Dave's Development Blog
Software Development using Borland / Codegear / Embarcadero RAD Studio

Finding Open Tools API Interfaces

By David | August 11, 2011 0 Comment

I've seen recently while looking for OTA information many people asking how to get a particular interface from the IDE. What follows is a process that I follow to try and find these interfaces and could be used as a rule of thumb but is not always correct.

Services Interfaces

Most of the `IOTAXxxxxServices` interfaces are exposed through casting the `BorlandIDEServices` global variable as follows:

```
Procedure Something;

Var
  CompileServices : IOTACompileServices;

Begin
  CompileServices := (BorlandIDEServices As IOTACompileServices);
  CompileServices.DisableBackgroundCompilation;
End;
```

There is one known exception to this and that is the `IOTASplashScreenServices` interface. This is exposed by another global variable `SplashScreenServices`. The reasons for this other variable is that the `BorlandIDEServices` variable at the point in time when the splash screen is being displayed is not set up and available.

Finding Interfaces

This method I use to find interfaces is quite simple. For example, lets take the `IOTAEditView` interface. I'll explain in a minute why I was looking for this interface as its an exception to the rule of thumb I'm describing here. Anyway, what I need to to find another interfaces that has a property or method that returns this interface. You can do this via a number of methods. You can use a Find/Search method in the

IDE to searching the ToolsAPI.pas file. My preferred method is to use a key-stroke exposed by GExperts (Ctrl+Alt+Up or Ctrl+Alt+Down) on the interface name and these keystrokes will move you to the previous or next instance of this interface in the source code.

The **IOTAEditView** interface is exposed by the following interfaces and methods / properties:

- **IOTASourceEditor70.GetEditView** a getter method for the property **IOTASourceEditor70.EditViews**;
- **IOTAEditBuffer60.GetTopView** a getter method for the property **IOTAEditBuffer60.TopView**;
- **IOTAEditorServices60.GetTopView** a getter method for the property **IOTAEditorServices60.TopView**.

The above give us 3 paths to this interface. The last one is completely resolved such that we can get the interface with the following code:

```
Procedure Something;

Var
  EditView : IOTAEditView;
  CP : TOTAEditPos;

Begin
  EditView := (BorlandIDEServices As IOTAEditorServices);
  CP := EditView.CursorPos;
  OutputDebugString(PChar(Format('Line %d, Column %d', [CP.Line, CP.Col])));
End;
```

For the other 2 in the list above we would need to repeat the exercise of finding the interface by looking for interfaces and their method that return the secondary interface. So for instance with the first item in the list above we would have to look for methods and property that return a **IOTASourceEditor** interface.

The reason that I don't look for the interface with the number on the end is that in each release of Delphi the interfaces that are implemented by **BorlandIDEServices** implements the highest interface without the number which in turn implements these previous IDE versions of the interfaces.

The above could be resolved in the following manner:

```
Procedure Something;

Var
  CM : IOTAModule;
  i : Integer;
  SourceEditor : IOTASourceEditor;

Begin
  CM := (BorlandIDEServices as IOTAModuleServices).CurrentModule;
```

```
  For i := 0 To CM.ModuleFielCount - 1 Do
    If ModuleFileEditors[i].QueryInterface(IOTASourceEditor, SourceEditor) = S_OK Then
      Begin
        EditView := SourceEditor.EditViews[0];
        CP := EditView.CursorPos;
        OutputDebugString(PChar(Format('Line %d, Column %d', [CP.Line, CP.Col])));
        Break;
      End;
End;
```

This one is a bit awkward as the **IOTAModule** interface property **ModuleFileEditors** only returns a **IOTAEditor** interface BUT is actually a **IOTASourceEditor** interface. To get the interface we must query the **IOTAEditor** interface to see if it does implement **IOTASourceEditor** which we can then use.

Now I come back to the reason for wanting the **IOTAEditView** interface is that this interface according to the comments in ToolsAPI.pas exposes the **IOTAElideServices** interface for folding and unfolding code but like the above example the interface is not exposed explicitly and must be obtained through a **QueryInterface** call as below:

```
EV := (BorlandIDEServices As IOTAEditorServices).TopView;
{$IFDEF D2006}
EV.QueryInterface(IOTAElideActions, EA);
{$ENDIF}
C.Col := iIdentCol;
C.Line := iIdentLine;
{$IFDEF D2006}
If EA <> Nil Then
  EA.UnElideNearestBlock;
{$ENDIF}
```

I hope this helps people gain better access to the Open Tools API.

Dave.

Category: Open Tools API Tags: Borland, BorlandIDEServices, CodeGear, Delphi, Embarcadero, Experts, OTA