

Brew MP™ - Practical Insights from Some Brew Die-hards

OpenPath Products

Christopher Axthelm, Caitlyn Paley, Rich Smith
February 25, 2011

OpenPath Products has had opportunity to work with Brew® since its initial launch in the U.S. over nine years ago with the Sharp Z-800 followed shortly by the ground-breaking Motorola T-720. 300 phones and 5,000 Brew ports later, we are still very glad to participate in the Brew ecosystem. As with most Brew die-hard developers, we were eager to embrace the Brew MP tools when they were formally released last year. To us, Brew MP demonstrated Qualcomm's commitment to the platform and provided new tools that we hoped would let us do more in less time.

As a service to the development community, our team felt it would be helpful to share some of our opinions and insights from having worked extensively with Brew MP over the past seven months. This paper addresses the topics that we felt would be most relevant for both Brew and non-Brew developers who are about to embrace Brew MP.

Backward Compatibility

The most common question we receive about Brew MP has to do with backward compatibility. Most content providers that already have Brew content are concerned that Brew MP will require a complete rewrite from their existing Brew code. Content providers considering new Brew MP development are similarly concerned that significant reworking will be required in order to support the millions of legacy Brew handsets already in the market.

Fortunately, with a little planning and with the support of the new Brew MP tools, it is not a problem for most applications to support Brew and Brew MP phones. It is our experience that, while some classes are deprecated for Brew MP phones, all of the older Brew v3.1.5 APIs are present and functional under the Brew MP phones that we have seen. Backward compatibility for Brew MP, therefore, is not as much of a problem as some may have feared. We scope the effort as akin to a challenging port rather than at the complexity of a full conversion.

Even so, please be aware that there are several notable deprecated classes that, while still usable, should be avoided. These classes include ISoundPlayer, ITAPI, IFile, ITextCtl, IAddrBook and IDBRecord. Even though they are available for backward compatibility, Brew MP defines alternatives that should be used for new development. In the case of ITextCtl, you are encouraged to use the BUIW TextWidget and StaticWidget, which demonstrates the prevalence of the BUIW style. The link "deprecated_replacements" in the API Introduction Section of the Brew MP API Reference guide provides a concise summary of the deprecated APIs.

Carefully crafted code written for Brew MP can also work well on many legacy Brew 3.x phones. Of course, if the application relies on Brew MP APIs that are not present in the older phones, then the application will not run. For

this reason, when writing new applications we try to determine the long-term distribution strategy for the application to ensure that it is not relying on APIs that are not present on all the target phones.

When writing new Brew MP applications, the question of how to write the user interface is often one of the first to be addressed. Most games use custom UIs through the graphics canvas and are portable between Brew and BREW MP. Brew MP implicitly encourages the use of its Brew UI Widgets (BUIW) by bundling the BUIW library. Since earlier Brew versions can also use BUIW as an extension, BUIW also provides a decent mechanism for creating backward-compatible user interfaces.

Most Brew veterans will agree that there is much more to the Brew ecosystem than just the technology. True Brew Testing (TBT) certification is a particularly important part of the process and affects backward compatibility. In order to pass TBT certification, both Brew and Brew MP applications must now define four application icons. The original Brew devices only supported three icons and the non-Brew MP SDK is only capable of generating MIF files with three icons. Since four icons must now be defined even for older phones, the Brew MP SDK must be used to create at least the MIF files for legacy devices. Failure to do so could result in TBT failure.

Also relevant to TBT requirements are application privileges that are defined within the Brew MIF file. The new Brew MP Resource Manager shows two types of privileges: one for “legacy” definitions and one for definitions using the new format. The “legacy” privileges are shown if you open the MIF file in an older (non Brew-MP SDK) MIF Editor. To ensure compatibility and consistency, we recommend filling out both areas with the same permissions using the Brew MP SDK Resource Manager. Note that permissions are now defined using bit-masks, which can be challenging to independently verify. To help validate the permissions, we created a simple web tool to decode the Brew permissions. This tool can be found at:

http://openpathproducts.com/OPP_Hex2Perms/

Finally, even if you are planning to only target non-Brew MP phones, it is worthwhile to use the Brew MP SDK for your development. The new SDK provides the ability to target different Brew versions and include a variety of tools, plug-ins, and utilities that we find help with the development process.

The remainder of this document describes some aspects of the Brew MP SDK that we find are most relevant to those embracing the tools.

Noteworthy Brew MP Tools

Brew MP SDK Manager

We like the new Brew MP SDK Manager as it serves as a one-stop shop for Brew tools and utilities. The manager provides a decent UI interface that not only shows your installed platforms, plug-ins, and tools, but also allows you to launch the Brew websites, download new platform versions, download new devices for the Simulator, and install (or uninstall) plug-ins. The tools contain...

- Loader and Logger,
- the new Resource Manager,

- the Simulator and its Target Manager,
- and the new AppCreator.

Loader and Logger

Unlike earlier versions, the new Loader and Logger tools do not support EFS version selections. This can be an issue with certain older phones whose EFS version is not detected properly. The Loader can connect to these phones but any file transfers will fail due to an EFS error. Another lesser change is that there are three options for your connection type. The naming conventions of the connection types are a bit confusing: two of the three are called “Brew MP Targets.” The “Brew MP Targets [USB]” option is for physical Brew MP phones that are in developer mode and simulation targets. The “Brew MP Targets [COM/BTIL]” can also load Brew MP phones and simulation targets using BTIL. The third option (“Brew Devices [COM/DIAG]”) is for non-Brew MP phones.

Be aware that the new Loader often will not create a folder for you when loading an application. In previous editions you could drag & drop mif, .mod, .bar, and SIG files into any white space while viewing the mod directory. A new folder would then be created with the name of the application and the MIF would be sent to the correct directory. While we have seen the drag and drop installation work occasionally, most of the time you have to create the folder separately from importing your files. Other than these minor points, the Loader and Logger operate the same as earlier versions.

Resource Manager

In earlier SDKs, MIFs and Resource (bar) files were created using distinct tools. The new Resource Manager provides a single tool for these tasks. This new tool also introduces new file formats. The old .BRX and .BRI file formats are now replaced with a .CAR file for your resources. In this format the “Dialogs” and “Arbitrary Resources” sections have been eliminated. Fortunately, in our experience, these formats were seldom used.

A new feature for the Resource Manager is a way to view the raw source in its own text editor. Some developers reported that the previous Resource Editor could crash if you modified its source text while the editor was open. This change makes it easier to edit resources around while staying in the same program. Keep in mind that the source is now Lua instead of XML based. Compiling a CAR file produces a header file (now *.h instead of *.brh) and the standard .bar file.

MIFs are now compiled from CIF files (instead of MFX files) and while you can load a MIF into the Resource Manager you will be unable to edit it. The designers at Qualcomm thought ahead for those older projects that need to be updated. If you are converting an older project and need to create a CIF you can load the old MIF or MFX, save it as a CIF and use that CIF going forward. A new MIF can then be generated from the Resource Manager as needed. Old icons from the MIF are automatically saved during this conversion.

The Resource Manager and new CIF format are a major reason to install the SDK Manager. As discussed above, the TBT now requires four application icons for all Brew submissions and the fourth icon is only accessible in Brew MP. Also relevant to NSTL requirements is the fact the Resource Manager shows two places for privileges: one in

the applet section and one in the module section. The difference is that the “Legacy” privileges will display if you open the MIF file in an old MIF Editor. Again, we recommend filling out both areas. The last point about the CIF is that it was designed to allow several applications or widgets in one MIF (see TrigML for more information). The side effect is that the class ID cannot be changed from within the Resource Manager GUI and must be changed using the source text editor.

Simulator and Target Manager

Another major tool of interest is the Simulator with its new Target Manager. In the older Brew SDK, a device skin needed to be assigned to the simulator and the simulator had to reference an application directory on your computer’s hard drive. The Brew MP SDK provides device targets that act as a complete phone. Under the Brew MP SDK, the phone simulator device feels much more like an emulator than a basic simulator.

The Target Manager is where you now create device targets. The Target Manager itself is functional, but we hope new versions will provide the ability to edit the target for memory size or screen size. The device targets are written in Flash and allow you to close the phone, turn it around, power it off and on, and a variety of things that either were not possible or not easily done in the earlier simulators. These added features come at the cost of needing to install the application onto the simulator instance. This is more work than simply pointing to the compiler’s output directory as with earlier simulator versions. Fortunately, the Brew MP Visual Studio plug-in automates this task. Note that old application installations remain on the device, so be careful that the new build successfully installs.

Visual Studio Plug-in

Beyond the standard tools, your SDK Manager also handles the installation of IDE plug-ins. One of the biggest improvements in the new Brew MP is the functionality of the Visual Studio plug-in. In addition to providing a quick way to debug projects, the plug-in also makes it much easier to manage makefiles. While there are few ways to compile the module (.mod) for the phone, most projects that we have encountered use the standard makefile format. However, we would see a variety of build processes to use the makefile. Often this was an attempt to streamline the building for variety of devices. The new Visual Studio plug-in recreates the makefile for you on each build and will run your compiler to create the module, the DLL, MIF, and any BAR files necessary. It then distributes these application files into a folder structure of your creation. The Visual Studio plug-in is customizable for using various compilers, folder structures, custom build commands, preprocessor defines, and more. This eliminates the need for several BAT files that are often run in the Visual Studio build events.

When you originally install Brew MP, some environment variables are added to your system. These variables are then used by the plug-in to easily build your project and its makefile with all of the correct information. A slight downside is that the Visual Studio’s plug-in attempts to include your Brew MP directories to every project with no clear way of differentiating what should not be considered a Brew MP project. *For this reason, the plug-in must be uninstalled before compiling any non-Brew MP projects.* As mentioned before, the SDK Manager allows you to install and uninstall plug-ins easily. So far we have not seen any ill effects from quickly uninstalling the Visual Studio plug-in on those days when not programming in Brew.

Final Thoughts

Having worked with many evolving mobile technologies over the past ten years, we recognize the challenges of innovating while preserving backward compatibility. Brew MP and its SDK represent one of the most graceful evolutions that we have seen. Because of this, our Brew MP porting efforts have been going smoothly and our new development has the potential to use some exciting new APIs. The SDK is more cohesive and helps us to more effectively create both Brew MP and legacy Brew builds.

We hope this paper has been helpful as you work with Brew MP. Please feel free to contact Rich Smith, OpenPath's CTO (rich.smith@openpathproducts.com) if you have any questions or feedback.

About OpenPath

Founded in March 2001, OpenPath Products is one of the most experienced mobile development, porting and QA companies in the world. By forming a collaborative partnership with its customers, OpenPath's U.S. based development and QA teams provide timely delivery of the highest quality Brew, Brew MP, Android, Blackberry, J2ME, Windows Phone, iPhone, Widgets, and other mobile products. For more information visit: www.openpathproducts.com or call 1.410.897.0406.