Route Data To Oracle & More With File Plug-Ins

By David Wallwork

BBj® 2.0 includes a powerful new tool for developers: the BBj File System Plug-in Interface. What makes this tool so powerful is that the BBj developer can now define what happens when his or her program executes READs and WRITEs on a channel.

We developed the concept of the file system plug-in interface in response to a Customer request. i-flex solutions limited, a development arm spun off from Citicorp, is updating a large legacy application that stores data in BBx® MKEYED files, but the company had a new requirement that some of the data be stored in Oracle.

There was a need for data to be written to Oracle instead of to an MKEYED file and to have other data written to both Oracle and an MKEYED file. i-flex also needed to be able to easily change the destination of data as requirements evolved. And, of course, all this had to be done with an absolute minimum of change to the existing BBX code. Allowing application developers to directly specify what will happen in a program's READs and WRITEs with a file plug-in was the BASIS solution - one that can benefit all BBj developers.

The file plug-in itself is Java code that performs whatever action is desired in response to the various forms of READ and WRITE. Although this requires the developer to write some Java code, it gives the developer complete control over what READ and WRITE mean. Usually the amount of Java code that is required is quite small. But, of course, it can be as extensive as the developer chooses to make it.

Using File Plug-Ins

In order for a BBj program to use a file plug-in, a new MODE must be added within the OPEN statement for a channel. For example, the statement

```
OPEN (dataChannel ) "filename"
```

changes to

```
OPEN (dataChannel , MODE = "plugin=someCOOLplugin") "filename"
```

The BBj File System Plug-in Interface is actually a Java interface. Any class that implements com.basis.plugin.FilePlugin can be used within an existing BBj program by adding a MODE phrase to the OPEN statement of a channel. (A complete description of the methods can be found in our documentation under "OpenVerb - OpenFile BBj.") If a channel is opened with a MODE indicating a file plug-in, any subsequent call to READ or WRITE will simply pass the data on to the appropriate method of the file plug-in. How the file plug-in obtains data for a READ, or what the file plug-in does with the data for a WRITE is independent of BBj.

In addition to being able to route data between Oracle and MKEYED files, there are other plug-in uses that can be implemented with the BBj 2.0 release. The file plug-in **com.basis.plugin.HashPlugin** is a simple example that reads and writes from a HashMap. The methods that are not appropriate in this example (such as a write without a key) would simply throw exceptions, causing the BBj program to

enter its error-handling routine. The Microsoft Windows-based file plug-in **com.basis.plugin.RegPlugin** implements native C code through the Java Native Interface (JNI). This file plug-in can be used to manipulate the Windows registry, allowing the developer to read, write, create or delete registry keys.

We expect that BASIS Customers will find some very interesting ways to use file plug-ins. If you want to use BBj file plug-ins, you can read more about them in the BBj documentation under the chapter BBj Components. But even if you don't have an immediate use for them in your applications, they are proof of how BBj is growing, and how it continues to improve its ability to interact with the rest of the computing world. And when you need BBj file plug-ins, they are readily available.