Errata

This document describes the use of the Telos 2101 AP component from the Borland C++ Builder environment. The example uses version 1 or the AP component. The general approach remains the same for version 2 but the name of the control and some function prototypes are changed.

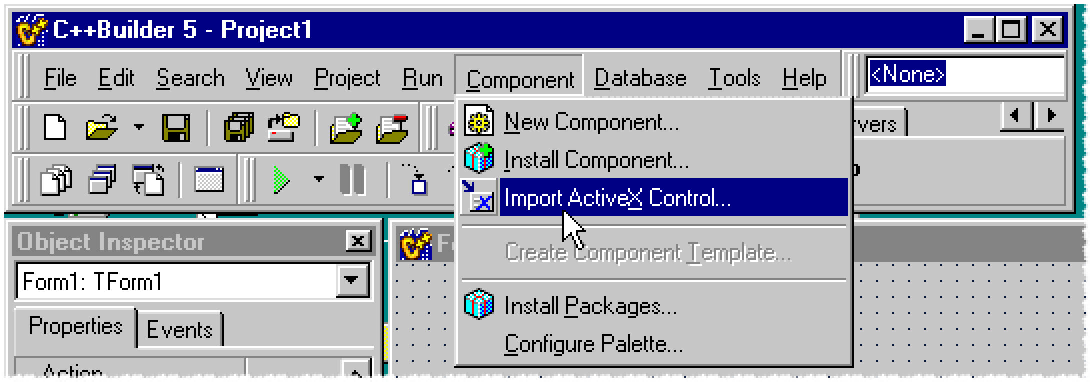
Telos 2101 ActiveX Control

The Telos 2101 ActiveX (TLSAPX) control exposes the application programming interface (API) for System 2101. This document describes how to use this control from Borland C++ Builder 5.0.

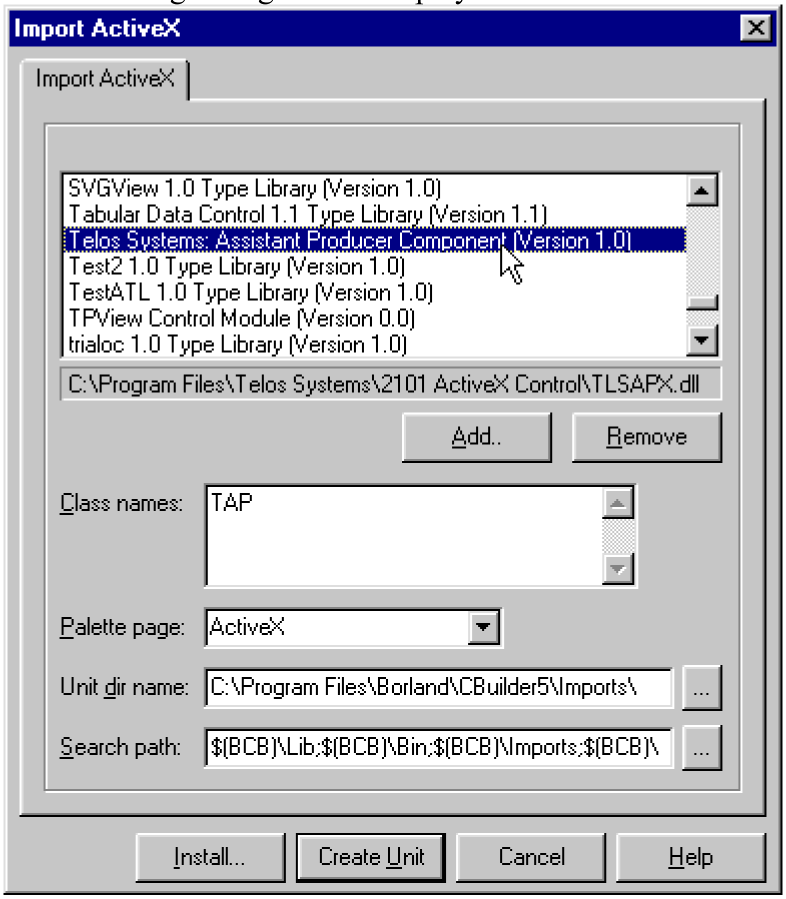
Our example uses a simple dialog-based application to exercise the API functions provided by the TLSAPX control. The example illustrates the basic approach to using the control but it does not cover the entire API. Although you will have to adapt this example to suit your specific needs, the basic steps should remain the same.

Using the TLSAPX from Borland C++ Builder

STEP 1: Start C++ Builder. From the "Component" menu select "Import ActiveX Control…"



The following dialog will be displayed:



SVGView 1.0 Type Library (Version 1.0)
Tabular Data Control 1.1 Type Library (Version 1.1)
Telos Systems: Assistant Producer Component (Version 1.0)
Test2 1.0 Type Library (Version 1.0)
TestATL 1.0 Type Library (Version 1.0)
TPView Control Module (Version 0.0)
trialoc 1.0 Type Library (Version 1.0)

C:\Program Files\Telos Systems\2101 ActiveX Control\TLSAPX.dll

Class names: TAP

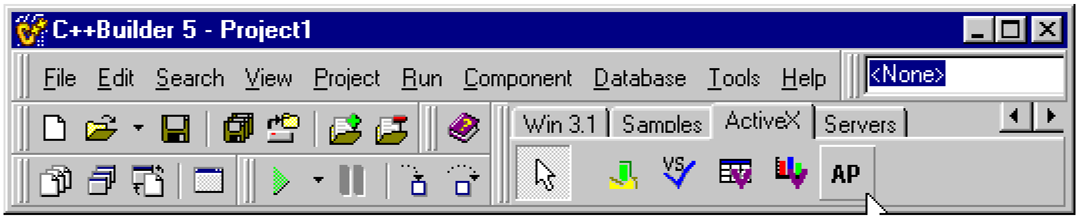
Palette page: ActiveX

Unit dir name: C:\Program Files\Borland\CBuilder5\Imports\

Search path: $(BCB)\Lib;$(BCB)\Bin;$(BCB)\Imports;$(BCB)\

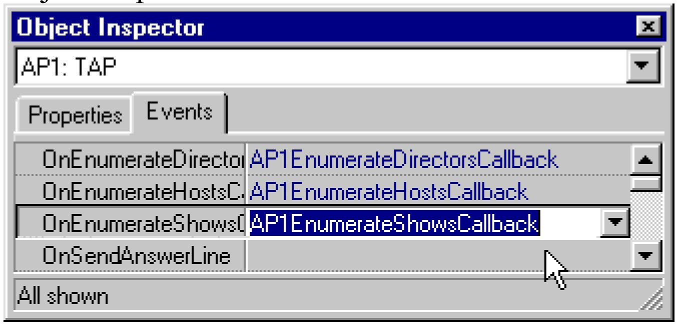
Scroll the list until you find “Telos Systems: Assistant Producer Component”. Select it then click the “Create Unit” button.

STEP 2: After this go back to the form dialog. Select the AP control from the “ActiveX” tab:



Click anywhere on your form to place the control.

STEP 3: Make sure that the AP control is selected and click the “Events” tab of the Object Inspector:



AP1: TAP

OnEnumerateDirector	AP1EnumerateDirectorsCallback
OnEnumerateHostsC	AP1EnumerateHostsCallback
OnEnumerateShows(	AP1EnumerateShowsCallback
OnSendAnswerLine	

Enter an event handler for each event listed.

STEP 4: Add the appropriate code for each handler. The code below shows the event handler for OnSendError:

```
void __fastcall TForm1::AP1SendError(TObject *Sender, long nErrorNumber,
      BSTR strErrorText)
{
  AnsiString aStr=strErrorText;

  MessageBox(0,aStr.c_str(),"SendErrorEvent",0);
}
```

Since all strings passed by the control are BSTR types we need to convert this to an AnsiString for display. This handler only displays all events received to the user. Although this is useful when developing and debugging your application you should have a more refined implementation. In particular error numbers under 1000 are not errors but notifications of completed operations.

Here is the event handler for OnEnumerateShows:

```
void __fastcall TForm1::AP1EnumerateShowsCallback(TObject *Sender,
      BSTR strShowName, long bRequiresPass, long bIsActive,
      BSTR strHostName, long bIsLast)
{
  AnsiString aStr=strShowName;

  MessageBox(0,aStr.c_str(),"EnumerateShowsEvent",0);
}
```

Please note that this example handler only displays the information received to the screen. Your application may want to store the information received to display the available shows in a friendlier manner.

STEP 5: Go back to the form and add OnCreate and OnDestroy handlers. Use this code for the OnActivate handler:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  HRESULT nRet=AP1->ControlInterface->OpenControl();
  if(nRet<0)
    MessageBox(0,"Unable to open control","Error",0);
}
```

This handler will initialize the control when the form is created. We're going to close the control when the form is destroyed. Use this code for the OnDestroy handler:

```
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
  HRESULT nRet=AP1->ControlInterface->CloseControl();
  if(nRet<0)
    MessageBox(0,"Unable to close control","Error",0);
}
```

STEP 6: Add a "Connect" button to the form. Add an OnClick handler for this button. In the handler use the following code:

```
void __fastcall TForm1::ConnectClick(TObject *Sender)
{
  WideString wstrHost="192.168.123.5";
  WideString wstrUser="ioan";
  WideString wstrPass="pass";

  HRESULT nRet=AP1->ControlInterface->Connect(wstrHost,wstrUser,wstrPass);
  AnsiString Msg;
  Msg.printf("Connect=%d!",nRet);
  MessageBox(0,Msg.c_str(),"Result",0);
}
```

You should replace the connect parameters (host, user name, password) with values that are valid for the 2101 system you are using.

Note that we're using the WideString data type since the control expects the BSTR type. WideString has a convenient BSTR conversion operator that makes it easy to use directly. Also note that we're using the ControlInterface property which exposes the COM interface for the ActiveX control. This allows us to get the HRESULT return code for the function call.

Add another button for "Enumerate shows" with the following handler:

```
void __fastcall TForm1::EnumerateShowsClick(TObject *Sender)
{
  HRESULT nRet=AP1->ControlInterface->EnumerateShows();
  AnsiString Msg;
  Msg.printf("EnumShows=%d!",nRet);
  MessageBox(0,Msg.c_str(),"Result",0);
}
```

Both ConnectClick and EnumerateShowsClick display the result returned even on success. Note that the HRESULT return type is a 32-bit data type.

The following table lists the values of common HRESULT values. More values are contained in the header file winerror.h.

Name	**Description**	**Value**
S_OK	Operation successful	0x00000000
E_UNEXPECTED	Unexpected failure	0x8000FFFF

E_NOTIMPL	Not implemented	0x80004001
E_OUTOFMEMORY	Failed to allocate necessary memory	0x8007000E
E_INVALIDARG	One or more arguments are invalid	0x80070057
E_NOINTERFACE	No such interface supported	0x80004002
E_POINTER	Invalid pointer	0x80004003
E_HANDLE	Invalid handle	0x80070006
E_ABORT	Operation aborted	0x80004004
E_FAIL	Unspecified failure	0x80004005
E_ACCESSDENIED	General access denied error	0x80070005

STEP 7: We're now ready to put it all together. Compile the project and run it. Click the "Connect" button on the dialog. You should receive a "Connect=0" on success. If the operation fails you will receive a negative result code. If connect fails then check to make sure the 2101 system is up and running. Also check the host name, user name and password to make sure they are valid.

Once connect succeeds click the "Enumerate Shows" button. You should receive one message for each show defined in the system. The message box will indicate if the show is active and if so the name of the host where it is running.

You should also notice that the SendError handler receives events from the control. Even on success the SendError handler receives messages to indicate success. Success codes are less than 1000 while error codes are greater than 1000.

Summary

This example application showed how to use the 2101 ActiveX control from the Borland C++ Builder environment. The specific details are for C++ Builder 5.0 but the same ideas can be applied to other C++ tools.

Although this example does not cover the entire 2101 API it illustrates the basic techniques for communicating with the ActiveX control. The example can be easily extended to cover all functionality needed by your application.