



BREW 2005
conference

Debugging BREW® Porting Layer

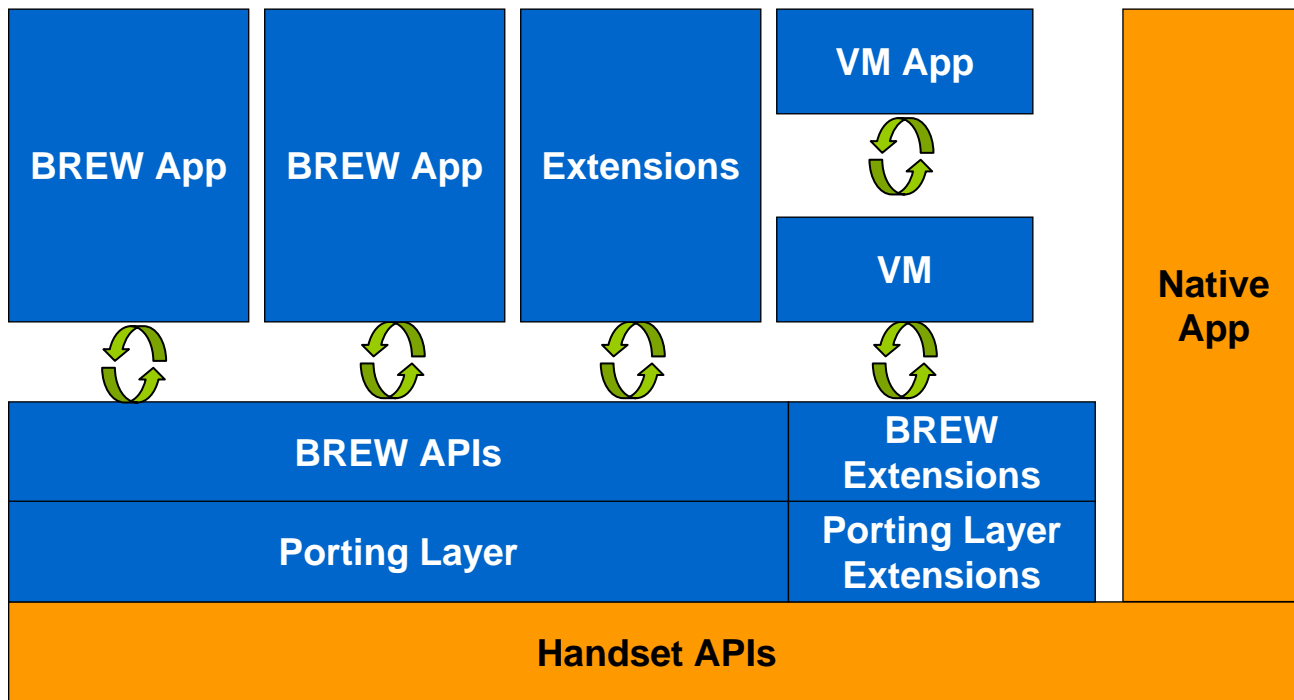
Sathish S. Ambley
Staff Engineer
QUALCOMM Incorporated

June 2, 2005

Session Overview

- **BREW architecture**
- **Simulating & debugging on Windows**
- **Interpreting BREW logs**
- **Enabling & disabling BREW debug sequences**
- **Debugging heap corruption**

BREW Architecture



**BREW Applet
Architecture**

Other

BREW Architecture (Cont.)

- **Device Layer**

- Provides Primitive OS & Device Driver functionality

- **OEM Layer**

- Provides services to the AEE layer natively or through the Device Layer

- **AEE Layer**

- Provides a uniform API to applications

Simulating on Windows

- **Porting Kit includes the BREW Simulator for Windows environments**
- **Microsoft Visual Studio workspace for Simulator that includes:**
 - **OEM sources that can be modified to simulate the behavior**
 - **BREW library for Windows**
 - **Other libraries to build the BREW Simulator**
- **BREW Applications & Extensions can be added to the workspace**

Simulating on Windows (Cont.)

The screenshot displays the Microsoft Visual C++ Emulator interface. The main window shows the source code for the `AEEMedia_Play` function. The code includes a `break;` statement and a `return SUCCESS;` statement. The left sidebar shows the project structure for 'Emulator files'. The bottom-left pane shows the local variables table for the current context.

| Name | Value |
|-------------------------------|-------------------------|
| <code>m_pUser</code> | <code>0x00dc30c8</code> |
| <code>m_pszFile</code> | <code>0x00dc34c8</code> |
| <code>m_pszFileAppPath</code> | <code>0x00dc3364</code> |
| <code>po</code> | <code>0x00dc50a4</code> |

The bottom-right pane shows the debug console output, which includes several `BREW_EMULATOR!` messages and a stack trace for `AEEMedia_Play` at line 266.

Interpreting BREW Logs

BREW Debug Messages

| | |
|---|---|
| #*gBI | BREW Initialization |
| #*gEX | BREW Exit |
| #*gST=<ClassID> | Application Start: EVT_APP_START (with class ID of the application being started) |
| #*gSU=<ClassID> | Application Stop: EVT_APP_SUSPEND |
| #*gRE=<ClassID> | Application Resume: EVT_APP_RESUME |
| #*gCL=<ClassID> | Application Close: EVT_APP_STOP |
| #*gXX | Close all BREW applications |
| #*g**=<ErrorNum> | BREW Exception |
| #*g*C=<ClassID>: <ErrorNum> | Error occurred while creating an instance of the specified class ID |
| #*p:ECode:<Event >, Key:<Code> | Key Press & Release events |

Interpreting BREW Logs

Examples:

| | |
|--------------------------------------|--|
| <code>#*gST=16809984</code> | Start of BREW application with class ID of 16809984 |
| <code>#*p:ECode:101, Key:e035</code> | EVT_KEY_PRESS event for AVK_SELECT |
| <code>#*p:ECode:100, Key:e035</code> | EVT_KEY event for AVK_SELECT |
| <code>#*p:ECode:102, Key:e035</code> | EVT_KEY_RELEASE event for AVK_SELECT |
| <code>#*gSU=16809984</code> | Suspend of BREW application with class ID of 16809984 |
| <code>#*gCL=16809984</code> | Stop of BREW application with class ID of 16809984 as a result of not handling suspend |
| <code>#*g*C=101402c:3</code> | ECLASSNOTSUPPORT error returned when creating instance with class ID of 0x101402c |
| <code>#*gST=16846582</code> | Start of BREW application with class ID of 16846582 |

BREW Debug Key Mode

- **Debug features supported are:**

| | |
|---|---|
| 0 | RESET: Clear all debugs |
| 1 | MEMORY: Walk heap when checking pointers |
| 2 | NETWORK: Network Diagnostics |
| 3 | MEMORYAVAIL: Display memory available |
| 4 | SYNCDDBGPRINTF: Toggle synchronous dbgprintf |
| 5 | DUMPMODULES: Dump module list (comes out dbgprintf) |
| 6 | DUMPHEAP: Dump heap (comes out dbgprintf) |

BREW Debug Key Mode (Cont.)

- **Debug features supported are:**

| | |
|-----|--|
| 7 | DUMPRESCACHE: Dump resource cache (comes out dbgprintf) |
| 8 | DUMPFILCACHE: Dump file cache (comes out dbgprintf) |
| 9 | CLOSERESCACHE: Closes all cached BREW resource files |
| 10 | WEIGHTFARF: Performance weight output |
| 11 | BREWSMS: To test BREW-directed SMS |
| 999 | PERSIST: Save debug flags to disk |

Debugging Heap Corruption

- **BREW Memory Management**
 - BREW's Applet heap is shared among all running BREW applications
 - BREW keeps track of memory that Applets allocate by marking heap nodes with a 32-bit ID that associates the node with the calling Applet
 - When an Applet exits, BREW sifts through the heap, looking for leaked nodes marked with the Applet's 32-bit ID & frees them

Debugging Heap Corruption (Cont.)

- **Consider the following heap corruption scenario:**
 - **BREW MediaPlayer Application that plays various media formats**
 - **Browses the media directory & lists all supported media formats which is played when selected**
 - **Number of media files present in the directory exceed the allocated space to hold them**
 - **Absence of check to limit the storage results in corruption of the heap**

Debugging Heap Corruption (Cont.)

Number of media files to hold

```
#define MP_MAX_FILES    32
```

Media File data structure

```
char *    m_szFileArray[MP_MAX_FILES];
```

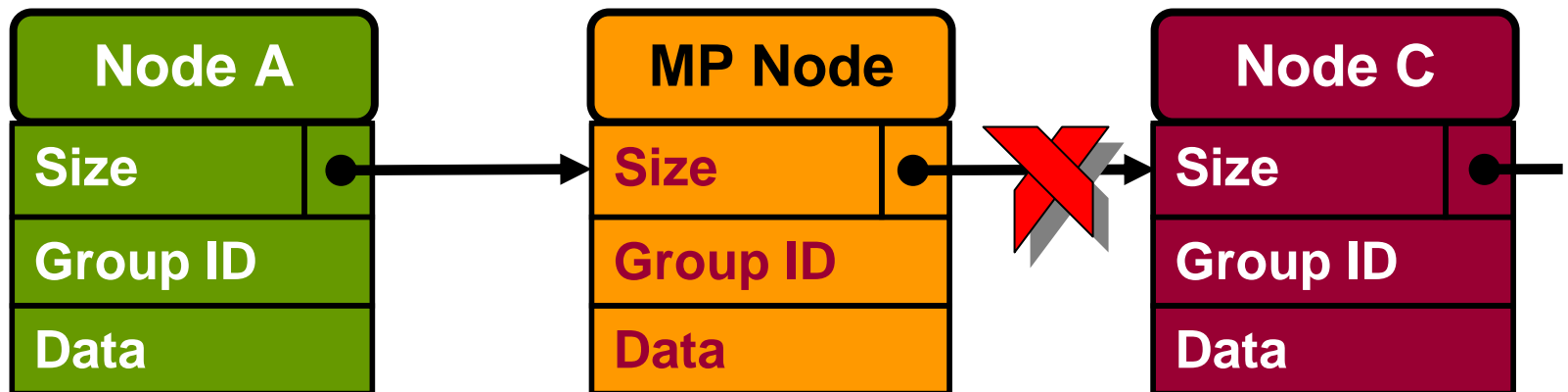
Enumerating the media files to the array

```
// pme structure that contains the member m_szFileArray
// is allocated from BREW heap using MALLOC
IFILEMGR_EnumInit(pFileMgr, MP_MEDIA_DIR, FALSE);
while ( /* wItemID < MP_MAX_FILES && */
        IFILEMGR_EnumNext(pFileMgr, &fi))
{
    char * szName;
    pme->m_szFileArray[wItemID] = STRDUP(fi.szName);
    wItemID++;
}
```

Debugging Heap Corruption (Cont.)

- **Heap Node Corruption**

- **MP Node: MediaPlayer heap node overflow results in corrupting the next node's header breaking the link between the heap nodes**



- **How to debug this heap corruption?**

Debugging Heap Corruption (Cont.)

- **Debugging Steps:**

- Turn on the BREW debug key mode sequence for memory checks to enable validation of heap
- Identifying the address of the node that is getting corrupted
- Add a Write breakpoint at this address location to identify who is writing into this location
- When the Write breakpoint hits, examine the source to determine the reason for writing to this location
- Identifying & correcting the source causing heap corruption

BREW OEM Support

- **On-Site visits or visit BREW Integration Lab (BIL) by appointment**
- **Send queries through E-Service:**
<http://qishelp.qualcomm.com>
- **Regional Offices In Japan, Korea, China, Israel & Brazil**
- **BREW OEM Extranet**
<http://brewx.qualcomm.com/oem>



Questions & Answers