

## **TeamCall Telephony Access Platform**

- TeamCall CSTA Server**

- TeamCall Express**

## **STLI Reference Guide for Developers**

TeamCall is a registered trademark of ilink Kommunikationssysteme GmbH (ilink).

All other trademarks, service marks, and trade names mentioned in this publication are owned by their respective owners.

While the information in this document is believed to be accurate, ilink makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

ilink shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance or use of this document.

#### COPYRIGHT NOTICE

No part of this document may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of ilink. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. ilink assumes no responsibility for errors or omissions contained in this document. This document and the features described herein are subject to change without notice.

Copyright © 2000-2021, 2025 by ilink Kommunikationssysteme GmbH. All rights reserved.

This manual describes STLI for TeamCall CSTA Server version 3.x / 5.0 and for TeamCall Express 2.  
November 2025.

ilink Kommunikationssysteme GmbH  
Grazer Strasse 10  
30519 Hannover, Germany  
[www.ilink.de](http://www.ilink.de)

# Contents

## Overview ..... 6

Document conventions .....	6
Connecting to the server .....	7
STLI version 2.....	7
Using STLI version 2.....	8
Additional information in STLI version 2 .....	8
Information about devices .....	9
Basic requirements for communication with the CTI Server .....	10

## Functional index of STLI requests..... 11

Monitoring a device .....	11
MonitorStart.....	11
MonitorStop .....	12
Setting up a simple connection between two devices.....	12
MakeCall .....	12
AnswerCall .....	13
ClearConnection .....	14
Setting up a telephone conference.....	14
ConsultationCall .....	14
ConferenceCall .....	15
Alternating calls .....	15
AlternateCall .....	15
Placing a call on hold .....	16
HoldCall .....	16
Reconnecting a call on hold .....	16
ReconnectCall .....	16
Retrieving a call on hold .....	17
RetrieveCall.....	17
Transferring a call .....	17
TransferCall.....	17
SingleStepTransfer .....	18
Diverting a call.....	18
DeflectCall .....	18
PickupCall .....	19
GroupPickupCall .....	19
Forwarding a call.....	20
SetForwarding .....	20
Completing a call.....	21
CallBack.....	21
CampOn .....	21
Intrude .....	22
Setting device features for connected devices .....	22

SetDeviceFeature.....	22
Setting agent states for agents logged to an ACD in the phone system .....	23
SetAgentState .....	23
Requesting agent state and device feature states.....	25
QueryDevice .....	25
Clearing a call .....	26
ClearCall.....	26
Call associated features.....	26
AssociateData .....	26
SendDTMFTones.....	28
Other requests .....	28
BYE .....	28
GetVersion .....	28

## **Events and Event Reports ..... 29**

Call events.....	29
Marking of incoming external calls .....	29
Feature events .....	30
Agent state events .....	30
Maintenance events.....	30
BackInService .....	31
OutOfService .....	31
List of events and event reports.....	31
AgentBusy.....	31
AutoAnswer.....	31
Conferenced .....	32
ConnectionCleared.....	32
Delivered.....	33
Diverted .....	33
DoNotDisturb .....	33
Established .....	34
Failed.....	34
Forwarding .....	35
Held.....	35
Initiated .....	35
LoggedOn .....	35
LoggedOff .....	35
MessageWaiting.....	36
MicrophoneMute .....	36
NetworkReached.....	36
NotReady .....	36
Originated .....	36
Queued .....	37
Ready .....	37
Retrieved.....	37
SpeakerMute .....	37
SpeakerVolume .....	37
Transferred .....	38
WorkingAfterCall .....	38
WorkNotReady .....	38
WorkReady .....	38

## **Error messages..... 39**

ILINK-INTERNAL DEMOVERSION .....	39
INVALCMD .....	39
INVALIDAGENTSTATE .....	39
INVALIDDEVICEFEATURE .....	39
INVALIDFORWARDINGFEATURE .....	39
INVALNUMPARAM .....	40
INVALPARAM .....	40
LINKOUTOFSERVICE .....	40
NOCALL .....	40
NOCONNECTION .....	40
NODEVICE .....	40
NOMONITOR .....	41
NOMULTIPLEINSTANCE .....	41
PENDINGREQUEST .....	41
SUCCESS .....	41
UNAVAILABLEREQUEST .....	41

## **STLI Best Practices..... 42**

Heartbeat .....	42
Identifying the ConnectionCleared event of the remote call connection .....	42
Phone number formats in STLI requests .....	43
The local extension number .....	43
Internal extension targets.....	43
External phone number targets .....	44
Local phone number targets .....	44
Long-distance and mobile phone number targets.....	45
International phone number targets .....	45
Using STLI with Avaya Definity and Unify OpenScape Voice .....	46
Controlling Devices .....	46
STLI Requests .....	46
STLI Events .....	47
Examples of an outbound call .....	47
Example for a typical phone system .....	47
Example for a Unify OpenScape Voice .....	48
Examples of an inbound call.....	49
Example for a typical phone system .....	49
Example for a Unify OpenScape Voice .....	49
Practical advice for STLI on Definity or OpenScape Voice .....	50
Configuration of the switch type.....	50
Configuration of extension numbers.....	50
Rewriting of target phone numbers in various STLI requests .....	50
Evaluating STLI events .....	52
Working around OpenScape Voice signaling issues .....	52

# 1

## Overview

The STLI interface (Simple Telephony Interface) is a CTI application programming interface of ilink's CTI middleware servers TeamCall CSTA Server and TeamCall Express.

**TeamCall CSTA Server** offers STLI v1 and v2 in addition to its other APIs (CSTA ASN.1, CSTA/XML, JTAPI, and TAPI).

**TeamCall Express** offers STLI v1.

STLI is based on CSTA (Computer Supported Telecommunication Applications). CSTA is an industry standard, described by ECMA, an international Europe-based industry association founded in 1961 and dedicated to the standardization of information and communication systems. For more information visit their web site at [www.ecma.ch](http://www.ecma.ch).

STLI allows the user to control calls and monitor devices within a phone system using a network connection.

Being a human readable plaintext API, it offers the application programmer an easy method to implement complex CTI applications.

This document describes all STLI requests, responses, events and errors.

## Document conventions

This manual uses the following typographical conventions:

The `monospace font` in body text represents portions of code and names of items, commands, requests and keywords.

Whole paragraphs in `monospace font` represent samples of code. In examples, user input is formatted in **bold type**, on-screen output in `standard fontstyle`.

Variables are placed in `<>`.

Optional parameters are placed in `[]`.

Mutually exclusive parameter options are separated by `|`.

# Connecting to the server

The CTI server provides a TCP port for applications to connect to. The port number is defined in the main configuration file of TeamCall CSTA Server and set in the configuration GUI of TeamCall Express. In both servers, the default value is 26535.

An application would open a TCP connection to this port.

On the command line, you could use programs like `telnet` on Windows and Linux or `nc` on macOS and Linux.

When the connection has been established, an STLI session can be started via the `STLI` command.

**NOTE (Windows):** Please note that the Windows command line does not echo this first command, so you'll have to type it blindly if you use the Windows command line.

The STLI session allows an application (or a command line user) to make call control requests and receive call control events for devices of the phone system that TeamCall connects to.

The server responds with a return code ("`error_ind`") indicating success.

## Example

Starting an STLI session:

```
telnet 127.0.0.1 26535
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
STLI
error_ind SUCCESS STLI
```

## STLI version 2

In addition to the standard STLI (version 1) **TeamCall CSTA Server** also offers STLI version 2. It only differs from the standard STLI concerning the information signaled in some events. In the section *Events and Event Reports* on page 28 you will find descriptions for STLI V1 and STLI V2 for the concerning events. These are:

Conferenced, ConnectionCleared, Delivered, Diverted, Established, Failed, Held, Initiated, NetworkReached, Originated, Queued, Retrieved and Transferred.

Please note that **TeamCall Express** does not offer STLI v2.

## Using STLI version 2

(only supported on TeamCall CSTA Server, not on TeamCall Express).

To start an STLI V2 session, use `STLI;Version=2` instead of just `STLI` after having connected to TeamCall CSTA Server.

It is also possible to toggle between STLI V1 and STLI V2.

The following request switches to STLI version 2 signaling from within an STLI session:

```
STLI;Version=2
```

---

**Note:** STLI version 2 is now active for the current session. Other sessions are not affected.

---

In order to switch back to STLI version 1 (standard) signaling, enter the following command:

```
STLI;Version=1
```

## Additional information in STLI version 2

The following information is additionally signaled in STLI V2:

`call identifier (<callId>)`

This information represents the call identifier of a connection, which is generated by the phone system. It is given in hexadecimal notation. If no call identifier information is given by the phone system, an empty string "" is signaled.

`unique device identifier (<deviceId>)`

This information represents the unique device identifier of a connection. This identifier is given by the phone system.

There are three types of identifiers:

- **dynamic id:** identified by the prefix "\$"; e.g., "\$00040100"
- **logical id:** identified by the prefix "#"; e.g., "#-2130640809"
- **device number:** no prefix; e.g., "7801"

Which type is used depends on the phone system; furthermore, each phone system supports its own notation.

Generally, this device identifier is unique within the phone system. I.e., no other device (internal or external) is associated with this unique device identifier. This unique device identifier is especially helpful when trying to track an external device (i.e., outside the scope of the phone system) which did not transmit its dialing number (i.e., phone number).

Some phone systems only assign a unique device identifier if the associated device is an external device (i.e., outside the scope of the phone system). Otherwise, the internal dialing number is signaled, which is unique within the phone system.

If no unique device identifier information is given by the phone system, an empty string "" is signaled.



# Information about devices

After every call event, a status message about the concerned device is returned within the STLI session.

```
DeviceInformation <DeviceId> <# of calls>
([<CallId>:<state>[,<CallId>:<state>]])
```

This message shows the number of calls which are connected to the device (Not the number of the parties to which the device is connected to.) The calls and their status are displayed in the following example:

## Example

### **MakeCall 111 4100**

```
error_ind SUCCESS MakeCall
Initiated 111 makeCall
DeviceInformation 111 1 (01e6:initiate)
Originated 111 newCall 111 4100 ""
DeviceInformation 111 1 (01e6:connect)
Delivered 111 newCall 4100 111 4100 ""
DeviceInformation 111 1 (01e6:connect)
Established 111 newCall 4100 111 4100 ""
DeviceInformation 111 1 (01e6:connect)
ConnectionCleared 111 normalClearing 111 {}
DeviceInformation 111 0 ()
```

To switch this off, you can enter the following command in a STLI session at any time (please note exact spelling)

**STLI;DeviceInformation=Off**

This suppresses the status messages for the current session. A status message can be sent asynchronously to a device, if the internal recovery mechanism is activated and the condition of a device was changed without receiving a call event.

The standard status message can be switched on at any time using the following statement:

**STLI;DeviceInformation=Standard**

This is switched on by default when a STLI session is started.

Additionally, the extended version of the status message can be switched on at any time using the following statement:

**STLI;DeviceInformation=Extended**

This also displays the uniqueId (dynamicId or deviceNumber or trunkId).

```
DeviceInformation <DeviceId>/<own uniqueId> <# of calls>
([<CallId>:<state>/<uniqueId of the
party>[,<CallId>:<state>]]/<uniqueId of the party>)
```

## Example

### **STLI;DeviceInformation=Extended**

```
error_ind SUCCESS STLI DeviceInformation "Extended"
MakeCall 111 4100
error_ind SUCCESS MakeCall
Initiated 111 makeCall
DeviceInformation 111/$00010100 1 (067b:initiate)
Originated 111 newCall 111 4100 ""
DeviceInformation 111/$00010100 1 (067b:connect)
Delivered 111 newCall 4100 111 4100 ""
DeviceInformation 111/$00010100 1 (067b:connect/$013c0100)
Established 111 newCall 4100 111 4100 ""
```

```
DeviceInformation 111/$00010100 1 (067b:connect/$013c0100)
ConnectionCleared 111 normalClearing 4100 {111}
DeviceInformation 111/$00010100 1 (067b:connect)
ConnectionCleared 111 normalClearing 111 {}
DeviceInformation 111/$00010100 0 ()
```

---

**Note:** The <uniqueId of the party> can only be displayed if it exists. For example, after a ConnectionCleared event no party is present.

---

## Basic requirements for communication with the CTI Server

A device can be monitored and a device can receive requests. Synchronous messages (responses or acknowledgements) and asynchronous messages (events) will occur, depending on the phone system's or a device's state.

All following descriptions are seen from the point where the CTI server has been started and is running.

```
telnet <CTI server host> <API port of CTI server>
```

establishes a telnet session connected to the running server as shown in the following example:

```
telnet ctihost 26535
Trying x.x.x.x...
Connected to x.x.x.x.
Escape character is '^]'.
```

Finally, the telnet session with the CTI server will be closed by typing

```
BYE
error_ind SUCCESS BYE
Connection closed by foreign host
```

Using this user interface, devices connected to the phone system can either be controlled using requests or can be monitored, which means, after setting/unsetting a monitoring point using the MonitorStop and MonitorStop requests, any change in the state of a monitored device will be reported.

Any following device number must only contain digits (i.e., 0..9). The number may not contain any blanks. Otherwise it will be interpreted as two numbers. Local device numbers are a representation of the full internal name. External device numbers must be preceded by the prefix used to place external calls ("operator"; e.g., 0 or 9).

Requests received by the phone system are handled as first come, first served, so any acknowledgement from the phone system is related to the oldest not acknowledged request.

For each request you will get a response of the type "error\_ind <parameters>". Please refer to section *Error messages* page 39.

# 2

## Functional index of STLI requests

This section describes the available requests in a functional order, grouping requests by actions like establishing and closing a connection, making a conference etc.

In general, events, indications and all other messages from TeamCall may provide a device number or another identifier for an internal or external device. If there is an incoming external call with no caller identification TeamCall will provide an empty string "".

As a service response, the server will always provide an acknowledgement after issuing a request. Positive responses are listed in the following section. Negative responses always include the CSTA error value. For more information about CSTA specific error messages visit the web site of ECMA at [www.ecma.ch](http://www.ecma.ch).

You can find a list of ilink specific error messages at the end of this chapter.

### Monitoring a device

A `MonitorStart` request creates an instance, that reports any change of the device's state, until it is ended by a `MonitorStop` request. For every change in the device's state, an event report will be generated.

---

**Note:** It may not be possible to monitor all internal devices. Restrictions may apply to groups, trunks, etc. Please refer to your phone system manual for more details.

---

#### MonitorStart

<b>Synopsis</b>	<code>MonitorStart &lt;localDevice&gt;</code>
<b>Parameter</b>	<code>&lt;localDevice&gt;</code> indicates the local device to be monitored.
<b>Example</b>	<b>MonitorStart 12345</b>
<b>Positive response</b>	<code>error_ind SUCCESS MonitorStart &lt;localDevice&gt;</code> is valid and monitoring has been started successfully.
<b>Description</b>	<p>The Monitor Start service enables the receipt of event reports for a local device. With <code>MonitorStart</code> you start monitoring a local device.</p> <p>All events reported for this monitor will be identified by the local device number of the monitored device.</p>

Only local devices can be monitored. Attempting to start monitoring on other devices like non-existing or external devices will cause an error message. The acknowledgement to a `MonitorStart` request does not contain a reference to the monitored device. When sending a couple of `MonitorStart` requests without waiting for an answer, the first `error_ind <error value> MonitorStart` response is related to the first previous not answered `MonitorStart` request.

Service termination can result from a client request (`MonitorStop`) or it can be initiated by TeamCall.

By using the `MonitorStart` function, an event report will be generated each time the call state, feature state or agent state of the monitored device/agent changes. Therefore, the event reports might be call events, feature events, agent state events or maintenance events.

## MonitorStop

<b>Synopsis</b>	<code>MonitorStop &lt;localMonitoredDevice&gt;</code>
<b>Parameter</b>	<code>&lt;localMonitoredDevice&gt;</code> indicates the currently monitored local device, where to stop monitoring.
<b>Example</b>	<b><code>MonitorStop 12345</code></b>
<b>Positive response</b>	<code>error_ind SUCCESS MonitorStop &lt;localMonitoredDevice&gt;</code> will no longer be monitored. I.e., no more event reports will be sent.
<b>Description</b>	The Monitor Stop service is used to terminate a previously initiated Monitor Start service. With <code>MonitorStop</code> you stop monitoring a local device. The state changes transmitted by event reports will no longer be sent.

---

**Note:** If more than one session monitors the same device, the monitor stop will only affect the session which issued the request.

---

## Setting up a simple connection between two devices

A `MakeCall` request initiates the connection by ringing up the local device. Depending on the phone system the local device will either auto answer the ringing or an `AnswerCall` request must be sent or someone must answer (i.e., off hook) the phone to make the second device ring. After an `AnswerCall` request has been sent to the second device or it becomes off-hook, both devices get connected. When one of them gets a `ClearConnection` request or hangs up, both will be disconnected.

## MakeCall

<b>Synopsis</b>	<code>MakeCall &lt;callingDevice&gt; &lt;calledDirectoryNumber&gt; [prompt   doNotPrompt]</code>
<b>Parameters</b>	<p><code>&lt;callingDevice&gt;</code> indicates the device from which the call originates. <code>&lt;callingDevice&gt;</code> must be a valid local device.</p> <p><code>&lt;calledDirectoryNumber&gt;</code> indicates the device to which the call should be directed. <code>&lt;calledDirectoryNumber&gt;</code> can be a local or an external device.</p>

The optional parameters `prompt` and `doNotPrompt` are mutually exclusive.

`prompt` means that the calling device will always ring first and that the outbound call will only be made once the user has picked up the phone.

`doNotPrompt` means that outbound calls will be made immediately, while the calling phone enters hands-free mode.

If no optional parameter is being used, the phone system's default behavior will be used. The CSTA specification states that the default shall be `prompt`, but this can often be changed in the phone system configuration.

Please note that `doNotPrompt` is not possible on all phone systems or for calls on all phone. E.g., this does not work on phones that do not support hands-free mode.

**Example** `MakeCall 12345 1111`

To connect a device in alerting state to the call (at `callingDevice` or `calledDevice`) an `AnswerCall` Request might be sent.

**Positive response** `error_ind SUCCESS MakeCall`

Both devices are valid; `callingDevice` will start to ring.

**Description** The Make Call service initiates a call between two devices and allows to set up a call between a calling device and a called device. Both device numbers have to be valid devices. Depending on the phone system type, after processing the `MakeCall` request, `callingDevice` will be either in alerting or connected state.

## AnswerCall

**Synopsis** `AnswerCall <calledLocalDevice>`

**Parameter** `<calledLocalDevice>` indicates the device which is presently called by CSTA. `<calledLocalDevice>` must be a valid local device.

**Example** `AnswerCall 12345`

**Positive response** `error_ind SUCCESS AnswerCall`

The call has been answered successfully by `<calledLocalDevice>`. Now `<calledLocalDevice>` is off-hook.

**Description** The Answer Call service connects an alerting call at a called local device.

A device must receive an incoming call to be able to issue an `AnswerCall` request. This service is typically associated with devices, that have attached speakerphone units and headset telephones, in order to connect to a call via hands free operation. For example, when the call is answered, one of the following actions may occur:

- If the specified device has a speaker and a microphone, the speaker and the microphone are turned on.
- If the specified device only has a speaker, the speaker is turned on. The handset has to be picked up in order to have a two-way conversation.
- If there is no speaker, then the handset has to be picked up in order to have a two-way conversation.
- If the specified device has a headset, the headset is turned on.

## ClearConnection

<b>Synopsis</b>	<code>ClearConnection &lt;localConnectedDevice&gt;</code>
<b>Parameter</b>	<code>&lt;localConnectedDevice&gt;</code> indicates the device to be disconnected. <code>&lt;localConnectedDevice&gt;</code> must be a valid local device.
<b>Example</b>	<code>ClearConnection 12345</code>
<b>Positive response</b>	<code>error_ind SUCCESS ClearConnection</code> <code>&lt;localConnectedDevice&gt;</code> has been disconnected.
<b>Description</b>	<code>ClearConnection</code> clears an existing connection. The Clear Connection service releases a specified device from a designated call.  This service releases the specified connection and CSTA Connection Identifier from the designated call. The result is the same, as if the device had hung up on the call. Note that the device might not be physically returned to on-hook and this might result in silence, a dial tone, or some other condition. Generally, if only two connections are in the call, the effect of <code>ClearConnection</code> is the same as that of <code>ClearCall</code> .

## Setting up a telephone conference

For set up, a connection between two devices or a telephone conference must have been established before. From this connection one device initiates a `ConsultationCall` to a new device. Meanwhile, the other connected devices will be placed on hold. When the connection to the new device has been established, a `ConferenceCall` retrieves the other devices from the hold state and establishes the telephone conference.

## ConsultationCall

<b>Synopsis</b>	<code>ConsultationCall &lt;callingDevice&gt; &lt;consultedDirectoryNumber&gt;</code> <code>[&lt;consultationMode&gt;]</code>
<b>Parameters</b>	<code>&lt;callingDevice&gt;</code> indicates the device from which the consultation call originates. <code>&lt;callingDevice&gt;</code> must be a valid local device.  <code>&lt;consultedDirectoryNumber&gt;</code> indicates the device to which the consultation call should be made. <code>&lt;calledDirectoryNumber&gt;</code> can be a local or an external device.  The optional parameter <code>&lt;consultationMode&gt;</code> is used for phone systems like the Nortel Meridian which require to specify the followup action at the time when the consultation call will be made.  These consultation modes are available: <ul style="list-style-type: none"><li>• <code>conferenceOnly</code> The consultation call can be turned into a conference call including the held call (via <code>ConferenceCall</code>) or it can be terminated. But the held call cannot be transferred to the consulted party (this is a phone system restriction).</li><li>• <code>transferOnly</code> From within the consultation call, the held call can be transferred to the consulted party (via <code>TransferCall</code>), or the consultation call can be terminated. It cannot however be turned into a conference call including the held call (this is a phone system restriction).</li></ul>

- `consultOnly`  
The consultation call can neither be turned into a conference call including the held call, nor can the held call be transferred to the consulted party (this is a phone system restriction).

This parameter will only be used for those phone systems that require it.

<b>Example</b>	<code>ConsultationCall 12345 1111</code>
<b>Positive response</b>	<code>error_ind SUCCESS ConsultationCall</code> ConsultationCall has been initiated successfully.
<b>Description</b>	The Consultation Call service places an existing active call at a device on hold and initiates a new call from the same device. Both device numbers have to be valid devices. <code>&lt;consultedDirectoryNumber&gt;</code> will ring. If <code>&lt;consultedDirectoryNumber&gt;</code> answers the call, a connection will be established. The second device from the previous call is still on hold.

## ConferenceCall

<b>Synopsis</b>	<code>ConferenceCall &lt;callingDevice&gt;</code>
<b>Parameter</b>	<code>&lt;callingDevice&gt;</code> indicates the device which originates the conference. <code>&lt;callingDevice&gt;</code> must be a valid local device.
<b>Example</b>	<code>ConferenceCall 12345</code>
<b>Positive response</b>	<code>error_ind SUCCESS ConferenceCall</code> The telephone conference has been established successfully. All devices are connected.
<b>Description</b>	The Conference Call service creates a conference between an existing held call or held conference and another active call at a conferencing device. The two calls at the conferencing device will be resolved into a single call.

## Alternating calls

The Alternate Call service places an existing active call on hold and then retrieves a previously held call.

At this point, a connection must already have been established and another call must have been placed on hold.

## AlternateCall

<b>Synopsis</b>	<code>AlternateCall &lt;callingDevice&gt;</code>
<b>Parameter</b>	<code>&lt;callingDevice&gt;</code> indicates the device which originated the call alternation. <code>&lt;callingDevice&gt;</code> must be a valid local device.
<b>Example</b>	<code>AlternateCall 12345</code>
<b>Positive response</b>	<code>error_ind SUCCESS AlternateCall</code> Alternation done. The hold device is reconnected to the origin device, the previous connected device is placed on hold.
<b>Description</b>	The Alternate Call service places an existing active call on hold and then retrieves a previously held call.

## Placing a call on hold

The Hold Call service places an existing active call (connected call) on hold.

### HoldCall

<b>Synopsis</b>	<code>HoldCall &lt;callingDevice&gt;</code>
<b>Parameter</b>	<code>&lt;callingDevice&gt;</code> indicates the device which initiated the hold.
<b>Example</b>	<code>HoldCall 12345</code>
<b>Positive response</b>	<code>error_ind SUCCESS HoldCall</code> The device, connected to <code>&lt;callingDevice&gt;</code> , has been successfully placed on hold.
<b>Description</b>	The Hold Call service places a connected connection into the hold state. This service interrupts communication for an existing call at a device.

## Reconnecting a call on hold

The Reconnect Call service clears a specified connection at the reconnecting device and retrieves a specified held connection at the same device.

### ReconnectCall

<b>Synopsis</b>	<code>ReconnectCall &lt;callingDevice&gt;</code>
<b>Parameter</b>	<code>&lt;callingDevice&gt;</code> indicates the device which originates the reconnect. <code>&lt;callingDevice&gt;</code> must be a valid local device. <code>&lt;callingDevice&gt;</code> must have at least one connection on hold.
<b>Example</b>	<code>ReconnectCall 12345</code>
<b>Positive response</b>	<code>error_ind SUCCESS ReconnectCall</code> Reconnection finished successfully. The previous connection has been cleared and the call on hold has been reconnected.
<b>Description</b>	The Reconnect Call service clears a specified connection at the reconnecting device and retrieves the held connection at the same device.



## Retrieving a call on hold

The Retrieve Call service connects a specified connection that is currently on hold.

### RetrieveCall

<b>Synopsis</b>	<code>RetrieveCall &lt;retrievingDevice&gt;</code>
<b>Parameter</b>	<code>&lt;retrievingDevice&gt;</code> indicates the device where to reconnect the held call to.
<b>Example</b>	<b>RetrieveCall 12345</b>
<b>Positive response</b>	<code>error_ind SUCCESS RetrieveCall</code> The previous call on hold has been successfully reconnected.
<b>Description</b>	The Retrieve Call service retrieves a held connection. <code>&lt;retrievingDevice&gt;</code> has to be in held state.

## Transferring a call

Transferring a call means passing a call to a different phone so that it can be continued there. There are two ways to transfer a call.

### TransferCall

<b>Synopsis</b>	<code>TransferCall &lt;transferringDevice&gt;</code>
<b>Parameter</b>	<code>&lt;transferringDevice&gt;</code> indicates the device which transfers the held call from itself to the remote device of the active call and disconnects itself.
<b>Example</b>	<b>TransferCall 12345</b>
<b>Positive response</b>	<code>error_ind SUCCESS TransferCall</code> Transferring has been completed successfully. The call has been transferred to the held call and <code>&lt;transferringDevice&gt;</code> has been disconnected from the call.
<b>Description</b>	The Transfer Call service transfers a call held at a device to an active (consultation) call at the same device. The held and active calls at the transferring device will be merged into a new call. The connections of the held and active calls at the transferring device will no longer be involved in the call.

## SingleStepTransfer

<b>Synopsis</b>	<code>SingleStepTransfer &lt;transferringDevice&gt; &lt;transferredToDevice&gt;</code>
<b>Parameters</b>	<code>&lt;transferringDevice&gt;</code> indicates the device which transfers an active call from itself to another device and disconnects itself.  <code>&lt;transferredToDevice&gt;</code> indicates the device to which the call is transferred.
<b>Example</b>	<b>SingleStepTransfer 12345 12346</b>
<b>Positive response</b>	<code>error_ind SUCCESS SingleStepTransfer</code> The transfer has been completed successfully. The call has been transferred to <code>&lt;transferredToDevice&gt;</code> and <code>&lt;transferredToDevice&gt;</code> has been disconnected from the call.
<b>Description</b>	The Single-Step-Transfer service transfers an established call at the transferring device to a new device. At the new device the call is in the alerting state. While the regular TransferCall service requires to create a consultation to the new device before completing the transfer, the SingleStepTransfer service does not require this. It transfers the call in a single step without setting up a consultation call.

## Diverting a call

Diverting a call means routing a call to a new destination. There are three different ways to divert a call.

---

**Note:** Depending on your phone system, restrictions may apply to this service. The use of this service may depend on the state of the deflecting device. E.g., if forwarding is enabled, if the call is alerting or queued, etc. Please refer to your phone system manual for more information.

---

## DeflectCall

<b>Synopsis</b>	<code>DeflectCall &lt;callToBeDeflected&gt; &lt;newDestination&gt;</code>
<b>Parameters</b>	<code>&lt;callToBeDeflected&gt;</code> indicates the device where to deflect the call. <code>&lt;callToBeDeflected&gt;</code> must be a valid local device.  <code>&lt;newDestination&gt;</code> indicates the destination, the call will be deflected to. <code>&lt;newDestination&gt;</code> can be a local or external device.
<b>Example</b>	<b>DeflectCall 12345 1111</b>
<b>Positive response</b>	<code>error_ind SUCCESS DeflectCall</code> Both devices are valid; the call has been successfully deflected.
<b>Description</b>	The Deflect Call service deflects a typically alerting call at a device and sends it to a new destination, that may be inside or outside the switching subdomain.

## PickupCall

<b>Synopsis</b>	<code>PickupCall &lt;callToBePickedUp&gt; &lt;requestingDevice&gt;</code>
<b>Parameters</b>	<code>&lt;callToBePickedUp&gt;</code> indicates the device where to deflect the call. <code>&lt;callToBePickedUp&gt;</code> must be a valid local device.  <code>&lt;requestingDevice&gt;</code> indicates the local device, which picks up the call. <code>&lt;requestingDevice&gt;</code> must be a valid local device.
<b>Example</b>	<b><code>PickupCall 12345 1111</code></b>  In opposition to <code>DeflectCall</code> , which notices the <code>&lt;newDestination&gt;</code> state, <code>PickupCall</code> connects to a new local device immediately. For the <code>&lt;newDestination&gt;</code> , all features such as <code>Forwarding</code> and <code>DoNotDisturb</code> for this device will be ignored when the call is being redirected to it.
<b>Positive response</b>	<code>error_ind SUCCESS PickupCall</code> The call has been successfully picked up.
<b>Description</b>	The Pickup Call service picks up an alerting or queued call at a local device and connects it to another destination inside the switching subdomain.

## GroupPickupCall

<b>Synopsis</b>	<code>GroupPickupCall &lt;requestingDevice&gt;</code>
<b>Parameter</b>	<code>&lt;requestingDevice&gt;</code> indicates the device which picks the call from a pickup group.
<b>Example</b>	<b><code>GroupPickupCall 12345</code></b>
<b>Positive response</b>	<code>error_ind SUCCESS GroupPickupCall</code> The call has been successfully picked up.
<b>Description</b>	The Group Pickup Call service diverts an alerting call at a device, which is a member of a specified or default pickup group, to the specified destination.

---

**Note:** The difference between this service and the Pickup Call service is that the Directed Pickup Call service specifies the actual connection to be picked up, but the Group Pickup Call service does not.

---

## Forwarding a call

Forwarding means automatically redirecting all incoming calls to a new destination. The Forwarding Call service allows to redirect all incoming calls to another destination that may be inside or outside the switching subdomain. Forwarding can be set immediately or depending on called devices state (busy or absent) and it can be set for internal only, external only or all calls.

The Forwarding service allows to set/unset different forwarding types.

---

**Note:** The phone system must support forwarding and the Set Forwarding service (which is a part of Set Device Feature service) to use this feature. Some phone systems support this request but do not send a feature event.

---

### SetForwarding

**Synopsis**      `SetForwarding <forwardingDevice> <forwardingType>`  
                 `[<newForwardingDestination>]`

**Parameters**   `<forwardingDevice>` indicates the device to be forwarded. `<forwardingDevice>` must be a valid local device.

`<forwardingType>` indicates the forwarding type.

Possible values for `<forwardingType>` are as follows:

```
forwardImmediateOn
forwardImmediateOff
forwardBusyOn
forwardBusyOff
forwardNoAnsOn
forwardNoAnsOff
forwardBusyIntOn
forwardBusyIntOff
forwardBusyExtOn
forwardBusyExtOff
forwardNoAnsIntOn
forwardNoAnsIntOff
forwardNoAnsExtOn
forwardNoAnsExtOff
forwardImmIntOn
forwardImmIntOff
forwardImmExtOn
forwardImmExtOff
```

`<newForwardingDestination>` indicates the new destination where to forward incoming calls to. If forwarding is set, `<newForwardingDestination>` must be specified. If forwarding is unset, `<newForwardingDestination>` must not be specified, as in

```
SetForwarding <forwardingDevice> forward*On
[<newForwardingDestination>] SetForwarding <forwardingDevice>
forward*Off
```

**Example**      `SetForwarding 111 forwardImmediateOn 1254`

**Positive response**   `error_ind SUCCESS SetForwarding`

**Description** `SetForwarding` allows forwarding of incoming calls at the device. Depending on the value of `<forwardingType>`, all incoming calls will be forwarded, or the phone system will check for the state of the called device and forward the calls depending on the result of the check.

## Completing a call

The Call Completion service invokes features which complete a call that may otherwise fail or terminate before being answered. Generally, this service is invoked when a call is set up and encounters a busy called device or no answer. There are three different types for completing a call. All three start from the point where a `MakeCall` failed because the called device is busy.

### CallBack

**Synopsis** `CallBack <callingDevice>`

**Parameter** `<callingDevice>` indicates the device which tried to call another device.

**Example** `CallBack 12345`

**Positive response** `error_ind SUCCESS CallBack`

**Description** The Call Back service requests the called device to return the call when the called device is in an appropriate state to accept the call or returns to idle. `CallBack` is similar to `CampOn`, but with `CallBack` the call may be hung up after the service is invoked. The CSTA switching function calls both parties, when the called device becomes available.

### CampOn

**Synopsis** `CampOn <callingDevice>`

**Parameter** `<callingDevice>` indicates the device which tries to call another device.

**Example** `CampOn 12345`

**Positive response** `error_ind SUCCESS CampOn`

**Description** The Camp On Call service allows to queue a connection for a device (which is typically busy), until that device becomes available (after finishing a current call or any previously queued calls, for example).

`CampOn` allows queueing for availability. Usually, `CampOn` makes the call wait until the called party finishes a current call and any previously camped calls.

To cancel a Camp On Call service

- issue the Clear Connection service or Clear Call service with the camp on connection or
- set the calling device on-hook.

Note that if the Camp On Call service is successfully cancelled, normal call progress messages of `ConnectionCleared` (with an event cause of normal clearing) will be generated.

## Intrude

**Synopsis** `Intrude <callingDevice>`

**Parameter** `<callingDevice>` indicates the device which tried to call another device.

**Example** `Intrude 12345`

**Positive response** `error_ind SUCCESS Intrude`

**Description** The Intrude Call service adds the calling device to a call at a busy called device. Depending upon the switching function, the result will be that the calling device is either actively or silently participating in the called device's existing call or consulting with the called device with a new call.

There are two cases specified for the Intrude Call service:

- Case A: The calling device (D1) joins call C2 with devices D2 and D3.
- Case B: The called device (D2) places its existing active call on hold first and then connects to the new calling device (D1).

To cancel an Intrude Call service,

- issue the Clear Connection or Clear Call service with the ConnectionID of either the calling device (cases A or B) or called device (case B) or
- set the calling device on-hook.

If the called device has more than one call during the execution of this service, the switching function will choose which call to intrude.

## Setting device features for connected devices

`SetDeviceFeature` modifies special features of devices connected to phone system like speaker or microphone mute, messaging etc.

The phone system must support the Set Device Feature service to use this service.

### SetDeviceFeature

**Synopsis** `SetDeviceFeature <deviceToBeSet> <aFeature> <...further parameters depending on the feature>`

**Parameters** `<deviceToBeSet>` Represents the device, where to set or modify features.

`<aFeature>` – Represents the feature to be set or modified.

Depending on the feature, there will be further parameters to be provided as arguments. It depends on the phone system and connected devices, whether all listed features can be set. Available features are:

`msgWaiting <On/Off>` – Allows to enable/disable messaging for specified device.

`doNotDisturb <On/Off>` – Allows to set/unset do not disturb state. Depending on the phone system and the connected devices, the result is, that the specified device is unreachable.

`enableRouting <On/Off>` – Determines, if route requests are made from a device. If enabled, in addition to route requests, made by the switching function, the device may request routes with a call in any state, including NULL.

`autoAnswer <On/Off>` – Determines, if an arriving call is answered automatically.

microphoneMute <On/Off> – Determines, if the microphone at specified devices is active or muted.

speakerMute <On/Off> – Determines, if the speaker at specified device is active or muted.

speakerVolume <[0..100]> – Determines the speaker volume for the specified device. The level to be set must be an integer within the range from 0 to 100.

---

**Note:** Some features are only available in CSTA Phase I.

---

<b>Examples</b>	<code>SetDeviceFeature 111 autoAnswer Off</code>
	<code>SetDeviceFeature 111 microphoneMute Off</code>
	<code>SetDeviceFeature 111 speakerMute Off</code>
	<code>SetDeviceFeature 111 speakerVolume 100</code>
<b>Positive response</b>	<code>error_ind SUCCESS SetDeviceFeature</code> <code>SetDeviceFeature was successful.</code>
<b>Description</b>	<code>SetDeviceFeature</code> allows to enable/disable or modify device specific settings like speaker level or microphone mute for specified devices. Some settings like <code>speakerMute</code> are executed directly at the device. Other settings like <code>forwarding</code> are set for a specific device, but not executed at the device itself, because call forwarding is done by the phone system.

## Setting agent states for agents logged to an ACD in the phone system

`SetAgentState` modifies states for agents at ACD like logging in and logging out, busy etc.

The phone system must support agents and the Set Agent State service to use this feature.

### SetAgentState

<b>Synopsis</b>	<code>SetAgentState &lt;deviceID/agentID&gt; &lt;requestedAgentState&gt; [&lt;agentID&gt; &lt;password&gt; &lt;group&gt;]</code>
<b>Parameters</b>	<code>&lt;deviceID/agentID&gt;</code> – Indicates the agent (i.e., the device associated to the agent), where to change the settings.  <code>&lt;requestedAgentState&gt;</code> – Indicates the agent state, the agent may take in relation to an ACD. It is possible, that an agent has several states with respect to different ACD devices. Alternatively, an agent may use a single state to describe its relationship to all ACD devices. Please refer to ECMA CSTA standards for further information. Agent states are reported in agent state reports.

Depending on whether the phone system supports CSTA Phase I or Phase II, there are different values available for `<requestedAgentState>`.

- Values for both Phase I and Phase II:

`loggedIn`  
`loggedOut`  
`notReady`  
`ready`

- CSTA Phase I:

`workNotReady`  
`workReady`

- CSTA Phase II:

`busy`  
`workingAfterCall`

`<agentID>` – Represents the agent identifier. If `<requestedAgentState>` is `loggedIn` or `loggedOut`, `<agentID>` must be specified. An agent identifier is allocated to each agent by the switching function, when each agent becomes visible across the CSTA service boundary for the first time. It may or may not be identical to the device identifier of the device used by the agent.

`<password>` – Represents the agent password, associated to each agent. If `<requestedAgentState>` is `loggedIn` or `loggedOut`, `<password>` must be specified. The password authenticates the agent to the CSTA application and/or one or more of its component functions.

If no group is specified as the last parameter, the password is optional.

But a password must be given if a group is specified (because the group is expected to be provided as the fifth parameter).

If you need to specify a password but the agent does not have one, the value `NO_PASSWORD` can be used.

`<group>` – Represents the group, the agent is related to. If `<requestedAgentState>` is `loggedIn` or `loggedOut`, `<group>` must be specified. This parameter specifies the ACD group, into which the agent is logging on.

<b>Example</b>	<code>SetAgentState 111 busy</code>
<b>Positive response</b>	<code>error_ind SUCCESS SetAgentState</code>
<b>Description</b>	<code>SetAgentState</code> allows to change settings for agents at an ACD. To use this service, the phone system must support agents.



## Requesting agent state and device feature states

QueryDevice allows to retrieve the last received events/settings for AgentState, DeviceFeatureState and Forwarding.

The phone system must support agent state event monitoring and device feature monitoring to use this feature.

### QueryDevice

**Synopsis** QueryDevice <localDevice> <feature>

**Parameter** <localDevice> indicates the local device that shall be queried.

<feature> indicates the feature which shall be queried.

**Features for CSTA I:** msgWaiting, doNotDisturb, forward, lastDialedNumber, deviceInfo, agentState

**Features for CSTA II:** msgWaiting, doNotDisturb, forward, deviceInfo, agentState, routing, autoAnswer, microphoneMute, speakerMute, speakerVolume

**Positive response** error\_ind SUCCESS QueryDevice <feature> <parameter>

**Examples**

**QueryDevice 111 deviceInfo**  
error\_ind SUCCESS QueryDevice deviceInfo deviceType=0 deviceClass=-1

**QueryDevice 111 lastDialedNumber**  
error\_ind UniversalFailure operationalError  
requestIncompatibleWithObject

**QueryDevice 100 forward**  
error\_ind SUCCESS QueryDevice forward forwardImmediateOff

**QueryDevice 100 forward**  
error\_ind SUCCESS QueryDevice forward forwardImmediateOn 103

**QueryDevice 100 doNotDisturb**  
error\_ind SUCCESS QueryDevice doNotDisturb Off

**QueryDevice 100 msgWaiting**  
error\_ind SUCCESS QueryDevice msgWaiting On

---

**Note:** Not all phone systems send agent state or feature events, although the feature requests may be supported.

---

**Description** The Query Device service provides information about the state of agents and device features, associated with a given device. If the underlying phone system does not support agent events or feature events, the state is not traced.

## Clearing a call

The Clear Call service releases all devices from an existing call.

### ClearCall

<b>Synopsis</b>	<code>ClearCall &lt;callToBeCleared&gt;</code>
<b>Parameter</b>	<code>&lt;callToBeCleared&gt;</code> indicates the call to be closed by releasing all devices.
<b>Example</b>	<code>ClearCall 12345</code>
<b>Positive response</b>	<code>error_ind SUCCESS ClearCall</code> The call has been cleared.
<b>Description</b>	The Clear Call service releases all devices from an existing call. In case of a conference call, all devices in the conference call will be removed from the call.

## Call associated features

Using `AssociateData`, user specified data can be appended to an active call.  
`SendDTMFTones` allows to send DTMF tones to a connection on a specific device.

### AssociateData

<b>Synopsis</b>	<code>AssociateData &lt;device&gt; &lt;data&gt;</code>
<b>Parameters</b>	<code>&lt;device&gt;</code> indicates the device.  <code>&lt;data&gt;</code> can contain any character except control sequences. The following characters are interpreted as delimiters and are substituted with <code>&lt;Space&gt;</code> : <code>'</code> , <code>&lt;TAB&gt;</code> , <code>&lt;CR&gt;</code> , <code>&lt;LF&gt;</code> .
<b>Example 1</b>	<code>AssociateData 100 My Personal attached Data !</code>
<b>Description</b>	This request appends user data to a call which is active at <code>&lt;device&gt;</code> .  <code>AssociateData</code> is a CSTA Phase II only feature and it is only displayed when using STLI version 2.  Associated data is only transmitted in these events: <code>Delivered</code> , <code>Established</code> , <code>Conferenced</code> , <code>Transferred</code> .  Associating and transmitting user data with a call is very phone system specific. Please refer to the documentation of your phone system for further detailed information.
<b>Example 2</b>	(Executed on an Alcatel A4400)  <code>STLI;Version=2</code> <code>error_ind SUCCESS STLI Version "2"</code>  <code>MonitorStart 111</code> <code>error_ind SUCCESS MonitorStart</code>  <code>MonitorStart 529</code> <code>error_ind SUCCESS MonitorStart</code> <code>Delivered 111 newCall 111 003028526530 111 "" 0125 \$00010100</code> <code>\$006b0104 ""</code> <code>DeviceInformation 111 1 (0125:alerting)</code>

Append the string "User Specific Data !" to the call

**AssociateData 111 User Specific Data !**

error\_ind SUCCESS AssociateData

**AnswerCall 111**

error\_ind SUCCESS AnswerCall

This Established event does not return any data: ""

Established 111 newCall 111 003028526530 111 "" 0125 \$00010100  
\$006b0104 ""

DeviceInformation 111 1 (0125:connect)

Call back from 111 to 529

**ConsultationCall 111 529**

Held 111 consultation 111 0125 \$00010100

DeviceInformation 111 1 (0125:hold)

error\_ind SUCCESS ConsultationCall

Originated 111 newCall 111 529 "" 0126 \$00010100

DeviceInformation 111 2 (0125:hold,0126:connect)

Delivered 111 newCall 529 111 529 "" 0126 \$014e0100 "" ""

DeviceInformation 111 2 (0125:hold,0126:connect)

Delivered 529 newCall 529 111 529 "" 0126 \$014e0100 "" ""

DeviceInformation 529 1 (0126:alerting)

**AnswerCall 529**

error\_ind SUCCESS AnswerCall

Established 529 newCall 529 111 529 "" 0126 \$014e0100 "" ""

DeviceInformation 529 1 (0126:connect)

111 is connected to 529 and received the first call with associated data.

Established 111 newCall 529 111 529 "" 0126 \$014e0100 "" ""

DeviceInformation 111 2 (0125:hold,0126:connect)

111 transfers the first call to 529

**TransferCall 111**

error\_ind SUCCESS TransferCall

Transferred 111 noCause 111 529 {} 0125 0126 "" ""

DeviceInformation 111 0 ()

529 receives a Transferred event which contains the associated data.

Transferred 529 noCause 111 529

{529/\$014e0100,3028526530/\$006b0104} 0126 "" 0127 User Specific  
Data !

DeviceInformation 529 1 (0127:connect)

## SendDTMFTones

**Synopsis**     `SendDTMFTones <device> <tones>`

**Parameters**     `<device>` indicates the device.

`<tones>` valid tones are: 0123456789\*#ABCD

                 Some phone systems also support ",," (pause)

---

**Note:**     Some phone systems only support this request on external calls (i.e., to a number outside of the phone system). The device must have an active connection in a connected state.

---

**Example**     `SendDTMFTones 123 98756`

`error_ind SUCCESS SendDTMFTones`

**Description**     SendDTMFTones is used to send DTMF tones to an active connection of a device.

## Other requests

### BYE

**Synopsis**     `BYE`

**Example**     `BYE`

`error_ind SUCCESS BYE`

**Description**     Close the STLI session and disconnect.

### GetVersion

**Synopsis**     `GetVersion`

**Example**     `GetVersion`

`error_ind SUCCESS GetVersion "R202009283"`

**Description**     This request returns TeamCall's internal version number.  
                 Since it does not alter the state of TeamCall it can also be used to implement a heartbeat for situations where network components might close the TCP connection after extended periods of inactivity.

# 3

## Events and Event Reports

There are different kinds of event reports: call events, feature events, agent state events and maintenance events.

### Call events

Call events are asynchronous. They are delivered for every monitored device (see *MonitorStart* on page 11 for details on monitoring). Every monitored device receives its own call event, indicating a change of state.

The Call Event Report service gives detailed information about the state and changes of the state of a device being monitored.

### Marking of incoming external calls

Sometimes external incoming calls are signaled with a leading + in the events: *delivered*, *established*, *queued*.

When an external incoming call (i.e., a call from outside of the phone system) is signaled, the dialing number of the calling party starts with the character + (unless you have specified a *prefixPlan* in the main configuration file).

In these events, the CSTA protocol (Phase I and II) identifies the calling party as exactly one of these types:

- *device*: internal or external calling device
- *implicitPublic*: external calling device
- *explicitPublic*: external calling device
- *implicitPrivate*: internal calling device
- *explicitPrivate*: internal calling device
- *other*: internal or external calling device (i.e., unspecified)

In order to identify external and internal devices within the events signaled via STLI/TAPI, the character + is prepended to the calling device number if its type is either *implicitPublic* or *explicitPublic*.

Which type is used for signaling depends on the phone system. Usually, external devices are of the type *implicitPublic*. Please refer to the document ECMA-218 for more information.

Sometimes external incoming calls are not signaled with the character + in the events: *transferred*, *conferenced*.

The connection list, which is transmitted via the `transferred` or `conferenced` events, does not contain any information about the origin of the included devices. They are simply transmitted as `DeviceID` without further specification. For example, there is no distinction of types in the `delivered` event.

## Feature events

The phone system must support feature events and feature event monitoring to use this feature.

Each device feature event report is a message, that indicates a change in the feature state of a device. The feature events occur asynchronously, generated by an event report service, initiated by calling `MonitorStart`.

When the feature associated with the monitored device changes its state, an event will be provided.

Each feature event report is a message that indicates a change in the feature state of a call or device in the CSTA network. Like call event reports, each feature event report indicates the new state that the feature enters independent of any previous state.

## Agent state events

The phone system must support agents and agent state event monitoring to use this feature. Each agent state event report is a message, that indicates a change in the state of the agent it is referred to.

The agent state events, which occur asynchronously, are generated by the Agent State Event Report service. They are initiated by calling `MonitorStart`.

The phone system must support the Agent State Event Report service to be able to get agent state events. Like call event reports, each agent state event report indicates the new state, that the agent enters, independent of any previous state.

## Maintenance events

The phone system must support maintenance events and maintenance event monitoring to use this feature.

Each device maintenance event report is a message, that indicates a change in maintenance state of a device.

The maintenance events occur asynchronously, generated by an event report service, initiated by calling `MonitorStart`.

When the monitored device changes its state, an event will be provided. Each maintenance event report is a message that indicates a change in state of a device in the CSTA network. Like call event reports, each maintenance event report indicates the new state that the feature enters independently of any previous state.

These are the maintenance events:

## BackInService

- Synopsis**      `BackInService <monitoredDevice> <cause> <device>`
- Example**      `BackInService 8801 noCause 8801`
- Description**    This is a maintenance event. It signals <monitoredDevice> that the device <device> is now back in service. I.e., available. Usually, this event is sent when a device is (re-)connected to the phone system or when an agent logs off from a device.

## OutOfService

- Synopsis**      `OutOfService <monitoredDevice> <cause> <device>`
- Example**      `OutOfService 3057 noCause 3057`
- Description**    This is a maintenance event. It signals <monitoredDevice> that the device <device> is now out of service. I.e., no longer available. Usually, this event is sent when a device is disconnected from the phone system or when an agent logs on to a device.

## List of events and event reports

Responses to previous requests and especially events can contain sequences of escape characters or identifiers which contain an empty string. Such sequences will be provided in double quotes ("..." or "" if the string is empty).

Events will be reported in the following format:

`<eventName> <monitoredDevice> <specific Parameters>`

Non-submitted parameters will be reported as an empty string "".

## AgentBusy

- Synopsis**      `AgentBusy <monitoredDevice> <cause> <agentDevice> <agentID>  
                 <agentGroup>`
- Example**      `AgentBusy 8803 "" 8803 8803 8851`
- Description**    This is an CSTA Phase II only agent state event.

## AutoAnswer

- Synopsis**      `AutoAnswer <monitoredDevice> <device> <on/off>`
- Description**    This is a feature event.

## Conferenced

**Synopsis V1** Conferenced <monitoredDevice> <cause> <confController> <addedParty> {<conferenceConnectionsList>}

**Example V1** Conferenced 100 newCall 100 104 {100,101,102,103}

**Synopsis V2** Conferenced <monitoredDevice> <cause> <confController> <addedParty> {conferenceConnectionsList} <primaryOldCall> <secondaryOldCall> <newCall> correlatorData

The devices in <conferenceConnectionsList> are identified by their <dialing number> *and* their unique <deviceId>. Each entry in this list has the format <dialing number>/<deviceId>.

Note that the <dialing number> is the phone number of the device and may not be transmitted.

Note that the string for <secondaryOldCall> can be empty ("").

Note that the <newCall> may not be set for the transferring device.

correlatorData represents the data eventually attached to the call using the AssociateData request.

**Example V2** Conferenced 123 noCause 123 168 {123/\$00040100,168/\$00030100,0017712345678/\$006c0104} 0310 0311 0312 ""

**Description** This is a call event. It indicates that the conferencing device has conferenced itself or another device with an existing call. For more information see also *Marking of incoming external calls* on page 29.

## ConnectionCleared

**Synopsis V1** ConnectionCleared <monitoredDevice> <cause> <releasingDevice> {activeConnectionsList}

**Example V1** ConnectionCleared 101 normalClearing 100 {101,102}

**Synopsis V2** ConnectionCleared <monitoredDevice> <cause> <releasingDevice> {activeConnectionsList} callId deviceId

The devices in the <activeConnectionsList> are identified by their <dialing number> *and* their unique <deviceId>. Each entry in this list has the format <dialing number>/<deviceId>.

Note that the <dialing number> is the phone number of the device and may not be transmitted.

The <callId> and <deviceId> represent the dropped connection. Please refer to ECMA-180/218 for more information.

**Example V2** ConnectionCleared 123 normalClearing 0017712345678 {123/\$00040100,168/\$00030100} 0312 \$006c0104

**Description** This is a call event. It indicates that a device in a call has been disconnected or was dropped from a call.



## Delivered

**Synopsis V1** Delivered <monitoredDevice> <cause> <alertingDevice>  
<callingDevice> <calledDevice> <lastRedirectionDevice>

**Example V1** Delivered 168 newCall 185 168 185 ""

**Synopsis V2** Delivered <monitoredDevice> <cause> <alertingDevice>  
<callingDevice> <calledDevice> <lastRedirectionDevice> callId  
deviceId originatingDeviceId correlatorData

The <callId> and <deviceId> usually represent the connection to the alerting device. Please refer to ECMA-180/218 for more information.

The <originatingDeviceId> represents the unique deviceId contained in the optional information originatingConnection. Please refer to ECMA-218 for more information. It is CSTA Phase II specific and identifies the device which set up the connection. I.e., the calling device. Depending on the phone system, this information may be empty (i.e., "") if the calling device is an internal device (i.e., within the phone system) and/or the monitored device is the calling device.

The correlatorData represents the data eventually attached to the call using the AssociateData request.

**Example V2** Delivered 123 newCall 123 0017712345678 123 "" 030b \$00040100  
\$006c0104 ""

**Description** This is a call event. It indicates that a call is being presented to a device in either the ringing state or entering distribution modes of the alerting state.  
lastRedirectionDevice may be empty.

Please note that the <calledDevice> may be formatted differently than the other devices. For outbound calls it contains the number in the format as it had been dialed (e.g., including a trunk access code) and for inbound calls it contains the number in the format as signaled by the telephone network. All other events are formatted in a phone system specific standardized way. Please note that in many cases both formats will be identical. I.e., they will contain the numbers in dialable form.

## Diverted

**Synopsis V1** Diverted <monitoredDevice> <cause> <divertingDevice>  
<newDestination>

**Example V1** Diverted 111 callNotAnswered 111 168

**Synopsis V2** Diverted <monitoredDevice> <cause> <divertingDevice>  
<newDestination> callId deviceId

The <callId> and <deviceId> usually represent the diverted connection. Please refer to ECMA-180/218 for more information.

**Description** This is a call event. It indicates that a call has been diverted successfully.

## DoNotDisturb

**Synopsis** DoNotDisturb <monitoredDevice> <device> <onOff>

**Description** This is a feature event.

## Established

**Synopsis V1** Established <monitoredDevice> <cause> <answeringDevice>  
<callingDevice> <calledDevice> <lastRedirectionDevice>

**Example V1** Established 185 newCall 185 168 185 ""

**Synopsis V2** Established <monitoredDevice> <cause> <answeringDevice>  
<callingDevice> <calledDevice> <lastRedirectionDevice> callId  
deviceId originatingDeviceId correlatorData

The <callId> and <deviceId> usually represent the connection to the answering device. Please refer to ECMA-180/218 for more information.

The <originatingDeviceId> represents the unique deviceId contained in the optional information originatingConnection. Please refer to ECMA-218 for more information. Established is a CSTA Phase II specific feature and identifies the device which set up the connection. I.e., the calling device. Depending on the phone system, this information may be provided as an empty string "" if the calling device is internal (i.e., within the phone system) and/or the monitored device is the calling device.

correlatorData represents the data eventually attached to the call using the AssociateData request.

**Example V2** Established 123 newCall 123 0017712345678 123 "" 030b \$00040100  
\$006c0104 ""

**Description** This is a call event. It indicates that a device has answered or has been connected to a call. lastRedirectionDevice may be empty.

Please note that the <calledDevice> may be formatted differently than the other devices. For outbound calls it contains the number in the format as it had been dialed (e.g., including a trunk access code) and for inbound calls it contains the number in the format as signaled by the telephone network. All other events are formatted in a phone system specific standardized way. Please note that in many cases both formats will be identical. I.e., they will contain the numbers in dialable form.

## Failed

**Synopsis V1** Failed <monitoredDevice> <cause> <failedDevice> <calledDevice>

**Example V1** Failed 1254 destNotObtainable 1254 168

**Synopsis V2** Failed <monitoredDevice> <cause> <failedDevice> <calledDevice>  
callId deviceId

The <callId> and <deviceId> usually represent the failed connection. Please refer to ECMA-180/218 for more information.

**Example V2** Failed 111 destNotObtainable 111 00177999999999 0330 \$00010100

**Description** This is a call event. It indicates that a call could not be completed and/or a connection has entered the fail state.

## Forwarding

**Synopsis** Forwarding <monitoredDevice> <device> <type> <forwardDN>  
<forwardedTo>

**Examples** Forwarding 100 100 forwardImmediateOn 101 101  
Forwarding 100 100 forwardImmExtOff "" ""

**Description** This is a feature event.

## Held

**Synopsis V1** Held <monitoredDevice> <cause> <holdingDevice>

**Example V1** Held 185 noCause 185

**Synopsis V2** Held <monitoredDevice> <cause> <holdingDevice> <callId> <deviceId>  
The <callId> and <deviceId> usually represent the hold connection. Please refer to ECMA-180/218 for more information.

**Example V2** Held 123 consultation 123 030b \$00040100

**Description** This is a call event. It indicates that an existing call has been put on hold.

## Initiated

**Synopsis V1** Initiated <monitoredDevice> <cause>

**Example V1** Initiated 168 newCall

**Synopsis V2** Initiated <monitoredDevice> <cause> <callId> <deviceId>  
The <callId> and <deviceId> usually represent the newly initiated connection. Please refer to ECMA-180/218 for more information.

**Example V2** Initiated 123 newCall 030a \$00040100

**Description** This is a call event. It indicates that <monitoredDevice> has gone off-hook for service or is being prompted to go off-hook.

## LoggedOn

**Synopsis** LoggedOn <monitoredDevice> <cause> <agentDevice> <agentID>  
<agentGroup> <password>

**Example** LoggedOn 8803 "" 8803 8803 8851 ""

**Description** This is an CSTA Phase I + II agent state event.

## LoggedOff

**Synopsis** LoggedOff <monitoredDevice> <cause> <agentDevice> <agentID>  
<agentGroup> <password>

**Example** LoggedOff 8803 "" 8803 8803 "" ""

**Description** This is an CSTA Phase I + II agent state event.

## MessageWaiting

**Synopsis** MessageWaiting <monitoredDevice> <device> <invoking> <onOff>

**Description** This is a feature event.

## MicrophoneMute

**Synopsis** MicrophoneMute <monitoredDevice> <device> <onOff>

**Description** This is a feature event.

## NetworkReached

**Synopsis V1** NetworkReached <monitoredDevice> <cause> <calledDevice>  
<trunkDevice>

**Synopsis V2** NetworkReached <monitoredDevice> <cause> <calledDevice>  
<trunkDevice> <callId> <deviceId>

The <callId> and <deviceId> usually represent the (newly created) outbound connection. Please refer to ECMA-180/218 for more information. The <deviceId> may be used to unambiguously identify the called party within the phone system.

**Example V2** NetworkReached 123 newCall "" "" 030a \$006c0104

**Description** This is a call event. It indicates that a call has been connected to an external network using a network interface device (e.g., trunk).

Please note that the <calledDevice> may be formatted differently than the other devices. For outbound calls it contains the number in the format as it had been dialed (e.g., including a trunk access code) and for inbound calls it contains the number in the format as signaled by the telephone network. All other events are formatted in a phone system specific standardized way. Please note that in many cases both formats will be identical. I.e., they will contain the numbers in dialable form.

## NotReady

**Synopsis** NotReady <monitoredDevice> <cause> <agentDevice> <agentID>

**Example** NotReady 8803 "" 8803 8803

**Description** This is an CSTA Phase I + II agent state event.

## Originated

**Synopsis V1** Originated <monitoredDevice> <cause> <callingDevice> <calledDevice>  
<originatingDevice>

**Example V1** Originated 168 newCall 168 185 ""

**Synopsis V2** Originated <monitoredDevice> <cause> <callingDevice> <calledDevice>  
<originatingDevice> <callId> <deviceId>

The <callId> and <deviceId> usually represent the originating connection. I.e., the connection of the calling device. Please refer to ECMA-180/218 for more information.

**Example V2** Originated 123 newCall 123 0017712345678 "" 030a \$00040100

**Description** This is a call event. It indicates that a call is being attempted from a device.  
Please note that the <calledDevice> may be formatted differently than the other devices. For outbound calls it contains the number in the format as it had been dialed (e.g., including a trunk access code) and for inbound calls it contains the number in the format as signaled by the telephone network. All other events are formatted in a phone system specific standardized way. Please note that in many cases both formats will be identical. I.e., they will contain the numbers in dialable form.

## Queued

**Synopsis V1** Queued <monitoredDevice> <cause> <queue> <callingDevice>  
<calledDevice> <lastRedirectionDevice> <numberQueued>  
<callsInFront>

**Example V1** Queued 100 campOn 100 102 100 "" "" ""

**Synopsis V2** Queued <monitoredDevice> <cause> <queue> <callingDevice>  
<calledDevice> <lastRedirectionDevice> <numberQueued>  
<callsInFront> <callId> <deviceId>

The <callId> and <deviceId> usually represent the queued connection. Please refer to ECMA-180/218 for more information.

**Description** This is a call event. It indicates that a call has been queued. I.e., device <callingDevice> is enqueued in <queue>.

## Ready

**Synopsis** Ready <monitoredDevice> <cause> <agentDevice> <agentID>

**Example** Ready 8803 "" 8803 8803

**Description** This is an CSTA Phase I + II agent state event.

## Retrieved

**Synopsis V1** Retrieved <monitoringDevice> <cause> <retrievingDevice>

**Synopsis V2** Retrieved <monitoringDevice> <cause> <retrievingDevice> callId  
deviceId

The <callId> and <deviceId> usually represent the retrieved connection. Please refer to ECMA-180/218 for more information.

**Description** This is a call event. It indicates that a previously held call has been retrieved.

## SpeakerMute

**Synopsis** SpeakerMute <monitoredDevice> <device> <onOff>

**Description** This is a feature event.

## SpeakerVolume

**Synopsis** SpeakerVolume <monitoredDevice> <device> <volume>

**Description** This is a feature event.

## Transferred

**Synopsis V1** Transferred <monitoredDevice> <cause> <transferringDevice>  
<transferredDevice> {<transferredConnectionsList>}

**Example V1** Transferred 185 noCause 111 168 {168, 185}

**Synopsis V2** Transferred <monitoredDevice> <cause> <transferringDevice>  
<transferredDevice> {transferredConnectionsList} <primaryOldCall>  
<secondaryOldCall> <newCall> <correlatorData>

The devices in the <transferredConnectionsList> are identified by their <dialing number> *and* their unique <deviceId>. Each entry in this list has the format <dialing number>/<deviceId>.

Note that the <dialing number> is the phone number of the device and may not be transmitted.

Note that the <secondaryOldCall> may *only* be set for the transferring device.

Note that the <newCall> may not be set for the transferring device.

The correlatorData represents the data eventually attached to the call using the AssociateData request.

**Example V2** Transferred 168 noCause 123 168  
{168/\$00030100,17712345678/\$006c0104} \$006c0104 \$006c0105 "" ""

**Description** This is a call event. It indicates that an existing call has been transferred to another device and that the device transferring the call has been dropped from the call. For more information see also *Marking of incoming external calls* on page 29.

## WorkingAfterCall

**Synopsis** WorkingAfterCall <monitoredDevice> <cause> <agentDevice> <agentID>  
<agentGroup>

**Example** WorkingAfterCall 8803 "" 8803 8803 8851

**Description** This is an CSTA Phase II agent state event.

## WorkNotReady

**Synopsis** WorkNotReady <monitoredDevice> <cause> <agentDevice> <agentID>

**Description** This is an CSTA Phase I agent state event.

## WorkReady

**Synopsis** WorkReady <monitoredDevice> <cause> <agentDevice> <agentID>

**Description** This is an CSTA Phase I agent state event.

# 4

## Error messages

The following error messages can occur when issuing STLI requests:

### **ILINK-INTERNAL DEMOVERSION**

**Response** `error_ind ILINK-INTERNAL DEMOVERSION MonitoringPointsExceeded`  
**Description** Only one device can be monitored with this demo version.

### **INVALCMD**

**Response** `error_ind INVALCMD <Request>`  
**Description** Unknown request.  
**Example** `MonStart 111`  
`error_ind INVALCMD MonStart`

### **INVALIDAGENTSTATE**

**Response** `error_ind INVALIDAGENTSTATE SetAgentState <requestedState>`  
**Description** Unknown parameter `<requestedState>` for request `SetAgentState`.

### **INVALIDDEVICEFEATURE**

**Response** `error_ind INVALIDDEVICEFEATURE SetDeviceFeature <feature>`  
**Description** Unknown parameter `<feature>` for request `SetDeviceFeature`.

### **INVALIDFORWARDINGFEATURE**

**Response** `error_ind INVALIDFORWARDINGFEATURE SetForwarding <forwardingType>`  
**Description** Unknown parameter `<forwardingType>` for request `SetForwarding`.

## **INVALNUMPARAM**

**Response**     `error_ind INVALNUMPARAM <Request>`  
**Description**   Invalid number of parameters for this request.  
**Example**       `MonitorStart 223 333`  
                  `error_ind INVALNUMPARAM MonitorStart`

## **INVALPARAM**

**Response**     `error_ind INVALPARAM <Request>`  
**Description**   Wrong parameter(s) for this request.  
**Example**       `SetForwarding 111 forwardOn 102`  
                  `error_ind INVALPARAM SetForwarding`

## **LINKOUTOFSERVICE**

**Response**     `error_ind LINKOUTOFSERVICE <Request>`  
**Description**   The request cannot be executed because there is no connection to the phone system at this moment.  
**Example**       `MakeCall 100 101`  
                  `error_ind LINKOUTOFSERVICE MakeCall`

## **NOCALL**

**Response**     `error_ind NOCALL <Request>`  
**Description**   The request cannot be executed because there is no call at the device at this moment.  
**Example**       `DeflectCall 111 123`  
                  `error_ind NOCALL DeflectCall`

## **NOCONNECTION**

**Response**     `error_ind NOCONNECTION <Request>`  
**Description**   The request cannot be executed because there is no connection at the device at this moment.  
**Example**       `ClearConnection 111`  
                  `error_ind NOCONNECTION ClearConnection`

## **NODEVICE**

**Response**     `error_ind NODEVICE <Request>`  
**Description**   The device has to be monitored before the request can be executed.  
**Example**       `DeflectCall 100 123`  
                  `error_ind NODEVICE DeflectCall`



## **NOMONITOR**

**Response** `error_ind NOMONITOR MonitorStop`  
**Description** The device is not being monitored during this session.  
**Example** **MonitorStop 333**  
`error_ind NOMONITOR MonitorStop`

## **NOMULTIPLEINSTANCE**

**Response** `error_ind NOMULTIPLEINSTANCE MonitorStart`  
**Description** The request `MonitorStart` has already been issued for this device during this session.  
**Example** **MonitorStart 111**  
`error_ind SUCCESS MonitorStart`  
**MonitorStart 111**  
`error_ind NOMULTIPLEINSTANCE MonitorStart`

## **PENDINGREQUEST**

**Response** `error_ind PENDINGREQUEST <Request>`  
**Description** Another session is currently waiting for the same request with the same parameters. Usually, this error is sent when another session already started a monitoring request for the same device and the request has not yet been answered by the phone system (i.e., the DeviceMonitor is not active).  
**Example** **MonitorStart 111**  
`error_ind PENDINGREQUEST MonitorStart 111`

## **SUCCESS**

**Response** `error_ind SUCCESS <Request>`  
**Description** No problems (this "error\_ind" does not indicate an error).  
**Example** **MonitorStart 111**  
`error_ind SUCCESS MonitorStart`

## **UNAVAILBLEREQUEST**

**Response** `error_ind UNAVAILBLEREQUEST <request> ""`  
**Description** Request at this time not supported. It could be not supported, disabled or invalid.

# 5

## STLI Best Practices

### Heartbeat

Depending on the network setup, there might be network components like firewalls between the application that uses STLI and the TeamCall / TeamCall Express server that the application connects to.

If these network components are configured to automatically close open TCP connections after extended periods of inactivity, the application should use a heartbeat mechanism in order to prevent its TeamCall connection from being closed.

For this, the STLI request `GetVersion` is ideally suited. It returns the server's internal version number without altering the server state.

Periodically calling this request makes sure that there is activity on the open TCP connection, so that the connection will not be closed.

### Identifying the `ConnectionCleared` event of the remote call connection

The STLI `ConnectionCleared` event is sent when a call has been terminated.

However, as per the CSTA call model, phone systems typically send two `ConnectionCleared` events when the remote party terminates the call: one for the termination of the remote call connection, and one for the termination of the local call connection.

If an application wants to ignore events for the remote call connection, it needs to identify if a given `ConnectionCleared` event is for the local or for the remote call connection.

A typical way of doing so is to compare the event's `releasingDevice` (the event's third parameter) to the user's local extension number. If there is a match, the event is for the local call connection.

Please note that the `releasingDevice` may be formatted differently from how the local extension is being used in STLI requests like `MonitorStart` (e.g., it may include the trunk number). So, the `releasingDevice` may have to be standardized before this connection is being made.

# Phone number formats in STLI requests

Most STLI requests contain the local extension number as their first parameter.

In addition to the local extension, several STLI requests also contain phone numbers in other parameters. For example, the target numbers in `MakeCall`, `ConsultationCall`, or `SingleStepTransfer` requests. These parameters could contain either internal extension numbers or external numbers.

Depending on the application and the use case, target numbers could come from a variety of origins, including:

- a database. E.g., in applications that make contact phone numbers clickable for dialing
- user input. E.g., in applications that allow the user to enter a phone number for dialing
- arbitrary documents. E.g., in applications that allow selected text to be dialed via a hotkey function

TeamCall passes the phone numbers in STLI parameters to the phone system without any modifications. That is why these phone numbers need to be formatted in the way expected by the phone system.

Depending on the application this might require more or less reformatting.

## The local extension number

Most STLI requests contain the local extension number as their first parameter. Most phone systems expect these numbers in the short form of just the extension number. E.g., 123. (See the section *Using STLI with Avaya Definity and Unify OpenScape Voice* below for an example of a different case.)

A typical application allows to configure the user's extension so that it can be monitored. This configuration should use the number format expected by the phone system.

## Internal extension targets

When an internal extension number shall be called, most phone systems expect this number in the short form of just the extension number. E.g., 555.

Sometimes the phone system accepts the internal extension target in various formats, but some phone systems only accept the short form. So it is recommended that an application shall always assume that this is the case.

If the data source provides this target number in a different form. E.g., +493028526555, it is the application's responsibility to:

- detect that this is an internal extension target
- reformat the target number to the short form (e.g., 555) before using it in the STLI request.

One way of doing so is to have a configuration value of the user's trunk number (e.g., +493028526) or the components contained in the trunk number (country code, area code, local trunk number) so that it can easily be determined that the target number starts with the trunk number.

All of this assumes that the externally dialable number can be derived by appending the extension number to the trunk number. This is often the case, but not always – some phone systems do not provide a DID (direct inward dialing) number for each internal extension or use DIDs for the internal extensions that do not end with the extension numbers.

*Examples (for apps connected to a phone system in Berlin, Germany with trunk number +49 30 28526)*

Input from the data source	Converted for use in STLI requests
<b>+49 (30) 28526</b> -555	555
<b>030 28526</b> -555	555
<b>28526</b> -555	555
555	555

## External phone number targets

When an external number shall be called (local number, long distance number, mobile phone, international number), many phone systems use a trunk access code (e.g., 0 or 9), to differentiate between internal and external targets.

If the phone system has a trunk access code, and if the data source does not already include this code in the number to be dialed, it is the application's responsibility to add the trunk access code to the target number before using it in the STLI request.

## Local phone number targets

Local phone number targets (phone numbers within the same area code) may or may not require the area code.

In some countries or locations (e.g., in Switzerland or New York City), the area code needs to be included in all external calls, including in local calls. If this is required, and this area code is not already included in the number to be dialed, it is the application's responsibility to add the area code before the number is used in the STLI request.

In other countries or locations (e.g., in Germany or many US locations) local numbers can be dialed without their area code.

If an application targets international audiences, this may need to be a configuration parameter.

If the target number is provided in full international format, including country code, it is the responsibility of the application to remove the country code and to add the long-distance call prefix to the local area code (if such a prefix is required in the phone numbering plan of the local country. E.g., 0 in Germany).

In all cases, local phone numbers are external phone numbers, so the information about trunk access codes in the section *External phone number targets* above applies as well.

*Examples (for apps connected to a phone system in Berlin, Germany with trunk access code 0)*

Input from the data source	Converted for use in STLI requests
<b>+49 (30)</b> 5551234	<b>0</b> 5551234
<b>030</b> 5551234	<b>0</b> 5551234
5551234	<b>0</b> 5551234

## Long-distance and mobile phone number targets

Long-distance phone number targets (phone numbers outside of the same area code) as well as mobile phone number targets may require a long-distance call prefix to be added to the area code (or mobile phone number equivalent) if that prefix is not already included in the number's area code.

If the target number is provided in full international format, including country code, it is the responsibility of the application to remove the country code and to add the long-distance call prefix to the local area code (if such a prefix is required in the phone numbering plan of the local country. E.g., 0 in Germany).

In all cases, long-distance and mobile phone numbers are external phone numbers, so the information about trunk access codes in the section *External phone number targets* above applies as well.

*Examples (for apps connected to a phone system in Berlin, Germany with trunk access code 0)*

Input from the data source	Converted for use in STLI requests
<b>+49 (89)</b> 5551234	<b>00</b> 895551234
089 5551234	<b>0</b> 0895551234

## International phone number targets

International phone number targets need to include an international call prefix (e.g., 00 in European countries, or 011 in the USA). If this prefix is not already included in the number to be dialed, it is the responsibility of the application to add this prefix (or to replace a leading plus character with this prefix) before the number is used in the STLI request.

In all cases, international phone numbers are external phone numbers, so the information about trunk access codes in the section *External phone number targets* above applies as well.

*Examples (for apps connected to a phone system in Berlin, Germany with trunk access code 0 and international call prefix 00)*

Input from the data source	Converted for use in STLI requests
+1 (202) 456-1414	00012024561414
001 202 456-1414	00012024561414

# Using STLI with Avaya Definity and Unify OpenScape Voice

## Controlling Devices

Almost all STLI commands as well as all STLI events contain a parameter that specifies the so-called *controlling device*. I.e., the device that the request or event pertains to.

Most phone systems address these devices using internal extension numbers, which are short numbers like 1414 or 555.

However, some phone systems, specifically Avaya Definity and Unify OpenScape Voice, address the controlling devices using the international phone number format. E.g., +12024561414 or +493028526555.

It is the responsibility of the application to use the correct format of the controlling device in STLI requests sent to TeamCall and to be able to handle the format in which the controlling device is indicated in STLI events received from TeamCall.

## STLI Requests

The following STLI requests use controlling devices. They are specified in the first parameters of each request (marked in blue text).

```
AlternateCall <callingDevice>
AnswerCall <calledLocalDevice>
AssociateData <device> <data>
CallBack <callingDevice>
CampOn <callingDevice>
ClearConnection <localConnectedDevice>
ConferenceCall <callingDevice>
ConsultationCall <callingDevice> <consultedDirectoryNumber>
    [<consultationMode>]
DeflectCall <callToBeDeflected> <newDestination>
GroupPickupCall <requestingDevice>
HoldCall <callingDevice>
Intrude <callingDevice>
```

```

MakeCall <callingDevice> <calledDirectoryNumber> [prompt |
doNotPrompt]
MonitorStart <localDevice>
MonitorStop <localMonitoredDevice>
PickupCall <callToBePickedUp> <requestingDevice>
QueryDevice <localDevice> <feature>
ReconnectCall <callingDevice>
RetrieveCall <retrievingDevice>
SendDTMFTones <device> <tones>
SetAgentState <deviceID/agentID> <requestedAgentState> [<agentID>
<password> <group>]
SetDeviceFeature <deviceToBeSet> <aFeature> <...further parameters
depending on the feature>
SetForwarding <forwardingDevice> <forwardingType>
[<newForwardingDestination>]
SingleStepTransfer <transferringDevice> <transferredToDevice>
TransferCall <transferringDevice>

```

## STLI Events

The situation for STLI events is similar: one parameter of each event contains the controlling device. For example:

```

ConnectionCleared <monitoredDevice> <cause> <releasingDevice>
{activeConnectionsList}
Delivered <monitoredDevice> <cause> <alertingDevice>
<callingDevice> <calledDevice> <lastRedirectionDevice>

```

## Examples of an outbound call

The following examples show the STLI requests and events of an outbound call from the internal extension 555 to the German mobile number +491705551234.

The inbound call is accepted by the called party and is later terminated via STLI.

The assumption is that the application is connected to a phone system in Berlin, Germany with the trunk access code 0 and the long-distance call prefix 0.

### Example for a typical phone system

#### STLI

```

error_ind SUCCESS STLI;Version=1;DeviceInformation=Standard
MonitorStart 555
error_ind SUCCESS MonitorStart
MakeCall 555 001705551234
error_ind SUCCESS MakeCall
Initiated 555 normal
DeviceInformation 555 1 (3247358556:initiate)

```

```

Originated 555 normal 555 001705551234 ""
DeviceInformation 555 1 (3247358556:connect)
Delivered 555 normal 001705551234 555 001705551234
DeviceInformation 555 1 (3247358556:connect)
Established 555 newCall 001705551234 555 001705551234
DeviceInformation 555 1 (3247358556:connect)

ClearConnection 555
ConnectionCleared 555 normalClearing 555 {}
DeviceInformation 555 0 ()
error_ind SUCCESS ClearConnection

MonitorStop 555
error_ind SUCCESS MonitorStop

BYE
error_ind SUCCESS BYE

```

## Example for a Unify OpenScape Voice

```

STLI
error_ind SUCCESS STLI;Version=1;DeviceInformation=Standard

MonitorStart +493028526555
error_ind SUCCESS MonitorStart

MakeCall +493028526555 001705551234
error_ind SUCCESS MakeCall
Initiated +493028526555 makeCall
DeviceInformation +493028526555 1
    (FF00010000000000F2F031616B690100:initiate)
Originated +493028526555 newCall +493028526555 491705551234 ""
DeviceInformation +493028526555 1
    (FF00010000000000F2F031616B690100:connect)
NetworkReached +493028526555 normal 491705551234
DeviceInformation +493028526555 1
    (FF00010000000000F2F031616B690100:connect)
Delivered +493028526555 networkSignal 491705551234 +493028526555
    491705551234
DeviceInformation +493028526555 1
    (FF00010000000000F2F031616B690100:connect)
Established +493028526555 networkSignal 491705551234 +493028526555
    491705551234
DeviceInformation +493028526555 1
    (FF00010000000000F2F031616B690100:connect)

ClearConnection +493028526555
ConnectionCleared +493028526555 normalClearing +493028526555 {}
DeviceInformation +493028526555 0 ()
error_ind SUCCESS ClearConnection

MonitorStop +493028526555
error_ind SUCCESS MonitorStop

BYE
error_ind SUCCESS BYE

```



## Examples of an inbound call

The following examples show the STLI requests and events of an inbound call from the German mobile number +491705551234 to the internal extension 555.

The inbound call is accepted on the telephone device of the local user and is later terminated by the caller.

The assumption is that the application is connected to a phone system in Berlin, Germany with the trunk access code 0 and the long-distance call prefix 0.

### Example for a typical phone system

#### STLI

```
error_ind SUCCESS STLI;Version=1;DeviceInformation=Standard
```

#### MonitorStart 555

```
error_ind SUCCESS MonitorStart
```

#### MakeCall 555 001705551234

```
error_ind SUCCESS MakeCall
```

```
Delivered 555 normal 555 001705551234 555
```

```
DeviceInformation 555 1 (3247358556:alerting)
```

```
Established 555 newCall 555 001705551234 555
```

```
DeviceInformation 555 1 (3247358556:connect)
```

```
ConnectionCleared 555 normalClearing 555 {}
```

```
DeviceInformation 555 0 ()
```

```
error_ind SUCCESS ClearConnection
```

#### MonitorStop 555

```
error_ind SUCCESS MonitorStop
```

#### BYE

```
error_ind SUCCESS BYE
```

### Example for a Unify OpenScape Voice

#### STLI

```
error_ind SUCCESS STLI;Version=1;DeviceInformation=Standard
```

#### MonitorStart +493028526555

```
error_ind SUCCESS MonitorStart
```

```
Delivered +493028526555 newCall +493028526555 491705551234  
+493028526555
```

```
DeviceInformation +493028526555 1  
(FF0001000000000002AF1316170690100:alerting)
```

```
Established +493028526555 newCall +493028526555 491705551234  
+493028526555
```

```
DeviceInformation +493028526555 1  
(FF0001000000000002AF1316170690100:connect)
```

```
ConnectionCleared +493028526555 normalClearing 491705551234  
{+493028526555}
```

```
DeviceInformation +493028526555 1  
(FF0001000000000002AF1316170690100:connect)
```

```
ConnectionCleared +493028526555 normalClearing +493028526555 {}
DeviceInformation +493028526555 0 ()

MonitorStop +493028526555
error_ind SUCCESS MonitorStop

BYE
error_ind SUCCESS BYE
```

## Practical advice for STLI on Definity or OpenScape Voice

### Configuration of the switch type

An application that uses STLI and that wants to support the Avaya Definity or OpenScape Voice phone systems in addition to other phone systems should add a configuration switch that determines the switch type:

- A switch type that requires the international format to be used for all phone numbers (e.g., Avaya Definity or OpenScape Voice)
- A switch type that does not require this (e.g., most other phone systems)

Depending on the value of this configuration switch, the app can decide on which behavior is appropriate for making STLI requests and for handling STLI events.

### Configuration of extension numbers

Each of the STLI requests listed above requires the controlling device to be specified. On the Definity and OpenScape Voice phone systems, this needs to be done in the international format.

An easy way to be able to do this is to let the app be flexible enough in the configuration of the users' extensions to allow either the internal number short form (e.g., 555) or in the international format (e.g., +493028526555). For each installation, the extensions could then be configured as is appropriate for the specific phone system.

Assuming that the app will use the user's configured extension as is in the STLI requests, this should be all that needs to be done.

### Rewriting of target phone numbers in various STLI requests

Many STLI requests also contain phone numbers in other parameters. For example, the target numbers in `MakeCall` or `ConsultationCall` requests.

As described in the section *Phone number formats in STLI requests* above, applications may need to reformat these target phone numbers.

Avaya Definity and OpenScape Voice could potentially pose some challenges because they are highly configurable.

In a typical configuration they behave like other phone systems in allowing the target numbers to be provided in a dialable format. In this case, the application does not need to do anything beyond what is described in the section *Phone number formats in STLI requests* above.

Typically, they allow the target numbers to be provided in the international format as well, but there is no advantage for the application to use this format.

If an Avaya Definity or OpenScape Voice system is configured to **only** allow target numbers to be provided in the international format.

**In such a case we would suggest to modify the phone system configuration to also allow target numbers to be provided in a dialable format.**

If such a configuration change is not possible, the application would have to convert the target numbers to international format before using them in STLI request. See the examples below.

The assumption in the following examples is that the app is connected to a phone system in Berlin, Germany.

#### Internal extension targets

Input from the data source	Converted for use in STLI requests if international format is required
+49 (30) 28526-555	+493028526555
<b>0</b> 30 28526-555	<b>+49</b> 3028526555
28526-555	<b>+4930</b> 28526555
555	<b>+493028526</b> 555

#### Local phone number targets

Input from the data source	Converted for use in STLI requests if international format is required
+49 (30) 5551234	+49305551234
<b>0</b> 30 5551234	<b>+49</b> 305551234
5551234	<b>+4930</b> 5551234

#### Long-distance and mobile phone number targets

Input from the data source	Converted for use in STLI requests if international format is required
+49 (89) 5551234	+49895551234
<b>0</b> 89 5551234	<b>+49</b> 895551234

#### International phone number targets

Input from the data source	Converted for use in STLI requests if international format is required
+1 (202) 456-1414	+12024561414
<b>00</b> 1 202 456-1414	<b>+1</b> 2024561414

## Evaluating STLI events

Different from other phone systems, the Avaya Definity and OpenScape Voice Systems typically use the international format (instead of dialable) for phone numbers in CTI events.

So in order to support these phone systems, the application needs to be able to evaluate the event parameters in the international format as well.

**Note:** As an exception, the `<calledDevice>` element in the `Delivered`, `Established`, `NetworkReached`, and `Originated` events is typically provided in the form in which it was given by the app (for outbound calls) or by the telephone network (for inbound calls). This may be a different format.

For example, if the target phone number of a `MakeCall` request was provided in dialable form, the `<calledDevice>` element in the `Established` event will also be provided in dialable format, but all other phone numbers in the event will be provided in international format.

## Working around OpenScape Voice signaling issues

The OpenScape Voice may send phone numbers in some event parameters in international format without the leading plus character, instead of the full international format (see examples of inbound and outbound calls above).

If this is the case and if these parameters are to be evaluated by the application and if the lack of a leading plus character would cause problems in this evaluation, the best way to proceed is to use TeamCall CSTA Server's *PrefixPlan* to add the missing + character before the STLI event is sent to the application.

The *PrefixPlan* is configured in TeamCall CSTA Server's configuration file *Default.conf*. It would be set up to replace all phone numbers that start with the local trunk number (but without a plus) by the same prefix including the plus.

As an example, if the phone system is located in Berlin, Germany with a trunk number of +49 (30) 28526, the *PrefixPlan* configuration rule would look like this:

```
prefixPlan "493028526/+493028526"
```

If an event sent from the OpenScape Voice includes the phone number 493028526, this number would be changed to +493028526. Phone numbers that do not start with 493028526 as well as phone numbers that correctly start with +493028526 are not modified.