

Writing an Efficient BREW[®] Porting Layer

Viswanathan “Vishy” Kalyanakrishnan
Staff Engineer/Manager
QUALCOMM



Overview

- **BREW OEM Layer**
 - Components & Function
 - Where it fits in
 - Interfaces it deals with
- **Need for optimizing the OEM layer**
- **Operator requirements & importance**
- **Optimization techniques**
 - Task prioritization
 - Display layer
 - File System layer
 - Network operations
 - Resource files

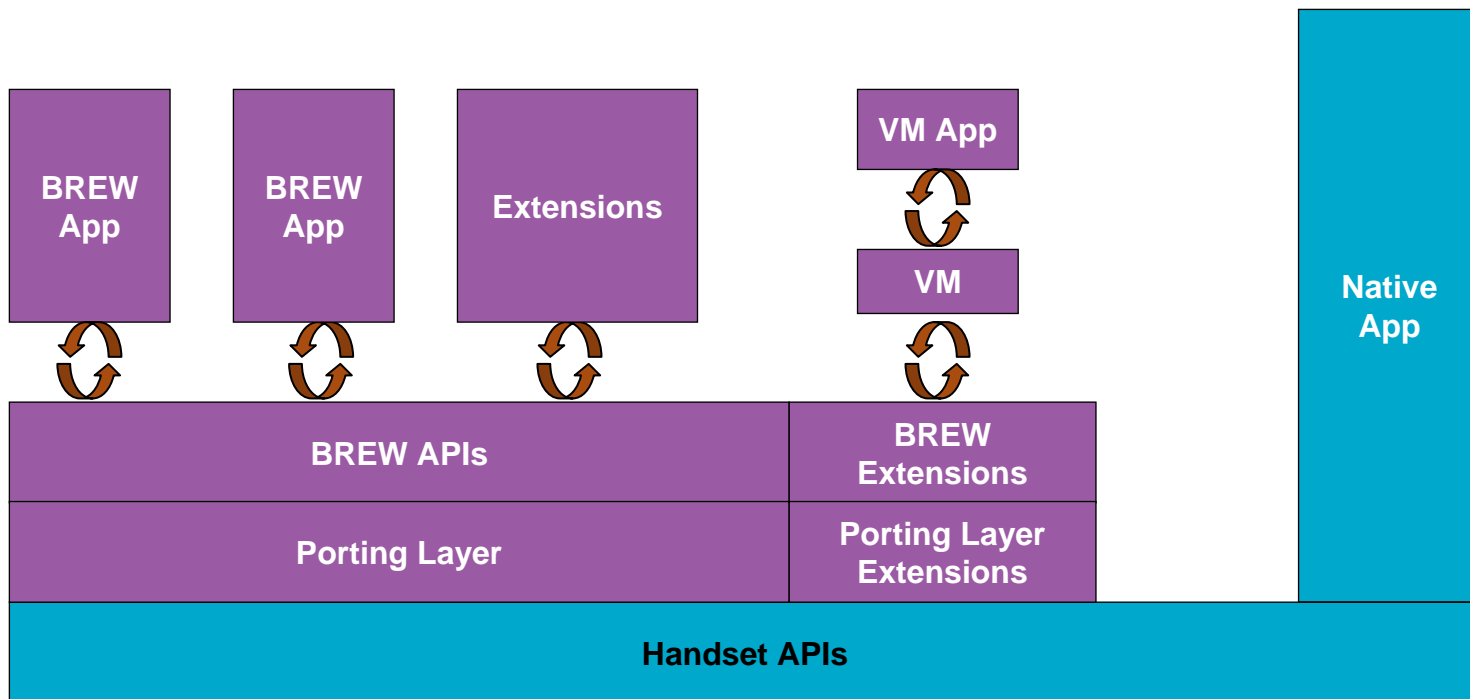
Overview (continued)

- **Handling Memory constraints**
- **Q & A**

What is the BREW OEM Layer?

- **Porting layer typically referred to as OEM Layer**
- **Glue between the BREW APIs & native device functions**
- **Contains a set of portability APIs**
- **Customizable by OEMs per device features & operator requirements**

Where it fits



 Dynamic Link

Applies to BREWv3.x & prior

Components of the OEM layer

- **BREW AEE header Files**
- **OEM Layer header files**
- **OEM Layer source files**
- **Resource files (strings, images, themes etc.)**
- **Configuration items**
- **Libraries**
- **Data files**

OEM Layer Interfaces

- **Display**
- **File System**
- **Networking/TAPI**
- **Device configuration**
- **Media**
- **Addressbook**
- **Database**
- **Security**
- **Etc.**

Need for OEM Layer Optimization

- **Improve BREW API performance**
- **Meet operator requirements**
- **Enhanced end user experience**
- **Obtain optimal performance given device constraints**
- **Expose additional functionality unique to the device**
- **Maintain backward compatibility**

Operator Requirements & Importance

- **Features required**
- **Memory requirements**
- **Benchmark tests**
- **Hardware**
- **File system performance**
- **Networking performance**

Optimization Techniques (1 of 5)

- **Task Prioritization**

- Pick a task context
- Identify the task priority
- If no separate BREW task, recommend using UI task context
- If running as a separate task, recommend a priority that is equal to or just lower than the UI task
- Very low priority could cause BREW task starvation
- Very high priority could cause other important tasks to starve

Optimization Techniques (1 of 5) (continued)

- Analyze task interactions
- Prevent CPU hogging – potential for watchdogs to be set off
- Periodically give up CPU to other tasks

Optimization Techniques (2 of 5)

• Display Layer

- One of the first layers to be ported for BREW
- Significant impact on end user experience
- Hardware
 - TFT Active Matrix, TFD, DSTN, CSTN etc.
 - Display Technology & Bus Architecture affect performance
- Hardware refresh rate & device buffer refresh rates
- Optimization in display device driver software

Optimization Techniques (2 of 5) (continued)

- Support the right bit depth
- Exploit processor caches
- Dirty rectangle optimizations
- Exploit hardware decoders
- BREWStone[®] can help

Optimization Techniques (3 of 5)

• File System Layer

- Significant impact on device performance
- Choosing the right flash part
 - NAND – Fast erase, slow single byte write, fast multibyte write, slow read, low power consumption, needs additional RAM for code execution
 - NOR – Slow erase, fast single-byte write, slow multibyte write, fast read, high power consumptions, does not require additional memory for code execution
- Take advantage of BREW's file caching mechanism
- File system block size variations

Optimization Techniques (3 of 5) (continued)

- **Network Operations**

- Optimize TCP window size
- Implement OEMSocket_SetSndBuf() & OEMSocket_SetRcvBuf() APIs
- Use OATWeb & OATDownload tests to obtain performance statistics

Optimization Techniques (4 of 5)

- **Configuration Items**

- Provide device information & OEM configuration
- Frequently accessed by BREW
(OEM_GetConfig/OEM_GetDeviceInfo)
- Typically maintained in persistent storage (NV/FS)
- Cache data for faster access
- Send out notifications on device item changes
- Maintain persistent storage & cache in sync

Optimization Techniques (4 of 5) (continued)

• Resource File Management

- BREW Resource files typically contain images, strings & dialogs
- BREW uiOne™ Widgets (BUIW) kit files contain theme information
- Loaded using ISHELL_LoadResXXX() APIs
- Exploit CFGI_CACHED_RESOURCES item
- Pros – Faster access to resources
- Cons – Increased heap usage

Optimization Techniques (5 of 5)

- **Resource File Management**

- Use BMPClean tool to reduce bitmap image sizes
- Preload resources at startup
- Move resource files to code space

- **Event Notifications**

- If registering for BREW notifications, make sure they are consumed as needed to prevent unwanted additional notifications
- Investigate potential to pre-launch applications in the background to reduce application load time.

Handling Memory Constraints

- **Devices such as handsets constrained on memory (RAM, ROM)**
- **ROM contains Read Only Data, Read Only Code & non zero initialized data**
- **RAM contains Read/Write data & Zero initialized data**
- **BREW libraries & OEM layer consume RAM & ROM**

Handling Memory Constraints

- **Identify minimum set of BREW features required**
- **Typically enabled/disabled via OEMFeatures.h**
- **Unlink libraries & remove code that is not required**
- **Analyze BREW feature interdependencies to identify all possible candidates for removal**
- **Re-use libraries (e.g., a PNG image decoder on the native side can also be used by BREW OEM layer)**
- **Inspect OEM Layer code & re-write more efficient code if needed**

Handling Memory Constraints

- **Explore moving code/data between file system and code space to adjust to device restrictions**
- **Exploit BREW features such as persistent files (files located in ROM area & accessed via standard BREW IFILEMGR APIs)**

Summary

- **Efficient, well-written BREW Porting layer translates to not just optimal API performance, but enhanced end user experience too**
- **Effectively apply optimizations to this layer**
- **Use applications like BREWStone, OATWeb & OATDownload during development to measure performance & optimize**
- **Understand operator requirements clearly so that you can actually take advantage of them**
- **Remember to maintain backward compatibility on OEM extensions also**

Q&A

- **Questions?**