Jump on the BUS – BASIS Update Service!

All Windows users have seen it... that annoying little tool tip in the system tray that says something like "An update is now available." Now, your application can be annoying too! Of course, I am being facetious, but an application feature that automatically looks for updates can be very desirable and often essential.

Along your development journey, jump on the BUS (BASIS Update Service) and travel light so end users can get a desired feature or an essential fix quickly, without waiting for a full release of your product.

Here Comes the BUS

The need for a BUS came about with the Barista® Application Framework, the BASIS RAD tool. Barista is both a development framework and a run-time engine for Barista-developed applications. So, the need is two-fold as both the developer and the end-user communities will benefit from the BUS. Barista developers wanted the ability to update their applications via a mechanism that would not require a lot of work or be too intrusive for the users. Previously, Barista developers had to wait for the next full release to get a bug fix or the developers had to manually create a .zip or .tar file with the new versions of the programs and files for the customers to manually install. Combining input from the Barista developers with an awareness of the flexibility of Web Services development, BASIS engineers agreed the perfect solution would be Web-based.

By Brian Hipple
Quality Assurance Supervisor

Get On Board

Typically, implementing Web Services takes a great deal of time and effort. Coding the BBj® service application is the easy part; the time consuming part is completing the overall configuration as well as writing and debugging the ancillary Java code. BBj 9.0 introduced an embedded Jetty Web Server that easily deploys BBj applications as Web Services without a great deal of configuration effort or the need to write Java code. This feature really does remove the pain and effort, allowing developers to set up a Web Service faster than they could ever imagine.

The BUS is a BBj CustomObject that utilizes the new Web Services functionality. The BASIS DBMS provides the back end of the BUS (that's where all the cool people sat when I was a kid!) using ESQL files and their support for foreign key relationships to provide an extra level of data integrity.

To take advantage of the public interface, the application passes information to the BUS such as vendor, name, revision, and current update level. The BUS then notifies the client application whether an update is available. If an update is pending, the service provides the application with the new update level, date, and a description about what the update includes. If the application wants the update, the BUS provides the FTP location of the update from which the client can download the JAR and install its contents. BASIS' design decision to deliver updates via an FTP server was predicated on two facts: 1) FTP servers are a proven and reliable way to provide files to users 2) FTP servers are accessible to any FTP client. This is especially important if the application has no access to the internet (yes it is true!), users can still download the update on another machine with internet access, copy the update to the target machine via 'sneaker net' or a USB drive, and apply the update.

To accept this automatic update, the developer must configure the application to allow updates, similar to the way Barista has, shown in **Figure 1** and **Figure 2**.

Any client with proper credentials can add or remove an update from the BUS. For convenience, the BASIS Installation Suite installs the BUS Administration >>

module along with the BUS Web Service and database structure. The BUS Admin module requires a user name and password for validation, allowing authenticated clients to add a patch. To add a patch to the BUS service, the client must provide such application information as vendor, name, revision, location. Given this information, the BUS can then create and return the new update level. To remove an update, the developer must specify a vendor, application name, revision, and update level as shown in **Figure 3.**

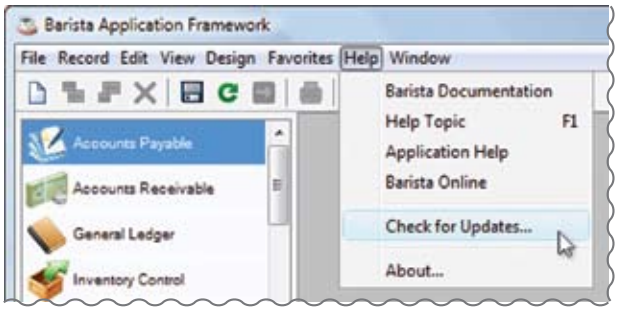

Figure 1. Barista's "Check for Updates" using the BUS

Since the BUS is implemented as a Web Service, the client can be a BBj, Java, Perl, C++ program, or any application that supports Web Services consumption. For example, the Barista development team chose a client Perl program to add updates. This Perl program runs on demand, checking files out from an update branch of the source code repository, and then adds the update via the BUS. A Barista developer only needs to check in the code to the appropriate SVN branch to be included in the next update. How easy is that?

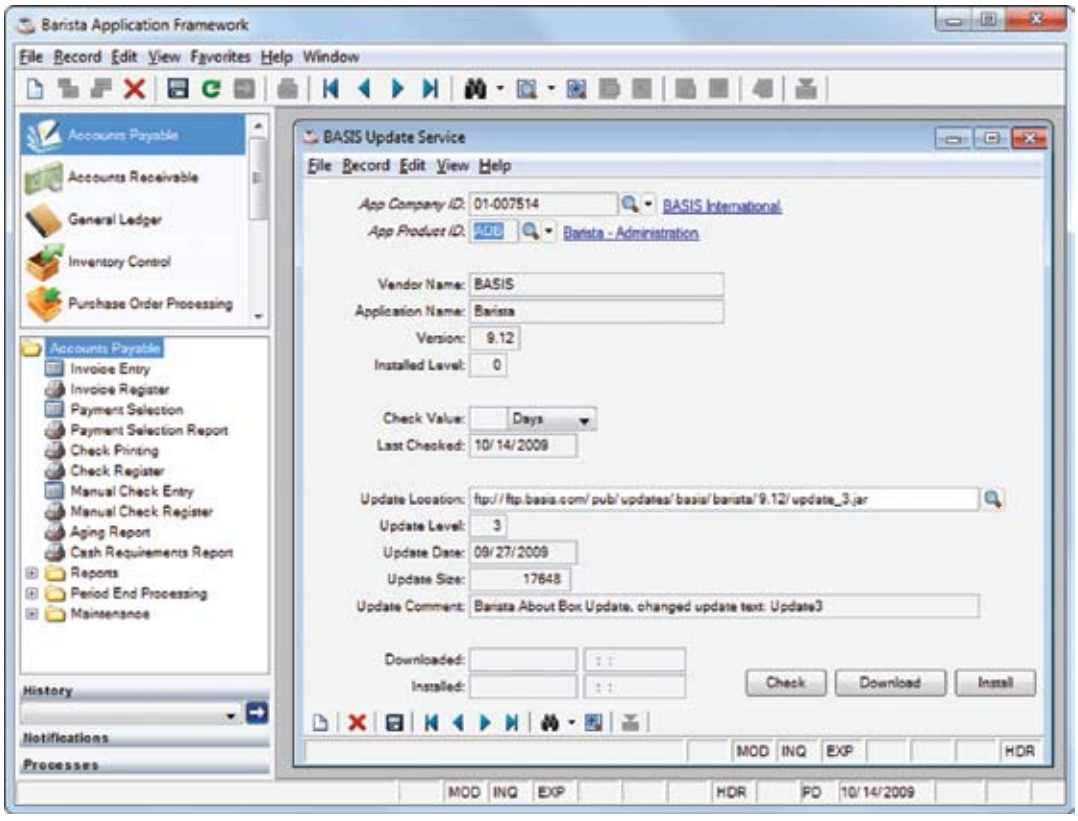

Figure 2. Barista using the BUS to provide updates

Recent Barista enhancements make it possible for end users to check for updates via the BUS and install any pending updates. Alternatively, Barista can easily install an update downloaded manually from an FTP server.

Summary

Empowering end users with the ability to update their applications with the BUS eliminates the need to manually distribute updates or create a new revision of the product. Whether adding a 'check for updates' prompt at program startup or invoking it manually, easily updating an application reduces maintenance costs and gives end users the confidence that they can get updates in a timely and convenient manner improving the quality and value of your product and service offering. The BUS Utility is a BBj 10.0 feature available today for preview in 9.x.

Isn't it time for you to offer your customers a ride on the BUS? ■

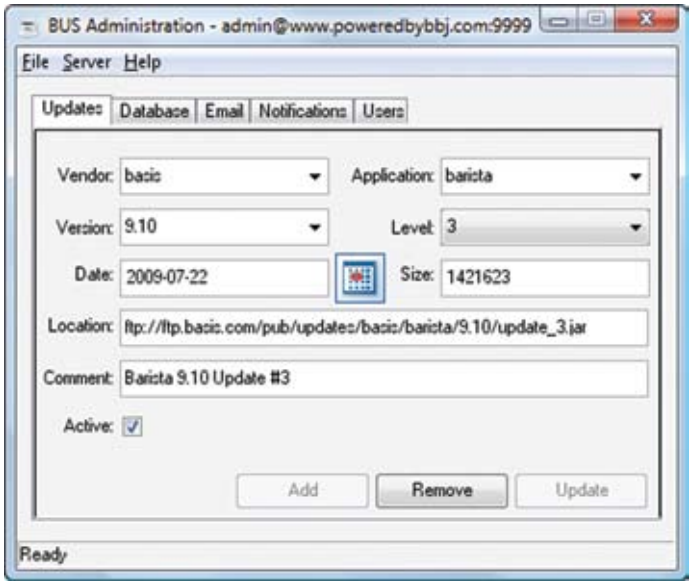

Figure 3. BUS Admin module showing details about an available update