ilink TeamCall Server

Guide and Reference for Programmers and Administrators

ilink
Kommunikationssysteme GmbH

TeamCall Server was created by Carsten J. Gericke, Patrick Kielhöfer and Ralf Brauwers.

The manual was written and edited by Helge Ebsen, Tillmann Krenz and Thilo Roscher.

This manual describes TeamCall Server version 3. Manual version 1.1.

ilink Kommunikationssysteme GmbH
Große Hamburger Straße 32
D-10115 Berlin, Germany

+49-30-285 26 - 0
+49-30-285 26 - 199 Fax

info@ilink.de
www.ilink.de

January 2001.

Content

1

Introduction

Welcome to ilink TeamCall Server.

TeamCall is the open Computer Telephony Integration (CTI) environment from ilink. Because of its open architecture, other developers can deploy TeamCall to use professional telephony features in their applications. The middleware layer of ilink TeamCall Server enables a simple and platform-independent communication between CSTA interfaces of PBXs from Deutsche Telekom, Siemens and Alcatel.

The TeamCall Server Development Kit supplies developers with all necessary information and tools to enable their applications to communicate with TeamCall Server.

TeamCall Server

TeamCall Server is the intelligent and complete solution for controlling a growing range of PBX products. Based upon STLI (Simple Telephony Interface) or NetTSPI/TAPI, TeamCall Server perfectly connects any CTI application to a PBX by translating telephony commands into the language of PBXs: 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`. For information on TAPI, please refer to `www.microsoft.com`. For further information on supported PBXs, see “PBX requirements” on page 15.

For CTI application development, TeamCall Server features a powerful set of APIs, which allow you to build your own new CTI applications, increase

features of existing ones or test new concepts in a rapid prototyping process.

Using the supported interfaces, TeamCall Server fits the requirements of any existing hardware and software. In most cases, a connection between TeamCall Server and a PBX will be established using a network socket connection. When no network connection is available, a V24 or an ISDN connection can be used.

This allows just-in-time development of CTI applications as well as installation and modification of CTI installations like call centers.

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 <>.

How this book is organized

- **Chapter 1: Introduction**

  This chapter provides a general overview of TeamCall Server.
- **Chapter 2: Installation and Setup**

  Lists the system requirements (hardware, software, PBX) and network requirements; describes the installation of TeamCall ServerTelasServer, necessary settings etc.
- **Chapter 3: Configuration**

  A description of TeamCall Server´s configuration files.

- **Chapter 4: SuperVisor Interface**

  Describes how to make changes to the setup while TeamCall Server is running, modification of logging behavior and debugging modes.

- **Chapter 5: STLI**

  Describes the Simple Telephony Interface.

- **Chapter 6: TeamCall LinkV24 - an IP-to-Serial-Router**

  Describes the installation of TeamCall LinkV24. A program that is necessary when using a PBX with only a serial interface.

- **Chapter 7: Log files**
- A list of logging files that are created when TeamCall Server is running.
- **Chapter 8: Troubleshooting**

  Describes how to solve certain problems when running TeamCall Server.

2

Installation and Setup

This chapter lists the hardware, software and PBX requirements and describes how to install, configure and test TeamCall Server.

Requirements

This section describes the PBX, computer and operating system requirements for running TeamCall Server.

PBX requirements

TeamCall Server currently supports the following PBX types with an installed CSTA interface:

- Alcatel Office E (4200) (CSTA I)

  Installed Ethernet interface required.
- Alcatel 4400 (CSTA I and CSTA II)

  Installed Ethernet interface required.
- Siemens Hicom 150 (CSTA II)

  ISDN connection: Installation of the ilink TeamCall software is required prior to installing TeamCall ServerTelasServer software. A computer running Windows NT with an installed ISDN interface card from Eicon (Diva Pro) with User-to-User signaling is also necessary.

Link via serial connection: Installation of the ilink TeamCall LinkV24 software is required prior to installing TeamCall Server software. After the ilink TeamCall LinkV24 package has been installed, install TeamCall Server package as outlined in this chapter. Edit the configuration file `Default.conf` according to section "Main configuration file entries" on page 29 and start TeamCall LinkV24 as described in chapter "TeamCall Link V24" on page 97. Next, start TeamCall Server following the steps in this chapter.

- Siemens Hicom 300 (CSTA I)

  The ACLC module plus Siemens CallBridge for WorkGroups software and a computer running Windows NT are required.

- Siemens A6

  see Telekom Octopus E300/800 below

- Telekom Octopus E300/800 (CSTA I)

  ISDN connection: Installation of the ilink TeamCall LinkISDN software is required prior to installing TeamCall Server software. In addition, a computer running Windows with an installed ISDN interface card from AVM with User-to-User signaling is required.

  Link via serial connection: Installation of the ilink TeamCall LinkV24 software is required prior to installing TeamCall Server software. After the ilink TeamCall LinkV24 package has been installed, install TeamCall Server package as outlined in this chapter. Edit the configuration file `Default.conf` according to section "Main configuration file entries" on page 29 and start TeamCall LinkV24 as described in chapter "TeamCall Link V24" on page 97. Next, start TeamCall Server following the steps in this chapter.

Hardware requirements

There are certain minimum hardware requirements for installing and running TeamCall Server. To run TeamCall Server in a network configuration, a network interface card has to be installed on the computer TeamCall Server will be installed on.

Windows NT 4

- Pentium II 300
- 128MBytes RAM

- Network interface card to connect to the PBX
- 5MBytes of available hard drive space plus around 50MBytes of additional space for log files

UNIX

- Pentium II 300, Sun Ultra 1, Sun Netra or Macintosh G3
- 128MBytes RAM
- Network interface card to connect to the PBX
- 5MBytes of available hard drive space plus around 50MBytes of additional space for log files.

Network requirements

In most cases, a connection between TeamCall Server and a PBX will be established via Ethernet and TCP/IP. TeamCall Server will connect to a PBX by addressing the PBX's TCP/IP address at different ports for different types of connections (e.g. STLI, SuperVisor etc.).

In order to connect TeamCall Server to a PBX with a network interface, TeamCall Server needs an installed TCP/IP protocol, provided by the OS.

If the PBX is connected to the computer via the serial interface, the package TeamCall LinkV24 and a special zero modem cable are needed. See chapter “TeamCall Link V24” on page 97.

Software requirements

You need one of the versions (or later) of the following operating systems and the mentioned software as the minimum system software configuration to run TeamCall Server.

Windows NT

- Windows NT 4.0 Workstation or Server
- Service Pack 3
- TCP/IP Services
- Unzip tool like WinZip

UNIX

- RedHat Linux 6.1, Linux Mandarke 7.1, S.u.S.E. Linux 6.3, FreeBSD 4, Sun Solaris 2.x, MacOS X Server or Unixware 2.x, Openstep/ NeXTStep
- TCP/IP
- Gnutar
- Gnuzip

Installing TeamCall Server

This section describes the installation of the TeamCall Server software package including the steps of the setup, e.g. settings for network and logging.

Installing on Windows NT

To install TeamCall Server on a computer running Windows NT, follow these steps:

Note: Before you begin, make sure you have the TCP/IP-address and the TCP/IP port number of the PBX handy.

1. Log in as `Administrator`.
2. Use `WinZip` or a similar tool to unpack the content of the `TeamCall.zip` package to a temporary location of your hard disk like `c:\temp\teamcall`.
3. Double-click on `setup.exe` to start the installation program.
4. Follow the on-screen instructions.

The following directories and files will be created in the installation path:

- `CSTAServer.exe` (TeamCall Server binary.)
- `Default.conf` (The main configuration file.)
- `tcdoc` (Directory containing TeamCall Server documentation in PDF format.)

- `log` (Directory containing the log files.)

The setup program installs TeamCall Server as a service. This means TeamCall Server will be started when Windows NT is started. For the configuration of TeamCall Server see the following sections.

Installation on UNIX platforms

To install TeamCall Server, only the files of the TeamCall Server package need to be copied to the installation directory. Additionally, two configuration files should be provided. See the next two sections for more details.

To unpack and install TeamCall Server under UNIX, follow these steps:

1. Log in as `root`.
2. Copy TeamCall Server binary package to a temporary location on your computer, e.g. `/tmp`.
3. Extract the package using the tools `gtar` and `gzip`:

   `gtar -xvpzf tcserver.tgz`

   A directory `teamcall` will be created which contains the following two files and single directory:

   `CSTAServer` (TeamCall Server binary.)
   `Default.conf` (The main configuration file.)
   `csta` (Script used by `init`.)
   `tcdoc` (Directory containing TeamCall ServerTelasServer documentation file in PDF format.)
4. Change to the directory `teamcall` using the `cd` command:

   `cd teamcall`
5. Copy the binary file to a location where you usually install 3rd party software, e.g. `/usr/local/bin` or `/opt/bin` using the `cp` command:

   `cp -p CSTAServer /usr/local/bin/`
6. Copy the configuration file to a location where you usually install 3rd party configuration files, e.g. `/usr/local/etc/`:

   `cp -p Default.conf /usr/local/etc/`
7. Copy the directory containing the documentation to a location where you normally install 3rd party documentation files, e.g. `/usr/local/doc` using the `cp` command:

   `cp -pR tcdoc /usr/local/doc/`

8. Edit the configuration file `Default.conf` according to the description below to match your local network configuration.

Setting up TeamCall Server

For the configuration of TeamCall Server, two files should be created and/or modified:

- the main configuration file and
- a configuration file for the local devices to be monitored.

Every TeamCall Server software package contains a main configuration file, named `Default.conf`.

It includes basic settings for the configuration of TeamCall Server. This file should be used as a source for generating your own new configuration files. Using this file, TeamCall Server can be started right after shipping without modifying any configuration. For further details see the chapter "SuperVisor interface" on page 41.

The main configuration file

The main configuration file contains the necessary data for TeamCall Server to communicate with the PBX.

The character # will be interpreted as beginning of a remark, which means that everything written after a # will be ignored.

The last character in the file must be a new-line (see bottom of the file `Default.conf`).

Each entry is identified by a key word, followed by a whitespace (i.e. SPACE or TAB) and the value for the key. The entries may be placed in any order, even though we suggest to keep the structure of the current file for better readability.

Note: Unknown keys are ignored.

Editing the main configuration file

Under UNIX, the TeamCall Server main configuration file must be modified to match your local network requirements. Under Windows NT this is done during the setup process.

Note: Before editing the files, make sure to have the TCP/IP address and the TCP/IP port number of the PBX handy.

To edit the main configuraton file, follow these steps:

1. Locate the main configuration file `Default.conf`.
2. Launch your favorite text-editing application e.g. `vi`.
3. Open the file `Default.conf`.
4. Find the entry `cstaLinkAddress`.
5. Change the IP-address after `cstaLinkAddress` to the IP-address of your PBX.
6. Find the entry `logDir`.
7. Change the path after `logDir` to the directory where TeamCall Server log files will be created.
8. Find the entry `cstaLinkPort`.
9. Change the TCP/IP port after `cstaLinkPort` to the TCP/IP port number of your PBX. The default TCP/IP port number is 2555.
10. Find the entry `license`.
11. Enter your personal license string after `license`.

    If you don't enter a license string, TeamCall Server will run in demonstation mode.
12. Find the entry `switchVersion`.
13. Enter an appropriate value after `switchVersion` according to the table below.
14. Save your changes and exit the text-editing application.

Values for `switchVersion`

switchVersion	default	switchVendor	switchId	switchRelease
100		Alcatel	A4200	any
150	*	Alcatel	A4400	any
200		Siemens	A6	6.3
201	*	Siemens	A6	6.4

Values for `switchVersion`

switchVersion	default	switchVendor	switchId	switchRelease
210	*	Siemens	Hicom150	any
300	*	Siemens	Hicom300E-intl	2.0
310		Siemens	Hicom300E/ Hicom300H	3.0
311		Siemens	Hicom300E-UK	3.1
320		Siemens	Hicom300E-US/Canada	6.6

Testing TeamCall Server

After you have installed and configured TeamCall Server you need to check that it works properly. To do so, you have to run TeamCall Server and perform a quick test.

Starting TeamCall Server under Windows NT

If you are logged in after restarting your computer as suggested during installation, TeamCall Server should run. If you have not already restarted your computer, please do so now.

Starting TeamCall Server under UNIX

Under UNIX, you can either start the TeamCall Server binary manually when you need it, or you can configure it to start automatically while booting.

Starting manually

Follow these steps to start TeamCall Server manually using the main configuration file `Default.conf`:

1. Change to the directory where you installed the TeamCall Server binary using the `cd` command.

   `cd /usr/local/bin`

2. Start the binary `CSTAServer` using the `-c` and `-PBX` switches to provide the absolute path to the main configuration file and the PBX type used.

Note: Find out possible values for `-PBX (and other options)` by starting `CSTAServer` with the `-?` switch: **`./CSTAServer -?`**.

`CSTAServer -c /usr/local/etc/Default.conf -PBX a6`

If the startup was successful, the following messages should appear:

```
-22071556.931882: Connecting to x.x.x.x:zzzz (IP
x.x.x.x)...
-22071556.922619: Waiting for connection to PBX/Link.
-22071556.916075: Login port 26535 added.
-22071554.348725: Connected to PBX/Link, sending
login sequence .
-22071553.609504: Log in to PBX/Link successful
-22071553.587936: PBX link up
```

Starting automatically

You can also install TeamCall Server to start while your OS boots. Please note that this option is for experts only, since we only provide an example init script named `csta` that must be configured and installed manually. It also requires knowledge of the specific boot process of the type of UNIX you are using. This will only work on Linux or Solaris systems. (On BSD-style systems, TeamCall Server can be started automatically by adding a line to an rc script like `/etc/rc.local`,
e.g. `/usr/local/bin/CSTAServer -d -PBX a6 &`)

1. Log in as `root`.
2. Change to the directory, where you unpacked the TeamCall Server

package.

```
cd /tmp/teamcall
```

3. Copy the file `csta` to the location where start-up scripts for the `init` process are kept.

   ```
   cp -p csta /etc/rc.d/init.d
   ```

4. Install the start-up script according to your version of `init`.
5. Customize it according to your needs.

   You can also start and stop TeamCall Server manually using the script's start/stop switches, e.g.

   ```
   /etc/rc.d/init.d csta start
   /etc/rc.d/init.d csta stop
   ```

Note: Be sure to run TeamCall Server in daemon mode by using the `-d` switch when starting TeamCall Server from within an rc script.

Testing the installation using STLI

Using a few simple STLI commands you can check if TeamCall Server works properly. Using STLI, you will place a call from one telephone to a second telephone, answer the call and hang up.

You will need to know the TCP/IP-address of the computer where TeamCall Server runs and the TCP/IP port number of the login port. You can check the configured TCP/IP port number of Login port next to the entry `loginPort` within the main configuration file `Default.conf`. The default TCP/IP port number is 26535. Also, you will need to have access to two telephones connected to the PBX. Windows NT users should start the `Command Prompt` application.

Use the following steps to connect to the STLI interface using the `telnet` command, place a call from one telephone to a second, answer the call and hang up:

1. Start telnet with the IP-address of the computer on which TeamCall Server runs, followed by a space and the login port number.

   ```
   telnet ctihost 26535
   ```

   The following messages should appear:

```
Trying x.x.x.x...
Connected to x.x.x.x.
```

2. 2. Start the STLI Session

   STLI

   ```
   error_ind SUCCESS STLI
   ```

3. Start a monitor for two telephones connected to the PBX, e.g. 100 and 101.

   MonitorStart 100

   The following message should appear:

   ```
   error_ind SUCCESS MonitorStart
   ```

   MonitorStart 101

   The following message should appear:

   ```
   error_ind SUCCESS MonitorStart
   ```

4. Place a call from telephone 100 to telephone 101 using the `MakeCall` command.

   MakeCall 100 101

   The following messages should appear and telephone 101 should ring.

   Please note that the numbers shown in this example may differ from the messages you receive.

   ```
   error_ind SUCCESS MakeCall
   Initiated 100 makeCall
   DeviceInformation 100 1 (0070:initiate)
   ```

5. Pick up the receiver from telephone 100 using the `AnswerCall` command.

   When using a PBX from Alcatel, please note that this step is only necessary, if the auto-answer mode is disabled for this telephone. When using a PBX from Siemens/Deutsche Telekom, this step can be omitted.

   AnswerCall 100

   The following messages should appear and telephone 100 should ring:

   ```
   error_ind SUCCESS AnswerCall
   Originated 100 newCall 100 101 ""
   DeviceInformation 100 1 (0070:connect)
   ```

```
Delivered 100 newCall 101 100 101 ""
DeviceInformation 100 1 (0070:connect)
Delivered 101 newCall 101 100 101 ""
DeviceInformation 101 1 (0070:alerting)
```

6. Answer the call at telephone 101 using the `AnswerCall` command.

   `AnswerCall 101`

   The following messages should appear:

```
error_ind SUCCESS AnswerCall
Established 101 newCall 101 100 101 ""
DeviceInformation 101 1 (0070:connect)
Established 100 newCall 101 100 101 ""
DeviceInformation 100 1 (0070:connect)
```

   Now, telephone 100 and telephone 101 are connected.

7. Terminate the connection using the `ClearConnection` command.

   `ClearConnection 100`

   The connection should terminate and the following messages should appear:

```
error_ind SUCCESS ClearConnection
ConnectionCleared 100 normalClearing 100 {}
DeviceInformation 100 0 ()
ConnectionCleared 101 normalClearing 100 {101}
DeviceInformation 101 1 (0070:connect)
ConnectionCleared 101 normalClearing 101 {}
DeviceInformation 101 0 ()
```

8. Quit the STLI session using the `BYE` command.

   `BYE`

   The following messages should appear:

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

Troubleshooting

General troubleshooting

If the installation and start-up of TeamCall Server was successful, you should be able to connect to TeamCall Server's SuperVisor interface, STLI and NetTSPI.

To check the availability of the above mentioned interfaces, follow these steps:

1. Launch the text editing application of your choice.
2. Open the `Error.log` file.
3. Look for the following three entries:

   ```
   SV added.
   STLI added.
   TAPI added.
   ```

If one of the entries `STLI added.` or `TAPI added.` is missing, the connection to STLI and/or NetTSPI was unsuccessful. A networking problem may have occurred. Check the IP-address of TeamCall Server and the TCP/IP port numbers of STLI and NetTSPI within the main configuration file. You should also check your networking configuration.

To find more information about possible issues, see also chapter “Troubleshooting” on page 113.

Log files

TeamCall Server logs all of its activities in different log files when activated and configured within the main configuration file. For information on how to activate logging and for descriptions of all options, see chapter “Log files” on page 101.

3

Configuration

TeamCall Server uses two different configuration files:

`Default.conf` — The main configuration file. Every TeamCall Server software package contains this configuration file. It includes basic settings for the configuration of TeamCall Server. This file should be used as a source for generating your own new configuration files.

`DeviceMonitor.conf` — Device configuration file for permanently monitored devices. A device can be monitored from every startup until shut down of TeamCall Server by adding it to `DeviceMonitor.conf`. The path to this configuration file is set within the main configuration file using the key `deviceMonitorConfFile`.

Main configuration file entries

This section describes all entries of the configuration file sorted in alphabetical order.

adminAddress

Key `adminAddress <TCP/IP address>`

Example `adminAddress 127.0.0.1`

Description Specifies the TCP/IP address, at which the NetTSPI Service Provider can be reached. If no address has been specifed or the key does not exist, the default values will be used. The default value is 127.0.0.1. This key is related to the key `securityEnabled`.

baudRate

Key `baudRate <baudRate>`

Example `baudRate 9600`

Description Specifies the baud rate for the connection to the PBX. Default value is `9600`.

callLogEnabled

Key `callLogEnabled <0/1>`

Example `callLogEnabled 1`

Description Specifies, whether calls shall be logged or not. `0` turns logging off, `1` turns logging on. Default value is `0`.

cstaLinkAddress

Key `cstaLinkAddress <TCP/IP address>`

Example `cstaLinkAddress 192.168.1.1`

Description Specifies the TCP/IP address of the PBX. If the `cstaLinkAddress` is not specified or the key does not exist, the default value will be used. The default value is `127.0.0.1` (localhost).

If the `cstaLinkAddress` has not been specified in the main configuration file it may be specified by starting TeamCall Server using the `-a` option. For further details on TeamCall Server command line parameters see “Starting and shutting down TeamCall Server” on page 41.

cstaLinkPort

Key `cstaLinkPort <TCP/IP port number>`

Example `cstaLinkPort 2555`

Description Specifies the port number for TCP/IP socket connection. If the port number is not specified or the key does not exist, the default value will be used. The default value is 2555.

cstaLogEnabled

Key `cstaLogEnabled <0/1>`

Example `cstaLogEnabled 1`

Description Specifies, whether plain CSTA messages shall be logged or not. `0` turns logging off, `1` turns logging on. Default value is `0`.

dataBits

Key `dataBits <7/8>`

Example `dataBits 8`

Description Specifies the number of data bits used for the connection to the PBX via TeamCall LinkV24. Default value is `8`.

debugLevel

Key `debugLevel <0-9>`

Example `debugLevel 0`

Description Specifies the debug level at which TeamCall Server will be started. The default value for running TeamCall Server is `0`. For different levels of debugging, the `debugLevel` may be set to values between `0` and `9`. If `debugLevel` has not been specified or the key does not exist, the default value will be used.

Possible values for the `debugLevel` are:

`0`: enable logging for performance analysis

`1`: allow logging of CSTA calls

`2`: allow logging of devices (adding and removing)

`3`: allow logging of NetTSPI

`5`: ASN1 message decoding and logging (CSTAxxxAdaptor)

`6`: system status messages, eg. `link down` (CSTAxxxAdaptor)

`7`: messages between model and adaptor (CSTAController)

`8`: protocol trace between adaptor and PBX (CSTAxxxAdaptor)

`9`: messages between adaptor and PBX (CSTAxxxAdaptor)

deviceMonitorConfFile

Key `deviceMonitorConfFile <absolutePath>`

Example `deviceMonitorConfFile /opt/teamcall/DeviceMonitor.conf`

Description Specifies the name and path of the device configuration file, which specifies the devices to be monitored. An absolute path should be specified. The default value for this key is

`<installationPath>/DeviceMonitor.conf`

There are three ways to monitor a device at the PBX:

A device can be monitored from every startup of TeamCall Server until shut down by adding it to the device configuration file. It also can be added in a SuperVisor session for the lifetime of the currently running TeamCall Server or it can be added temporarily by calling the appropriate requests in a STLI session or a NetTSPI session. For further details, see "Device configuration file" on page 38.

license

Key `license <licenseKey>`

Example
```
license 3ac6d7af928aeda3d7c40fff4e1d12a42f094de7a61d72
07e0d7bf4008fabae6bb17c3975b99abc1137d5cf3a253e141c21918
5307dc4c2e2c50e0f2111ca419c806ad72b014c0f93960609f9ea261
64cb4c51bb9978e87862352c1580a2419032738cc8frz5394b49
```

Description Specifies the license key for TeamCall Server. There is no default value. If no license key is specified, TeamCall Server will run in demonstration mode.

linkedPBX

Key `linkedPBX <a6/hicom150>`

Example `linkedPBX a6`

Description Specifies the PBX type for TeamCall LinkV24. Default is `hicom150`.

logDir

Key `logDir <directory>`

Example `logDir /tmp/logs`

Description Directory where to log to, prepended to logfiles. If `logDir` has not been specified, the logfiles will be created within the default path. The default path for logging is the installation path.

If the path specified in `logDir` does not exist or is not writeable, or if the key `logDir` does not exist in the main configuration file, TeamCall Server log files will be created in the installation path of TeamCall Server.

logFileMaxBackups

Key `logFileMaxBackups <0-n>`

Description Number of backup files per log file. Default is `1` (one log file and one backup file). If the value of `logFileMaxBackups is 0`, no backups will be made. After reaching `logFileMaxSize`, the current log file will be deleted, newly created, then logging starts again.

logFileMaxSize

Key `logFileMaxSize <Kbytes/-1>`

Example `logFileMaxSize 300`

Description Maximum size of a log file in Kbytes. The default value is 100 Kbytes.

If 0 <= `logFileMaxSize` <= 100 Kbytes, the value will be set to 100 Kbytes.

If the value of `logFileMaxSize` is `-1`, there will be no maximum limit for the size of the log file. It is up to the administrator to control the size of the log file.

If the value of `logFileMaxSize` is `-1` no backup file will be created.

loginPort

Key `loginPort <TCP/IP port number>`

Example

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

Description To start a certain session, a `loginPort` has to be used. After establishing a login session, the user decides what session type to start. When a session has been started, it is kept up during the connection. To start a different session, the login session has to be initiated again.

For example, after starting a login session via telnet, one of these three sessions can be started by issueing the appropriate command:

`STLI` — starts a STLI session

`SuperVisor` — starts a SuperVisor session

`NetTSPI` — starts a NetTSPI/TAPI session (not further documented)

A correct entry gets the following response:

`error_ind SUCCESS <Session>`

The user has to wait for a positive response before new commands can be issued. The login session can be closed with the command BYE. The default value for `loginPort` is 26535.

maxPendingRequests

Key `maxPendingRequests <seconds>`

Example maxPendingRequests 27

Description Key to determine the length of the list of pending requests. Default is 20; minimum is 20, maximum is 1000.

Every request which is sent from TeamCall Server (via the session) to the PBX is usually followed with a response (indication) from the PBX.

Example in STLI:

`MakeCall 100 101`

`error_ind SUCCESS MakeCall`

There is always the possibility, that the PBX cannot handle the amount of requests anymore and

- a response is sent very late or
- no response is sent at all.

To avoid this, additional requests from the session are put on a waiting list to take some pressure off the PBX.

All requests from the sessions are received by a central object (CSTAController) of TeamCall Server and then passed on to the PBX. Every request is put on a list of pending requests. After a response was received (`error_ind SUCCESS, error_ind UniversalFailure`, etc.) the request is taken off the list. If the list of pending requests has reached a certain size which is set by the key `maxPendingRequests`, additional requests are temporarily stored in another list, which is called the list of queued requests. After a request got a response and was removed from

the list of pending requests, the next request from the list of queued requests is passed on.

The length of the list of queued requests is not limited. The length of the list of pending requests is set with this key. See also key `requestTimeout`.

Note: The value for `maxPendingRequests` should not be set too high. Tests with Siemens Hicom150 and Alcatel 4400, after a certain number of requests, resulted in requests not beeing answered or rejected.

parity

Key `parity <even/odd/none>`

Example `parity even`

Description Specifies if and what type of parity is used for the connection to the PBX via TeamCall LinkV24. Default value is none.

prefixPlan

Key `prefixPlan "<pattern>/<replacement>?[:<pattern>/<replacement>]*"`

Example `prefixPlan "+90/90:+9/9:+/90"`

Description Specifies replacement rules for leading strings of dialing numbers in call events. A rule consists of two strings, separated by a slash character: the string to be replaced and the replacing string. Rules are separated by colons. This defines how the PBX reports all numbers of calling and called parties.

requestTimeout

Key `requestTimeout <seconds>`

Example `requestTimeout 30`

Description Key to set the maximum time in seconds which a request stays on the pending or queued list. The default value is `30`, the minimum value is `20` and the maximum is `-1` (unlimited).

To avoid having requests on the pending or queued list forever if the PBX is overloaded, there is a time limit for requests on both lists which is set by the key `requestTimeout`. Both lists are constantly monitored and if a

request has reached the time set with `requestTimeout`, it is removed. In this case, the following error message is sent to the session:

`error_ind REQUESTTIMEOUT <Request>`

See also key `maxPendingRequests`.

securityEnabled

Key `securityEnabled <0/1>`

Example `securityEnabled 0`

Description This key is related to the key `adminAddress`. Possible values are `0` (disabled) and `1 (enabled)`. If the key exists and is set to `1`, Admin Server has to be running to establish a NetTSPI connection to TeamCall Server. The default value is `1`.

serialDevice

Key `serialDevice <serialDevice>`

Example `serialDevice COM1`

Description Interface to establish a serial connection to the PBX. The value depends on the OS, for example: `Com1` for Windows platforms, `/dev/ttya` for Unix platforms.

serialLogEnabled

Key `serialLogEnabled <0/1>`

Example `serialLogEnabled 1`

Description Log all serial port traffic messages from and to TeamCall LinkV24 to the file `<logdir>/SerialTokenizer.log`. If the key is set to `1`, serial port logging is enabled, if it is set to `0`, serial port logging is disabled.

stopBits

Key `stopBits <1/2>`

Example `stopBits 1`

Description Specifies the number of stop bits used for the connection to the PBX via TeamCall LinkV24. Default value is `1`.

switchVendor

Key `switchVendor <switchVersion>`

Example `switchVendor 210`

Description Specifies the vendor, model and software release of the PBX used..

Values for `switchVersion`

switchVersion	default	switchVendor	switchId	switchRelease
100		Alcatel	A4200	any
150	*	Alcatel	A4400	any
200		Siemens	A6	6.3
201	*	Siemens	A6	6.4
210	*	Siemens	Hicom150	any
300	*	Siemens	Hicom300E-intl	2.0
310		Siemens	Hicom300E/ Hicom300H	3.0
311		Siemens	Hicom300E-UK	3.1
320		Siemens	Hicom300E-US/Canada	6.6

tcpLogEnabled

Key `tcpLogEnabled <0/1>`

Example `tcpLogEnabled 1`

Description Log all TCP/IP messages from and to TeamCall LinkV24 to the file `<logdir>/TCPTokenizer.log`. If the key is set to `1`, TCP/IP logging is enabled, if it is set to `0`, TCP/IP logging is disabled.

Example main configuration file

```
adminAddress            10.10.6.13
##### TCP/IP Addresses and Ports
# TCP address of PBX
cstaLinkAddress         10.1.1.1
# Port of PBX
cstaLinkPort            2555
# TCP Port for login
loginPort               19000
##### Logging
logDir                  /tmp/debug
#directory where to log to, prepended to logfiles
callLogEnabled          1
#0=false, 1=true ; logs to callLogFile
cstaLogEnabled          1
#0=false, 1=true ; logs to cstaLogFile.
debugLevel              9
logFileMaxBackups       3
logFileMaxSize          250
##### Initial Monitoring
deviceMonitorConfFile   /opt/pbx//DeviceMonitorDebug.conf
#should be absolute path
##### Replacement Rules for replacing leading strings in
Dialing Numbers
prefixPlan              "+00/00:+0/0:+/00"
```

Device configuration file

When starting TeamCall Server, the device configuration file containing device numbers will be read. These devices are monitored from startup of TeamCall Server. No further request is necessary.

Note: The initially monitored devices may be removed at runtime via a SuperVisor session using the request: `RemoveDeviceMonitor <device>`

This request does not modify the device configuration file. For further details on `RemoveDeviceMonitor` see “"sticky" adding/removing new devices to be monitored” on page 51.

Every device connected to a PBX has a unique internal number. The device configuration file must contain an entry for each device, a monitoring point shall be set for. The following restrictions apply to this file:

- one device number per line
- no blank lines in between
- a new line after the last entry
- no remarks

Example

```
111
185
123
168
```

The device entries can be placed in any order.

Refer to “More about device monitoring” on page 51 for details.

The name of this file (default: `DeviceMonitor.conf`) must be specified using its absolute path as a value of the key `DeviceMonitorConfFile` in the main configuration file.

Every valid device listed in the device configuration file will be monitored from startup until shut down of TeamCall Server. Further devices may be monitored by using the appropriate requests within different sessions. See “Monitoring a device” on page 60.

These devices will only be monitored as long as at least one session is running.

Important notice

Please note that it may take a while before a monitoring point has been added to a device. For every entry in the device configuration file a request is sent to the PBX. The response time until these requests are acknowledged depends on many circumstances:

- network load

- load of the PBX
- load of the TeamCall Server computer

Due to too much traffic, it may be possible that the responses of the PBX will be delayed or not sent at all. This could cause synchronization problems.

As long as a monitoring point is not active, no event reports for this device can be received.

Additionally, no sessions will be able to add a monitoring point to this device (an attempt gets the response `error_ind PENDINGREQUEST`).

The progress of adding monitoring points can be supervised from the SuperVisor session with the request `ShowDeviceMonitors`, see section “More about device monitoring” on page 51.

For adding a greater number of monitoring points (more than 40), we recommend using the request `AddDeviceMonitor` from within the SuperVisor session. After issueing each `AddDeviceMonitor` request you should wait to get the response of the PBX. This way you make sure that monitoring points will be added correctly in the right order. Also you receive feedback from the PBX if it gets too busy. In that case any command will receive a negative response.

4

SuperVisor interface

Part one of this chapter describes starting and shutting down TeamCall Server, running it at different debugging levels and all the administrative tasks which can be fulfilled when TeamCall Server is not running.

Part two of this chapter describes the the SuperVisor interface. The SuperVisor interface is, like STLI, a network user interface, which allows to connect to TeamCall Server with supervisor status.

Starting and shutting down TeamCall Server

In the following section you will find the command line options and default settings for starting TeamCall Server.

As mentioned in the chapter “Installation and Setup” on page 15, there are default settings, which allow you to run TeamCall Server without modifying the shipped TeamCall Server software package.

Starting TeamCall Server

Syntax for all operating systems:

```
CSTAServer [-a <TCP/IP address>] [-c <configFile>] -PBX
<pbxType>
```

Additional options for Microsoft Windows NT:

```
[ -i | -u | s ]
```

Additional option for UNIX systems:

`[-d]`

Possible startup scenarios

Syntax `CSTAServer -PBX <pbx>`

Description Starting TeamCall Server with default settings. In this case, TeamCall Server looks for the main configuration file `Default.conf` in its installation path (i.e. where the TeamCall Server binary is installed).

Use the -PBX switch to specify the PBX type you are using. It is mandatory.

Note: Find out possible values for `-PBX (and other options)` by starting `CSTAServer` with the `-?` switch: **`./CSTAServer -?`**.

If the main configuration file was not found, TeamCall Server terminates with an error-message.

If the main configuration file was found, TeamCall Server starts up with the settings loaded from this file.

Note: The default main configuration file shipped with the TeamCall Server software package has no entries set for the TCP/IP address of the PBX and the device configuration file (in fact these entries are inserted as remarks). To run TeamCall Server without making modifications, it must be started with the command line option `-a` (see next section).

The device configuration file contains entries for all devices which are to be monitored while TeamCall Server is running.

If the device configuration file does not exist (or is empty), no monitoring points for monitoring devices will be set. In this case, monitoring points can be added in a "sticky" way, using the `AddDeviceMonitor` request or by sending

- a `MonitorStart` request (see “MonitorStart” on page 61) from a STLI session or
- if `AddDeviceMonitor` has been used, the specified device will be monitored until TeamCall Server is shut down or the monitoring will be stopped using the `RemoveDeviceMonitor` request. In the other two cases, state changes of the specified device will be traced

as long as this device is monitored in at least one STLI/NetTSPI session (see lower section for details).

If the main configuration file exists, but does not have entries for `cstaLinkPort`, `loginPort`, the following internal default values will be used:

```
cstaLinkPort            2555
loginPort               26535
TAPITCPPort 26535       # TCP Port for NetTSPI
STLITCPPort 8000        # TCP Port for STLI
```

Syntax `CSTAServer -a <cstaLink TCP/IP address> -pbx <pbxType>`

Description TeamCall Server receives the TCP/IP address of the PBX from the command line. If `<cstaLink TCP/IP Address>` is invalid, TeamCall Server will terminate with an error message. All other values are loaded from the main configuration file. If it does not exist, TeamCall Server will start using the default values.

```
cstaLinkPort 2555
loginPort 26535
```

Syntax `CSTAServer -c <configFile> -pbx <pbxType>`

Description TeamCall Server starts up using the settings as specified in the main configuration file `<configFile>` as an absolute path. If this file was not found, TeamCall Server will terminate with an error message.

Syntax `CSTAServer -a <cstaLink TCP/IP address> -c <configFile> -pbx <pbxType>`

Description TeamCall Server receives the TCP/IP address of the PBX from the command line and all other settings from the specified main configuration file `<configFile>` as an absolute path. If the `<cstaLink TCP/IP Address>` is invalid, TeamCall Server will terminate with an error message.

Note: The following command line option is unly valid for the UNIX version of TeamCall Server.

Syntax `CSTAServer -d -pbx <pbxType>`

Description Run TeamCall Server as a daemon. When using `-d` the `CSTAServer` process gets forked and sent into the background. This is especially useful when starting TeamCall Server automatically using a start-up script.

Note: The following command line options are for the Windows NT version of TeamCall Server only.

Syntax `CSTAServer -i -pbx <pbxType>`

Description Install as Windows NT Service option. This option creates an entry for TeamCall Server in the list of all processes to be started when Windows NT boots. This option cannot be combined with the following options: `-a`, `-c`, `-u` and `-s`. The main configuration file has to be placed in the same directory where the TeamCall Server binary is installed.

Syntax `CSTAServer -u -pbx <pbxType>`

Description Uninstall as Windows NT Service option. This option removes the entry from the list of all processes to be started, when Windows NT boots. This option cannot be combined with the following options: `-a`, `-c`, `-i` and `-s`.

Syntax `CSTAServer -s -pbx <pbxType>`

Description Do not run as Windows NT Service option. When starting TeamCall Server from within the `Command Prompt` application on Windows NT, it will run as a background process. Starting it with the `-s` option runs TeamCall Server as a normal process. This option can be combined with the `-a` and `-c` options, but not with `-i` and `-u` options.

Example session: Starting TeamCall Server

For normal use (i.e. `debugLevel=0`) TeamCall ServerTelasServer should be run as a background process.

To start TeamCall Server from within a command prompt (`Command Prompt` application or UNIX shell), type:

`CSTAServer -pbx <pbxType>`

under Windows NT or

`CSTAServer -pbx <pbxType> &`

under UNIX.

When starting, TeamCall Server returns status messages.

Shutting down TeamCall Server

In order to shut down TeamCall Server, a SuperVisor session must be established via the login port. Typing SHUTDOWN in the SuperVisor session will shutdown TeamCall Server. In the example below, the login port is 26535.

Example

```
telnet localhost 26535
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
SuperVisor
error_ind SUCCESS SuperVisor
SHUTDOWN
error_ind SUCCESS SHUTDOWN
Connection closed by foreign host.
```

TeamCall Server has been shut down.

Administering TeamCall Server using the SuperVisor interface

The SuperVisor interface allows to open a network connection to TeamCall Server via telnet and remotely administer settings of TeamCall Server.

For more information about CSTA specific error messages visit the web site of ECMA at www.ecma.ch. You will find a list of ilink specific error messages below in the section “Error messages” on page 53.

In order to open a connection to TeamCall Server, you have to know the login port number. The login port number is defined in the main configuration file. In the example below, the port number is 26535.

When a telnet connection has been established, an administrator can view the current configuration, enable/disable logging, change paths for logging, reload a new configuration from a specified configuration file (for example to change the debugging level) and manually add and/or remove devices to be monitored.

Example Starting a SuperVisor session:

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

Viewing the current configuration

To view the current configuration, in the SuperVisor session type `Config`.

Example

```
Config
error_ind SUCCESS Actual Config:
logDir : /Network/Users/csta/a4400-2/logs
binDir : /opt/ilink/CSTAServer/etc/
errorLogFile : Error.log
deviceLogFile : Devices.log
callLogFile : Calls.log
cstaLogFile : CSTA.log
sysLogFile : Sys.log
cstaLinkAddress : 10.1.1.1
cstaLinkPort : 2555
loginPort : 26535
adminAddress : 10.1.1.2
debugLevel : 9
callLogEnabled : 0
cstaLogEnabled : 1
logFileMaxSize : 100
logFileMaxBackups : 30
securityEnabled : 1
snapshotDevicePeriod : 300
deviceMonitorConfFile : /Network/Users/csta/a4400-2/
DeviceMonitor.conf
```

```
prefixPlanAsString : +00/00:+0/0:+/00
prefixPlan : ((+00,00),(+0,0),(+,00))
```

TeamCall Server reports the current settings as listed above.

Activating the changes of a configuration file

To run TeamCall Server with new settings from a changed configuration file, TeamCall Server must be shut down and started up again.

Viewing supported requests

In general, a PBX supports only some of the requests that are defined in CSTA Phase I, II or III. On top of that, TeamCall Server does not yet support all of the requests defined in CSTA Phase I, II or III. Therefore, TeamCall Server has a built-in list of supported and unsupported requests of each PBX specified using the key `switchVersion`.

Syntax `IsSupported <Request>`

Description Returns `1` (true) or `0` (false), if the request is supported by the PBX.

Examples `AnswerCall` is supported:

```
IsSupported AnswerCall
error_ind SUCCESS IsSupported AnswerCall 1
```

`SnapshotDevice` is supported:

```
IsSupported SnapshotDevice
error_ind SUCCESS IsSupported SnapshotDevice 1
```

`MakePredictiveCall` is unsupported:

```
IsSupported MakePredictiveCall
error_ind SUCCESS IsSupported MakePredictiveCall 0
```

Syntax `SupportedRequests`

Description Returns a list with all supported requests. The number right before each request indicates that the request is enabled (`1`) or disabled (`0`).

Example

```
SupportedRequests
error_ind SUCCESS SupportedRequests
0 AlternateCall
1 AnswerCall
```

```
1 ClearConnection
1 ConferenceCall
1 ConsultationCall
0 DivertCallDeflect
0 HoldCall
1 MakeCall
1 QueryDevice
1 ReconnectCall
1 RetrieveCall
1 SetDeviceFeatureMessageWaiting
1 SetDeviceFeatureForwardImmediateOn
1 SetDeviceFeatureForwardImmediateOff
1 SetDeviceFeatureAgentLoggedIn
1 SetDeviceFeatureAgentLoggedOut
1 SetDeviceFeatureAgentNotReady
1 SetDeviceFeatureAgentReady
1 SetDeviceFeatureAgentBusy
1 SetDeviceFeatureAgentWorkingAfterCall
0 SingleStepTransfer
1 TransferCall
0 EscapeService
0 ChangeMonitorFilter
1 MonitorStart
1 MonitorStop
0 SnapshotDevice
```

Logging

Logging allows viewing a couple of different events. However, to enable logging, a valid path for creating logfiles must be specified. If no path has been set within the main configuration file or the path is invalid, log files will be created in the same folder where the TeamCall Server binary is stored.

Note: The size of the log files will only be limited by the available disk space of the volume, where the log files were created and the OS's file size restrictions. To ensure that logging will not stop the whole system by running out of space when logging a long period of time, the size of log files should be watched or a separate volume with limited size for logging should be used.

The log files

When logging has been activated, up to five different log files will be created depending on the chosen debug level:

An error log file named `Error.log`, a device log file named `Devices.log`, a calls log file named `Calls.log`, a system log file named `Sys.log` and a CSTA log file named `CSTA.log`.

The error log file contains everything that will be prompted to `stdout`.

The device log file contains information about the state of the monitored devices.

The calls log file contains information about the calls and the devices which are connected to these calls and their states.

The sys log file contains status, warning and error messages.

The CSTA log file contains the plain ASN1 Messages that were exchanged between TeamCall Server and the PBX.

The level of detail of the logged messages depends on the debug level. Refer to the upper sections for details about the debug levels.

Temporary changing path for log files

Typing `SetLogDir <Path>` temporarily changes the path for logging. `<Path>` must be a valid absolute path. After shutting down and starting up TeamCall Server, the log file path will be reset to the value of the main configuration file.

Example

```
SetLogDir /tmp/a4400
error_ind SUCCESS SetLogDir
```

Enable/disable logging

Logging of call events and CSTA events can be enabled and disabled by issuing the following requests:

Example

```
EnableLog call
error_ind SUCCESS EnableLog
DisableLog call
error_ind SUCCESS DisableLog
EnableLog csta
error_ind SUCCESS EnableLog
DisableLog csta
error_ind SUCCESS DisableLog
```

In contrast to the CSTA log and call log, which can be enabled and disabled, an error log file will always be generated if `logDir` exists. All messages from TeamCall Server which are sent to `stderr` will also be stored in this file.

Debugging modes

The standard debug level of TeamCall Server is `0`. Only for troubleshooting purposes the debug level should be changed to a value different to `0`. See chapter “Troubleshooting” on page 113.

Note: Debugging affects the performance of TeamCall Server.

The currently supported debug levels are as follows (excerpt from `Default.conf`).

```
# 0: enable logging for performance analysis
# 1: allow logging of csta calls
# 2: allow logging of devices (adding and removing)
# 3: allow logging of NetTSPI
# 5: ASN1 message decoding and logging (CSTAxxxAdaptor)
# 6: System status messages, eg. "link down"
(CSTAxxxAdaptor)
# 7: Messages between model and adaptor (CSTAController)
```

```
# 8: Protocol trace between adaptor and pabx
(CSTAxxxAdaptor)
# 9: Messages between adaptor and pabx (CSTAxxxAdaptor)
```

"sticky" adding/removing new devices to be monitored

When starting TeamCall Server, a monitoring point in the PBX will be created for every device specified within the device configuration file. These devices will be monitored from startup to shut down of TeamCall Server. To add further devices to monitor until shut down of TeamCall Server without shutting down TeamCall Server, the requests:

`AddDeviceMonitor <Device>` and

`RemoveDeviceMonitor <Device>`

that must be issued in a SuperVisor session, allow to manually add or remove monitoring points. The monitoring points are "sticky", that means they persist until TeamCall Server is shut down.

Example

```
AddDeviceMonitor 123
## Add a Monitoring Point for Device 123
RemoveDeviceMonitor 456
## Remove Monitoring Point for device 456
```

Note: To successfully add a monitoring point for a device, the (internal) device number must be valid. Only one monitoring point per device can be set.
To successfully remove a monitoring point for a device, the device number must be valid and there must be an existing monitoring point for it. Otherwise an error will be reported.

These settings are temporary settings. After shutting down and starting-up TeamCall Server, the monitoring points within the PBX will be reset according to the values loaded from the device configuration file.

More about device monitoring

Device monitoring allows you to gather continuous information about device changes. To view these changes, a `MonitorStart` request (see

"MonitorStart" on page 61) must be issued in a STLI session. For important additional information, refer to "Important notice" on page 39.

As mentioned in the previous sections, there are three levels of device monitoring which affect the period of time TeamCall Server monitors a device.

There are three different ways to monitor a device:

- permanently by adding this device to the device configuration file
- "sticky" by calling an `AddDeviceMonitor` request in a SuperVisor session
- dynamically by only calling a `MonitorStart/TSPI_lineOpen` request in a STLI/NetTSPI session.

If there is no entry in the device configuration file and no device was added using a SuperVisor session, TeamCall Server only monitors the devices which are dynamically introduced using the `MonitorStart/TSPI_lineOpen` request. Use the `MonitorStop/TSPI_lineClose` request to remove the dynamically introduced monitoring points.

The second way to configure monitoring points within the TeamCall Server configuration is to add the devices to be monitored to the device configuration file.

At last, monitoring points can also be added in a "sticky" way using the `AddDeviceMonitor` request within a SuperVisor session.

The difference between these two ways is that if a device has been added to the device configuration file, TeamCall Server internally traces change information about this device from startup to shut down.

Also the specified device will be monitored every time TeamCall Server starts up, while a device which is added using the `AddDeviceMonitor` request will only be monitored as long as TeamCall Server is running.

Hint:	Add a new device using the request `AddDeviceMonitor` and additionally add it to the device configuration file (also see "Important notice" on page 39) to avoid restarting TeamCall Server.

Syntax `ShowDeviceMonitors`

Description Lists all currently monitored devices in the following format: `<device> <isSticky> <isActive> <number of observers>`. For more information about the flag `isSticky` see ""sticky" adding/removing new devices to be monitored" on page 51. The flag `isActive` shows if the monitoring request has already been answered by the PBX. The last

parameter shows how many observers (i.e. sessions) are currently monitoring the device.

Example **ShowDeviceMonitors**

```
error_ind SUCCESS ShowDeviceMonitors
111 sticky:true active:true observers:11
101 sticky:true active:true observers:25
168 sticky:false active:true observers:3
185 sticky:true active:true observers:9
123 sticky:true active:false observers:0
187 sticky:true active:true observers:9
172 sticky:true active:true observers:0
173 sticky:false active:true observers:1
```

Note: Any attempt to start monitoring on a non-active DeviceMonitor will be answered with `error_ind PENDINGRQUEST <Request>`.

Error messages

The following error messsages can occur when issuing SuperVisor requests:

FILEERROR

Response `error_ind FILEERROR SetLogDir`

Description Directory does not exist or is read-only.

Example

```
SetLogDir /doesNotExist
error_ind FILEERROR SetLogDir
```

ILINK-INTERNAL DEMOVERSION

Response `error_ind ILINK-INTERNAL DEMOVERSION MonitoringPointsExceeded`

Description Only one device can be monitored with this demo version.

INTERNALERROR

Response `error_ind INTERNALERROR <Request>`

Description Internal error. Please analyze the `Error.log` file.

INVALCMD

Response `error_ind INVALCMD <Request>`

Description Unknown request.

Example

```
Add 123
error_ind INVALCMD Add
```

INVALNUMPARAM

Response `error_ind INVALNUMPARAM <Request>`

Description Invalid number of parameters for this request.

Example

```
AddDeviceMonitor 524 123
error_ind INVALNUMPARAM AddDeviceMonitor
```

INVALPARAM

Response `error_ind INVALPARAM <Request>`

Description Wrong parameter(s) for this request.

Example

```
EnableLog ddd
error_ind INVALPARAM EnableLog
```

LINKOUTOFSERVICE

Response `error_ind LINKOUTOFSERVICE <Request>`

Description The request cannot be executed because there is no connection to the PBX at this moment.

Example

```
AddDeviceMonitor 524
error_ind LINKOUTOFSERVICE AddDeviceMonitor
```

NOMULTIPLEINSTANCE

Response `error_ind NOMULTIPLEINSTANCE AddDeviceMonitor`

The request `AddDeviceMonitor` has already been issued for this device during this session.

Example **`AddDeviceMonitor 524`**

`error_ind SUCCESS AddDeviceMonitor`

`AddDeviceMonitor 524`

`error_ind NOMULTIPLEINSTANCE AddDeviceMonitor`

NOTREGISTERED

Response `error_ind NOTREGISTERED RemoveDeviceMonitor`

Description The device is not (or no longer) monitored during this session.

Example **`RemoveDeviceMonitor 524`**

`error_ind SUCCESS RemoveDeviceMonitor`

`RemoveDeviceMonitor 524`

`error_ind NOTREGISTERED RemoveDeviceMonitor`

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 PBX (i.e. the DeviceMonitor is **not** active).

Example **`AddDeviceMonitor 111`**

`error_ind PENDINGREQUEST AddDeviceMonitor 111`

SUCCESS

Response `error_ind SUCCESS <Request>`

Description Not an error at all.

Example **`AddDeviceMonitor 524`**

`error_ind SUCCESS AddDeviceMonitor`

UNAVAILABLEREQUEST

Response `error_ind UNAVAILABLEREQUEST <request> ""`

Description The issued request is not supported. It could be not supported at all, disabled or invalid.

5

STLI

This chapter describes all STLI (Simple Telephony Interface) requests, responses, events and errors implemented in TeamCall Server.

STLI is based on CSTA (Computer Supported Telecommunication Applications). CSTA is an industry standard, desribed 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`.

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

It uses the ilink CSTA Framework and offers the application programmer an easy method to implement complex CTI applications.

The application programmer opens a socket connection to the telephony server, sends requests and receives events, errors, and event indications. Every description of a request contains an example, showing the use of it.

STLI version 2

In addition to the standard STLI (version 1) there is also STLI version 2. It only differs from the standard STLI concerning the information signalled in some events. In the section about events “List of events and event reports” on page 82 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`.

Switching to STLI version 2

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

After having established a telnet connection to the STLI port (e.g. 8000), the following request switches on the STLI version 2 signalling (SV2):

```
STLI;Version=2
```

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

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

```
STLI;Version=1
```

Additional information in STLI version 2

The following information is additionally signalled in STLI V2:

`call identifier (<callId>)`

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

`unique device identifier (<deviceId>)`

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

There are three types of identifiers:

- **dynamic id** - identified by the prefix "$"; e.g. "$00040100" (Alcatel A4400)
- **logical id** - identified by the prefix "#"; e.g. "#-2130640809" (Siemens A6/Telekom Octopus)
- **device number** - no prefix; e.g. "7801" (Siemens Hicom 150)

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

Generally, this device identifier is unique within the PBX, 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 PBX) which did not transmit its dialing number (i.e. phone number).

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

If no unique device identifier information is given by the PBX, an empty string "" is signalled.

Basic requirements for communication with TeamCall 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 PBX's or a device's state.

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

`telnet <TeamCall Server Host> <login port of TeamCall 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 TeamCall Server will be closed by typing

BYE

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

Using this user interface, devices connected to the PBX 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 preceeded by the prefix used to place external calls ("operator", e.g. 0 or 9).

Requests received by the PBX are handled as first come, first served, so any acknowledgement from the PBX 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” on page 92.

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 Server 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 Server 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.

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: Not all internal devices may be monitored. Restrictions may apply to groups, trunks, etc. Please refer to your PBX manual for more details.

MonitorStart

Synopsis `MonitorStart <localDevice>`

Parameter `<localDevice>` - indicates the local device to be monitored.

Example **`MonitorStart 12345`**

Positive response `error_ind SUCCESS MonitorStart`

`<localDevice>` is valid and monitoring has been started successfully.

Description The Monitor Start service enables the receipt of event reports for a local device. With `MonitorStart` you start monitoring a local device.

All events reported for this monitor will be identified by the local device number of the monitored device.

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 Server.

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

Synopsis `MonitorStop <localMonitoredDevice>`

Parameter `<localMonitoredDevice>` - indicates the currently monitored local device, where to stop monitoring.

Example **`MonitorStop 12345`**

Positive response `error_ind SUCCESS MonitorStop`

<localMonitoredDevice> will no longer be monitored, i.e. no more event reports will be sent.

Description The Monitor Stop service is used to terminate a previously initiated Monitor Start service. With MonitorStop 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 PBX 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

Synopsis MakeCall <callingDevice> <calledDirectoryNumber>

Parameter <callingDevice> - indicates the device from which the call originates. <callingDevice> must be a valid local device.

<calledDirectoryNumber> - indicates the device to which the call should be directed. <calledDirectoryNumber> can be a local or an external device.

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 at the PBX 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 sucessfully 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

Synopsis ClearConnection <localConnectedDevice>

Parameter <localConnectedDevice> - indicates the device to be disconnected. <localConnectedDevice> must be a valid local device.

Example **ClearConnection 12345**

Positive response error_ind SUCCESS ClearConnection

<localConnectedDevice> has been disconnected.

Description ClearConnection 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 `ClearConnection` is the same as that of `ClearCall`.

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

Synopsis `ConsultationCall <callingDevice> <calledDirectoryNumber>`

Parameters `<callingDevice>` - indicates the device from which the call originates. `<callingDevice>` must be a valid local device.

`<calledDirectoryNumber>` - indicates the device to which the call should be directed. `<calledDirectoryNumber>` can be a local or an external device.

Example **`ConsultationCall 12345 1111`**

Positive response `error_ind SUCCESS ConsultationCall`

`ConsultationCall` has been initiated successfully.

Description 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. `<calledDevice>` will ring. If `<calledDevice>` answers the call, a connection will be established. The second device from the previous call is still on hold.

ConferenceCall

Synopsis `ConferenceCall <callingDevice>`

Parameter `<callingDevice>` - indicates the device which originates the conference. `<callingDevice>` must be a valid local device.

Example **`ConferenceCall 12345`**

Positive response `error_ind SUCCESS ConferenceCall`

The telephone conference has been established successfully. All devices are connected.

Description 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

Synopsis `AlternateCall <callingDevice>`

Parameter `<callingDevice>` - indicates the device which originated the call alternation. `<callingDevice>` must be a valid local device.

Example **`AlternateCall 12345`**

Positive response `error_ind SUCCESS AlternateCall`

Alternation done. The hold device is reconnected to the origin device, the previous connected device is placed on hold.

Description 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

Synopsis `HoldCall <callingDevice>`

Parameter `<callingDevice>` - indicates the device which initiated the hold.

Example **HoldCall 12345**

Positive response `error_ind SUCCESS HoldCall`

The device, connected to <callingDevice>, has been successfully placed on hold.

Description 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

Synopsis `ReconnectCall <callingDevice>`

Parameter <callingDevice> - indicates the device which originates the reconnect. <callingDevice> must be a valid local device. <callingDevice> must have at least one connection on hold.

Example **ReconnectCall 12345**

Positive response `error_ind SUCCESS ReconnectCall`

Reconnection finished successfully. The previous connection has been cleared and the call on hold has been reconnected.

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

Retrieving a call on hold

The Retrieve Call service connects a specified connection on hold.

RetrieveCall

Synopsis `RetrieveCall <retrievingDevice>`

Parameter <retrievingDevice> - indicates the device where to reconnect the held call to.

Example **RetrieveCall 12345**

Positive response `error_ind SUCCESS RetrieveCall`

The previous call on hold has been successfully reconnected.

Description The Retrieve Call service puts a specified connection on hold. `<retrievingDevice>` has to be in hold state.

Transfering a call

The Transfer Call service transfers a call held at a device to an active call at the same device and disconnects the transferring device from the call.

TransferCall

Synopsis `TransferCall <transferringDevice>`

Parameter `<transferringDevice>` - indicates the device, which transfers an active call from itself to a held call and disconnects itself.

Example **TransferCall 12345**

Positive response `error_ind SUCCESS TransferCall`

Transferring proceeded successfully. The call has been transferred to the held call and `<transferringDevice>` has been disconnected from the call.

Description The Transfer Call service transfers a call held at a device to an active 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.

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 PBX, 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 PBX manual for more information.

DeflectCall

Synopsis `DeflectCall <callToBeDeflecteded> <newDestination>`

Parameters `<callToBeDeflecteded>` - indicates the device where to deflect the call. `<callToBeDeflecteded>` must be a valid local device.

`<newDestination>` - indicates the destination, the call will be deflected to. `<newDestination>` can be a local or external device.

Example **`DeflectCall 12345 1111`**

Positive response `error_ind SUCCESS DeflectCall`

Both devices are valid; the call has been successfully deflected.

Description 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 sub-domain.

PickupCall

Synopsis `PickupCall <callToBePickedUp> <requestingDevice>`

Parameters `<callToBePickedUp>` - indicates the device where to deflect the call. `<callToBePickedUp>` must be a valid local device.

`<requestingDevice>` - indicates the local device, which picks up the call. `<requestingDevice>` must be a valid local device.

Example **`PickupCall 12345 1111`**

In opposition to `DeflectCall`, which notices the `<newDestination>` state, `PickupCall` connects to a new local device immediately. For the `<newDestination>`, all features such as `Forwarding` and `DoNotDisturb` for this device will be ignored when the call is being redirected to it.

Positive response `error_ind SUCCESS PickupCall`

The call has been successfully picked up.

Description The Pickup Call service picks up an alerting or queued call at a local device and connects it to another destination inside the switching sub-domain.

GroupPickupCall

Synopsis `GroupPickupCall <requestingDevice>`

Parameter `<requestingDevice>` - indicates the device which picks the call from a pickup group.

Example **GroupPickupCall 12345**

Positive response `error_ind SUCCESS GroupPickupCall`

The call has been successfully picked up.

Description 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 sub-domain. 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 PBX must support forwarding and the Set Forwarding service (which is a part of Set Device Feature service) to use this feature. Some PBXs 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 PBX 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 PBX like speaker or microphone mute, messaging etc.

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

SetDeviceFeature

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

Parameter `<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 PBX 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 PBX 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.

Example

```
SetDeviceFeature 111 autoAnswer Off
SetDeviceFeature 111 microphoneMute Off
SetDeviceFeature 111 speakerMute Off
SetDeviceFeature 111 speakerVolume 100
```

Positive response `error_ind SUCCESS SetDeviceFeature`

`SetDeviceFeature` was successful.

Description `SetDeviceFeature` allows to enable/disable or modify device-specific settings like speaker level or microphone mute for specified devices. Some settings like `speakerMute` are executed directly at the device. Other settings like `forwarding` are set for a specific device, but not executed at the device itself, because call forwarding is done by the PBX.

Setting agent states for agents logged to an ACD at PBX

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

The PBX must support agents and Set Agent State service to use this feature.

SetAgentState

Synopsis `SetAgentState <deviceID/agentID> <requestedAgentState> [<agentID> <password> <group>]`

Parameter `<deviceID/agentID>` — Indicates the agent (i.e.the device associated to the agent), where to change the settings.

`<requestedAgentState>` — 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 PBX supports Phase I or Phase II, there are different values available for `<requestedAgentState>`.

- Values for 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 loggedOff, <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 loggedOff, <password> must be specified. The password authenticates the agent to the CSTA application and/or one or more of its component functions.

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

Example **SetAgentState 111 busy**

Positive response error_ind SUCCESS SetAgentState

Description SetAgentState allows to change settings for agents at an ACD. To use this service, the PBX 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 PBX 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, where to report the agent state and device feature state.

Positive response error_ind SUCCESS QueryDevice <feature> <parameter>

features for CSTA I: msgWaiting, doNotDisturb, forward, lastDialedNumber, deviceInfo, agentState

features for CSTA II: `msgWaiting`, `doNotDisturb`, `forward`, `deviceInfo`, `agentState`, `routing`, `autoAnswer`, `microphoneMute`, `speakerMute`, `speakerVolume`

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 PBXs 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 PBX 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

Synopsis `ClearCall <callToBeCleared>`

Parameter `<callToBeCleared>` - indicates the call to be closed by releasing all devices.

Example **`ClearCall 12345`**

Positive response
`error_ind SUCCESS ClearCall`

The call has been cleared.

Description
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.

Appending user data to a call

Using `AssociateData`, user specified data can be appended to an active call.

AssociateData

Synopsis
`AssociateData <device> <data>`

Parameter
`<device>` - indicates the device.

`<data>` - can contain any character except control sequences. These characters are interpreted als delimiters and substituted with <Space>: ';', <TAB>, <CR>, <LF>.

Example
`AssociateData 100 My Personal attached Data !`

Description
This request appends user data to a call which is active at `<device>`.

`AssociateData` 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: `Delivered`, `Established`, `Conferenced`, `Transferred`.

Associating and transmitting user data with a call is very PBX-specific. Please refer to the documentation of your PBX for further detailed information.

Example 2
(Executed on an Alcatel A4400, # comments)

`STLI;Version=2`

```
error_ind SUCCESS STLI Version "2"
```

`MonitorStart 111`

```
error_ind SUCCESS MonitorStart
```

`MonitorStart 529`

```
error_ind SUCCESS MonitorStart
Delivered 111 newCall 111 003028526530 111 "" 0125
$00010100 $006b0104 ""
```

```
DeviceInformation 111 1 (0125:alerting)
# 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 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 recieves 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)
```

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 61 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 outside the PBX) 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 signalisation depends on the PBX/Switch. 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 PBX 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 PBX 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 PBX 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 PBX 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 PBX 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 PBX 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> <onOff>

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} callId 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.

The <callId> represents the newly created conference call.

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} 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 79.

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 PBX, this information may be empty (i.e. `""`) if the calling device is an internal device (i.e. within the PBX) 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.

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 PBX, this information may be provided as an empty string `""` if the calling device is internal (i.e. within the PBX) 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.

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 0017799999999 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> 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 PBX.

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 (eg. trunk).

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.

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 79.

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.

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 supresses 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.

Error messages

The following error messsages 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 PBX 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 PBX (i.e. the DeviceMonitor is not active).

Example

```
MonitorStart 111
error_ind PENDINGREQUEST MonitorStart 111
```

SUCCESS

Response `error_ind SUCCESS <Request>`

Description No problems.

Example

```
MonitorStart 111
error_ind SUCCESS MonitorStart
```

UNAVAILABLEREQUEST

Response `error_ind UNAVAILABLEREQUEST <request> ""`

Description Request at this time not supported. It could be not supported, disabled or invalid.

6

TeamCall Link V24

TeamCall LinkV24 routes TCP/IP messages to a given serial device and vice-versa. If the PBX does not have an ethernet network connection, TeamCall LinkV24 allows to control the PBX with TeamCall Server using the advantages of an IP socket connection.

Setup and requirements of TeamCall LinkV24

The serial interface

In order to use TeamCall LinkV24, a computer must have an available serial port the PBX will be connected to.

Possible values for the baud rate of the serial interface are:

256000, 128000, 115200, 57600, 56000, 38400, 19200, 14400, 9600, 4800, 2400, 1200, 600, 300, 110

The default baud rate is 9600.

The maximum possible baudrate depends on both, PBX and computer. The chosen value must fit both requirements.

The serial cable

Hicom 150

The serial cable has to be a "zero modem cable" and should not exceed a total lenght of 15 meters without a special shielding. The pin assignments are as follows:

Pin assignments for Siemens Hicom 150

	Host 9 pins	Hicom 25 pins
DCS	1	8
RxD	2	2
TxD	3	3
DTR	4	6
DSR	6	20
RTS	7	5
CTS	8	4
GND	5	7

Note: Hardware handshake via DTR/DTS or RTS/CTS is not provided. The asynchronous V.24/RS-232 interface has to be operated with the following parameters:

```
Baud rate: 2400 or 9600
Data bits: 8
Parity: none
Stop-bit: 1
```

Octopus E300/800/Siemens A6

The cable needed depends on the operating system used. Some need a DTR/DTS signal and others do not. Please ask your PBX dealer for a suitable cabel.

The socket port

TeamCall LinkV24 requires an available TCP/IP port number at the local computer. Assigning this port number to TeamCall LinkV24, a socket connection to the computer can be established and TeamCall LinkV24 routes messages from this port to the assigned computer´s serial port.

This port cannot be used by other applications or services running on that computer.

One TCP/IP client is supported at a time. Further attempts to connect to the router will result in the message `connection refused`.

Any available port number can be assigned to TeamCall LinkV24. The default port number is 2555.

Configuration

To run TeamCall LinkV24, a main configuration file must be created. It will be loaded automatically if it is named `Default.conf` and stored in the binary's path.

If the main configuration file is not named `Default.conf` or stored at a different path then TeamCall LinkV24 must be started using the following option:

```
LinkV24 -c <absoutePath>/<ConfFile>
```

TeamCall LinkV24 loads its information from the main configuration file by parsing its keys.

As TeamCall Server's main configuration file can also be named `Default.conf` and both binaries can be stored in the same path, you can create one main configuration file named `Default.conf` for both programs.

The default settings will be used for starting TeamCall LinkV24 if no configuration file is found.

Default settings are:

- 9600 baud for serial connection and
- 2555 as port number reserved for TCP/IP connection.
- 8 data bits
- no parity

- 1 stop bit
- a Siemens Hicom 150 PBX is connected to the serial port

Note: The serial device of the Hicom150 is set to 8N1 (8 data bits, parity none, 1 stop bit) and the serial device of the A6 is usually set to 8E1 (8 data bits, parity even, 1 stop bit). If you are in doubt about the serial device setup of of your A6 PBX, contact a service engineer from Siemens or Deutsche Telekom.

The main configuration file

As mentioned above, the main configuration file for running TeamCall LinkV24 has to be named `Default.conf` to be loaded automatically. For more information see “Main configuration file entries” on page 29. For using TeamCall LinkV24 the keys `baudRate` and `serialDevice` are mandatory.

The file contains keys, which can be placed in any line order. The keys contain the settings for running TeamCall LinkV24.

Example

```
baudRate 9600           # baud rate for connection to PBX
serialDevice COM1       # interface for connection to PBX
dataBits 8              # 7 or 8; default: 8
parity none             # even, odd, none; default: none
stopBits 1              # 1 or 2; default: 1
serialLogEnabled 0      # 0=disabled, 1=enabled
tcpLogEnabled 0         # 0=disabled, 1=enabled
                        # the value depends on OS, f.e.:
                        # "Com1" for Win32-platforms,
                        # "/dev/ttya" for Unix-platforms
logDir /tmp             # path for logFiles
cstaLinkPort 2555       # Server Port, TeamCall LinkV24
                        # will use @ localhost
linkedPBX hicom 150     # a6 or hicom150;
                        # default: hicom150
```

7

Log files

TeamCall Server logs all of its activities in different log files when activated and configured within the main configuration file. You can turn on the logging mechanism to debug an error, but remember to turn this feature off, when it is no longer needed.

The number of messages written to the log files depend on the debug level set within the main configuration file. To learn more about debug levels, see the key `debugLevel` in chapter "Configuration" on page 29.

The log files have the following names:

Error.log — The logging mechanism does not need to be turned on as described above for the error logging feature. The error log file is created automatically when the server starts up.

Sys.log — Status messages, warnings and errors. For detailed information see "Sys.log" on page 102.

Device.log — It contains information about the status of the monitored telephones (devices).

Calls.log — The `Calls.log` file contains information about the calls and the telephones that are connected to these calls and their status.

CSTA.log — The `CSTA.log` file contains the plain ASN.1 messages that are exchanged between TeamCall Server and the PBX.

Activate logging

Note:	Activating the logging mechanism may affect the performance of the server.

To enable the logging mechanism, follow these steps:

1. Locate the the main configuration file (e.g. `Default.conf`).

   On a computer running Windows NT, the file `Default.conf` is located in the installation path.
2. Launch your favourite text-editing application.
3. Open the file `Default.conf`.
4. Find the entry `callLogEnabled`.
5. Change the value after `callLogEnabled` to `1` in order to activate call logging.
6. Find the entry `cstaLogEnabled`.
7. Change the value after `cstaLogEnabled` to `1` in order to activate CSTA logging.
8. Find the entry `debugLevel`.
9. Change the value after `debugLevel` to `9` in order to enable full logging.
10. Find the entry `logFileMaxSize`.
11. Change the value after `logFileMaxSize` to `-1` in order to disable log rotation.

Note: If any of these entries are not present in the file, please add them.

Sys.log

The following messages are logged in the file `Sys.log` in the log directory. There are three categories for messages:

- Status — just for your information
- Warnings — notes about problems that can occur under certain conditions
- Errors — errors that **have** to be corrected in order to run TeamCall Server correctly.

Information in <> represent variables.

Status

```
Connected to Admin Server.
Connected to PBX/Link, sending login sequence.
Connecting to <value>:<value> (IP <value>)...
Connecting to Admin Server at <aaa.bbb.ccc.ddd:port>
Connection terminated.
Connection to Admin Server terminated.
Log in to PBX/Link successful.
NetTSPI port <port> added.
PBX Link up
Reloading DeviceMonitors (PBX LinkUp)
STLI port <port> added.
Server shutdown.
SuperVisor port <port> added.
LinkV24 Serial Device initialized.
LinkV24: Closing TCP-client connection
LinkV24 ready.
Waiting for connection to PBX/Link.
```

Warnings (WRNXXXX)

WRN0003

Message `(WRN0003) DEMOVERSION restricted to one monitoring point.`

Description You are currently using a demo version of TeamCall Server which is restricted to one monitoring point only.

WRN0004

Message `(WRN0004) SuperVisor port <port> not available, retrying.`

Description The SuperVisor port entered in the main configuration file (key: `servicePort`, default: `7999`) is currently not available. TeamCall Server will retry to connect to the port every 2 minutes. If the port is still not available after 5 retries you should either use another port or check if

there is not another instance of TeamCall Server running on the same computer using the same port.

WRN0005

Message `(WRN0005) STLI port <port> not available, retrying.`

Description The STLI port entered in the main configuration file (key: `stliPort`, default: `8000`) is currently not available. TeamCall Server will retry to connect to the port every 2 minutes. If the port is still not available after 5 retries you should either use another port or check if there is not another instance of TeamCall Server running on the same computer using the same port.

WRN0006

Message `(WRN0006) NetTSPI port %d not available, retrying.`

Description The NetTSPI port entered in the main configuration file (key: `netTspiPort`, default: `26535`) is currently not available. TeamCall Server will retry to connect to the port every 2 minutes. If the port is still not available after 5 retries you should either use another port or check if there is not another instance of TeamCall Server running on the same computer using the same port.

WRN0007

Message `(WRN0007) DEMOVERSION restricted to one monitoring point: <value> not set`

Description You are currently using a demo version of TeamCall Server which is restricted to one monitoring point only.

WRN0008

Message `(WRN0008) Baudrate <value> for serial device not supported ! Setting to default (<value>)`

Description The baudrate you specified in the main configuration file (key: `baudRate`) is not supported. The value has been changed to the default value.

WRN0009

Message `(WRN0009) Error Processing line <value> in file <value>: <value>.`

Description An entry in the main configuration file could not be processed. The entry will be ignored. Please check the main configuration file.

WRN0010

Message `(WRN0010) SupportedRequest file <value> : invalid entry "<value>", ignoring`

Description An entry in your `SupportedRequest` file could not be processed. The entry has been ignored. Please check your `SupportedRequest` file if you experience problems.

WRN0011

Message `(WRN0011) SupportedRequest file <value> : invalid/ unsupported request in entry ``<value>'', ignoring`

Description An entry in your `SupportedRequest` file could not be processed. The entry has been ignored. Please check your `SupportedRequest` file if you experience problems.

WRN0012

Message `(WRN0012) SupportedRequest file <value> : invalid value in entry "<value>", ignoring`

Description An entry in your `SupportedRequest` file could not be processed. The entry has been ignored. Please check your `SupportedRequest` file if you experience problems.

WRN0013

Message `(WRN0013) Login port <value> not available, retrying.`

Description The login port given in the main configuration file (key: `loginPort`, default: `26535`) is currently not available. TeamCall Server will retry to connect to the port every 2 minutes. If the port is still not available after 5 retries you should either use another port or check if there is not

another instance of TeamCall Server running on the same computer using the same port.

Errors (ERRXXXX)

ERR0001

Message `(ERR0001) Connection to TeamCall Admin Server failed (err = <value>)`

Description Connection to TeamCall Admin Server failed. Please check the given IP-address in the main configuration file of TeamCall Server (key: `adminAddress`, default: `127.0.0.1`) and the installation of the TeamCall Admin Server.

ERR0002

Message `(ERR0002) Hostname for administration (<value>) could not be resolved to IP-Address, check your DNS.`

Description The DNS name for the TeamCall Admin Server set in the main configuration file (key: `adminAddress`) is not correct: the specified hostname cannot not be mapped to a TCP/IP address. Please check the entry in the main configuration file and also the DNS configuration of the TeamCall Admin Server computer.

ERR0003

Message `(ERR0003) PBX link down`

Description TeamCall Server has currently no connection to the PBX. Please check the network connectivity of the host where TeamCall Server is running. Also make sure that the PBX is still connected to the local network and that it is up und running.

ERR0004

Message `(ERR0004) PBX link idle`

Description TeamCall Server has currently no connection to the PBX. Please check the network connectivity of the host where TeamCall Server is running. Also

make sure that the PBX is still connected to the local network and that it is up und running.

ERR0005

Message `(ERR0005) Session login failed`

Description The entered password and/or userid is invalid. Please check your input and try again.

ERR0006

Message `(ERR0006) Session login failed password expired`

Description The entered password for that userId is no longer valid. Please check your input and try again.

ERR0009

Message `(ERR0009) Can't open device monitor configuration file <value>`

Description The value of the key `deviceMonitorConfFile` in the main configuration file is not correct: the file could not be found. Please enter the complete path of the file and restart TeamCall Server.

ERR0010

Message `(ERR0010) setsockopt for port <value> failed <value>: <value>`

Description The port is currently not useable. Please locate the port, edit the main configuration file, choose a different port number and restart TeamCall Server. If you still experience this problem, please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0011

Message `(ERR0011) Can't open stream socket for port <value>`

Description The port is currently not useable. Please locate the port, edit the main configuration file, choose a different port number and restart TeamCall

Server. If you still experience this problem, please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0012

Message `(ERR0012) Hostname <value> could not be resolved to IP-Address, check your DNS`

Description The DNS name for the PBX/Link entered in the main configuration file of TeamCall Server (key: `cstaLinkAddress`) is not set correctly. The specified host name cannot be mapped to a TCP/IP address. Please check the entry in the main configuration file and also the DNS configuration of the TeamCall Server computer.

ERR0013

Message `(ERR0013) Connection to PBX/Link failed (err = <value>)`

Description TeamCall Server will retry to connect to the PBX/Link every 2 minutes. If the connection still fails after 5 retries, you should make sure that:

- the key `cstaLinkAddress` in the main configuration file is set correctly
- the key `cstaLinkPort` in the main configuration file is set correctly
- the TeamCall Server computer is connected to the network
- the PBX/Link is available within the network
- the PBX/Link is up and runnig
- the TeamCall Link process is up and running
- there is no other TeamCall Server currently connected to the PBX/Link

ERR0014

Message `(ERR0014) PBX unavailable: received "U"`

Description This error is Alcatel-specific. The Alcatel-PBX is currently not available. TeamCall Server will retry to connect to the PBX/Link every 2 minutes. If the connection still fails after 5 retries, you should check your PBX.

ERR0015

Message `(ERR0015) Signal caught: timeout <signal>`

Description TeamCall Server received a signal and terminated. Please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0016

Message `(ERR0016) Signal caught: restart <signal>`

Description The server received a signal and terminated. Please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0017

Message `(ERR0017) Signal caught: alarm_handler <signal>`

Description TeamCall Server received a signal and terminated. Please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0018

Message `(ERR0018) Signal caught: just_die <signal>`

Description The server received a signal and terminated. Please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0019

Message `(ERR0019) Could not write login message`

Description This error is Hicom150-specific. The initial login message could not be sent to the Hicom150. Please check to make sure that:

- the key `cstaLinkAddress` in the main configuration file is set correctly
- the key `cstaLinkPort` in the main configuration file is set correctly
- the PBX and Link is available within the network

- the TeamCall Server computer is connected to the network
- the PBX and TeamCall Link is up and runnig
- the TeamCall Link process (either TeamCall LinkV24 or TeamCall LinkISDN) is up and running
- there is no other TeamCall Server currently connected to the TeamCall Link

ERR0020

Message `(ERR0020) Log in to PBX/Link failed: received LOGNACK`

Description This error is Hicom150-specific. Apparently, someone modified the password of the Hicom150. Either reset it to the original password or contact the TeamCall support department in order to get a modified version of TeamCall Server.

ERR0021

Message `(ERR0021) Could not open serial device <value>: <value> = <value>`

Description This error is specific to TeamCall LinkV24. The serial device set in the main configuration file cannot not be opened. Either it does not exist or another process is using it. Please modify the value of the key `serialDevice` in the main configuration file and restart the TeamCall LinkV24 process. You may also restart TeamCall Server.

ERR0022

Message `(ERR0022) Could not get attributes for serial device <value>: <value> = <value>`

Description This error is specific to TeamCall LinkV24. The serial device set in the main configuration file cannot not be initialized. Either it does not exist or another process is using it. Please modify the value of the key `serialDevice` in the main configuration file and restart the TeamCall LinkV24 process. You may also restart TeamCall Server.

ERR0023

Message `(ERR0023) Could not set attributes for serial device <value>: <value> = <value>`

Description This error is specific to TeamCall LinkV24. The serial device given in the main configuration file cannot not be initialized. Either it does not exist or another process is using it. Please modify the value of the key `serialDevice` in the main configuration file and restart the TeamCall LinkV24 process. You may also restart TeamCall Server.

ERR0024

Message `(ERR0024) LinkV24: received unknwon token`

Description This error is specific to TeamCall LinkV24. Please restart TeamCall LinkV24 and then TeamCall Server. If the problem persists, make sure that the connection from the TeamCall LinkV24 to the PBX works properly and the PBX is configured correctly. In case you experience further problems, please enable debugging, restart TeamCall Server, repeat the same scenario and send the log files to the TeamCall support department.

ERR0025

Message `(ERR0025) LinkV24: could not get TCP port <value>.Killing Process.`

Description This error is specific to TeamCall LinkV24. The TCP/IP port given in the main configuration file (key: `cstaLinkPort`, default: 2555) is not available. You should either use another port or make sure that there is no other TeamCall LinkV24 (or other process) running on the same computer using the same port.

ERR0026

Message `(ERR0026) Missing mandatory key "<key>" in Configuration File!Killing Process.`

Description The mandatory <key> has to be set in the main configuration file.

ERR0027

Message `(ERR0027) LinkV24: problem while processing serial message. Killing process.`

Description This error is specific to TeamCall LinkV24. The message received from the PBX could not be processed. Please restart TeamCall LinkV24 and then TeamCall Server. If the problem persists, make sure that the connection from the TeamCall LinkV24 to the PBX is correct and the PBX is configured correctly. In case you experience further problems, please enable debugging, restart the server, replay the same scenario and send the log files to the TeamCall support department.

ERR0030

Message `(ERR0030) start: Can't open stream socket.`

Description TeamCall Server cannot connect to the PBX/Link and will retry to connect to the PBX/Link every 2 minutes. If the connection still fails after 5 retries you should make sure that:

- the key `cstaLinkAddress` in the main configuration file is set correctly
- the key `cstaLinkPort` in the main configuration file is set correctly
- the PBX/Link is available within the network
- the TeamCall Server computer is connected to the network
- the PBX/Link is up and runnig
- the TeamCall Link process is up and running
- there is no other TeamCall Server currently connected to the PBX/Link

ERR0031

Message `(ERR0031) Can't open supportedRequest file %s <value>`

Description The value of the key `supportedRequest` in the main configuration file is not set correctly. The file could not be found. Please enter the file name with the absolute path and restart TeamCall Server.

8

Troubleshooting

In this chapter you find information about known issues with TeamCall Server. Not all of the issues apply to the current release. Issues not present in the current release may be of interest to developers who develop software based on the current release but plan to deploy it on a previous release.

Issues

After starting TeamCall Server, no connection to the PBX will be established

Operating systems: All

Description TeamCall Server has been started. The current configuration has been reported. Below this configuration listing, TeamCall Server reports

```
Connecting to 195.21.11.162:2555 ...
Waiting for connection to server.
```

(where 195.21.11.162:2555 is TCP/IP:PORT in this example) will be reported but no connection can be established

Work-around By any reason, the network connection to the PBX cannot be established. Check out the network in the following order:

Try to ping the TCP/IP address of the PBX. If all packets get lost, the TCP/IP address may be wrong, the physical network connections may not work correctly or the PBX may have no network port.

If pinging was successful, try a telnet to TCP/IP:PORT of the PBX. If it is refused, and you get an error message like

```
telnet: Unable to connect to remote host: Connection refused
```

the TCP/IP port number might be wrong; check the TCP/IP port number of the PBX. Another indicator for a wrong port number at the PBX is that additionally to the waiting to connect message, a message like

```
Connection terminated.
```

will be reported.

If both previous tests were successful, it is possible that TCP/IP:PORT is valid, but does not belong to the PBX.

In general, check the network configuration.

Local devices cause an error, when accessed with STLI requests or NetTSPI requests.

Operating systems: All

Description

After sending some requests (e.g. `MakeCall` or `ConsultationCall`) TeamCall Server responds

```
error_ind UniversalFailure operationalErrors requestIncompatibleWithObject
```

or after sending other requests (e.g. AnswerCall) TeamCall Server responds

```
Failed <monitoredDevice> destNotObtainable <failedDevice> <calledDevice>
```

Work-around

There are two possible reasons:

If a device is handled by TeamCall Server in a PBX installation, it has to be announced to TeamCall Server/PBX by setting a monitoring point for it.

This is can be done by adding its device number to the device configuration file. For temporary cases and test cases it can be added manually in SuperVisor mode, calling the `AddDeviceMonitor` request.

In the described error case, the first possible reason is, that there is no monitoring point for the device. A device without a monitor point can be accessed by a `MakeCall` request, but not by an `AnswerCall` request.

The second possible reason is, that the device has been physically disconnected (unplugged) or is in any other way not correctly connected to the PBX or setup at PBX.

First make sure that the device is connected to the PBX and works correctly in this environment, by taking it off-hook and manually calling another device.

If there is no dial tone or another device cannot be called manually, check the PBX and its configuration.

If the previous test was successful, the device cannot be monitored. Check the configuration of the PBX.

INDEX

L

M

N

T

U

W