Entire Connection

Programmierschnittstelle (API)

Version 4.5.4

November 2016

ADABAS & NATURAL

Dieses Dokument gilt für Entire Connection ab Version 4.5.4.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Dokument-ID: PCC-API-454-20161118DE

Inhaltsverzeichnis

Vorwort

Mit der Programmierschnittstelle (Application Programming Interface - API) können Sie Funktionen von Entire Connection direkt aus einem Programm heraus aufrufen. Ein ActiveX-Control stellt eine gemeinsame Schnittstelle für die Entwicklung mit Visual Basic .NET, C++ oder C# zur Verfügung.

Dieser Abschnitt behandelt die folgenden Themen:

Allgemeine Informationen

Übersicht der API-Aufrufe

Andere Ereignisse, Tasten-Codes und Fehler-Codes

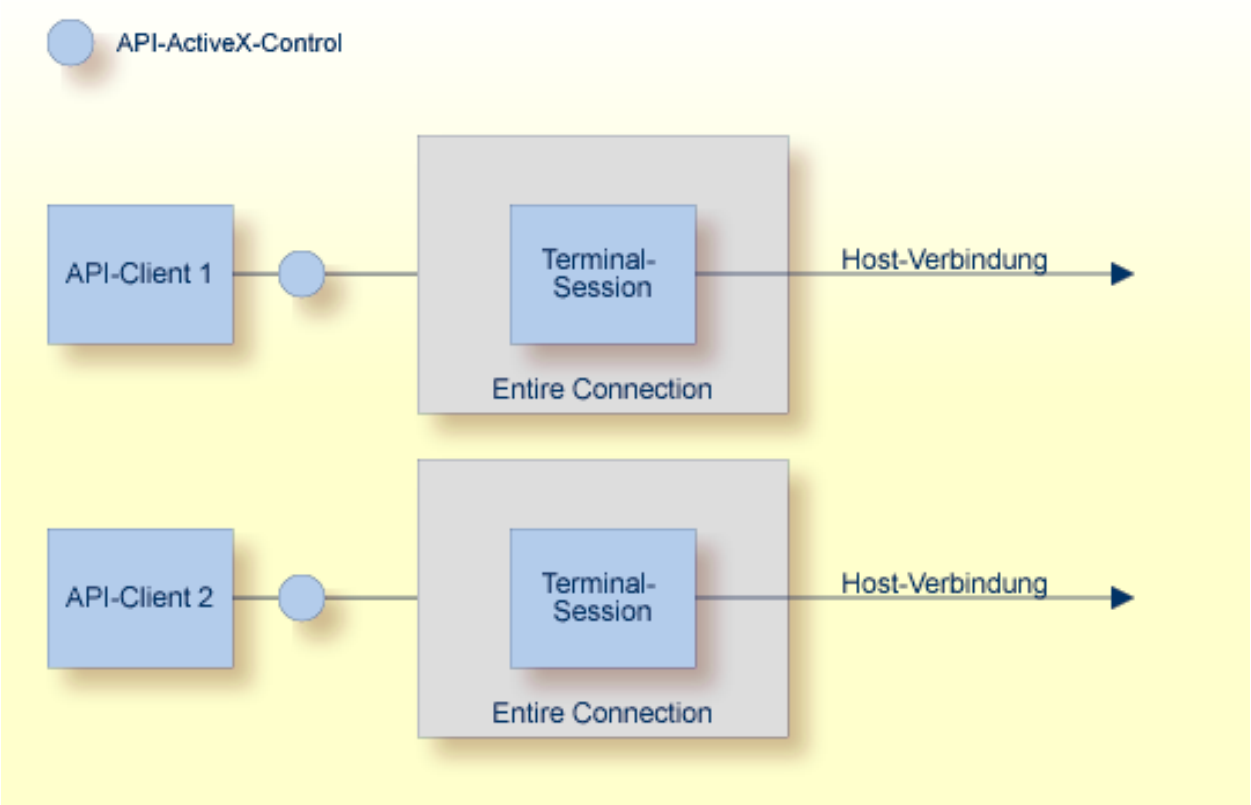
Es wird vorausgesetzt, dass Sie bereits mit ActiveX-Controls (entweder mit Visual Basic .NET, C++ oder C#) und Entire Connection vertraut sind.

Diese Beschreibung sollte zusammen mit dem Beispielcode gelesen werden, der sich auf dem Installationsmedium befindet. Sie finden den Beispielcode im Verzeichnis *Windows\API* des Installationsmediums.

1 Allgemeine Informationen

API-Controls und Terminal-Sessions

Jedes API-Control kann die Verbindung zu einer bestehenden Terminal-Session herstellen oder eine neue Terminal-Session erstellen. Jede Terminal-Session kann jederzeit mit einem API-Control verbunden sein. Die einzige Ausnahme ist der UA-Modus, bei dem das Verbinden nicht erlaubt ist. Außerdem ist es nicht möglich, den UA-Modus bei einem Terminal einzuschalten, das durch die Programmierschnittstelle kontrolliert wird.



Im API-Modus ist eine Terminal-Session in der Regel unsichtbar, um Benutzereingaben zu verhindern. Wenn das Terminal durch die Programmierschnittstelle sichtbar gemacht wird, hat der Benutzer die volle Kontrolle über das Terminal. Unter anderem kann er dann Prozedurdateien ausführen und die Terminal-Session beenden. Jeder Datentransfer und jede Prozedurdatei bleibt unter der Kontrolle des API-Client.

Synchrone und asynchrone Aufrufe

Synchrone (blockierende) und asynchrone (nicht-blockierende) Aufrufe stehen mit Visual Basic .NET, C++ und C# zur Verfügung. Während der Planungsphase entscheiden Sie, welche dieser beiden Aufrufarten für Ihre Zwecke am Besten geeignet ist.

Im asynchronen Modus kehren fast alle API-Aufrufe sofort mit einem entsprechenden Return-Code zurück. Ausnahmen sind die Initialisierungsfunktionen und die Funktionen zum Beenden der Verbindung mit einer Terminal-Session. Diese Funktionen blockieren immer, unabhängig vom gewählten Modus.

Nachdem ein Befehl bei der asynchronen API-Ausführung abgearbeitet wurde, sendet das Control ein Ereignis, das bestätigt, dass der Befehl abgearbeitet wurde. Die Parameter dieses Ereignisses enthalten das Ergebnis des Aufrufs (d.h. eine Rückmeldung wie der Befehl abgearbeitet wurde) und alle angeforderten Daten.

Die Beschreibungen in der ***Übersicht der API-Aufrufe*** informieren Sie darüber, ob ein Aufruf nur synchron ausgeführt werden kann. In allen anderen Fällen wird ein Ereignis gesendet, das bestätigt, dass der Befehl abgearbeitet wurde; `LogonEntireConnection` sendet zum Beispiel `LogonComplete`.

In bestimmten Situationen erzeugt das API-Control auch Benachrichtigungsereignisse ohne Berücksichtigung des Modus, in dem es gerade ausgeführt wird. Dies können Fehlermeldungen, Informationen und Datentransferdaten sein.

Glossar

API	Funktionalität, die Drittanbieteranwendungen zur Verfügung steht.
API-Client	Die Anwendung, die Entire Connection mit Hilfe der Programmierschnittstelle kontrolliert.
API-Control	Das vom API-Client verwendete ActiveX.
Terminal-Session	Die Terminal-Anwendung von Entire Connection.
Asynchron	Nicht-blockierender Modus. Die Programmierschnittstelle kehrt sofort zu der aufrufenden Anwendung zurück. Wenn die Verarbeitung abgeschlossen ist, sendet die Programmierschnittstelle eine Nachricht an die Anwendung.
Synchron	Die Programmierschnittstelle kehrt nur dann zu der aufrufenden Anwendung zurück, wenn die Verarbeitung des Funktionsaufrufs abgeschlossen ist.

2 Übersicht der API-Aufrufe

Dieser Abschnitt enthält einen Überblick über alle verfügbaren API-Aufrufe. Sie sind in die folgenden funktionalen Bereiche aufgeteilt:

- **Initialisierung**
  - GetRunningTerminalSessions
  - Initialize
  - LogonEntireConnection
- **Session öffnen**
  - GetAvailableSessions
  - OpenSession
- **Allgemeine Steuerung**
  - RunHostCommand
  - PutData
  - SetDataNotificationFlag
- **Bildschirmdaten**
  - GetScreenText
  - GetScreenRawText
  - GetScreenAttributes
  - GetExtendedAttributes
  - GetCursorPosition
  - SetCursorPosition
  - ClearScreenText
  - CheckForScreenText
- **Datentransfer**
  - SetAPIFileDetails
  - SetWorkFileDetails
  - GetFileName
  - CancelFileTransfer
- **Tasks und Prozedurdateien**
  - RunEntConTask
  - SetGlobalParameter
  - GetGlobalParameter
  - CancelRunningTask
- **Session beenden**

- `CloseSession`
- `CloseAllSessions`
- `BreakConnection`

- **Andere Methoden**
  - `GetScreenSize`

Diese API-Aufrufe (und alle damit verbundenen Ereignisse) sind nachstehend ausführlich beschrieben.

Initialisierung

Beim Starten einer Session kann der API-Client entweder die Verbindung zu einem aktiven Terminal herstellen oder ein neues Terminal erstellen.

Session-Namen aller aktiven Terminals abfragen (nur synchroner Aufruf)

- Rufen Sie Folgendes auf:

```
APIReturn = GetRunningTerminalSessions(TerminalNames, NumTerminals)
```

Dies gibt alle zur Zeit aktiven Terminals, zu denen eine Verbindung hergestellt werden kann, in einem Array aus. Der Aufruf der Funktion `GetRunningTerminalSessions` ist der einzige Aufruf, der vor dem Aufruf von `Initialize` erfolgen kann.

Verbindung zu einem Terminal herstellen

- Rufen Sie Folgendes auf:

```
APIReturn = Initialize(CreateSession, LinkSessionName, UserLoggedOn, OpenSession)
```

Die Parameter sind:

`CreateSession`	Boolean. "true" bedeutet, dass ein neues Terminal erstellt werden soll.
`LinkSessionName`	String. Name eines vorhandenen Terminals, zu dem eine Verbindung hergestellt werden soll. Der Name ist einer der Terminal-Namen, der von der Funktion `GetRunningTerminalSessions` geliefert wird.
`UserLoggedOn`	Boolean. Gibt "true" zurück, wenn die Anmeldung bei Entire Connection auf der Workstation bereits erfolgt ist. Damit ein Terminal benutzt werden kann, muss sich der Benutzer pro Arbeitsplatz einmal anmelden. Wenn `UserLoggedOn` "false" ist, muss sich der API-Client jetzt anmelden.

`OpenSession`	String. Normalerweise leer. In einer besonderen Situation wird der Name einer geöffneten Session eingetragen.

Wenn für `CreateSession` "true" übergeben wird oder keine Verbindung zum angegebenen Terminal hergestellt werden kann, erstellt das API-Control ein neues Terminal.

Wenn die Verbindung zu einem vorhandenen Terminal hergestellt wurde und in der Zwischenzeit eine Session in diesem Terminal geöffnet wurde, enthält der Parameter `OpenSession` den Namen der Session. Der API-Client muss in dieser speziellen Situation entscheiden, ob er mit dieser Session, die nicht unter seiner Kontrolle geöffnet wurde, arbeiten will. Diese Situation kann nur entstehen, wenn zum Zeitpunkt der Abfrage der vorhandenen Terminals ein Terminal gerade im Begriff war, eine Session zu öffnen, der Vorgang jedoch länger dauerte und noch nicht abgeschlossen war.

Bei Entire Connection anmelden

- Rufen Sie Folgendes auf:

```
APIReturn = LogonEntireConnection(UserName, Password)
```

Session öffnen

Der API-Client kann entweder die zur Verfügung stehenden Session-Namen aus der Share-Datei abfragen oder eine namentlich bekannte Session direkt öffnen.

Alle Sessions abfragen, die für einen Benutzer von Entire Connection definiert sind

- Rufen Sie Folgendes auf:

```
APIReturn = GetAvailableSessions(SessionNames, DefaultSession)
```

Die Parameter sind:

`SessionNames`	Variant Array(Strings). Die Namen aller definierten Sessions.
`DefaultSession`	String. Der Name der Standard-Session.

Eine dieser Sessions öffnen

- Rufen Sie Folgendes auf:

```
APIReturn = OpenSession(SessionName)
```

Der Parameter ist:

`SessionName`	String. Der Name der zu öffnenden Session.

Die Session ist nun offen und kann benutzt werden.

Hiermit verbundene Ereignisse:

- `FirstScreenArrived`

  Wird gesendet, wenn die Session die ersten Daten vom Host erhält.

- `ScreenSizeChanged(NumRow, NumColumns)`

  Gibt die anfängliche Bildschirmgröße an und auch, ob sich das Terminal während der Session dynamisch verändert.

- `SessionOpened(SessionName)` Wird gesendet, wenn eine Session geöffnet wird ohne dass der API-Client die Methode `OpenSession` aufruft. Dies kann zum Beispiel durch einen Start-Task geschehen.

  Der Parameter ist:

`SessionName`	String. Der Name der geöffneten Session.

Allgemeine Steuerung

Befehle an die geöffnete Session senden

- Rufen Sie Folgendes auf:

```
APIReturn = RunHostCommand(CommandName)
```

Der Parameter ist:

`CommandName`	String. Der Name des Befehls, der auf dem Host aufgerufen werden soll.

Der String wird an den Host gesendet und anschließend an die Funktionstaste ENTER.

Beliebigen Text und Tasten-Codes senden

- Rufen Sie Folgendes auf:

```
APIReturn = PutData(Text, KeyCode)
```

Die Parameter sind:

`Text`	String. Text, der an den Host übertragen werden soll.
`KeyCode`	Integer. Taste, die nach der Übertragung des Textes gesendet werden soll.

Der Text, der mit diesem Befehl gesendet wird, kann Zeilenvorschübe enthalten. Diese werden wie das Drücken der Funktionstaste NEWLINE interpretiert. Wenn Sie nur einen Tasten-Code senden wollen, müssen Sie als Text eine leere Zeichenkette übergeben.

Datenbenachrichtigungen erlauben (nur synchroner Aufruf)

- Rufen Sie Folgendes auf:

```
APIReturn = SetDataNotificationFlag(Enable)
```

Der Parameter ist:

`Enable`	Boolean. Wenn man dies auf "true" setzt, werden Datenbenachrichtigungen eingeschaltet. Vorgabe: ausgeschaltet.

Anzeige des Terminal-Fensters ein- und ausschalten

- Setzen Sie die API-Control-Eigenschaft `TerminalInteractive` (boolean).

  Wenn Sie die Verbindung zu einem Terminal herstellen, bleibt es solange sichtbar bis dieser Wert auf "false" gesetzt wird.

  Wenn Sie ein neues Terminal erstellen, ist es solange unsichtbar bis dieser Wert auf "true" gesetzt wird.

Hiermit verbundene Ereignisse:

- `CursorPositionChanged(XPosition, YPosition)`

  Wird gesendet, wenn sich das Terminal im interaktiven Modus befindet und die Cursor-Position mit der Maus verändert wird (nicht wenn sich der Cursor durch Tippen bewegt).

- `NewScreenDataArrived()`

  Wenn eingeschaltet bedeutet dies, dass neue Daten vom Host angekommen sind.

Bildschirmdaten

Bildschirmtext steht als Rohtext zur Verfügung wie er vom Host empfangen wird und als verarbeiteter Text wie er im Terminal angezeigt wird. Der Rohtext enthält alle Zeichen - auch die, die nicht angezeigt werden sollen (z.B. Passwort) - und kann auch Nullwerte enthalten.

Da der Rohtext Nullwerte enthalten kann, kann er nur in ein Array mit vorzeichenlosen Zeichen ausgegeben werden. Der Bildschirmtext wird in ein Array mit Zeichenketten ausgegeben.

Bildschirmtext abfragen

- Rufen Sie Folgendes auf:

```
APIReturn = GetScreenText(ScreenTextArray, TopLeftX, TopLeftY, BottomRightX, ↩
BottomRightY)
```

Die Parameter sind:

`ScreenTextArray`	Variant Array(Strings). Eine Zeichenkette pro angeforderter Textzeile.
`TopLeftX`	Integer. Startkoordinate.
`TopLeftY`	Integer. Startkoordinate.
`BottomRightX`	Integer. Endkoordinate.
`BottomRightY`	Integer. Endkoordinate.

Wenn eine dieser Koordinaten auf -1 gesetzt wird, wird der gesamte Bildschirm ausgegeben.

Rohdaten abfragen

- Rufen Sie Folgendes auf:

```
APIReturn = GetScreenRawText(ScreenTextArray)
```

Der Parameter ist:

`ScreenTextArray`	Variant Array(Unsigned chars). Rohdaten-Buffer.

Bildschirmattribute abfragen

- Rufen Sie Folgendes auf:

```
APIReturn = GetScreenAttributes(Attributes, AttributesDescription)
```

Die Parameter sind:

`Attributes`	Variant Array(Unsigned chars). Attribut-Buffer.
`AttributesDescription`	Variant Array(Unsigned chars). Die Beschreibung eines Attributs ist ein Array mit 6 Werten, das die Bit-Muster für die Attributeigenschaften enthält:

Member 0:	Attribut
Member 1:	Geschützt
Member 2:	Numerisch
Member 3:	Keine Anzeige
Member 4:	Erhöhte Helligkeit
Member 5:	Feldinhalt wurde geändert

Erweiterte Bildschirmattribute abfragen

- Rufen Sie Folgendes auf:

```
APIReturn = GetExtendedAttributes(ExtendedAttributes)
```

Der Parameter ist:

`ExtendedAttributes`	Variant Array(Unsigned chars). Buffer mit erweiterten Attributen.

Aktuelle Cursor-Position ermitteln und verändern

- Rufen Sie Folgendes auf:

```
APIReturn = GetCursorPosition(XPosition, YPosition) APIReturn = ↩
SetCursorPosition(XPosition, YPosition)
```

Die Parameter sind:

`XPosition`	Integer. X steht für die Spalte der Cursor-Position.
`YPosition`	Integer. Y steht für die Zeile der Cursor-Position.

Editierbaren Text im angegebenen Bereich entfernen

- Rufen Sie Folgendes auf:

```
APIReturn = ClearScreenText(TopLeftX, TopLeftY, BottomRightX, BottomRightY)
```

Die Parameter sind:

TopLeftX	Integer. Startkoordinate.
TopLeftY	Integer. Startkoordinate.
BottomRightX	Integer. Endkoordinate.
BottomRightY	Integer. Endkoordinate.

-1 in einem dieser Werte heißt: der gesamte Bildschirm.

IF-Befehl zum Suchen von Bildschirmtext aufrufen

- Rufen Sie Folgendes auf:

```
APIReturn = CheckForScreenText(Text, Result, Position, TopLeftX, TopLeftY, ↩
Length, CaseSensitive)
```

Die Parameter sind:

Text	String. Zu suchender Text.
Result	Boolean. "true" wenn der Text gefunden wurde.
Position	Integer. Bildschirmposition, in der der Text gefunden wurde.
TopLeftX	Integer. Startkoordinate.
TopLeftY	Integer. Startkoordinate.
Length	Integer. Textlänge.
CaseSensitive	Boolean. "true" wenn Überprüfung auf Groß-/Kleinschreibung.

Datentransfer

Datentransfer vorbereiten für den direkten Transfer mit dem API-Client

- Rufen Sie Folgendes auf:

```
APIReturn = SetAPIFileDetails(WorkFileNumber, UploadFlag, BinaryFlag, ReportFlag)
```

Die Parameter sind:

`WorkFileNumber`	Integer. Nummer des Work File.
`UploadFlag`	Boolean. Wird zum Hochladen gesetzt.
`BinaryFlag`	Boolean. Wird zum Übertragen von Binärdateien gesetzt.
`ReportFlag`	Boolean. Wird für das Report-Format gesetzt.

Die folgenden Ereignisse werden beim Hochladen gesendet:

```
GetAsciiUploadFileBuffer(ErrorCode, FileNumber, Data, DataLength, DataFormat)

GetBinaryUploadFileBuffer(ErrorCode, WorkFileNumber, Data, DataLength)
```

Die folgenden Ereignisse werden beim Herunterladen gesendet:

```
AsciiFileDataArrived(ErrorCode, FileNumber, DataLength, Data, DataFormat)

BinaryFileDataArrived(ErrorCode, FileNumber, DataLength, Data, DataFormat)
```

Die Ereignisparameter sind:

`ErrorCode`	Integer. Muss von der Programmierschnittstelle auf 0 gesetzt werden, um daraufhinzuweisen, dass der Vorgang ohne Fehler bearbeitet wurde.
`FileNumber`	Integer. Das zu verarbeitende Work File.
`DataLength`	Integer. Hochladen: übergeben wird die erwartete Größe; zurückgegeben wird die tatsächliche Größe. Herunterladen: wird auf die Größe der übertragenen Daten gesetzt.
`Data`	Variant Arry(unsigned char). Die zu übertragenden Daten.
`DataFormat`	String. Beschreibung des Datensatzformats.

Bei einem normalen Datentransfer muss der API-Client einen Dateinamen angeben. Dieser Name kann vordefiniert werden.

Dateiname vordefinieren

- Rufen Sie Folgendes auf:

```
APIReturn = SetWorkFileDetails(Name, FileNumber, Upload, Binary, Report)
```

Die Parameter sind:

`Name`	String. Zu verwendender Dateiname.
`FileNumber`	Integer. Das verwendete Work File.
`Upload`	Boolean. Wird zum Hochladen gesetzt.
`Binary`	Boolean. Wird zum Übertragen von Binärdateien gesetzt.
`Report`	Boolean. Wird für das Report-Format gesetzt.

Wenn keine vordefinierten Werte für das verwendete Work File gefunden werden, wird der API-Client nach einem Dateinamen gefragt.

Dateiname übergeben

- Reagieren Sie auf folgendes Ereignis:

```
APIReturn = GetFileName(ErrorCode, FileNumber, Upload, Binary, ToPrinter, ↩
Landscape, ControlChars, DosFormat, FileName)
```

Die Parameter sind:

`ErrorCode`	Integer. Bei Null wird der Dateiname benutzt und die Verarbeitung beginnt. Bei einem anderen Wert wird die Verarbeitung abgebrochen.
`FileNumber`	Integer. Das verwendete Work File.
`Upload`	Boolean. Wird zum Hochladen eines Dateinamens gesetzt.
`Binary`	Boolean. Wird zum Übertragen von Binärdateien gesetzt.
`ToPrinter`	Boolean. Wird zum Herunterladen auf einen Drucker gesetzt.
`Landscape`	Boolean. Wird zum Drucken im Querformat gesetzt.
`ControlChars`	Boolean. Wird zum Interpretieren der Steuerzeichen gesetzt.
`FileName`	String. Zu verwendender Dateiname.

Laufenden Datentransfer abbrechen

- Rufen Sie Folgendes auf:

```
APIReturn = CancelFileTransfer(FileNumber)
```

Der Parameter ist:

`FileNumber`	Integer. Die Nummer des Work File, für das der Datentransfer abgebrochen wird.

Dieser Aufruf ist synchron. Er stellt eine Anfrage zum Abbrechen in eine Warteschlange. Wenn der Datentransfer abgeschlossen ist, wird das Ereignis `FileTransferComplete` gesendet.

Hiermit verbundene Ereignisse:

- `FileTransferStarting(ErrorCode, FileNumber, Upload, Binary, Headings)`

  Die Parameter sind:

`ErrorCode`	Integer. Bei Null wird der Dateiname benutzt und die Verarbeitung beginnt. Bei einem anderen Wert wird die Verarbeitung abgebrochen.
`FileNumber`	Integer. Das verwendete Work File.
`Upload`	Boolean. Wird zum Hochladen eines Dateinamens gesetzt.
`Binary`	Boolean. Wird zum Übertragen von Binärdateien gesetzt.
`Headings`	Variant Array (Strings). Enthält die Feldnamen des Datensatzes für den Datentransfer.

- `FileTransferComplete(FileNumber, Upload, ErrorCode)`

  Die Parameter sind:

`FileNumber`	Integer. Das verwendete Work File.
`Upload`	Boolean. Wird gesetzt, wenn das Hochladen beendet ist.
`ErrorCode`	Integer. Wird auf Null gesetzt, wenn der Vorgang ohne Fehler bearbeitet wurde.

- `FileTransferProgress(ProgressMessage)`

  Der Parameter ist:

`ProgressMessage`	String. Nachricht, die normalerweise im Ausgabefenster der Terminal-Anwendung angezeigt wird.

Tasks und Prozedurdateien

Task oder Prozedurdatei ausführen

- Rufen Sie Folgendes auf:

```
APIReturn = RunEntConTask(TaskName)
```

Der Parameter ist:

`TaskName`	String. Der Name eines unter Entire Connection definierten Task oder einer Prozedurdatei.

Anmerkung: Bei einer synchronen Verbindung kehrt die Programmierschnittstelle erst dann zur aufrufenden Anwendung zurück, nachdem der `TaskName` geprüft und der Task oder die Prozedurdatei gestartet wurde (nicht erst wenn der Task oder die Prozedurdatei beendet wird). Bei einem asynchronen Aufruf kehrt die Programmierschnittstelle sofort zur aufrufenden Anwendung zurück.

Auf die globalen Parameter `+PARM0` bis `+PARM9` zugreifen

- Rufen Sie Folgendes auf:

```
APIReturn = SetGlobalParameter(ParamNumber, Value) APIReturn = ↩
GetGlobalParameter(ParamNumber, Value)
```

Die Parameter sind:

`ParamNumber`	Integer. Zwischen 0 und 9 für den erforderlichen Parameter.
`Value`	String. Wert des Parameters.

Prozedurdatei abbrechen (nur synchroner Aufruf)

- Rufen Sie Folgendes auf:

```
APIReturn = CancelRunningTask()
```

Die Programmierschnittstelle kehrt sofort zur aufrufenden Anwendung zurück. Die Prozedurdatei sendet beim Abbruch das Ereignis `EntConTaskComplete`.

Hiermit verbundene Ereignisse:

- `EntConTaskStarting(ErrorCode, TaskName)`

  Wird aufgerufen, wenn ein Task gestartet wird, der nicht explizit von der Programmierschnittstelle aufgerufen wurde (z.B. ein Task zum Anmelden).

  Die Parameter sind:

`ErrorCode`	Integer. Muss auf 0 (Null) gesetzt werden, damit der Task starten kann.
`TaskName`	String. Name des gestarteten Task.

- `EntConTaskComplete(ErrorCode, TaskName)`

  Die Parameter sind:

`ErrorCode`	Integer. Ist auf Null gesetzt, wenn der Task fehlerfrei ausgeführt wurde.
`TaskName`	String. Name des Task.

- `TaskInputRequest(ErrorCode, DisplayOne DisplayTwo, Flags, ReturnData)`

  Dieses Ereignis wird gesendet, wenn der Befehl `INPUT` in einer Prozedurdatei ausgeführt wird.

  Die Parameter sind:

`ErrorCode`	Integer. Wird auf Null gesetzt, wenn die Eingabe erfolgt ist.	
`DisplayOne`	String. Erste Zeile der Eingabeaufforderung.	
`DisplayTwo`	String. Zweite Zeile der Eingabeaufforderung.	
`Flags`	Variant Array. Flags für die Anzeige (siehe unten).	
	`Flags(0)`	Muss Daten zurückgeben; leer ist nicht erlaubt.
	`Flags(1)`	Nur numerische Daten.
	`Flags(2)`	Passwortfeld.
	`Flags(3)`	Maximale Länge der angeforderten Daten.
`ReturnData`	String. Daten, die an die Prozedurdatei übergeben werden sollen.	

- `TaskDisplayMessageRequest(ErrorCode, Text, DialogBox, MessageType, Response)`

  Dieses Ereignis wird gesendet, wenn der Befehl `WAIT` in einer Prozedurdatei ausgeführt wird.

  Die Parameter sind:

`ErrorCode`	Integer. Wird auf 0 (Null) gesetzt, wenn die Prozedur weiterlaufen soll. Wenn Null nicht gesetzt ist, wird die Prozedur abgebrochen.
`Text`	String. Anzuzeigende Nachricht.
`DialogBox`	Boolean. "true", wenn ein Dialogfeld mit einer Nachricht erwartet wird.
`MessageType`	Variant. Anzeigeparameter.
`Response`	Integer. Standard-Microsoft-Antwort-Code der MessageBox (z.B. "IDOK"), wenn `DialogBox` "true" ist.

- `TaskError(ErrorCode, ErrorText)`

  Die Parameter sind:

`ErrorCode`	Integer. Vom Task ausgegebener Fehler-Code.
`ErrorText`	String. Anzuzeigende Nachricht.

Session beenden

Geöffnete Session beenden, aber die Verbindung mit Entire Connection bestehen lassen

- Rufen Sie Folgendes auf:

```
APIReturn = CloseSession()
```

Alle Terminals schließen (nur asynchroner Aufruf)

- Rufen Sie Folgendes auf:

```
APIReturn = CloseAllSessions()
```

Hiermit wird jede Terminal-Session beendet, einschließlich der Terminal-Sessions, die direkt geöffnet wurden. Dieser Aufruf sollte mit Vorsicht benutzt werden. Er bricht auch die Verbindung mit dem Terminal ab. Es wird kein Ereignis gesendet, das bestätigt, dass der Befehl abgearbeitet wurde.

Verbindung mit dem Terminal beenden (nur synchroner Aufruf)

- Rufen Sie Folgendes auf:

```
APIReturn = BreakConnection(Closedown)
```

Der Parameter ist:

`Closedown`	Boolean. Ist auf "true" gesetzt, um das Terminal-Fenster beim Abbrechen der Verbindung zu schließen.

Wenn `Closedown` auf "false" gesetzt wird und das Terminal nicht angemeldet ist, wird das Terminal trotzdem geschlossen. Wenn das Terminal unsichtbar war, wird es beim Abbrechen der Verbindung automatisch angezeigt.

Hiermit verbundene Ereignisse:

- `CurrentSessionClosed`

  Die Session wurde beendet, aber nicht durch einen API-Aufruf. Dies kann passieren, wenn das Terminal interaktiv benutzt wird und der Benutzer die Session schließt oder wenn eine Zeitüberschreitung auftritt.

- `TerminalClosedown`

  Das Terminal wurde geschlossen, aber nicht durch einen API-Aufruf. Dies kann im interaktiven Modus passieren, wenn der Benutzer die Anwendung schließt oder wenn `CloseAllSessions` von einer anderen API-Session aufgerufen wird.

Andere Methoden

Aktuelle Größe des geöffneten Terminals ermitteln

- Rufen Sie Folgendes auf:

```
APIReturn = GetScreenSize(NumberOfRows, NumberOfColumns)
```

3 Andere Ereignisse, Tasten-Codes und Fehler-/Return-Codes

Andere Ereignisse

- `ServerRequestedFileName(ErrorCode, OpenFile, Flags, Title, DefExtension, Filter, InitFileName, InitDirectory, FileName)`

  Wird aufgerufen, wenn die Session einen Dateinamen benötigt.

  Die Parameter sind:

`ErrorCode`	Integer. Wird auf Null gesetzt, wenn der Dateiname angegeben wurde.
`FileName`	String. Zu verwendender Dateiname.

  Die anderen Parameter sind die, die im Standarddateiauswahldialog angegeben werden können.

- `TerminalWarningMessage(Message, DisplayFlag)`

  Die Parameter sind:

`Message`	String. Anzuzeigende Nachricht.
`DisplayFlag`	Boolean. Es wird erwartet, dass der Aufruf eine Nachricht in einem blockierenden Dialogfeld anzeigt (z.B. mit der Funktion MessageBox).

Tasten-Codes

Die folgende Tabelle enthält die Tasten-Codes, die mit `PutData` übergeben werden können. Die erste Spalte enthält den Namen der Funktionstaste. In der zweiten Spalte steht die Konstante für die Funktionstaste, so wie sie in der Include-Datei *ECAPI.H* definiert ist, und die dritte Spalte enthält den Wert des Tasten-Codes für die Funktionstaste. Nur diese Tasten-Codes sollten verwendet werden. Wenn andere Werte übergeben werden, kann dies unvorhergesehene Folgen haben.

Die Include-Datei *ECAPI.H* ist als Bestandteil der Beispiele auf dem Installationsmedium von Entire Connection enthalten.

Funktionstaste	Definition des Tasten-Codes	Wert des Tasten-Codes
PF1	EC_PF1	20
PF2	EC_PF2	21
PF3	EC_PF3	22
PF4	EC_PF4	23
PF5	EC_PF5	24
PF6	EC_PF6	25

Funktionstaste	Definition des Tasten-Codes	Wert des Tasten-Codes
PF7	EC_PF7	26
PF8	EC_PF8	27
PF9	EC_PF9	28
PF10	EC_PF10	29
PF11	EC_PF11	30
PF12	EC_PF12	31
PF13	EC_PF13	32
PF14	EC_PF14	33
PF15	EC_PF15	34
PF16	EC_PF16	35
PF17	EC_PF17	36
PF18	EC_PF18	37
PF19	EC_PF19	38
PF20	EC_PF20	39
PF21	EC_PF21	40
PF22	EC_PF22	41
PF23	EC_PF23	42
PF24	EC_PF24	43
ATTN	EC_ATTN	46
CLEAR	EC_CLEAR	16
CR	EC_CR	13
DEVCNCL	EC_DEVCNCL	50
EEOF	EC_EEOF	54
ERASEINP	EC_ERASEINP	44
INSERT	EC_INSERT	82
NEWLINE	EC_NEWLINE	48
PRINT	EC_PRINT	49
SYSREQ	EC_SYSREQ	47
HOME	EC_HOME	71
PA1	EC_PA1	17
PA2	EC_PA2	18
PA3	EC_PA3	19
DELETE	EC_DELETE	83
BACKSPACE	EC_BACKSPACE	8
TAB	EC_TAB	9
BACKTAB	EC_BACKTAB	15

Funktionstaste	Definition des Tasten-Codes	Wert des Tasten-Codes
LEFT	EC_LEFT	75
RIGHT	EC_RIGHT	77
UP	EC_UP	72
DOWN	EC_DOWN	80
DUE2	EC_DUE2	56
EM	EC__EM	84
AFZ	EC_AFZ	11
EFZ	EC_EFZ	165
LZE	EC_LZE	89
RU	EC_RU	163
SDZ	EC_SDZ	160
SZA	EC_SZA	85
K1	EC_K1	193
K2	EC_K2	194
K3	EC_K3	195

Fehler-/Return-Codes

Alle Fehler-/Return-Codes sind Integer-Werte. Die unten aufgelisteten Konstanten sind in der Include-Datei *ECAPI.H* definiert. Die Zahlen in Klammern sind die eigentlichen Code-Werte.

Die Include-Datei *ECAPI.H* ist als Bestandteil der Beispiele auf dem Installationsmedium von Entire Connection enthalten.

`API_SUCCESS (0)`
: Die meisten API-Funktionen geben `API_SUCCESS` zurück, wenn die Funktion erfolgreich war. Einige API-Funktionen haben einen spezifischen Return-Code im Erfolgsfall - siehe die 3 folgenden Funktionen.

`API_CALL_QUEUED (-1)`
: Dieser Return-Code wird im asynchronen (nicht-blockierenden) Modus benutzt. Er bedeutet, dass die Anforderung von der API-Anwendung erfolgreich an Entire Connection zur Bearbeitung übergeben wurde. Der eigentliche Return-Code für die Anforderung wird dann von Entire Connection in einem Ereignis (completion event) an die API-Anwendung gesendet.

`API_NEW_SESSION_OPENED (-2)`
: Return-Code der API-Funktion `Initialize`, wenn eine neue Host-Session erfolgreich geöffnet wurde.

`API_PROC_CANCELLED_OK (-3)`
: Return-Code im Ereignis (completion event) für die API-Funktion `CancelRunningTask`, wenn die Prozedurdatei oder der Task erfolgreich abgebrochen wurde.

`API_ERROR_CALL_BLOCKED (1)`
: Dieser Return-Code wird intern benutzt. Er wird nicht an die API-Anwendung zurückgereicht.

`API_ERROR_INCORRECT_PARAMETERS (2)`
: Alle API-Funktionen prüfen, ob die übergebenen Parameter gültig sind. Dieser Fehler-Code wird zurückgegeben, wenn die Parameter nicht gültig sind.

`API_ERROR_NO_USER (10)`
: Dieser Fehler-Code wird von API-Funktionen zurückgegeben, die eine Host-Session benötigen, aber noch kein Benutzer in Entire Connection angemeldet ist. Bevor eine Host-Session geöffnet werden kann, muss ein Benutzer in Entire Connection angemeldet sein. Benutzen Sie die API-Funktion `LogonEntireConnection` zur Anmeldung.

`API_ERROR_NO_OPEN_SESSION (11)`
: Dieser Fehler-Code wird von API-Funktionen zurückgegeben, die auf einer Host-Session arbeiten, wenn noch keine Host-Session geöffnet wurde. Sie müssen erst eine Host-Session öffnen. Hierzu stehen die API-Funktionen `GetAvailableSessions` und `OpenSession` zur Verfügung.

`API_ERROR_NO_FILE_TRANSFER (12)`
: Fehler-Code der API-Funktion `CancelFileTransfer`, wenn es keinen aktiven Datentransfer gibt.

`API_ERROR_NO_SESSIONS_DEFINED (13)`
: Fehler-Code der API-Funktion `GetAvailableSessions`, wenn für den angemeldeten Benutzer keine Sessions in der Share-Datei von Entire Connection definiert sind.

`API_ERROR_NO_SCREEN_PRESENT (14)`
: Fehler-Code der API-Funktion `GetScreenText`, wenn keine Bildschirmdaten vorhanden sind. Dies kann beim Öffnen der Session passieren, wenn der Host den ersten Bildschirm noch nicht geschickt hat.

`API_ERROR_NO_SESSION_NAME (15)`
: Fehler-Code der API-Funktion `OpenSession`, wenn kein Session-Name im Parameter `SessionName` übergeben wurde.

`API_ERROR_NO_TASK_RUNNING (16)`
: Fehler-Code der API-Funktion `CancelRunningTask`, wenn kein aktiver Task oder keine aktive Prozedur vorhanden ist.

`API_ERROR_NOT_CONNECTED (20)`
: Dieser Fehler-Code wird von den API-Funktionen zurückgegeben, wenn das API-ActiveX-Control nicht mit Entire Connection verbunden ist. Zum Beispiel, weil Entire Connection vom Benutzer beendet wurde.

`API_ERROR_ALREADY_CONNECTED (21)`
: Fehler-Code der API-Funktion `Initialize`, wenn die Funktion bereits vorher aufgerufen und erfolgreich durchgeführt wurde.

`API_ERROR_ALREADY_LOGGED_ON (22)`
Fehler-Code der API-Funktion `LogonEntireConnection`, wenn der Benutzer bereits in Entire Connection angemeldet ist.

`API_ERROR_ALREADY_INITIALIZED (23)`
Fehler-Code der API-Funktion `Initialize`, wenn das API-ActiveX-Control bereits mit Entire Connection verbunden ist.

`API_ERROR_SESSION_ALREADY_OPEN (24)`
Fehler-Code der API-Funktion `OpenSession`, wenn bereits eine Host-Session geöffnet ist.

`API_ERROR_SESSION_NOT_FOUND (30)`
Dieser Fehler-Code wird zur Zeit nicht benutzt.

`API_ERROR_API_CALL_ONLY (31)`
Dieser Fehler-Code wird in Entire Connection benutzt, wenn API-Funktionen aufgerufen werden, es aber keine aktive API-Anwendung gibt.

`API_ERROR_INITIALIZATION_FAILED (40)`
Fehler-Code der API-Funktion `Initialize`, wenn das API-ActiveX-Control nicht initialisiert oder nicht mit Entire Connection verbunden werden konnte.

`API_ERROR_CALL_FAILED (41)`
Dieser Fehler-Code wird von den API-Funktionen zurückgegeben, wenn Entire Connection die Anforderung nicht erfolgreich durchführen konnte und keinen spezifischen Fehler-Code zurückgegeben hat.

`API_ERROR_COMMS_ERROR (200)`
Dieser Fehler-Code wird zur Zeit nicht benutzt.

`API_ERROR_INTERNAL_ERROR (201)`
Dieser Fehler-Code wird von den API-Funktionen zurückgegeben, wenn ein unerwarteter Fehler oder Ausnahmefehler aufgetreten ist. Die Anforderung wurde abgebrochen. Entire Connection ist vermutlich instabil. Starten Sie Entire Connection neu und versuchen Sie es noch einmal.