CTI und Telefonie-Testprogramm

AID 018 DE

Autor: FA, KN, MW. Version 10.2. Zuletzt geändert 04.09.2017

Version	Bemerkung
10.2	Anpassung der Formatierungen
10.1	telephony durch cti ersetzt
10.0	Aktualisieren des Kapitels zum CTI-Testprogramm
3.2	Umstelllung des Formats
3.1	Letzte Version vor Umstellung der Versionshistorie im Dokument

Inhaltsverzeichnis

1. Telefonie mit ADITO

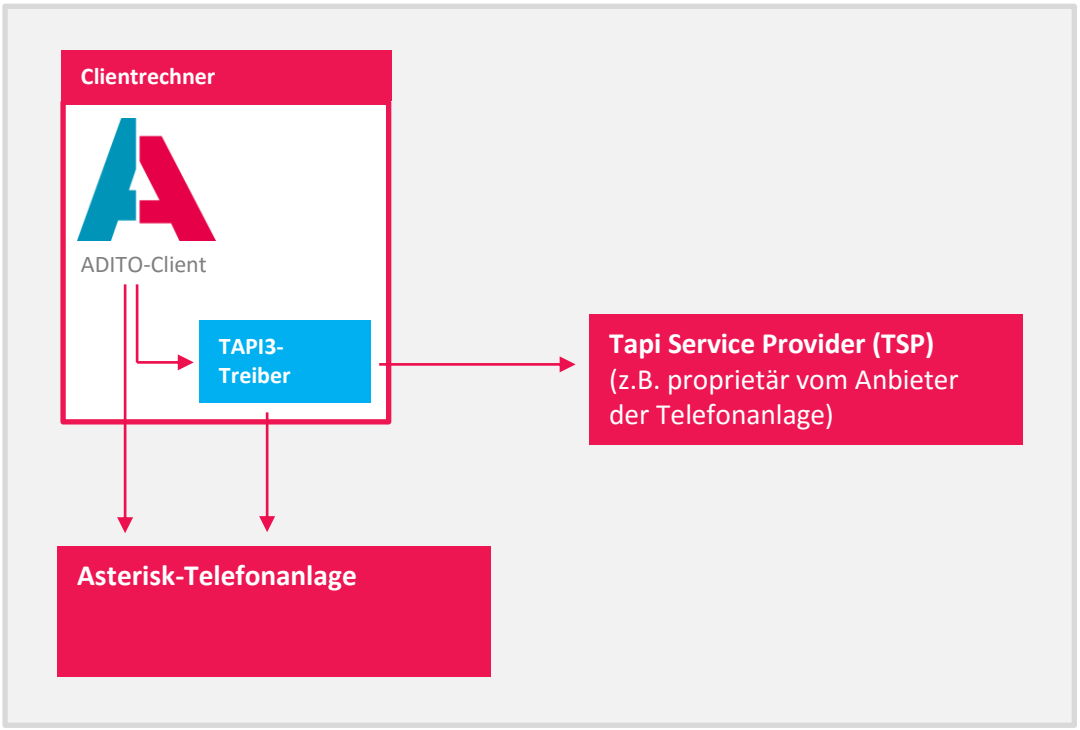
Die CTI-Funktionalität steht in ADITO4 zur Verfügung und ermöglicht die Anbindung an vorhandene TAPI3- oder Asterisk-Telefonsysteme. Die ADITO-Telefonie ist mit drei Systemen kompatibel:

- Einem Tapi3-kompatiblen Telefonsystem.
- Dem internen Emulator des Testprogramms.
- Einem Asterisk-Telefonserver.

In diesem Dokument werden die Alternativen beschrieben. Sie finden hier eine kurze Einführung zur Konfiguration und Installation von CTI am ADITO-Client, sowie eine ausführliche Beschreibung zum CTI-Testprogramm.

Tapi3 wird innerhalb der ADITO Telefonie-Schnittstelle nur clientseitig unterstützt. Das heißt, es muss auf jeden Fall lokal ein Tapi3-Treiber installiert sein! Fragen Sie hier im Zweifelsfall Ihren Telefonie-Spezialisten.

1.1. Anbindung



1.1.1. Über TAPI

In den meisten Fällen kommuniziert der ADITO-Client mit der Telefonanlage über einen lokal am Rechner installierten Tapi3-Treiber, der die Verbindung zum Tapi Service Provider selbst regelt. Geht ein Anruf bei dem Benutzer ein, so übernimmt der TAPI3-Treiber das Management von Rufnummer, Anruftyp und gibt dieses dem ADITO-Client weiter.

1.1.2. Asterisk

Steht eine Asterisk-Telefonanlage zur Verfügung, so kommuniziert der ADITO-Client direkt mit dieser Telefonanlage. ADITO verwendet zur Kommunikation GJTAPI in der Version 1.8 (http://gjtapi.sourceforge.net/).

Eine Verbindung von ADITO zum Asterisk-Server über einen Tapi3-Treiber ist ebenfalls möglich. Es existieren eine Reihe von kostenlosen oder kostenpflichtigen Treibern hierfür, z.B. Activa (http://activa.sourceforge.net/).

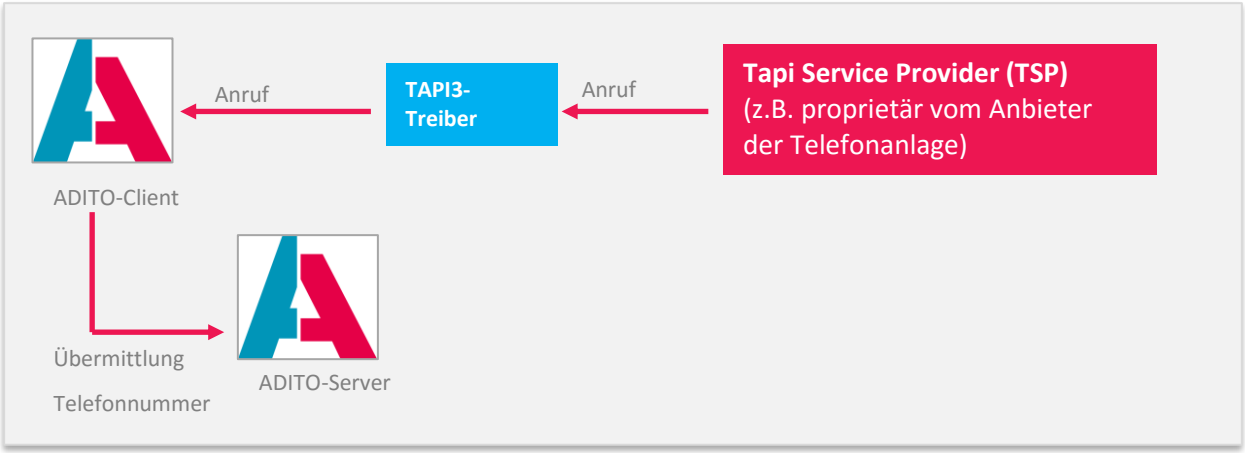
1.1.3. Protokolle, Sicherheit, Verbindungen

Welches Protokoll in Ihrer Telefonie-Kommunikation verwendet wird (z.B. bei Asterisk SIP, IAX…), oder welche Sicherheit in ihrer TSP-Infrastruktur verwendet wird, hängt stark von der Konfiguration ihrer Telefonanlage ab. ADITO kommuniziert nur auf Treiberebene mit diesen Schnittstellen. Ziehen Sie im Zweifel immer Ihren Telefonanlagen-Anbieter oder –Spezialisten hinzu, wenn es sich um Fragen der Einrichtung von ADITO und CTI handelt.

1.2. Kommunikationsablauf

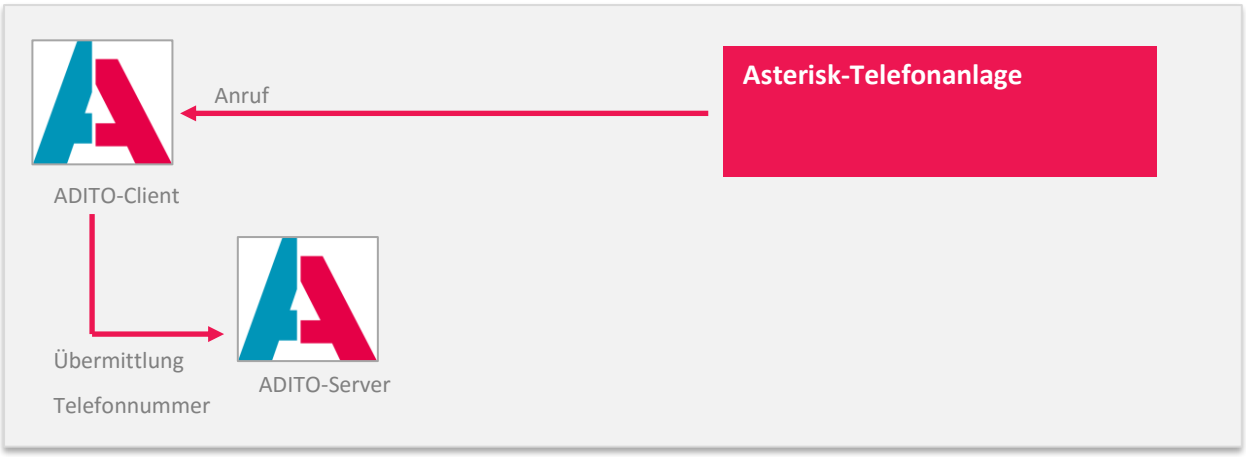
1.2.1. Eingehender Anruf

1.2.1.1. TAPI



Geht ein Telefonanruf ein, so übermittelt die Telefonanlage an den Tapi Service Provider die Informationen über den eingehenden Anruf. Der TSP übermittelt diese Daten an den lokalen Tapi-Treiber, der mit der Anrufer-Telefonnummer verknüpft ist. Der lokale ADITO Client reagiert auf den Anruf, der vom Tapi-Treiber registriert wurde, und sendet die Telefonnummer und die Notiz über den Anruf an den ADITO-Server über die bestehende ADITO Client-Server-Infrastruktur. Nun führt der ADITO-Server die JDito-Prozesse aus, die notwendig sind, damit der Client z.B. die Maske des Anrufers ansehen kann.

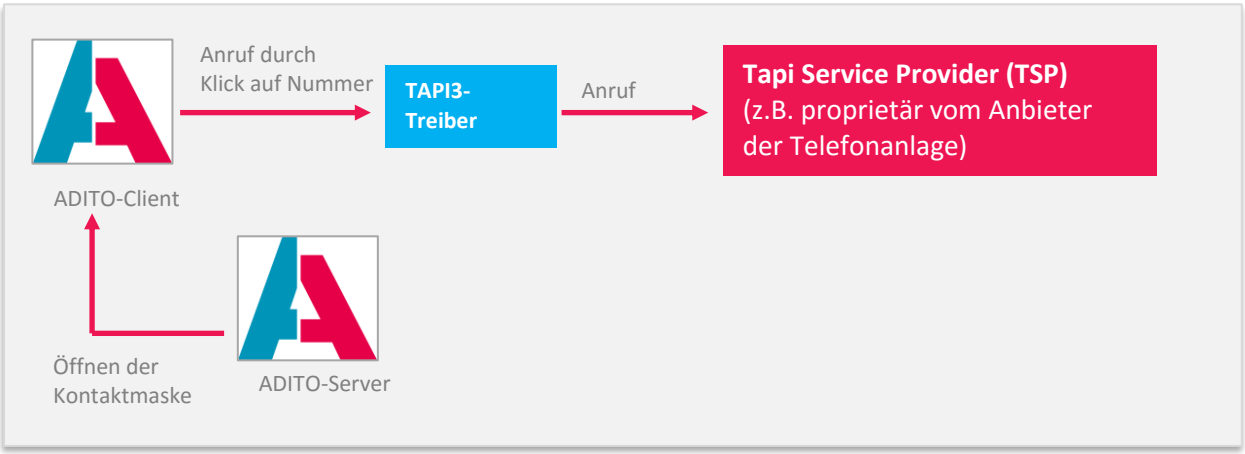
1.2.1.2. Asterisk



Geht ein Telefonanruf ein, so reagiert der lokale ADITO Client auf den Anruf, der von Asterisk registriert wurde, und sendet die Telefonnummer und die Notiz über den Anruf an den ADITO-Server über die bestehende ADITO Client-Server-Infrastruktur. Nun führt der ADITO-Server die JDito-Prozesse aus, die notwendig sind, damit der Client z.B. die Maske des Anrufers ansehen kann.

1.2.2. Ausgehender Anruf

1.2.2.1. TAPI



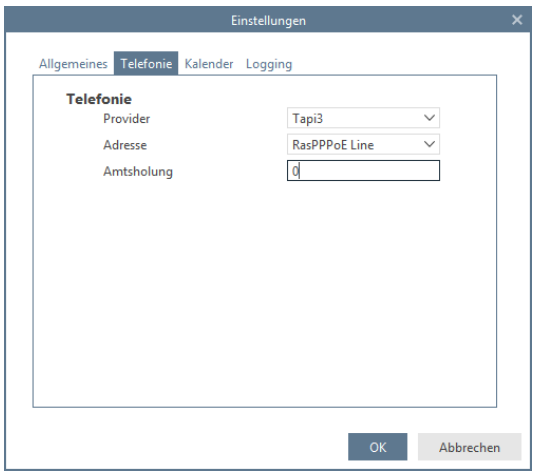
Um einen ausgehenden Anruf zu tätigen, klickt der Benutzer auf eine Telefonnummer, die auf einer Maske angezeigt wird. Diese Telefonnummer wird vom Server an den Client übermittelt, der lokale Client reicht diese Nummer weiter an den lokalen Tapi-Treiber. Die weitere Kommunikation mit dem Tapi Service Provider nimmt der Treiber vor.

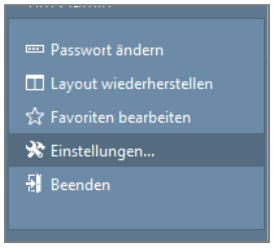
1.2.2.2. Asterisk

Um einen ausgehenden Anruf zu tätigen, klickt der Benutzer auf eine Telefonnummer, die auf einer Maske angezeigt wird. Diese Telefonnummer wird vom Server an den Client übermittelt, der lokale Client reicht diese Nummer weiter an die Asterisk-Telefonanlage.

1.3. Konfiguration im ADITO-Client

Die Telefonie kann immer im Client selbst eingestellt werden. Dazu muss der Punkt `Einstellungen` in der Sidebar gewählt werden.





Zuerst ist der **Provider** auszuwählen (Tapi3, Asterisk oder Emulator). Üblicherweise wird hier Tapi3 gewählt. Dazu muss allerdings Telefonie über Tapi3 am Client-Rechner installiert sein! Nur bei installiertem Tapi3-Treiber wird die Auswahlliste bei **Adresse** gefüllt. Der Inhalt der Auswahlliste erfolgt abhängig vom Tapi3-Client. Die **Amtsholung** hängt ebenfalls von der Konfiguration der Telefonanlage ab.

Bei Asterisk wird hier üblicherweise die Verbindung zum Telefonie-Server angegeben, z.B. `Asterisk; Server=192.168.13.37;P;Port=5038`. Auch hier wird die Liste der möglichen Adressen erst gefüllt, wenn eine gültige Verbindung angegeben wurde. Wie der Connection String aufgebaut wird, erfahren Sie unten – die Konfiguration der Telefonie entspricht im Testprogramm der des Clients.

2. Das CTI-Testprogramm

2.1. Installation

Um das CTI-Testprogramm zu installieren, entpacken Sie den Inhalt der Datei CTITest4.4.40.zip in ein beliebiges Verzeichnis auf Ihrer Festplatte. Sollten Sie nicht über diese Datei verfügen, setzen Sie sich mit dem ADITO Service in Verbindung (info@adito.de oder über den Service Client), die Ihnen das Archiv gerne zusendet.

2.2. Enthaltene Dateien

- adito-synthetica-lf-flat-4.4.0.jar
- asterisk-java-2.0.0.CI-adito-150728.jar
- ctitest.bat
- cti-4.4.jar
- dllfromjarloader-1.2.1.jar
- gjtapi-1.8.0-adito-150916.jar
- jtapi-1.3.1.jar
- synthetica-2.20.0.jar
- tapi3provider-1.8.0-adito-150916.jar

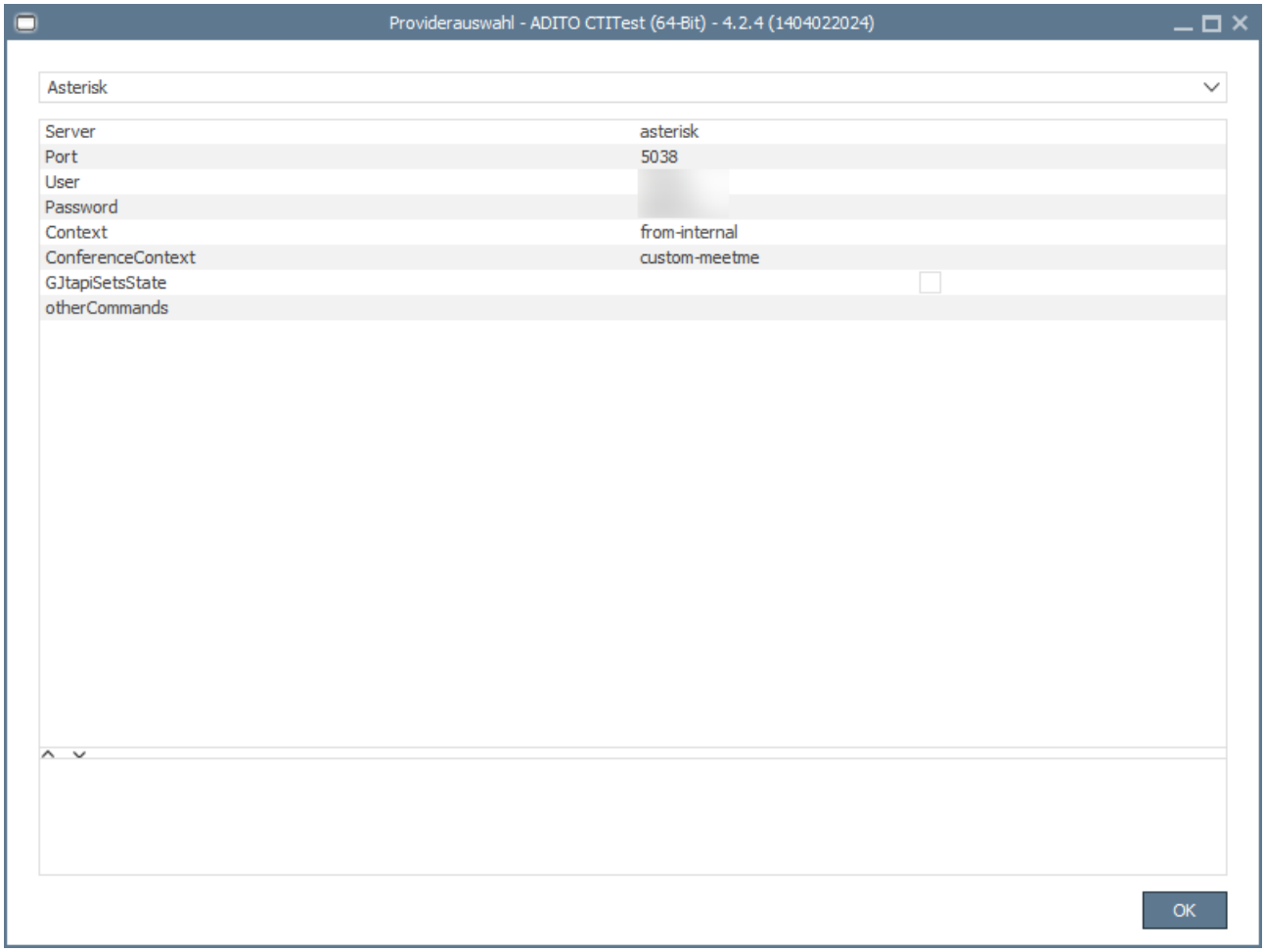
2.3. Start

Starten Sie die Anwendung durch einen Doppelklick auf die Datei `ctitest.bat`. Diese Batchdatei enthält den Start des Java-Laufzeitsystems und den Aufruf der CTI-Testanwendung. Beim Aufruf der Anwendung wird davon ausgegangen, dass sich das Verzeichnis im Suchpfad von Windows befindet. Falls dies nicht der Fall ist, müssen Sie die Angabe java.exe zu Beginn der Befehlszeile um den Verzeichnispfad der Java-Installation ergänzen oder das Java bin-Verzeichnis in die Windows-Umgebungsvariable path aufnehmen. Sollten Sie die Anwendung in anderen Betriebssystemen verwenden wollen, erstellen Sie auf Basis der .bat-Datei bitte ein äquivalentes Shell-Script (z.B. `ctitest.sh`).

2.4. Einrichten der Telefonie

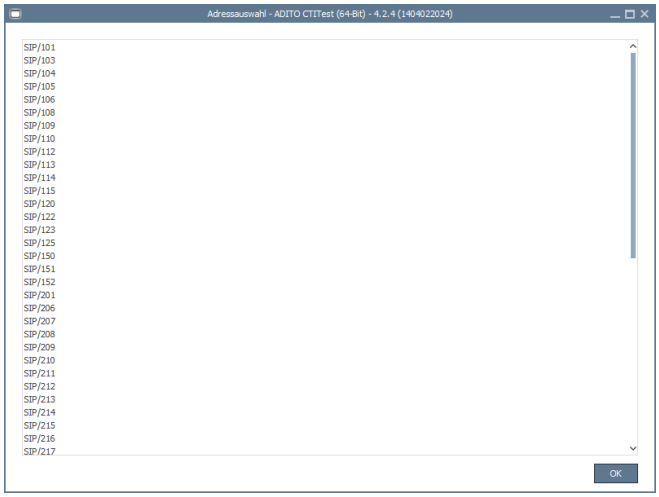
2.4.1. Auswahl des Providers

Wenn Sie das Programm starten, werden Sie nach dem Tapi-Provider gefragt. Hier können Sie den Provider einstellen, eventuell auch noch das Logging aktivieren. Geben Sie hier einfach die Zugangsdaten zum Telefonie-System ein.



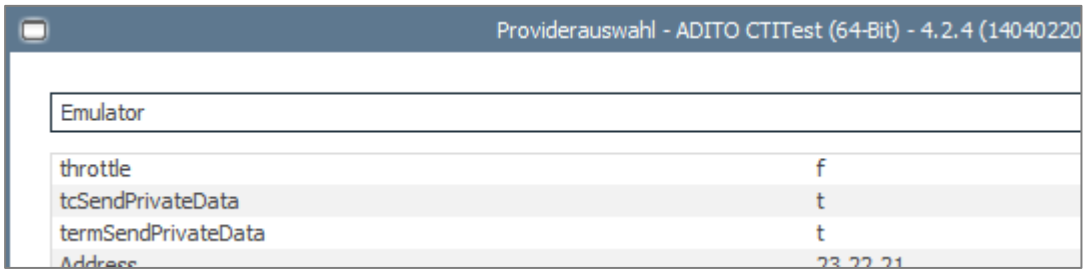
Es folgt eine Auswahl der vorhandenen Adressen, wählen Sie hier die gewünschte Adresse, um die Telefonie-Funktionalität durchzuführen.

Bei Asterisk:

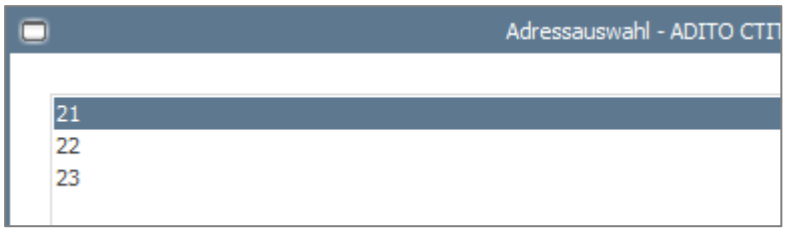


Bei Tapi wählen Sie die integrierte TAPI3-Schnittstelle aus. Die Auswahl der möglichen Adressen hängt von der Telefonanlage und dem TSP (Tapi Service Provider) ab.

Wollen Sie hingegen nur testen, wie die Telefonie im Allgemeinen funktioniert, können Sie das System auch über einen Emulator verwenden. Wählen Sie dazu beim Start des Programms in die Eingabebox „Emulator“ ein.



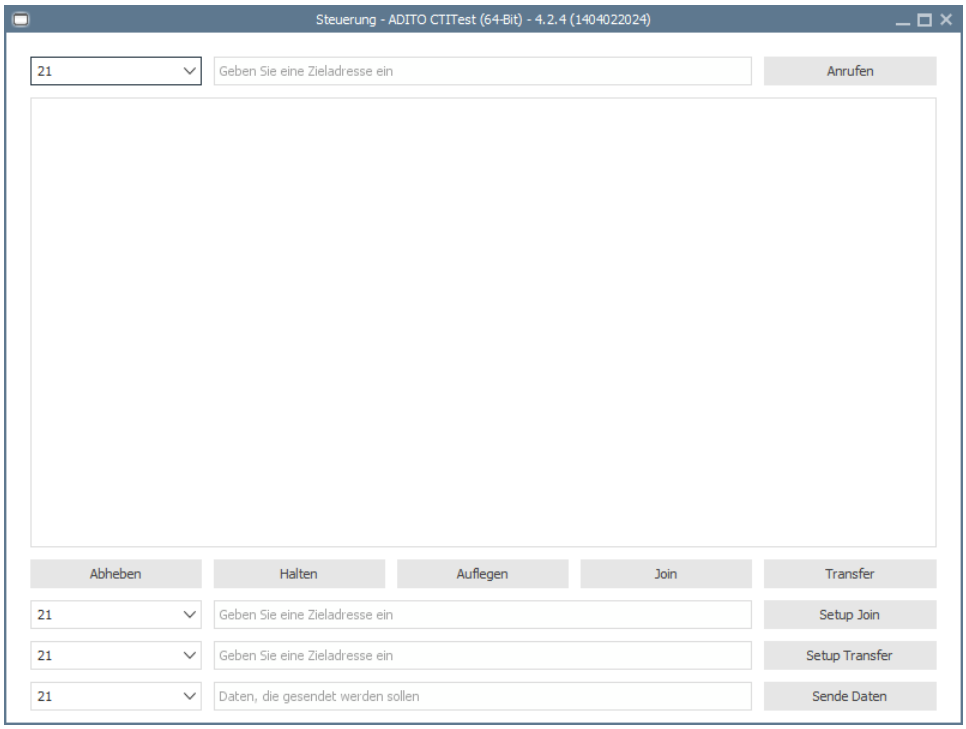
In diesem Fall stehen drei Demo-Adressen zur Auswahl, aus denen eine verwendet werden kann.



2.4.2. Die Programmfenster

Die Verwendung des Telefonie-Testprogramms wird anhand der Einstellung „Emulator“ beschrieben. Hier besteht die Möglichkeit alle Gesprächspartner in einem einzigen Fenster zu beobachten.

Nachdem Sie sich für eine Line entschieden haben, wird das Programm mit zwei Fenstern geöffnet.



Das Fenster `Steuerung` dient der Steuerung des verwendeten Anschlusses.

Sie haben hier die Möglichkeit, einen Anruf zu starten, zu beantworten, zu halten oder einer Konferenz beizutreten. Sie können darüber hinaus einen bestehenden Call weiterleiten.

`sende Daten` dient Funktionen, die in der Schnittstelle möglicherweise nicht definiert sind. Diese Funktionen müssen in der Spezifikation des TAPI Service-Providers angegeben sein – dadurch ermöglichen sich beispielsweise gewisse Erweiterungen der Telefonanlage ohne JDito-Funktionen.

Das unbenannte Fenster dient als Übersicht über alle vorhandenen Nummern. Es handelt sich hier um drei emulierte Telefone, die untereinander alle Telefonie-Möglichkeiten verwenden können.

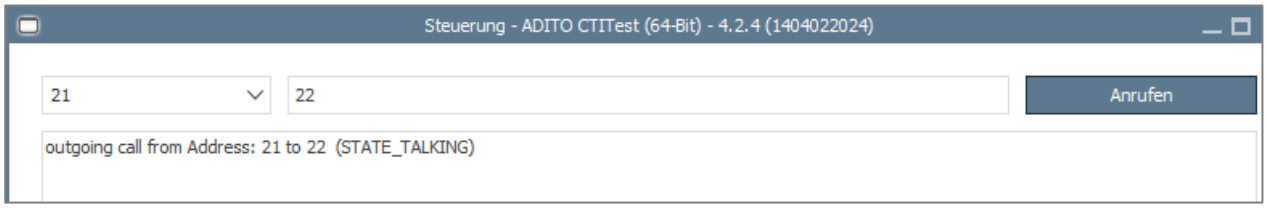
Die Schaltfläche `On Hook` simuliert dabei den Hörer. „On Hook“ bedeutet aufgelegt, bei Klick ändert sich die Beschriftung auf `Off Hook`, diese Einstellung entspricht einem abgenommenen Hörer.

2.5. Testen der Telefonie-Funktionen

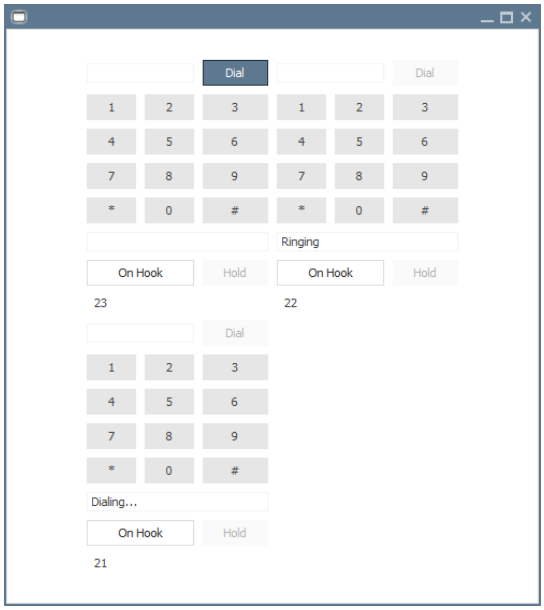
2.5.1. Anrufen eines Teilnehmers

Um in der Einstellung „Emulator“ jemanden anzurufen, verwenden Sie das Control Panel. Mit dem von Ihnen gewählten Anschluss können Sie hier mit den anderen Apparaten interagieren.

Tragen Sie in das Eingabefeld der `Steuerung` eine Nummer ein und drücken auf `Anrufen`, um einen Anruf durchzuführen.



Im oberen Teil des Fensters befindet sich eine Auswahlliste, in der nun der entsprechende Anruf erscheint.



Im Übersichtsfenster der drei Apparate sehen Sie bei beiden Anschlüssen nun, dass sich der Status der Telefonate geändert hat. Bei der Nummer, die gewählt hat, wird der Status `Dialing…` angezeigt, bei der angerufenen Nummer läutet das Telefon (`Ringing`).

Wird der Hörer durch Klick auf `On Hook` abgenommen, ändert sich wiederum der Status der beiden betroffenen Telefone. Beide Apparate bekommen den Status `Active`.

Wird nun versucht, mit dem dritten Telefon eine der beiden Nummern anzurufen, erscheint als Status `Busy Signal`, also das Besetztzeichen.

Sie können auch die anderen beiden Teilnehmer untereinander telefonieren lassen. Klicken Sie dazu zuerst auf `On Hook` eines Teilnehmers, geben die gewünschte Nummer ein und klicken auf `Dial`.

2.5.2. Weiterleiten eines Anrufs

Einen bestehenden Anruf können Sie nur als Teilnehmer der Leitung weiterleiten, die Sie Anfangs gewählt haben. Klicken Sie dazu in einem bestehenden Anruf auf `Halten` um ihn zu halten.

Wählen Sie in der Anrufliste ein gehaltenes Gespräch aus, geben eine Nummer in das Eingabefeld neben `Setup Transfer` ein und klicken auf `Setup Transfer`.

Der Gesprächspartner der eben eingegebenen Nummer muss nun abheben.

Nun kann auch ein Vermittlungsgespräch (Rücksprache) gehalten werden.

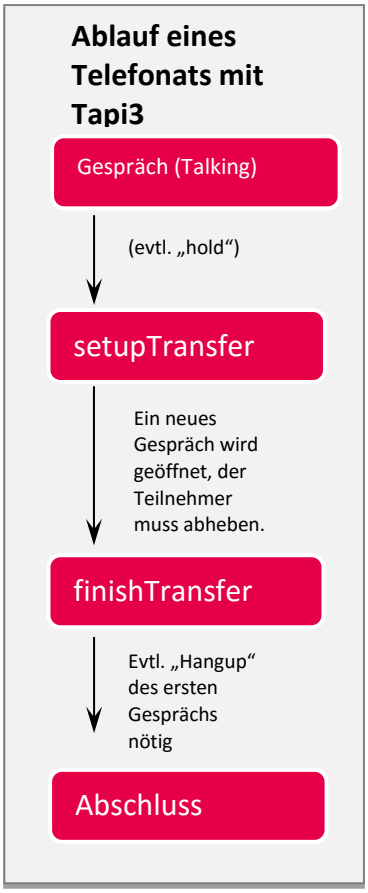
Jetzt können die beiden Gespräche (das gehaltene und das neue Gespräch) markiert werden und der Transfer über die Schaltfläche `Transfer` beendet werden.

2.5.3. Rücksprache und Weiterleitung mit Asterisk und Tapi3.

Je nach verwendeter Telefonanlage und Treiber kann der Ablauf einer Weiterleitung bzw. einer Rücksprache anders erfolgen.

Tapi3: Um eine Weiterleitung oder eine Rücksprache vorzunehmen, ist `setup join` bzw. `setup transfer` notwendig. Ob vorher setzen des Telefongesprächs auf „hold“ nötig ist, hängt von der Telefonanlage ab.

Asterisk: `Join` und `transfer` sind hier direkt, ohne setup möglich, auch `hold` ist in der Regel nicht nötig.



2.5.4. Testen der TAPI-Funktionalität bei Verwendung eines Asterisk-Telefonieservers

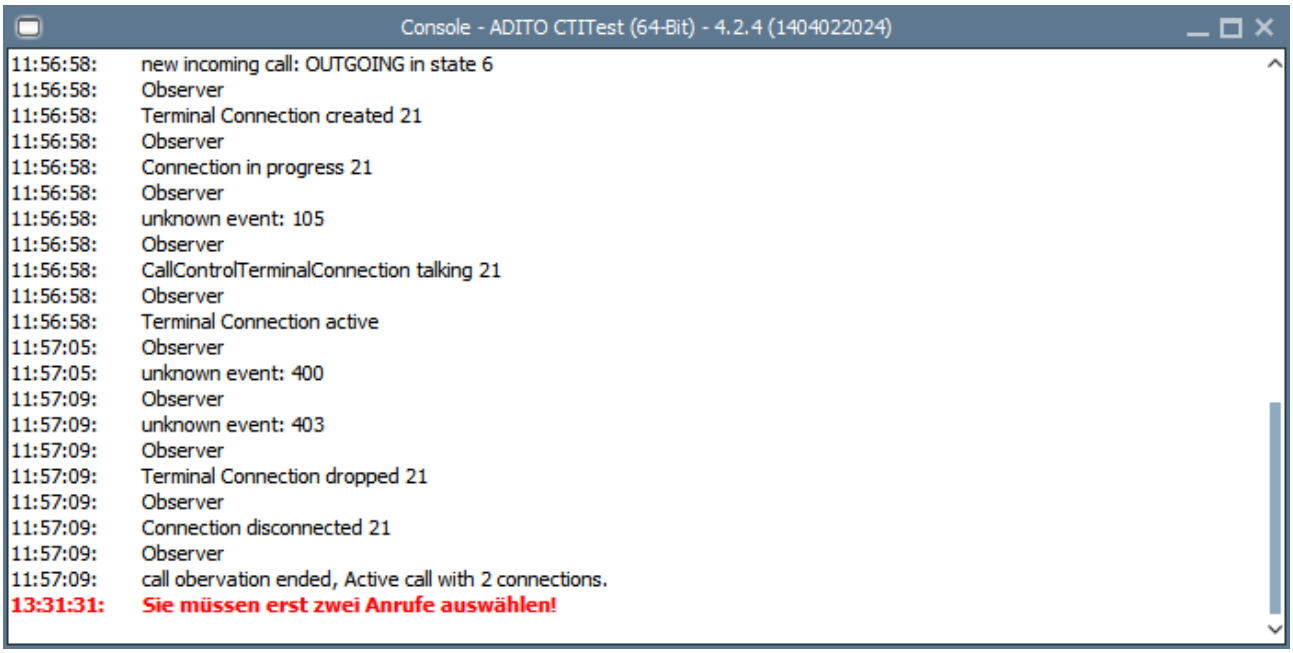
Sollte Ihnen zum Testen ein Asterisk-Telefonieserver zur Verfügung stehen, Sie aber die TAPI3-Schnittstelle von ADITO testen möchten, so können Sie den Activa TSP verwenden. Dieser verbindet sich wie ADITO direkt mit Ihrem Asterisk TSP, stellt aber einen Tapi3-Treiber zur Verfügung.

Diesen TSP können Sie unter http://activa.sourceforge.net/ herunterladen. Er ist mit ADITO4 und dem CTI-Testprogramm kompatibel.

2.6. Administration

2.6.1. Konsolenausgabe

Das CTI-Testprogramm schreibt alle Informationen über die jeweiligen Gespräche auf die Konsolenausgabe. Sollte ein Fehler auftauchen und ein bestimmter Status des Gesprächs nicht möglich sein, so erscheint auch hier in der Konsolenausgabe ein Fehler.



3. CTI und JDito

3.1. Liste der vorhandenen Methoden

Die für die Integration der Telefonie verfügbaren Methoden lassen sich in die drei Kategorien Verbindungsauf- und abbau, Darstellung von Verbindungen und Konfiguration einteilen.

3.1.1. Verbindungen

```
cti.answer()
cti.call()
cti.forward()
cti.hangup()
cti.join()
cti.switchHold()
```

3.1.2. Darstellung

```
cti.addRefreshable()
cti.removeRefreshable()
```

3.1.3. Konfiguration

```
cti.getAvailableAddresses()
cti.getCurrentCallCount()
cti.getCurrentCalls()
cti.getOutsideLineAccess()
cti.setCallEndProcess()
cti.setCallStartProcess()
cti.setCallStateChangedProcess()
cti.setPrivateDataCallbackProcess()
```

3.2. Konstanten und Statuswerte

Bitte denken Sie daran, bei der Nutzung im Code allen Namen von Konstanten oder Verbindungszuständen den Namensraum `telephony` voranzustellen. Aus `ISINCOMING` wird also ein `telephony.ISINCOMING`. Der JDito-Assistent übernimmt den Namensraum automatisch.

Konstante	Bedeutung
ADDRESS	Enthält die Nummer des Anrufers
CALLID	Eindeutige Kennung für diese Verbindung
INVALID	Ungültige Verbindung

ISINCOMING	Dies ist eine eingehende Verbindung
STATE	Enthält den Status der Verbindung (s.u.)
LOCALADDRESS	
LOCALID	

Tabelle: Schlüssel für die Map mit den aktiven Verbindungen

Der Status einer Verbindung ist über die Konstanten der untenstehenden Tabelle abfragbar.

Konstante	Bedeutung
CALLSTATE_UNKNOWN	Unbekannter Verbindungszustand
CALLSTATE_CONNECTING	Verbindung wird aufgebaut
CALLSTATE_TALKING	Aktive Verbindung, Gepräch läuft
CALLSTATE_RINGING	Klingelton, ausgehende Verbindung
CALLSTATE_CONNECTION_RINGING	Klingelton, ankommende Verbindung
CALLSTATE_ON_HOLD	Verbindung wird gehalten
CALLSTATE_CREATED	Verbindungsobjekt wurde erzeugt
CALLSTATE_DISCONNECTED	Verbindung wurde abgebaut
CALLSTATE_BUSY	Besetzt, Teilnehmer nicht erreichbar

Tabelle: Liste der möglichen Verbindungszustände

3.3. Prozesse und Prozessaufrufe

3.3.1. Automatischer Aufruf von Prozessen

Für die Annahme von Verbindungen bzw. die Reaktion des Systems auf Zustandswechsel wurde eine Möglichkeit geschaffen, eigene JDIto-Prozesse in das System einzuhängen. Diese werden dann bei Eintritt des entsprechenden Ereignisses automatisch ausgeführt. Derzeit sind drei solcher Einhängemöglichkeiten vorhanden.

- Bei Aufbau einer Verbindung.
- Bei Abbau einer Verbindung.
- Bei Statuswechsel einer Verbindung.

Um Ihre eigenen Prozesse zu nutzen, wird der Name des jeweiligen Prozesses einer der drei Prozessmethoden (s.u.) als Parameter übergeben.

3.3.2. setCallEndProcess(prozessname)

Legt den Prozess fest, der beim Beenden einer Verbindung ausgeführt werden soll. Als Parameter wird der Prozessname angegeben. Wird üblicherweise im `autostart`-Prozess gesetzt.

3.3.3. setCallStartProcess(prozessname)

Legt den Prozess fest, der beim Aufbau einer Verbindung ausgeführt werden soll. Als Parameter wird der Prozessname angegeben. Wird üblicherweise im `autostart`-Prozess gesetzt.

3.3.4. setCallStateChangedProcess(prozessname)

Diese Methode legt fest, welcher Prozess ausgeführt wird, falls sich der Zustand einer Verbindung ändert bzw. Verbindungen erstellt oder gelöscht werden. Als Parameter wird der Prozessname angegeben. Wird üblicherweise im `autostart`-Prozess gesetzt.

3.3.5. setPrivateDataCallbackProcess (prozessname)

In diesem Prozess können Sie auf privateData-Ereignisse reagieren. PrivateData-Ereignisse sind spezielle Ereignisse, welche die jeweilige Telefonanlage anbietet. In diesem Prozess können Sie auf alle Variablen zugreifen, die auch in anderen Prozessen existieren (CallStart / CallEnd / StateChanged). Zusätzlich steht hier eine lokale Variable `$local.privateData` und `$local.privateData2` (als Map mit ausführlicheren Informationen) zur Verfügung, welche die privaten Daten als String-Array enthält.

3.3.6. Übergabevariablen für Prozesse

Um die Feststellung der Verbindung zu erleichtern, auf die sich die Prozessaktivierung bezieht, stehen eine Reihe von Variablen zur Verfügung. Diese werden als prozesslokale Variablen automatisch erzeugt und können mit der Methode `a.valueof()` ausgelesen werden.

- `$local.callID` Diese Variable enthält die Call-ID für die Verbindung, die den Prozess aktiviert hat. „I“ und „D“ werden in Großbuchstaben geschrieben!
- `$local.callState` Diese Variable enthält den Status der Verbindung als einen der Werte aus der Tabelle mit den Verbindungszuständen weiter oben.
- `$local.callAddress` Diese Variable enthält die Nummer des Anrufers für die Verbindung, falls eine Übertragung der Rufnummer erfolgt.
- `$local.callIsIncoming` Diese Variable enthält den Wert `true`, falls es sich bei der Verbindung, die den Prozess aktiviert hat, um einen eingehenen Anruf handelt.
- `$local.privateData2` Diese Variable enthält diejenigen Daten der Telefonanlage, die nicht über die Standardwerte (callAddress usw.) abgefragt werden können. Diese Variable ist ein Objekt-Array. Die möglichen Einstellungen sind von der eingesetzten Telefonanlage abhängig.

3.3.7. Implementierungshinweise

Je nach Implementierung des TAPI-Treibers werden bei einem Anruf, bzw. beim Beenden eines Anrufs, sowohl `CallStartProcess` (bzw. `CallEndProcess`), als auch `CallStateChangedProcess` aktiviert oder nicht. Im Normalfall sollten beide Prozesse ausgeführt werden.

4. Umsetzung im xRM Basissystem

Im ADITO-Basissystem stehen die Funktionsbibliotheken

- `cti_log`
- `lib_telephony`

zur Verfügung. Im Prozess `autostart` wird festgelegt, dass der Process `cti_log` als CallState-Prozess verwendet wird:

```
cit.setCallStateChangedProcess("cti_log");
```

4.1. Systemoptionen im Client

Im Frame `Systemverwaltung` können Sie Systemoptionen für die Telefonie einstellen.

Details zu den Systemoptionen finden Sie in den ADITO-Informationsdokumenten AID096-DE.

4.2. Anrufprozess cti_log

Dieser Prozess liest zuerst alle Systemoptionen und lokalen Variablen aus.

4.2.1. Prüfung der privateData

privateData Events hängen von der Telefonanlage ab, an die ADITO angebunden ist. Im Referenzsystem werden standardmäßig einige Events abgefragt, und zwar:

- Anrufernummer
- Nummer des Angerufenen
- Nummer desjenigen Teilnehmers, der den Anruf verbunden hat

Weitere Events können abgefragt werden, aber hängen stark von der jeweiligen Telefonanlage ab.

4.2.2. Anruferidentifizierung und Gesprächsführung

Unabhängig von der privateData wird im Folgenden im Prozess abgearbeitet. Der Anrufer wird geöffnet, falls erkannt und gewünscht, die Historie wird geöffnet, das Log wird geschrieben.

4.3. Bibliothek lib_telephony

Diese Funktionsbibliothek lib_telephony bietet Funktionen an, um vereinfachte Telefonie-Zugriffe zu ermöglichen. Folgende Funktionen bietet diese Funktionsbibliothek an:

Option	Auswirkung
call	Dieser Funktion wird nur eine Telefonnummer übergeben. Sie versucht automatisch, auf Basis dieser übergebenen Nummer eine passende Telefonnummer zu generieren.

getPhoneNumber	Bereinigt die Telefonnummer eines übergebenen Strings und gibt eine sauber formatierte Telefonnummer zurück.
cleanPhoneNumber	Wie getPhoneNumber, nur wird hier die Vorwahl nicht ausgelesen.
getSearchAddr	Holt sich die Suchadresse zu einer bestehenden Telefonnummer.
openCTIContact	Öffnet den Kontakt zu einer übergebenen Telefonnummer auf Basis der CTILog-Tabelle.
getRelIDsfromCall	Gibt zu einer übergebenen Telefonnummer die RELATIONID zurück.
openCallerFrame	Öffnet den Anrufer auf Basis der RELATIONID und übergebenen Caller-Adresse.