected headset.
HeadsetType: Type of the disconnected headset.
HeadsetPath: Full device path of the disconnected device.

Commented [A(2)]: Need to mention that this API is available only the v2 of the protocol

Also the login message has to be updated to indicate the version 2 of the protocol

EPDS

Headset Removed

The SDK would send HeadsetRemoved notification when the headset is removed. This API is available only with WebSocket V2 protocol.

Commented [A(3)]: Need to mention that this API is available only the v2 of the protocol

Also the login message has to be updated to indicate the version 2 of the protocol

Event

```
{
  "Event": "HeadsetRemoved",
  "EventType": "Notification",
  "HeadsetName": "headset name",
  "HeadsetType": "headset type",
  "HeadsetPath": "PID_740"
}
```

Event parameters

Event: Name of the event, here "HeadsetRemoved".
EventType: Type of the event, here "Notification".
HeadsetName: Name of the disconnected headset.
HeadsetType: Type of the disconnected headset.
HeadsetPath: Full device path of the disconnected device.

EPOS

Active Device Change

Send active device changed event to SDK. This is used to select the active headset by the softphone. At a time only one headset can be active and call control can be done only using the active headset.

Event

```
{
  "Event": "SendActiveDeviceChanged",
  "EventType": "Request",
  <HSSTATUS>1</HSSTATUS>
  <HeadsetPath>VID_1395&PID_00c8</HeadsetPath>
}
```

Event parameters

Event: Name of the event, here "SendActiveDeviceChanged".

EventType: Type of the event, here "Request".

HSSTATUS: Set this to true if the device is EPOS/DSEA, false otherwise

HeadsetPath: Full device path

Acknowledgement

```
{
  "Event": "SendActiveDeviceChanged",
  "EventType": "Acknowledgement",
  <HSSTATUS>1</HSSTATUS>
  <HeadsetPath>VID_1395&PID_00c8</HeadsetPath>
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "SendActiveDeviceChanged".

EventType: Type of the event, here "Request".

HSSTATUS: Set this to true if the device is EPOS/DSEA, false otherwise

HeadsetPath: Full device path

ReturnCode: 0 on pass or the error code.

EPDS

Other SDK functions

Redial last user from headset

If the user redials the last user by using the headset, the SDK will send the following event information.

Event

```
{
  "Event": "RedialLastUser",
  "EventType": "Request",
  "IDForResponse": 1,
  "HeadsetPath": "PID_0043"
}
```

Event parameters

Event: Name of the event, here "RedialLastUser".

EventType: Type of the event, here "Request".

IDForResponse: When the SDK expects a response to the event it has sent it will provide this ID with the event. This ID must be used in the response that is sent for this event.

HeadsetPath: The value of this field indicates the full PID (Product ID) of the device on which Redial event is triggered.

Response

```
{
  "Event": "RedialLastUser",
  "EventType": "Acknowledgement",
  "IDForResponse": 1,
  "CallPlacedTo": "Caller 1",
  "ReturnCode": 0
}
```

Response parameters

Event: Name of the event, here "RedialLastUser".

EventType: Type of the event, here "Acknowledgement".

IDForResponse: The ID for the response.

CallPlacedTo: The name of the caller to whom the call is placed.

ReturnCode: 0 on pass or the error code.

Note: If the softphone mentions during registration that the "Redial" functionality is not supported, this message will not be sent to the softphone.

EPDS

Off-hook from headset

If the user does an Off-Hook event on the headset, the SDK will send the following event information.

Event

```
{
  "Event": "OffHook",
  "EventType": "Request",
  "IDForResponse": 2,
  "HeadsetPath": "\\?\\HID#VID_1395&PID_0043&MI_03&Col01#8&12eb85c3&0&0000#{4d1e55b2-f16f-11cf-88cb-00111000030}"
}
```

Event parameters

Event: Name of the event, here "OffHook".

EventType: Type of the event, here "Request".

IDForResponse: The id for the response.

HeadsetPath: The value of this field indicates the full PID (Product ID) of the device on which off-hook button is pressed.

Response

```
{
  "Event": "OffHook",
  "EventType": "Acknowledgement",
  "IDForResponse": 2,
  "ReturnCode": 0
}
```

Response parameters

Event: Name of the event, here "OffHook".

EventType: Type of the event, here "Acknowledgement".

IDForResponse: The ID for the response.

ReturnCode: 0 on pass or the error code.

When the user does an "Off-hook" event, it is expected that the user should be able to hear the dial-tone on the headset. This can be achieved by placing a call with no caller information. However, this is not possible in certain softphones (especially commercial phones like Skype, etc).

EPDS

On-hook from headset

If the previous sent Off-Hook event fails on the headset or cannot be executed, the SDK will send the following event information.

Event

```
{  
  "Event": "OnHook"  
  "EventType": "Notification",  
  "Reason": "Headset event failed"  
}
```

Event parameters

Event: Name of the event, here "OnHook".

EventType: Type of the event, here "Notification".

Reason: The reason the on-hook event is received.

EPOS

Send Presence status from softphone

If there is a presence status change in softphone, the following request must be sent to the SDK and the acknowledgement will follow. This is for the EPOS USB Busylight devices.

Event

```
{  
  "Event": "SetPresence",  
  "EventType": "Request",  
  "State": "Available"  
}
```

Event parameters

Event: Name of the event, here "SetPresence".

EventType: Type of the event, here "Request".

State: The string containing the Presence status. The list of possible values are Available, Away, Busy, DoNotDisturb and IMNotification.

Response

```
{  
  "Event": "SetPresence",  
  "EventType": "Acknowledgement",  
  "ReturnCode": 0  
}
```

Response parameters

Event: Name of the event, here "SetPresence".

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

EPOS

Send User Defined Custom Presence status from softphone

If there is a presence status change in softphone, the following request must be sent to the SDK and the acknowledgement will follow. This is for the EPOS USB Busylight devices.

Event

```
{
  "Event": "SetPresence",
  "EventType": "Request",
  "State": " presenceName, presenceColor , presencePattern"
}
```

Event parameters

Event: Name of the event, here "SetPresence".

EventType: Type of the event, here "Request".

State: The string containing the Presence status is the combination of presenceName, presenceColor, presencePattern. Sets a User Defined Custom Presence State on the Busylight device.

presenceName: Expected value is any string. This parameter has no validation; it can also be left empty.

presenceColor: Expected value is of the format "#RRGGBB".

presencePattern: Expected value is {0,1,2,3,4}:

0 -- Busylight will not Glow

1 -- Busylight will continuously Glow

2 -- Busylight will Flash

3 -- Busylight will Pulse (Breathing Light effect)

4 -- Busylight will glow with a Long Flash Pattern

Response

```
{
  "Event": "SetPresence",
  "EventType": "Acknowledgement",
  "ReturnCode": 0
}
```

Response parameters

Event: Name of the event, here "SetPresence".

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

EPoS

Presenting Status

Sets the Busylight device to Presenting state in an Active Call state if set to true

Event

```
{
  "Event": "Presenting",
  "EventType": "Request",
  "PresentingStatus": true
}
```

Event parameters

Event: Name of the event, here “Presenting”.
EventType: Type of the event, here “Request”.
PresentingStatus: Presenting (true/false)

Response

```
{
  "Event": "SetPresence",
  "EventType": "Acknowledgement",
  "ReturnCode": 0
}
```

Response parameters

Event: Name of the event, here “Presenting”.
EventType: Type of the event, here “Request”.
ReturnCode: 0 on pass or the error code.

EPICS

Enable Logging (Deprecated)

This request is deprecated and should not be used.

It is possible to generate logs of the messages and events that happen in the SDK. The following message needs to be sent to enable logging:

Request

```
{
  "Event": "EnableLogging",
  "EventType": "Request",
  "LogFileName": "Path of log file"
}
```

Request parameters

Event: Name of the event, here "EnableLogging".

EventType: Type of the event, here "Request".

LogFileName: The fully qualified path for the log file. If this is blank, the logs will not be generated.

Acknowledgement

```
{
  "Event": "EnableLogging",
  "EventType": "Acknowledgement",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "EnableLogging".

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

EPoS

Disable Logging (Deprecated)

This request is deprecated and should not be used.

The following request must be sent when the user wants to stop logging:

Request

```
{
  "Event": "DisableLogging",
  "EventType": "Request"
}
```

Request parameters

Event: Name of the event, here "DisableLogging"
EventType: Type of the event, here "Request".

Acknowledgement

```
{
  "Event": "DisableLogging",
  "EventType": "Acknowledgement",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "DisableLogging".
EventType: Type of the event, here "Acknowledgement".
ReturnCode: 0 on pass or the error code.

EPoS

Mute from softphone

If the call on the softphone is muted, the following request must be sent to the SDK, so that the headset can also be put in mute state:

Request

```
{  
  "Event": "MuteHeadset",  
  "EventType": "Request"  
}
```

Request parameters

Event: Name of the event, here "MuteHeadset".

EventType: Type of the event, here "Request".

Acknowledgement

```
{  
  "Event": "MuteHeadset",  
  "EventType": "Acknowledgement",  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "MuteHeadset".

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

Note: If the softphone mentions during registration that "Mute" functionality is not supported, the SDK will ignore this message.

EPDS

Mute from headset

If the user presses the mute button on the headset, the following message is sent by the SDK:

Event

```
{  
  "Event": "MuteSoftphone"  
  "EventType": "Notification"  
}
```

Event parameters

Event: Name of the event, here "MuteSoftphone"

EventType: Type of the event, here "Notification".

Note: If the softphone mentions during registration that "Mute" functionality is not supported, the SDK will ignore this message.

EPoS

Unmute from softphone

If the user un-mutes the call from the softphone UI, the following request must be sent:

Request

```
{  
  "Event": "UnmuteHeadset",  
  "EventType": "Request"  
}
```

Request parameters

Event: Name of the event, here "UnmuteHeadset"

EventType: Type of the event, here "Request".

Acknowledgement

```
{  
  "Event": "UnmuteHeadset"  
  "EventType": "Acknowledgement"  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "UnmuteHeadset"

EventType: Type of the event, here "Acknowledgement".

ReturnCode: 0 on pass or the error code.

Note: If the softphone mentions during registration that "Mute" functionality is not supported, the SDK will ignore this message.

EPoS

Unmute from headset

If the user un-mutes the headset, the event will be received as the following message:

Event

```
{  
  "Event": "UnmuteSoftphone",  
  "EventType": "Notification"  
}
```

Event parameters

Event: Name of the event, here "UnmuteSoftphone"

EventType: Type of the event, here "Notification".

Note: If the softphone mentions during registration that "Mute" functionality is not supported, the SDK will ignore the following message.

EPoS

Get softphone mute status

If the SDK wants to know if the softphone is muted or not, it will send the following event:

Request

```
{  
  "Event": "GetSPMuteStatus",  
  "EventType": "Request",  
  "IDForResponse": 3  
}
```

Request parameters

Event: Name of the event, here "GetSPMuteStatus"

EventType: Type of the event, here "Request".

IDForResponse: When the SDK expects a response to the event it has sent it will provide this ID with the event. This ID must be used in the response that is sent for this event.

Response

```
{  
  "Event": "SPMuteStatus",  
  "EventType": "Acknowledgement",  
  "IDForResponse": 3,  
  "Mute": "Yes",  
  "ReturnCode": 0  
}
```

Response parameters

Event: Name of the event, here "SPMuteStatus"

EventType: Type of the event, here "Acknowledgement".

IDForResponse: The unique id for the response.

Mute: 'Yes' if the softphone is Muted else 'No'. If the softphone does not support mute, please state 'No'.

ReturnCode: 0 on pass or the error code.

EPOS

Fail a call

If the call fails on the softphone due to known or unknown reason, the following request must be sent:

Request

```
{
  "Event": "CallFailed"
  "EventType": "Notification",
  "CallID": 1,
  "Reason": "Network connection lost"
}
```

Request parameters

Event: Name of the event, here "CallFailed"

EventType: Type of the event, here "Notification".

CallID: The id of the call to fail.

Acknowledgement

```
{
  "Event": "CallFailed",
  "EventType": "Acknowledgement",
  "CallID": "0",
  "ReturnCode": 0
}
```

Acknowledgement parameters

Event: Name of the event, here "CallFailed"

EventType: Type of the event, here "Acknowledgement".

CallID: ID of the call this event is about.

ReturnCode: 0 on pass or the error code.

NOTE: If the call failing is a conference call, please supply the conference ID in the CallID parameter.

EPoS

System Information

This API is to get the information about the SDK like product version, product name and product build version. To get that details the following request must be sent:

Request

```
{  
  "Event": "SystemInformation",  
  "EventType": "Request"  
}
```

Request parameters

Event: Name of the event, here "SystemInformation".

EventType: Type of the event, here "Request".

Acknowledgement

```
{  
  "Event": "SystemInformation",  
  "EventType": "Acknowledgement",  
  "ProductName": "Call control SDK",  
  "ProductVersion": "10.1.0.1998",  
  "ReturnCode": 0  
}
```

Acknowledgement parameters

Event: Name of the event, here "SystemInformation"

EventType: Type of the event, here "Acknowledgement".

ProductName: The name of the SDK.

ProductVersion: The version of the SDK.

ReturnCode: 0 on pass or the error code.

EPOS

Interfacing with the SDK using REST API

This REST API will be part of SDK application. SDK must run in the user machine after installation. The SDK will make the REST server active in localhost (127.0.0.1) on port number 41085 (HTTP) & 41086(HTTPS).

All requests must be sent to the following url;

`http://127.0.0.1:41085/DSEA/CallController/{softphone_id}`

or

`https://127.0.0.1:41086/DSEA/CallController/{softphone_id}`

Where **{softphone_id}** must be a string selected by the softphone that ensure uniqueness toward other softphones. We recommend using the name of your softphone.

To make documentation easier to read we will refer to

`http://127.0.0.1:41085/DSEA`

and

`https://127.0.0.1:41086/DSEA`

as the **{base_url}**

REST API follow request response architecture i.e. the client (SP) makes a request and server (SDK) responds to the request. The originating Headset events cannot be sent by REST server to softphones. The client must periodically request to get headset events by the below stated API.

Request

URL: **{base_url}/CallController/{softphone_id}/CallEvents**
Method: GET

Response

```
{
  "CallEvents":
  [
    {
      //Data for Call Event 1
    },
    {
      //Data for Call Event 2
    },
    .....
  ]
}
```

The events between two consecutive requests are kept in API as an array and passed as payload, on client request. The client must go through the events in same order.

The softphone must request these events continuously to keep the SDK and softphone synced. As we are polling and getting the events there will be a delay based on the selected interval.

If client want no delay in message, use web sockets instead of REST API.

EPDS

Initial Connection

Due to the nature of the REST API there is no need to make an initial connection or to shutdown a connection once done with it.

Connection is being closed

The SDK will let the softphone know, that it is about to shut down by posting the below call event.

Event

```
{  
  "Event": "ListenerClosed"  
  "Reason": "REASON FOR SHUTDOWN"  
}
```

Event parameters

Event: Name of the event, here "ListenerClosed".

Reason: The reason for the listener being shut down.

EPoS

Login and logout

Softphone login

Whenever Softphone wants to use the SDK, it must first register itself by making the following request with its supporting features as JSON message.

Request

```
URL: {baseurl}/CallController/{softphone_id}/Login-Logout
Method: POST
Data:
{
  "SPName": "Softphone::Vendor Softphone 1",
  "SPBrandName": "Softphone Vendor Brand",
  "SPVersion": "1.0.0",
  "SPIconImage": "SPImage.ico",
  "RedialSupport": "Yes",
  "OffHookSupport": "Yes",
  "MuteSupport": "Yes",
  "AudioDeviceChangesSupport": "Yes"
}
```

Request parameters

SPName: Name of the softphone. Name of the softphone should contain "Softphone::" followed by softphone name. E.g. "Softphone::Vendor Softphone 1". Here softphone name is Vendor Softphone 1.

SPBrandName: Brand Name of the softphone. Ex: "Softphone Vendor Brand"

SPVersion : Version of the softphone. Ex: "1.0.0".

SPIconImage: Fully qualified path of the icon image file of the softphone. The file must adhere to the following conditions: The file format is '.ico'. Files in any other format or files renamed as '.ico' may not give the same effect. The size of the image is 16*16 pixels. Softphone will have to provide the complete path of the icon in machine. If this icon is part of installer, provide only the file name. If above conditions are failing, then a default image will be used for this softphone.

RedialSupport: If the softphone provides support for placing an outgoing call to the last called user when the event is received from the headset, set the value as "Yes" else use "No". By default, SDK assumes the value to be 'Yes'.

OffHookSupport: Off-hook event is meant to provide a dial tone on the headset. If the softphone can provide this functionality set the value as "Yes" else set it as "No". By default, SDK assumes the value to be 'Yes'.

MuteSupport: If the softphone provides support for setting mute/unmute from headset and sends requests for setting mute/unmute to SDK, set this value to "Yes". By default, SDK assumes the value to be 'Yes'.

EPDS

Softphone logout

To stop using the services, user must make the following requests. For reusing the services, the softphone must reregister by softphone login.

Request

URL: {baseurl}/CallController/{softphone_id} /Login-Logout
Method: DELETE

EPDS

Incoming call

Making an incoming call

When an incoming call is received, the following request must be made:

Request

```
URL: {baseUrl}/CallController/{softphone_id}/Call
Method: POST
Data:
{
  "CallDirection": "Incoming",
  "CallID": "1"
}
```

Request parameters

CallDirection: for incoming call use "Incoming".

CallID: The unique id for the call. The uniqueness is the responsibility of the SF

EPoS

Accept and incoming call from softphone

If the incoming call is accepted from the softphone, the following request must be made to the SDK:

Request

```
URL: {baseurl}/CallController/{softphone_id}/Call
Method: PUT
Data
{
  "CallDirection": "Incoming",
  "CallID": "1"
}
```

Request parameters

CallDirection: for incoming call use "Incoming".
CallID: The id of the call accepted.

EPDS

Incoming call accepted from headset

If the incoming call is accepted by using the headset, the SDK will send the following call event

Event

```
{
  "Event": "InCallAccepted",
  "CallID": "1"
}
```

Event parameters

Event: Name of event, here "InCallAccepted".
CallID: ID of the call accepted.

Acknowledgement

```
URL: {baseUrl}/CallController/{softphone_id}/CallEventsAck
Method: PUT
Data:
{
  "Event": "InCallAcceptedAck",
  "CallID": "1"
}
```

Acknowledgement parameters

Event: Name of the event, here "InCallAcceptedAck".
CallID: ID of the call being accepted.

EPDS

Reject an incoming call from softphone

If the incoming call is rejected from the softphone UI, the following request must be made:

Request

```
URL: {baseurl}/CallController/{softphone_id} /Callrejected
Method: DELETE
Data:
{
  "CallDirection": "Incoming",
  "CallID": "1"
}
```

Request parameters

CallDirection: for incoming call use "Incoming".
CallID: The unique id for the call.

EPDS

Incoming call has been rejected from headset

If the incoming call is rejected by using the headset, the SDK will send the following call event:

Event

```
{  
  "Event": "InCallRejected",  
  "CallID": "1"  
}
```

Event parameters

Event: Name of the event, here "InCallRejected".
CallID: ID of the call this acknowledgement is about.

Acknowledgement

```
URL: {baseUrl}/CallController/SOFTPHONE_ID/CallEventsAck  
Method: PUT  
Data:  
{  
  "Event": "InCallRejectedAck",  
  "CallID": "1"  
}
```

Acknowledgement parameters

Event: Name of the event, here "InCallRejectedAck".
CallID: ID of the call rejected.

EPoS

Active calls

Ending a call from softphone

If the active call is ended from the softphone UI, the following request must be made:

Request

```
URL: {baseurl}/CallController/{softphone_id}/Call
Method: DELETE
Data:
{
  "CallID": "1"
}
```

Request parameters

CallID: The id of the call ended.

EPICS

Call has been ended from headset

If the active call is ended by using the headset, the SDK will send the following Event to the softphone

Event

```
{
  "Event": "CallEnded",
  "CallType": "Normal",
  "CallID": "1"
}
```

Event parameters

Event: Name of the event, here "CallEnded".
CallType: "Normal" for normal call, "Conf" for Conference call.
CallID: ID of the call this acknowledgement is about.

Acknowledgement

```
URL: {baseUrl}/CallController/{softphone_id}/CallEventsAck
Method: PUT
Data:
{
  "Event": "CallEndedAck",
  "CallType": "Normal",
  "CallID": "1"
}
```

Acknowledgement parameters

Event: Name of the event, here "callEndedAck".
CallType: Softphone must pass this value as "Normal" for normal call and "Conf" for conference call
CallID: The id of the call terminated.

EPoS

Putting a call on hold from softphone

If the active call is put on hold from the softphone UI, the following request must be made:

Request

```
URL: {baseurl}/CallController/{softphone_id}/HoldCall
Method: PUT
Data:
{
  "CallID": "1"
}
```

Request parameters

CallID: The id of the call put on hold.

EPDS

Call has been put on hold from headset

If the active call is put on hold by using the headset, the SDK will send the following call event

Event

```
{  
  "Event": "CallHold",  
  "CallID": "1"  
}
```

Event parameters

Event: Name of the event, here "CallHold".
CallID: ID of the call this event is about.

Acknowledgement

```
URL: {baseurl}/CallController/{softphone_id}/CallEventsAck  
Method: PUT  
Data:  
{  
  "Event": "CallHoldAck",  
  "CallID": "1"  
}
```

Acknowledgement parameters

Event: Name of the event, here "CallHoldAck".
CallID: The id of the call that is put on hold.

EPoS

Resume a call from softphone

When the call on hold is resumed from the softphone UI, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/ResumeCall
Method: PUT
Data:
{
  "CallID": "1"
}
```

Request parameters

CallID: The id for the call resumed.

EPoS

Call has been resumed from headset

On resuming the held call from headset, the SDK will send the following call event:

Event

```
{  
  "Event": "CallResume",  
  "CallID": "1"  
}
```

Event parameters

Event: Name of the event, here "CallResume".
CallID: ID of the call this event is about.

EPDS

Outgoing calls

Making an outgoing call

If an outgoing call is made from the softphone UI, the following request must be sent:

Request

```
URL: {baseUrl}/CallController/{softphone_id}/Call
Method: POST
Data:
{
  "CallDirection": "Outgoing",
  "CallID": "1"
}
```

Request parameters

CallDirection: this should be "Outgoing" for outgoing calls.

CallID: The unique id for the call. The uniqueness is the responsibility of the SF.

EPDS

Outgoing call has been accepted by remote party

If the incoming call is accepted by the softphone, the following request must be made to SDK:

Request

```
URL: {baseurl}/CallController/{softphone_id}/Call
Method: PUT
Data:
{
  "CallDirection": "Outgoing",
  "CallID": "1"
}
```

Request parameters

CallID: The id of the call accepted.

EPDS

Outgoing call has been rejected by remote party

If the remote party rejects the call, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/Callrejected
Method: DELETE
Data:
{
  "CallDirection": "Outgoing",
  "CallID": "1"
}
```

Request parameters

CallDirection: will be "Outgoing" for outgoing calls.
CallID: The id of the call rejected.

EPoS

Outgoing call has been ended by caller

If the caller ends the call before remote party accepts, the following request must be sent:

Request

```
URL: [baseurl]/CallController/{softphone_id}/CallEnded
Method: DELETE
Data:
{
  "CallDirection": "outgoing",
  "CallID": "1"
}
```

Request parameters

CallDirection: will be "outgoing" for outgoing calls.
CallID: The id of the call ended.

EPoS

Conference calls

Start a conference call

If the user starts a conference call, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/ConfCall
Method: POST
Data:
{
  "ConfID":1
}
```

Request parameters

ConfID: The unique id for the conference call. The uniqueness is the responsibility of the SF.
Irrespective of the number of participants in the conference, the SDK treats the conference as a single call. If participants are added to the conference after the conference has been started, do not send any additional messages to the SDK.
To start a conference, you need to end all the calls you wish to combine into a conference and then start the conference.

EPICS

End a conference call

When the conference call is ended, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/ConfCall
Method: DELETE
Data:
{
  "ConfID":1
}
```

Request parameters

ConfID: The id of the conference ended.

Assume there are 3 callers in a conference viz. Caller1, Caller2, and Caller3, where Caller1 is the host (initiator) of the conference. If Caller3 ends the call from the remote end, do not end the conference. This is because irrespective of the number of participants in the conference, the conference is treated as a single call. Only when the last participant in the conference ends the call you should end the conference.

EPoS

Put a conference call on hold

If the conference call is put on hold, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/ConfHold
Method: PUT
Data:
{
  "ConfID":1
}
```

Request parameters

ConfID: The id of the conference put on hold.

EPoS

Resume a conference call

If the conference call that was put on hold is resumed from the softphone UI, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/ConfResume
Method: PUT
Data:
{
  "ConfID":1
}
```

Parameters:

ConfID: The id of the conference resumed.

EPoS

Current headset

Headset connected

If the user connects the Headset the following event will be sent to the softphone

Event

```
{  
  "Event": "HeadsetConnected",  
  "HeadsetName": "name of headset",  
  "HeadsetType": "type of headset",  
  "HeadsetPath": "PID_740"  
}
```

Event parameters

Event: Name of the event, here "HeadsetConnected"
HeadsetName: Name of the connected headset
HeadsetType: type of the connected headset
HeadsetPath: Full device path of the connected device

EPoS

Headset disconnected

If the user disconnects the Headset the following event will be sent to the softphone

Event

```
{
  "Event": "HeadsetDisconnected",
  "HeadsetName": "name of headset",
  "HeadsetType": "type of headset",
  "HeadsetPath": "PID_740"
}
```

Event parameters

Event: Name of the event, here "HeadsetDisconnected"
HeadsetName: Name of the disconnected headset
HeadsetType: type of the disconnected headset
HeadsetPath: Full device path of the disconnected device

EPOS

Active Device Change

Send active device changed event to SDK. This is used to select the active headset by the softphone. At a time only one headset can be active and call control can be done only using the active headset.

Event

```
{  
  "Event": "ActiveDeviceChange",  
  "HSSTATUS": "1",  
  "HeadsetPath": "VID_1395&PID_006d"  
}
```

Event parameters

Event: Name of the event, here "SendActiveDeviceChanged".

HSSTATUS: Set this to true if the device is EPOS/DSEA, false otherwise

HeadsetPath: Full device path

EPDS

Other SDK functions

Redial last user from headset

If the user redials the last user by using the headset, the SDK will send the following event:

Event

```
{  
  "Event": "RedialLastUser"  
  "HeadsetPath": "PID_0043"  
}
```

Event parameters

Event: Name of the event, here "RedialLastUser".

HeadsetPath: The value of this field indicates the full PID (Product ID) of the device on which Redial event is triggered.

Response

```
URL: {baseurl}/CallController/{softphone_id} /Redial  
Method: PUT  
Data:  
{  
  "CallPlacedTo": "Caller 1"  
}
```

Response parameters

CallPlacedTo: The name of the caller to whom the call is placed.

Note: If the softphone mentions, during registration that the "Redial" functionality is not supported, this message will not be sent to the softphone.

EPDS

Off-hook from headset

If the user does an Off-Hook event on the headset, the SDK will send the following event

Event

```
{
  "Event": "OffHook",
  "HeadsetPath": "PID_0043"
}
```

Event parameters

Event: Name of the event, here "OffHook".

HeadsetPath: The value of this field indicates the full PID (Product ID) of the device on which off-hook button is pressed.

When the user does an "Off-hook" event, it is expected that the user should be able to hear the dial-tone on the headset. This can be achieved by placing a call with no caller information.

EPoS

On-hook from headset

If the previous sent Off-Hook event fails on the headset or cannot be executed, the SDK will send the following event information.

Event

```
{  
  "Event": "OnHook"  
}
```

Event parameters

Event: Name of the event, here "OnHook"

EPOS

Send Presence status from softphone

If there is a presence status change in softphone, the following request must be sent to the SDK and the acknowledgement will follow. This is required for EPOS USB Busylight devices.

Request

```
URL: {baseurl}/CallController/{softphone_id}/SetPresence
Method: POST
Data:
{
  "Event": "SetPresence",
  "State": "Available"
}
```

Parameters:

Event: Name of the event, here "SetPresence"

State: The string containing the Presence status. The list of possible values are Available, Away, Busy, DoNotDisturb and IMNotification

EPOS

Send User Defined Custom Presence status from softphone

If there is a presence status change in softphone, the following request must be sent to the SDK and the acknowledgement will follow. This is for the EPOS USB Busylight devices.

Request

```
URL: {baseUrl}/CallController/{softphone_id}/SetPresence
Method: POST
Data:
{
  "Event": "SetPresence",
  "State": "presenceName, presenceColor , presencePattern"
}
```

Parameters:

Event: Name of the event, here "SetPresence"
State: The string containing the Presence status is the combination of presenceName, presenceColor, presencePattern. Sets a User Defined Custom Presence State on the Busylight device.
presenceName: Expected value is any string. This parameter has no validation; it can also be left empty.
presenceColor: Expected value is of the format "#RRGGBB".
presencePattern: Expected value is {0,1,2,3,4}:
0 -- Busylight will not Glow
1 -- Busylight will continuously Glow
2 -- Busylight will Flash
3 -- Busylight will Pulse (Breathing Light effect)
4 -- Busylight will glow with a Long Flash Pattern

Presenting Status

Sets the Busylight device to Presenting state in an Active Call state if set to true.

Event

```
{
  "Event": "Presenting",
  "PresentingStatus": true
}
```

Event parameters

Event: Name of the event, here "Presenting".
PresentingStatus: Presenting (true/false)

EPDS

Enable Logging (Deprecated)

This request is deprecated and should not be used.

It is possible to generate logs of the messages and events that happen in the SDK. The following message needs to be sent to enable logging:

Request

```
URL: {baseurl}/CallController/{softphone_id}/Logging
Method: POST
Data:
{
  "LogFileName": "Path of log file"
}
```

Request parameters

LogFileName: The fully qualified path for the log file. If this is blank, the logs will not be generated.

EPoS

Disable Logging (Deprecated)

This request is deprecated and should not be used.

The following request must be sent when the user wants to stop logging:

Request

URL: [{baseurl}/CallController/{softphone_id}/Logging](#)
Method: DELETE

EPDS

Mute from softphone

If the call on the softphone is muted, the following request must be sent to the SDK, so that the headset can also be put in mute state:

Request

```
URL: {baseUrl}/CallController/{softphone_id}/Mute
Method: PUT
Data:
{
  "Event": "MuteHeadset"
}
```

Request parameters

Event: Name of the event, here "MuteHeadset"

EPDS

Mute from headset

If the user presses the mute button on the headset, the following message send to the softphone:

Event

```
{  
  "Event": "MuteSoftphone"  
}
```

Event parameters

Event: Name of the event, here "MuteSoftphone"

Acknowledgement

```
URL: {baseurl}/CallController/{softphone_id}/CallEventsAck  
Method: PUT  
Data:  
{  
  "Event": "MuteSoftphoneAck"  
}
```

Acknowledgement parameters

Event: Name of the event, here "MuteSoftphoneAck".

Note: If the softphone mentions during registration that "Mute" functionality is not supported, the SDK will ignore this message

EPoS

Unmute from softphone

If the user un-mutes the call from the softphone UI, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/Mute
Method: PUT
Data:
{
  "Event": "UnmuteHeadset"
}
```

Request parameters

Event: Name of the event, here "UnmuteHeadset".

EPDS

Unmute from headset

If the user presses the unmute button on the headset, the following call event is sent to the softphone:

Event

```
{  
  "Event": "UnmuteSoftphone"  
}
```

Event parameters

Event: Name of the event, here "UnmuteSoftphone"

Acknowledgement

```
URL: {baseUrl}/CallController/{softphone_id}/CallEventsAck  
Method: PUT  
Data:  
{  
  "Event": "UnmuteSoftphoneAck"  
}
```

Acknowledgement parameters

Event: Name of the event, here "MuteSoftphoneAck".

EPDS

Get softphone mute status

If the SDK wants to know current mute status of the softphone the following call event is sent to the softphone

Event

```
{  
  "Event": "GetSPMuteStatus"  
}
```

Event parameters

Event: Name of the event, here "GetSPMuteStatus"

Response

```
URL: {baseurl}/CallController/{softphone_id}/MuteStatus  
Method: PUT  
Data:  
{  
  "Mute": "Yes"  
}
```

Response parameters

Mute: 'Yes' if the softphone is Muted else 'No'. If the softphone does not support mute, please state 'No'.

EPDS

Fail a call

If the call fails on the softphone due to known or unknown reason, the following request must be sent:

Request

```
URL: {baseurl}/CallController/{softphone_id}/CallFailed
Method: PUT
Data:
{
  "CallID": "1"
  "Reason": "Network connection lost"
}
```

Request parameters

CallID: The unique ID for the call.

Reason: The reason for call fail.

NOTE: If the call failing is a conference call, please supply the conference ID in the CallID parameter.

EPDS

System Information

This API is to get information about the SDK like product version, product name and product build version. To get that details the following request must be sent:

Request

```
{  
  "Event": "SystemInformation"  
}
```

Request parameters

Event: Name of the event, here " SystemInformation "

Response

```
URL: {baseUrl}/CallController/{softphone_id}/SystemInformation  
Method: PUT  
Data:  
{  
  "Event": "SystemInformation",  
  "ProductName": "DSEASDK",  
  "ProductVersion": "10.4.0.7055"  
}
```

Response parameters

Event: Name of the event, here "SystemInformation"
ProductName: The name of the SDK.
ProductVersion: The version of the SDK.

EPoS

HTTP status Codes

This are the status / error code that will be send when you make any of the above request

HTTP_OK	= 200,
HTTP_NO_CONTENT	= 204,
HTTP_BAD_REQUEST	= 400,
HTTP_UNAUTHORIZED	= 401,
HTTP_NOT_FOUND	= 404,
HTTP_METHOD_NOT_ALLOWED	= 405,
HTTP_NOT_ACCEPTABLE	= 406,
HTTP_UNSUPPORTEDMEDIATYPE	= 415,
HTTP_INTERNAL_SERVER_ERROR	= 500,
HTTP_NOT_IMPLEMENTED	= 501,
HTTP_SERVICE_UNAVAILABLE	= 503,
HTTP_VERSION_NOT_SUPPORTED	= 505

EPoS

Application specific error codes

The following is the list of all the possible error codes that might be returned.

- 0 = NoError,
- 1 = DefaultError
- 2 = InvalidSoftphoneID,
- 3 = InvalidSoftphone,
- 4 = DuplicteSoftphone,
- 5 = SoftPhoneNotLoggedIn,
- 6 = InvalidCallID,
- 7 = InvalidCallState,
- 8 = DuplicateCallID,
- 9 = DuplicateSoftphoneName,
- 10 = InsufficientData,
- 11 = InvalidJsonMessage,
- 12 = RestHandlerNotSet,
- 13 = RedialFailed,
- 14 = OffHookFailed,
- 15 = InvalidRequestID.