

Overview

The 2101 AP component (TLSAPX2) is a COM/ActiveX control providing a high level API for controlling and getting information about the shows defined in the system.

The AP component is usable from many programming languages. The specific steps involved in using this component will depend on your choice of programming language. Separate documents describe access methods for Microsoft Visual C++, Microsoft Visual Basic, and Borland C++ Builder.

This document describes the API and the sequence of operations.

Installing the Control

The 2101 AP component is contained in a single DLL file named TLSAPX2.DLL. You need to include this DLL as part of your product installer. You may install this file either to your application directory or to the Windows system directory. After installation you must register the component. Most install packages have built-in facilities to register components. You may also register the control manually using the following command:

```
c:\> RegSvr32 <path>\TLSAPX2.DLL
```

Operation Sequence

1. Call *OpenControl* once to initialize the control. You must call *CloseControl* when you are done using the control.
2. Call *Connect* to establish a connection to an AP server. You must call *Disconnect* to close the connection when done.
3. Call *EnumerateShows* to get a list of shows in the system. You must create an event handler for *EnumerateShowsCallback* in order to receive the results. The *EnumerateShowsCallback* will be fired once for each show in the system (whether active or not).
4. Call *ConnectToShow* to establish a connection to an active show. If a password is required you must provide it when making the call.
5. Once connected to a show you may call *ChangeShow* to change the show running on the host to another show. Call *EnumerateShowsForHost* to determine which shows are allowed to run on this host. If the new show requires a password you must provide it when calling *ChangeShow*.
6. Once we connected to a show you can call *EnumerateDirectors* to get information about the desktop directors attached to this show.
7. At this point you can either attach to an existing (and free) desktop director by calling *AttachToDirector* or you can attach stand-alone in talent mode by calling *AttachAsTalent*.
8. At this point you may call any of the *Post...* functions listed below. You can also receive events from the control (as long as you provide handlers for them).

Commands

Once connected to a show you may issue any of the commands listed below.

```
PostBusyAllButton(int nSession);
PostDelayDumpButton(int nSession, bool bDown);
PostDialKey(int nSession, int chDialKey);
PostDropButton(int nSession, int nColumn);
PostHoldButton(int nSession, int nColumn);
PostLineButton(int nSession, int nLine, int nColumn);
PostMuteButton(int nSession);
PostNextButton(int nSession);
PostRecordButton(int nSession);
PostSpeakerButton(int nSession);
PostText(int nSession, int nTextType, int nLine, const char *pszText);
PostTransferButton(int nSession);
PostUpdateAll(int nSession);
```

Events

The following events are used by the control to send information back to the application:

```
EnumerateShowsCallback(BSTR strShowName,
                      BOOL bIsActive,
                      BSTR strHostName,
                      BOOL bIsLast);

EnumerateHostsCallback(BSTR strHostName,BOOL bIsLast);

EnumerateDirectorsCallback(int nSession,
                          BSTR strDirectorName,
                          BOOL bIsTalent,
                          BOOL bIsFree,
                          BOOL bIsLast);

SendShowInfo(int nSession,int nMaxLines,int nHybrids);

SendMode(int nSession, BOOL bTalent);

SendShowChange(int nSession,BSTR strNewShowName,BSTR strOldShowName);

SendRecord(int nSession,BOOL bOn);

SendBusyAll(int nSession,BOOL bOn);

SendTransfer(int nSession,BOOL bOn);

SendMute(int nSession,BOOL bOn);

SendRingMute(int nSession,BOOL bOn);

SendDelayDump(int nSession,BOOL bOn);

SendHandset(int nSession,BOOL bOn);

SendSpeaker(int nSession,BOOL bOn);

SendLineStyle(int nSession,
              int nLine,
              int nColumn,
              int nLineStyle,
              long lElapsedTimeSec);

SendSeizeLine(int nSession,int nLine,int nColumn);

SendRingLine(int nSession,BSTR strCallerId,int nLine);

SendDropLine(int nSession,BSTR strCallerId,int nLine);

SendAnswerLine(int nSession,BSTR strCallerId,int nLine,int nColumn);

SendSelectLine(int nSession,BSTR strCallerId,int nLine,int nColumn);

SendDialKey(int nSession,int chKey);

SendError(int nSession,int nErrorCode,BSTR strErrorText);

SendText(int nSession,int nTextType,BSTR strText,int nLine);
```

AP Line States

The nLineState parameter in SendLineState can have one of the following values:

LINE_STATE_INVALID	=0x0000,
LINE_STATE_RESERVED	=0x0100,
LINE_STATE_IDLE	=0x0200,
LINE_STATE_RINGING	=0x0301,
LINE_STATE_HOLD	=0x0402,
LINE_STATE_READY_HOLD	=0x0504,
LINE_STATE_READY_NEXT	=0x0604,
LINE_STATE_TALENT_HOLD	=0x0702,
LINE_STATE_ON_AIR	=0x0808,
LINE_STATE_ON_AIR_LOCKED	=0x0908,
LINE_STATE_HANDSET	=0x0A10,
LINE_STATE_HANDSET_LOCKED	=0x0B10,
LINE_STATE_SPEAKER	=0x0C10,
LINE_STATE_SPEAKER_LOCKED	=0x0D10,
LINE_STATE_USED_ELSEWHERE	=0x0E20,
LINE_STATE_BUSY_ALL	=0x0F20,
LINE_STATE_SHARED_UNAVAILABLE	=0x1020

AP Error Codes

The following error codes may be sent by the SendError event:

Error Code	Error Message
1	Connect: Success!
2	EnumerateShows: Success!
3	EnumerateHostsForShow: Success!
4	ChangeShow: Success!
5	ConnectToShow: Success!
6	EnumerateDirectors: Success!
7	AttachToDirector: Success!
8	AttachAsTalent: Success!
9	PostBusyAllButton: Success!
10	PostDelayDumpButton: Success!
11	PostDialKey: Success!
12	PostDropButton: Success!
13	PostHoldButton: Success!
14	PostLineButton: Success!
15	PostMuteButton: Success!
16	PostNextButton: Success!
17	PostRecordButton: Success!
18	PostSpeakerButton: Success!
19	PostText: Success!
20	PostTransferButton: Success!
21	PostUpdateAll: Success!
22	EnumerateShowsForHost: Success!
1001	Connect: No storage manager found!
1002	Connect: Unable to read user info from storage manager!
1003	Connect: Invalid password!
2001	EnumerateShows: No storage manager found!
3001	EnumerateHostsForShow: No storage manager found!
4001	ChangeShow: Not connected to a show!
4002	ChangeShow: Failed!
5001	ConnectToShow: Unable to find active show!
5002	ConnectToShow: Invalid show password!
6001	EnumerateDirectors: Not connected to a show!
7001	AttachToDirector: Not connected to a show!
7002	AttachToDirector: Attach failed!
8001	AttachAsTalent: Not connected to a show!
8002	AttachAsTalent: Attach failed!
9001	PostBusyAllButton: Not attached to show!
10001	PostDelayDumpButton: Not attached to show!
11001	PostDialKey: Not attached to show!
12001	PostDropButton: Not attached to show!
13001	PostHoldButton: Not attached to show!
14001	PostLineButton: Not attached to show!
15001	PostMuteButton: Not attached to show!
16001	PostNextButton: Not attached to show!
17001	PostRecordButton: Not attached to show!
18001	PostSpeakerButton: Not attached to show!
19001	PostText: Not attached to show!
20001	PostTransferButton: Not attached to show!
21001	PostUpdateAll: Not attached to show!
22001	EnumerateShowsForHost: No storage manager found!
99001	Bad session id!

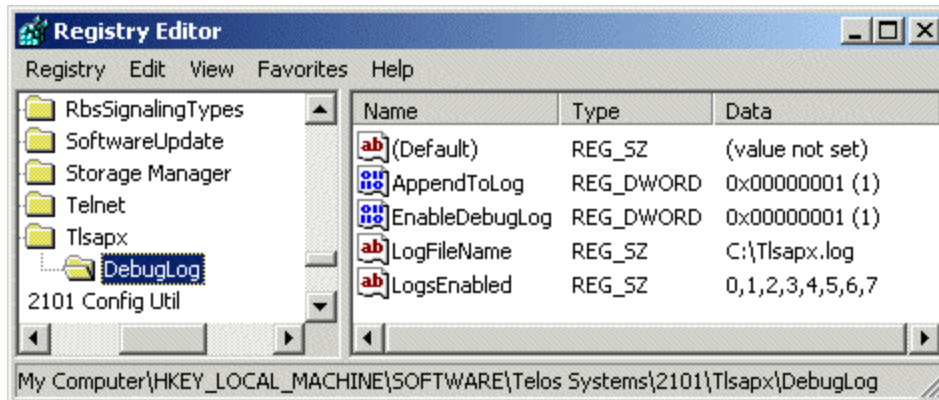
Multiple Session Support

The AP component supports multiple sessions through a single connection. This means that you may connect to multiple shows and/or desktop directors through a single connection. The session parameter works as a “channel” identifier allowing you to specify a particular show connection.

For example, suppose you want to control two shows in talent mode. After calling *OpenControl* and *Connect* you call *ConnectToShow* and specify zero as session identifier for the first show. You then call *ConnectToShow* again and specify one as the session identifier for the second show. Subsequently, any commands you issue using zero as the session identifier will only affect the first show while a session identifier of one will affect the second show. All events received from the first show will be marked with a session identifier set to zero and events received from the second show will be marked with a session identifier set to one.

Debug Output

The TLSAPX component can send debug output to a file. This may be useful during development of debugging. Debug output is enabled through the following registry options:



Set *EnableDebugLog* to 1 to enable debug output or to zero to disable it. The *LogFileName* specifies the name of the log file where the debug output is written while *AppendToLog* controls if the file is overwritten or appended to each time the control is opened. The *LogsEnabled* allows you to enable or disable groups of messages. The default string of “0,1,2,3,4,5,6,7” enables all available debug messages.

Operations – Alphabetical Order

AttachAsTalent

Attach to a show in Talent mode.

Declaration

```
AttachAsTalent(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method attaches to a show in Talent mode. You must already be connected to a show before using *AttachAsTalent*.

AttachToDirector

Attach to a director.

Declaration

```
AttachToDirector(int nSession, BSTR strDirectorName)
```

Parameters

nSession Session identifier.

strDirectorName Name of director as returned by *EnumerateDirectors()*.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method attaches to a desktop director. You can then control the desktop director from software. User actions on the director are reflected back as notification events.

CloseControl

Close control.

Declaration

```
CloseControl()
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Call this method to close the control when no longer needed.

Connect

Connect to 2101 server.

Declaration

```
Connect(int nSession,  
        BSTR strHostPort,  
        BSTR strUser,  
        BSTR strPassword  
        )
```

Parameters

<i>nSession</i>	Session identifier.
<i>strHostPort</i>	Host and port in this format: "host:port". If a port is not specified a default value of 9998 is assumed.
<i>strUser</i>	User name registered on the specified server.
<i>strPassword</i>	Password.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method opens a connection to a 2101 server using the specified user name and password. The user name and password must already be registered on the 2101 server.

ConnectToShow

Connect to an existing show.

Declaration

```
ConnectToShow(int nSession,  
              BSTR strShowName,  
              BSTR strShowPassword,  
              BSTR strHostName  
              )
```

Parameters

<i>nSession</i>	User specified session identifier.
<i>strShowName</i>	Show name
<i>strShowPassword</i>	Show password
<i>strHostName</i>	Host name

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method connects to a show. The session identifier specified will be used by all subsequent calls to refer to this particular session.

ChangeShow

Change the show.

Declaration

```
ChangeShow(int nSession,  
           BSTR strNewShowName,  
           BSTR strNewShowPassword  
           )
```

Parameters

nSession User specified session identifier.
strNewShowName New show name.
strNewShowPassword New show password.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method creates a show on the specified host. Use EnumHostsForShow() to determine all hosts where a particular show may be created. The session identifier specified will be used by all subsequent calls to refer to this particular session.

Disconnect

Close connection to 2101 server.

Declaration

```
Disconnect()
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method closes the connection to the 2101 server.

EnumerateDirectors

Enumerate directors.

Declaration

```
EnumerateDirectors(int nSession)
```

Parameters

nSession Session identifier as specified in the call to `ConnectToShow()`.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method enumerates the directors attached to the show we're connected to. This method can only be called after a `CreateShow()` or `ConnectToShow()` returned success.

EnumerateDirectorsCallback

Director event.

Declaration

```
void EnumerateDirectorsCallback(int nSession,  
                                const char *pszDirectorName,  
                                bool bIsTalent,  
                                bool bIsFree,  
                                bool bIsLast  
                                )
```

Parameters

nSession Session identifier as specified in the call to `ConnectToShow()`.

pszDirectorName Name of director.

bIsTalent true if in talent mode.

bIsFree true if not controlled by attached software.

bIsLast true if this is the last director.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This event is fired in response to a call to `EnumerateDirectors()`.

EnumerateHostsForShow

Request a host list for a show.

Declaration

```
EnumerateHostsForShow(BSTR strShowName)
```

Parameters

strShowName Show name

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method requests list of hosts where the specified show is allowed to run. If the show is not defined in the system this will return a list of all active hosts. The host callback event will be invoked for each host.

EnumerateHostsCallback

Host event.

Declaration

```
void EnumerateHostsCallback(char *pszHostName,  
                           bool bIsLast  
                           )
```

Parameters

pszHostName Name of the host.

bIsLast true if this is the last host.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This event is fired in response to a call to EnumerateHostsForShow().

EnumerateShows

Request a show list.

Declaration

```
EnumerateShows ( )
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method requests a show list known to 2101. The show callback event will be invoked for each show.

EnumerateShowsCallback

Show event.

Declaration

```
void EnumerateShowsCallback(const char *pszShowName,  
                           bool bRequiresPass,  
                           bool bIsActive,  
                           const char *pszHostName,  
                           bool bIsLast  
                           )
```

Parameters

pszShowName Name of the show.

bRequiresPass true if show requires a password.

bIsActive true if show is active.

pszHostName Name of the host where show is running.

bIsLast true if this is the last show.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This event is fired in response to a call to *EnumerateShows()*.

EnumerateShowsForHost

Request a list of show that may run on the specified host.

Declaration

```
EnumerateShowsForHost(BSTR strHostName)
```

Parameters

strHostName Host name

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method requests list of shows allowed to run on the specified host. If the host name is not defined in the system this will return a list of all shows. The host callback event will be invoked for each host.

EnumerateShowsForHostCallback

Host event.

Declaration

```
void EnumerateShowsForHostCallback(char *pszShowName,  
                                   bool bIsLast  
                                   )
```

Parameters

pszShowName Name of the show.
bIsLast true if this is the last show.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This event is fired in response to a call to EnumerateShowsForHost().

OpenControl

Open control.

Declaration

```
OpenControl()
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Call this method first before using any other methods.

PostBusyAllButton

Post desktop director key press.

Declaration

```
PostBusyAllButton(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostDelayDumpButton

Post desktop director key press.

Declaration

```
PostDelayDumpButton(int nSession, BOOL bDown)
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostDialKey

Post desktop director key press.

Declaration

```
PostDialKey(int nSession, int chKey)
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostDropButton

Post desktop director key press.

Declaration

```
PostDropButton(int nSession, int nColumn)
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostHoldButton

Post desktop director key press.

Declaration

```
PostHoldButton(int nSession, int nColumn)
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostLineButton

Post desktop director key press.

Declaration

```
PostLineButton(int nSession, int nLine, int nColumn)
```

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostMuteButton

Post desktop director key press.

Declaration

```
PostMuteButton(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostNextButton

Post desktop director key press.

Declaration

```
PostNextButton(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostRecordButton

Post desktop director key press.

Declaration

```
PostRecordButton(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostSpeakerButton

Post desktop director key press.

Declaration

```
PostSpeakerButton(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostText

Send a text string.

Declaration

```
PostText(int nSession,  
         int nTextType,  
         int nLine,  
         BSTR strText  
        )
```

Parameters

nTextType Application-defined text identifier.

nLine Line number associated with the text.

strText Text string.

Return Value

S_OK on success, E_FAIL on error.

Remarks

This method sends a text string to the 2101 server. The server then sends the string to all devices attached to the same show as us. The string is ignored by desktop directors but it is used by software programs to pass specific information about caller name, address, etc..

PostTransferButton

Post desktop director key press.

Declaration

```
PostTransferButton(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Calling this method has the same effect as pressing the corresponding key on the desktop director.

PostUpdateAll

Request a status update.

Declaration

```
PostUpdateAll(int nSession)
```

Parameters

nSession Session identifier.

Return Value

S_OK on success, E_FAIL on error.

Remarks

Invoke this method to request a status update from the server. All status information will be sent back including line states, button states, and etc. through the appropriate callback functions.