July, 2004

Advisor Answers

Making IntelliSense available all the time

VFP 9/8/7

Q: I love the way IntelliSense shows me the properties and methods of an object in method code of forms and visual classes. Can I make it work in .PRG files, too?

Also, I know that it's good practice to use WITH … ENDWITH when I'm working with many PEMs of the same object, but IntelliSense doesn't work inside a WITH statement. Is there a way to make it work there?

A: IntelliSense was one of the best enhancements of VFP 7. The List Members feature that shows you the properties, events and methods of the object you're working with makes it much easier to get code right the first time.

In the Form Designer and Class Designer, List Members works automatically for objects that are included in the form or class you're creating. As soon as you type a period after the object reference, the list of PEMs appears. However, when you're working with other objects or creating code outside those designers, you have to give IntelliSense a hand.

The secret is to declare the variable that holds the object reference with the appropriate type. For example, if you include this line in a .PRG file:

```
LOCAL oForm AS FORM
```

you'll have access to the List Members feature for that form, as in Figure 1.

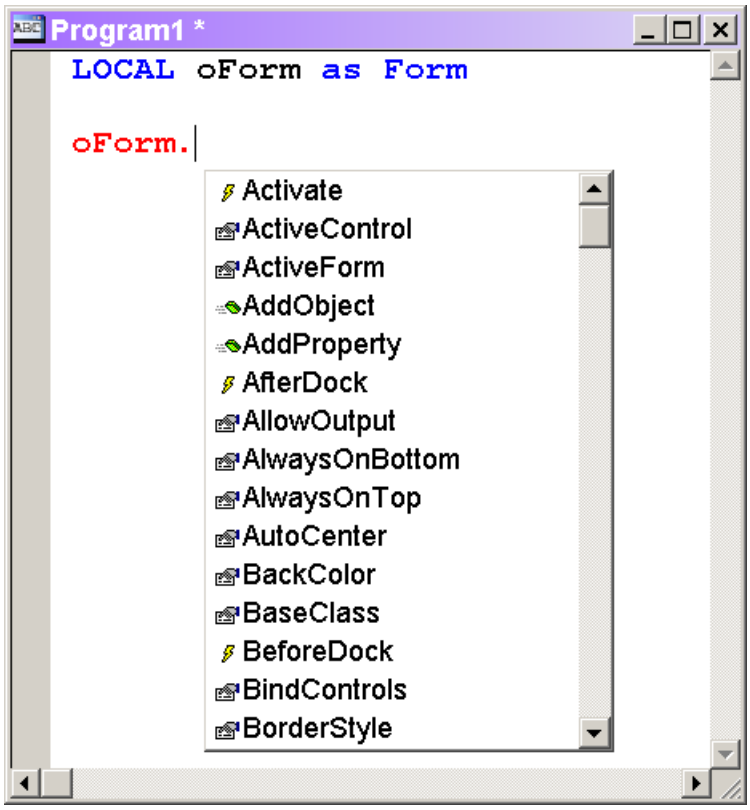

Figure 1. Making List Members available—When you declare a variable as a known object type, IntelliSense can help you write code.

Declaring the variable works for all kinds of classes, including the VFP base classes, classes you define in VFP, and Automation servers. However, for IntelliSense to work with classes other than the base classes, you have to provide more information.

For VFP classes, the secret is to declare not just the class name, but the class library, as well, using this syntax:

```
LOCAL oVariable AS cClassName OF cClassLib
```

For example, I have a class that lets me change the font and font size of a form and its controls on the fly. (See "Let Users Control Fonts" in the November, 2002 issue.) If I wanted to work with that class in code, I could declare the variable like this:

```
LOCAL oFontHandler AS cusfonthandler OF accessibility.vcx
```

Of course, I need to make sure to provide the full path to the class library. Once I've declared the variable this way, the List Members functionality is available, as in Figure 2.

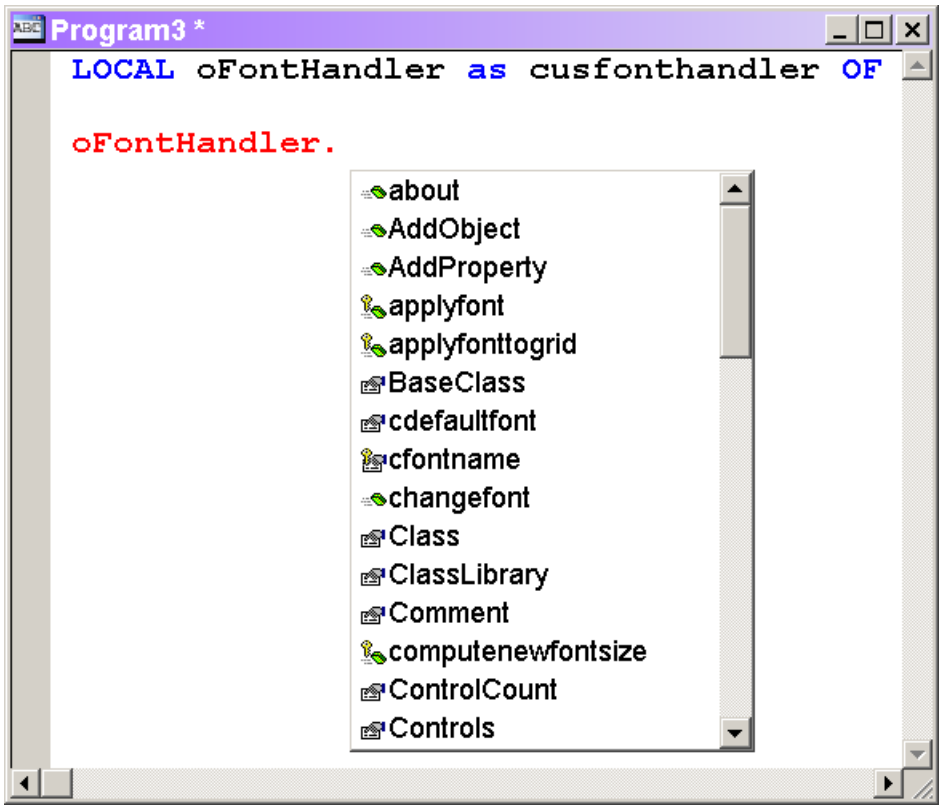

Figure 2. Listing members of a custom class—To use IntelliSense with custom classes, define the variable using the class name and class library.

A similar strategy works for Automation servers; in this case, you need to know the name of the server and the class. To provide IntelliSense for Word's Application object, I can define the variable like this:

```
LOCAL oWord AS Word.Application
```

and IntelliSense will kick in when I type oWord. With some Automation servers (including Word and PowerPoint), the same technique works for contained objects (like Word's Document and Table objects).

Some Automation servers (like Excel) make you work a little harder, though. For those servers, if you want List Members to work for contained objects, it's not sufficient just to declare the object variable. Before you use it, you also have to register the Automation server using the IntelliSense manager. Open this tool using Tools | IntelliSense Manager and click on the Types tab. Click the Type Libraries button to open the Type Library References dialog. (Figure 3 shows both the dialog and the IntelliSense Manager.)

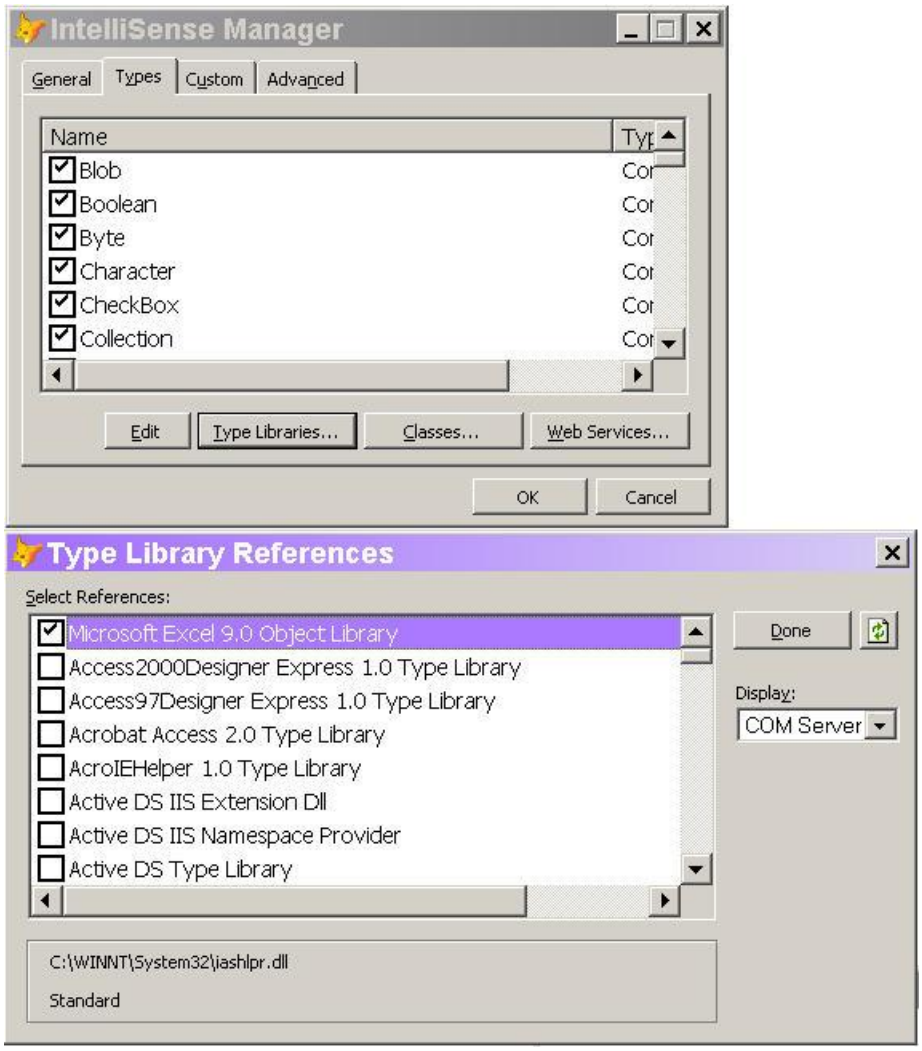

Figure 3. Registering Automation servers—When you register a type library with the IntelliSense Manager, all its objects are available and the various objects are listed when you declare variables.

Find each Automation server you want to register and check the checkbox. (In Figure 3, Excel is already registered.) Click Done to return to the IntelliSense Manager.

Registering type libraries has another benefit. The objects of the type library are available when you're declaring variables. Figure 4 shows an example.

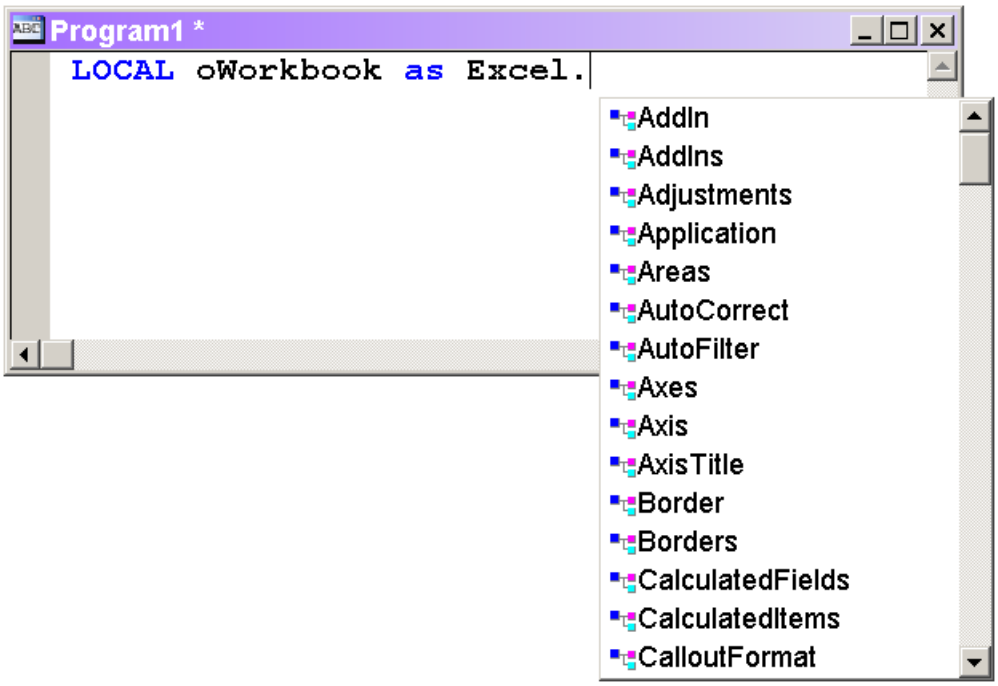

Figure 4. Benefits of registering—Once you've registered a type library, it's easy to declare variables of any of its classes.

As you mention, using WITH … ENDWITH in your code is generally considered to be good practice (though there are some situations where it's not a good idea). However, in VFP 8 and earlier, IntelliSense doesn't work for the WITH variable. That is, if you have something like:

```
WITH oForm
```

typing a period inside the WITH doesn't show you the PEMs of the oForm variable.

In VFP 9, WITH has been enhanced to support IntelliSense. However, as with other uses of IntelliSense, you have to give VFP a hint. You do so by adding an AS clause to the WITH statement:

```
WITH oForm AS Form
```

To use WITH for a custom class, add the OF clause (making sure to provide the path to the class library):

```
WITH oFontHandler as cusfonthandler OF accessibility.vcx
```

For Automation servers, use the server plus class name format:

```
WITH oDoc AS Word.Document
```

In VFP 8 and earlier, however, you have to use tricks to get IntelliSense inside a WITH statement. The easiest solution is to declare an extra variable of the appropriate type, use it while writing the code and then use Find-and-Replace to eliminate it. For example, you might do something like this:

```
LOCAL oForm AS Form
LOCAL oJunk AS Form

WITH oForm
   oJunk.Caption = "My Caption"
   oJunk.Width = 300
   …
ENDWITH
```

Then, replace "oJunk." with ".".

IntelliSense makes it much easier to write correct code the first time. It's worth spending some time learning about the various features, so you can make the most of this powerful tool.

–Tamar