What is a DispInterface?

Short answer: it is a specification for an *IDispatch* interface.

Long answer: The *IDispatch* interface is the basis of all automation. It has two methods that allow pointerless scripting languages to call methods by name, instead of using method pointers: *GetIDsOfNames*, and *Invoke*. *GetIdsOfNames* retrieves the numerical ID of a method with a given name (provided that the object implements another interface that has method with that name). *Invoke* uses the numerical ID of a method to call that method. The numerical ID of a method is called the *DispID*. For example, suppose you create an interface that looks like this:

```
type
  IMyIntf = Interface(IDispatch)
    ['{4D733280-C514-11D4-8481-A68F52CBDB56}']
    procedure DoThis;
  end;
```

and an object that implements it that looks like this:

```
type
  TMyDispatchObj = class(TAutoObject, IMyIntf)
  public
    procedure DoThis;
  end;
```

Delphi will call both *GetIDSOfNames* and *Invoke* for you – all you need to do is use a variant. Like this:

```
{ ... }
var
  AVar: OleVariant;
  { ... }

{ ... }
  AVar := CreateComObject(CLASS_TMyAutoObj) as IDispatch;
  AVar.DoThis;
  { ... }
```

Every time you call a method of an variant referencing an *IDispatch* interface, *GetIDsOfNames* and *Invoke* are called for you behind the scenes. However, calling *GetIdsOfNames* for every method is quite slow. And since all it does is find a numerical ID for a given method name, it might be nice if you could look up that ID in advance and pass it to *Invoke* directly, rather than go through *GetIdsOfNames* every time. Enter the *DispInterface*:

```
type
  IMyDispInterface = dispinterface
    ['{4D733284-C514-11D4-8481-A68F52CBDB56}']
    procedure DoThis; DispID 1;
  end;
```

This declaration tells Delphi the DispID of each method in an interface. So if you use a DispInterface variable, rather than a variant, Delphi can call *Invoke* directly using that, rather than go through *GetIdsOfNames*:

```
{ ... }
var
  Disp: IDispatch;
  Dispint: IMyDispInterface;
{ ... }
  Disp := CreateComObject(CLASS_TMyAutoObj) as IDispatch;
  Dispint := IMyDispInterface(Disp);
  Dispint.DoThis;
  { ... }
```

A DispInterface is not a true interface. When you use a DispInterface, you are actually using the *IDispatch* interface of an object, just as you are when you use a variant. That's why you can cast an *IDispatch* interface directly to a dispinterface, as in the second line above. All you're doing here is telling the compiler that you already know what other methods a particular object will implement, and what DispIDs can be used to invoke them with.

Original resource:	The Delphi Pool
Author:	Deborah Pate
Added:	2013-03-13
Last updated:	2013-03-13