Overview of API Calls

This section provides an overview of all available API calls, grouped according to the following functional areas:

- Initialization

  - `GetRunningTerminalSessions`

  - `Initialize`

  - `LogonEntireConnection`

- Opening a Session

  - `GetAvailableSessions`

  - `OpenSession`

- General Control

  - `RunHostCommand`

  - `PutData`

  - `SetDataNotificationFlag`

- Screen Data

  - `GetScreenText`

  - `GetScreenRawText`

  - `GetScreenAttributes`

  - `GetExtendedAttributes`

  - `GetCursorPosition`

  - `SetCursorPosition`

  - `ClearScreenText`

  - `CheckForScreenText`

- Data Transfer

  - `SetAPIFileDetails`

  - `SetWorkFileDetails`

  - `GetFileName`

- ○ `CancelFileTransfer`

- Tasks and Procedure Files

    - ○ `RunEntConTask`

    - ○ `SetGlobalParameter`

    - ○ `GetGlobalParameter`

    - ○ `CancelRunningTask`

- Closing a Session

    - ○ `CloseSession`

    - ○ `CloseAllSessions`

    - ○ `BreakConnection`

- Other Methods

    - ○ `GetScreenSize`

See the descriptions below for detailed information on these API calls (including associated events).

Initialization

When starting a session, the API client can either attach to a running terminal or create a new terminal.

▶ **To find out the session names of any running terminals (synchronous call only)**

- Call the following:

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

This returns an array of currently running terminals that can be attached.
`GetRunningTerminalSessions` is the only call that can be made before calling
`Initialize`.

▶ **To attach to a terminal**

- Call the following:

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

The parameters are:

`CreateSession`	Boolean. "true" indicates that a new terminal is to be created.
`LinkSessionName`	String. The name of an existing terminal to attach to. The name is one of the terminal names that is returned by the `GetRunningTerminalSessions` function.
`UserLoggedOn`	Boolean. Returns "true" if the logon to Entire Connection has already taken place on the workstation. In order to use a terminal, a user has to log on once per workstation. If `UserLoggedOn` is "false", the API client has to log on now.
`OpenSession`	String. Normally empty. In a special case, this contains the name of an open session.

If "true" was returned for `CreateSession` or if it is not possible to attach to the specified terminal, the API control creates a new session.

If the connection to an existing terminal has been established and if in the meantime a session has been opened in this terminal, the `OpenSession` parameter contains the name of the session. In this special case, the API client has to decide whether it wants to work with this session which has not been opened under its control. This can only happen if an existing terminal is attached that is currently in the process of opening a session, and this process takes a while and has not yet been completed.

▶ To log on to Entire Connection

- Call the following:

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

Opening a Session

The API client can either query the available session names from the share file or open a known session directly.

▶ To query all sessions defined for the Entire Connection user

- Call the following:

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

The parameters are:

`SessionNames`	Variant Array(Strings). The names of all defined sessions.
`DefaultSession`	String. The name of the default session.

▶ To open one of these sessions

- Call the following:

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

The parameter is:

SessionName	String. The name of the session that is to be opened.

The session is now open and can be used.

Associated Events:

- FirstScreenArrived

  Fired when the session receives the first data from the host.

- ScreenSizeChanged(NumRow, NumColumns)

  Notifies the initial screen size, and also whether the terminal changes dynamically during a session.

- SessionOpened(SessionName)

  Fired if a session opens without the API client calling the OpenSession method. This may happen, for example, when a startup task is used. The parameter is:

SessionName	String. The name of the open session.

General Control

▶ **To send commands to the open session**

- Call the following:

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

The parameter is:

CommandName	String. The name of the command that is to be executed on the host.

The string is sent to the host and then to the function key ENTER.

▶ **To send general text and key codes**

- Call the following:

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

The parameters are:

Text	String. The text that is to be transferred to the host.
KeyCode	Integer. The key that is to be sent after the text has been transferred.

The text that is sent with this command can contain line feeds. These are interpreted as if the function key NEWLINE has been pressed. If you only want to send a key code, you have to pass an empty string for the text.

▶ To enable data notifications (synchronous call only)

- Call the following:

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

The parameter is:

`Enable`	Boolean. When you set this to "true", data notifications are switched on. Default: off.

▶ To show and hide the terminal window

- Set the API control property `TerminalInteractive` (boolean).

  If you connect to a terminal, it stays visible until this value is set to "false".

  If you create a new terminal, it is invisible until this value is set to "true".

Associated Events:

- `CursorPositionChanged(XPosition, YPosition)`

  Fired when the terminal is in interactive mode and the cursor position is changed with the mouse (not when the cursor moves due to typing).

- `NewScreenDataArrived()`

  If enabled, this indicates that new data has arrived from the host.

Screen Data

Screen text is available as the raw text as it is received by the host and as the processed text as it is displayed on the terminal. The raw text contains all characters - including those that are not to be displayed (e.g. password) - and can contain zero values.

Since the raw text can contain zero values, it can only be returned as an array of unsigned characters. The screen text is returned as an array of strings.

▶ To return screen text

- Call the following:

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

The parameters are:

`ScreenTextArray`	Variant Array(Strings). One string per line of text requested.
`TopLeftX`	Integer. Starting coordinate.
`TopLeftY`	Integer. Starting coordinate.
`BottomRightX`	Integer. Ending coordinate.
`BottomRightY`	Integer. Ending coordinate.

If any of the coordinates is set to -1, the entire screen is returned.

▶ To return raw data

- Call the following:

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

  The parameter is:

`ScreenTextArray`	Variant Array(Unsigned chars). Raw data buffer.

▶ To return screen attributes

- Call the following:

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

  The parameters are:

`Attributes`	Variant Array(Unsigned chars). Attribute buffer.
`AttributesDescription`	Variant Array(Unsigned chars). The description of an attribute is an array of 6 values containing the bit patterns for the attribute properties: Member 0: Attribute Member 1: Protected Member 2: Numeric Member 3: No display Member 4: High display Member 5: Modify data tag

▶ To return extended screen attributes

- Call the following:

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

The parameter is:

`ExtendedAttributes`	Variant Array(Unsigned chars). Extended attribute buffer.

▶ To read and set the current cursor position

● Call the following:

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

The parameters are:

`XPosition`	Integer. X indicates the cursor position in the column.
`YPosition`	Integer. Y indicates the cursor position in the line.

▶ To remove all editable text in the specified area

● Call the following:

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

The parameters are:

`TopLeftX`	Integer. Starting coordinate.
`TopLeftY`	Integer. Starting coordinate.
`BottomRightX`	Integer. Ending coordinate.
`BottomRightY`	Integer. Ending coordinate.

-1 in any value indicates the whole screen.

▶ To call the IF command used to check for screen text

● Call the following:

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

The parameters are:

Text	String. Text to check for.
Result	Boolean. "true" if the text was found.
Position	Integer. Screen position where the text was found.
TopLeftX	Integer. Starting coordinate.
TopLeftY	Integer. Starting coordinate.
Length	Integer. Text length.
CaseSensitive	Boolean. True if case-sensitive check.

Data Transfer

▶ **To prepare for data transfer to be processed directly by the API client**

- Call the following:

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

The parameters are:

WorkFileNumber	Integer. Work file number.
UploadFlag	Boolean. Is set for upload.
BinaryFlag	Boolean. Is set for binary transfer.
ReportFlag	Boolean. Is set for report format.

This results in the following events being fired during upload:

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

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

and the following events being fired during download:

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

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

The event parameters are:

ErrorCode	Integer. Must be set to 0 by the API to indicate that all was processed without error.
FileNumber	Integer. The work file to be processed.
DataLength	Integer. Upload: the expected size is passed; the actual size is returned. Download: is set to the size of the transmitted data.
Data	Variant Arry(unsigned char). Data that are to be transferred.
DataFormat	String. Description of the record format.

For a normal transfer operation, the API client has to to provide a file name. This can be done by presetting a file name.

▶ To preset a file name

- Call the following:

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

  The parameters are:

Name	String. The file name that is to be used.
FileNumber	Integer. The work file being processed.
Upload	Boolean. Is set for upload.
Binary	Boolean. Is set for binary transfer.
Report	Boolean. Is set for report format.

If no preset values are found for the work file being processed, the API client will be asked for a file name.

▶ To return a file name

- Respond to the following event:

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

  The parameters are:

ErrorCode	Integer. If set to zero, the file name is used and processing starts. If set to any other value, processing is canceled.
FileNumber	Integer. Work file being processed.
Upload	Boolean. Is set for uploading a file name.
Binary	Boolean. Is set for binary transfer.
ToPrinter	Boolean. Is set to download to a printer.
Landscape	Boolean. Is set to print in landscape format.
ControlChars	Boolean. Is set to interpret control characters.
FileName	String. The file name to be used.

▶ To cancel a running data transfer

- Call the following:

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

The parameter is:

FileNumber	Integer. The number of the work file for which the data transfer is to be canceled.

This call is synchronous. It queues a cancelation request. When data transfer has completed, the FileTransferComplete event is fired.

Associated Events:

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

The parameters are:

ErrorCode	Integer. If set to zero, the file name is used and processing starts. If set to any other value, processing is canceled.
FileNumber	Integer. The work file being processed.
Upload	Boolean. Is set for uploading a file name.
Binary	Boolean. Is set for binary transfer.
Headings	Variant Array (Strings). Contains the field names of the record for the transfer.

- FileTransferComplete(FileNumber, Upload, ErrorCode)

The parameters are:

FileNumber	Integer. Work file being processed.
Upload	Boolean. Is set if upload is completed.
ErrorCode	Integer. Is set to zero if the data transfer was processed without error.

- `FileTransferProgress(ProgressMessage)`

The parameter is:

ProgressMessage	String. Message that normally appears in the output window of the terminal application window.

Tasks and Procedure Files

▶ **To run an Entire Connection task or procedure file**

- Call the following:

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

The parameter is:

TaskName	String. The name of an Entire Connection task or procedure file.

Note:
For a synchronous connection, the application programming interface returns to the calling application after the `TaskName` has been checked and the task or procedure file has been started (not when the task or procedure file is completed). For an asynchronous call, the application programming interface immediately returns to the calling application.

▶ **To access the global parameters `+PARM0` to `+PARM9`**

- Call the following:

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

The parameters are:

ParamNumber	Integer. From 0 to 9 for the required parameter.
Value	String. Value of the parameter.

▶ **To cancel a running procedure file (synchronous call only)**

- Call the following:

```
APIReturn = CancelRunningTask()
```

This will return immediately. The procedure file will notify termination by firing the `EntConTaskComplete` event.

Associated Events:

- `EntConTaskStarting(ErrorCode, TaskName)`

Is called when a task is started other than explicitly by the application programming interface (e.g. a logon task).

The parameters are:

ErrorCode	Integer. Has to be set to 0 (zero) so that the task can be started.
TaskName	String. The name of the task that has been started.

- `EntConTaskComplete(ErrorCode, TaskName)`

The parameters are:

ErrorCode	Integer. Is set to zero if the task has completed without error.
TaskName	String. Task name.

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

This event is fired if an `INPUT` statement is executed in a procedure file.

The parameters are:

ErrorCode	Integer. Is set to zero after input has been provided.
DisplayOne	String. First line of prompt text.
DisplayTwo	String. Second line of prompt text.
Flags	Variant Array. Display flags (see below). Flags(0) Must return some data; blank is invalid. Flags(1) Numeric data only. Flags(2) Password field. Flags(3) Maximum length of the requested data.
ReturnData	String. Data that are to be returned to the procedure.

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

This event is fired if a WAIT statement is executed in a procedure file.

The parameters are:

ErrorCode	Integer. Is set to 0 (zero) if the procedure is to continue. If zero is not set, the procedure is canceled.
Text	String. The message to be displayed.
DialogBox	Boolean. "true" if a message box is expected.
MessageType	Variant. Display parameters.
Response	Integer. Standard Microsoft response code of the MessageBox (e.g. "IDOK") if DialogBox is "true".

● TaskError(ErrorCode, ErrorText)

The parameters are:

ErrorCode	Integer. Error code from the task.
ErrorText	String. Message to be displayed.

Closing a Session

▶ **To close an open session and leave the connection to Entire Connection active**

● Call the following:

```
APIReturn = CloseSession()
```

▶ **To close all terminals (asynchronous call only)**

● Call the following:

```
APIReturn = CloseAllSessions()
```

This will close any terminal session on the workstation, including those opened directly. This call should be used with caution. It also breaks the connection to the terminal. There is no completion event.

▶ **To break the link to the terminal (synchronous call only)**

● Call the following:

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

The parameter is:

Closedown	Boolean. Is set to "true" to close the terminal window on disconnect.

If `Closedown` is set to "false" and the Entire Connection terminal is not logged on, the terminal will be closed anyway. If the terminal was hidden, it will be automatically shown when the connection is broken.

Associated Events:

- `CurrentSessionClosed`

  The session has closed down without a request from the application programming interface. This can happen if the terminal is interactive and the user closes the session, or if a session times out.

- `TerminalClosedown`

  The terminal has completely closed down with no request from the application programming interface. This can happen in interactive mode if the user closes the application, or if `CloseAllSessions` is called from another API session.

Other Methods

▶ **To return the current size of the open terminal**

- Call the following:

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