QUALCOMM®

Binary Runtime Environment for Wireless™

BREW™ Browser 2.0 Software Design Overview

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA. 92121-1714
U.S.A.

This manual was written for use with the BREW™ Browser software version 2.0.0. This manual and the BREW Browser software described in it are copyrighted, with all rights reserved. This manual and the BREW Browser software may not be copied, except as otherwise provided in your software license or as expressly permitted in writing by QUALCOMM Incorporated.

Printed in the United States of America

BREW™ Browser 2.0 Software Design Overview

80-D4526-1_A

April 4, 2003

Contents

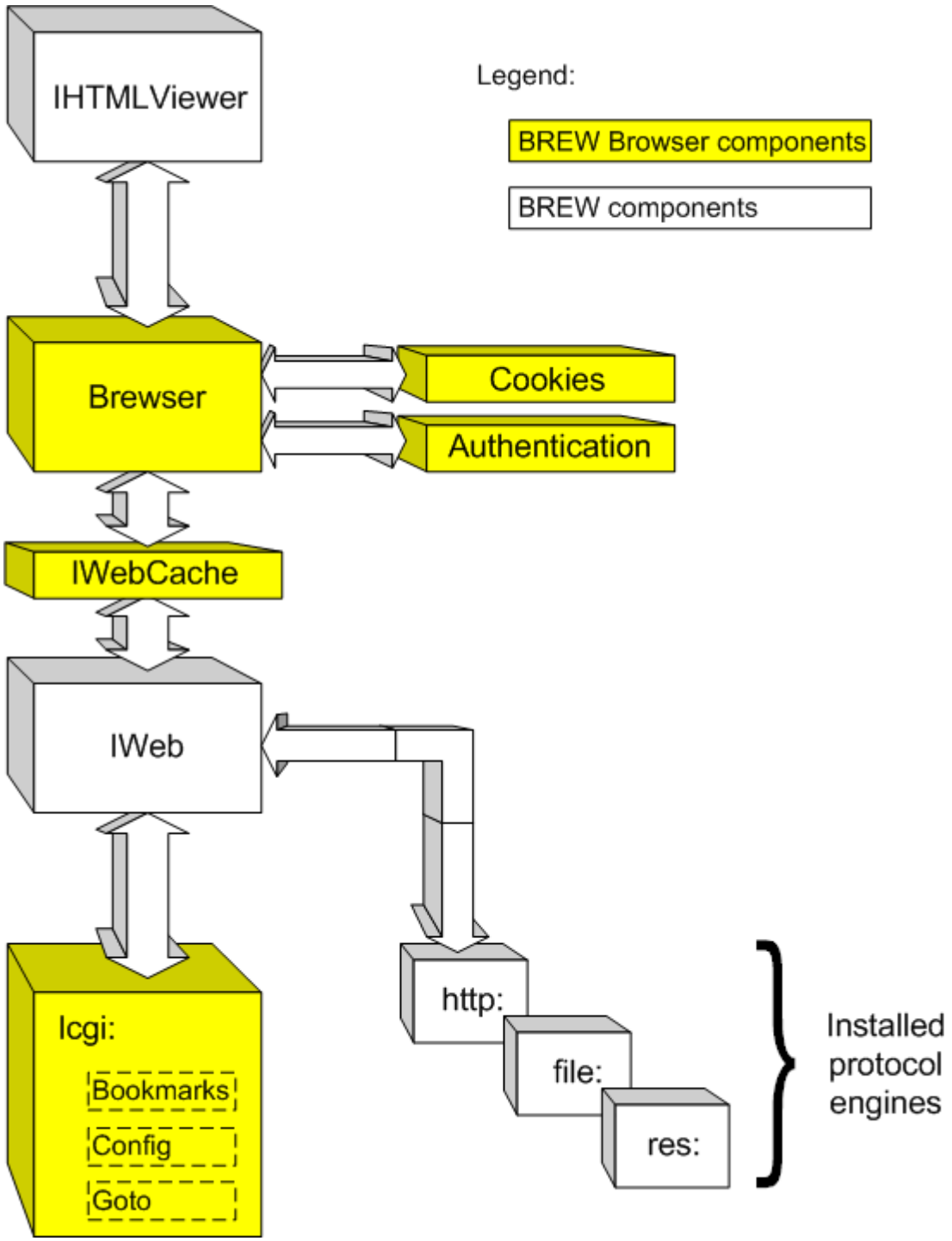
BREW™ Browser 2.0 Software Design Overview

The BREW™ Browser code is grouped into the following components.

Component	Location	Description
Brewser	brewser.c	In addition to handling startup, shutdown, and event handling, the Brewser component manages the UI Shell, navigation, and history functions. See Brewser component on page 6.
CookieMgr	brewser.c	The CookieMgr component manages all cookie-related functions, including expiration, loading at startup, and freeing of cookies in response to BREW Low-RAM callbacks. See CookieMgr component on page 7.
AuthMgr	brewser.c	The AuthMgr component maintains authentication cache-related functions and all associated logic for user password prompts. See AuthMgr component on page 8.
LCGI	bookmarks.c config.c lcgieng.c	The Local Common Gateway Interface (LCGI) component of BREW Browser intercepts web requests and generates the appropriate HTML screens for objects such as bookmarks, preferences, and URLs. See LCGI component on page 8.
IWebCache	webcache.c	The IWebCache component handles all caching-related functions in BREW Browser. See IWebCache component on page 10.

The following diagram summaries the interaction between these components and other BREW components.

Interaction between BREW Browser components and other BREW components



NOTE: Other, less core parts of the BREW Browser—tooltips, status box, slider, headermaker, refresher, prefmgr, and utility code (util)—have been separated from brewser.c in the interest of efficiency.

Brewser component

Brewser is the application's main object. It implements the IApplet interface that is used directly by BREW, so it receives events from BREW. In addition to handling startup, shutdown, event dispatching, and providing the glue for all the other components, Brewser also performs the following two specific roles.

Role	Description
UI Shell	Brewser handles drawing and placement of controls that constitute the UI, including the main HTML pane, toolbar, tooltips, status box, and sundry dialogs.
Navigation and history management	Brewser manages the history list, what is shown during navigation, additions to the history list, and what “Back” means. Brewser is the object that receives notifications from the IHtmlViewer to take a link.

UI Shell

BREW Browser's UI is made up of the following four objects:

- An IHtmlViewer that is the main viewing pane. All fetched documents are displayed here. BREW Browser can currently view only objects that BREW's HTML viewer can handle.
- A soft key menu with tooltips that gives access to buttons, such as Back, Help, Bookmarks, and so forth.
- A secondary IHtmlViewer used to open dialogs that don't belong in the main IHtmlViewer mentioned above.
- A status bar image to show navigation progress.

BREW Browser does not have native dialogs. Most of the browser's own UI is implemented in HTML using the lcgi: extension. For more information, see LCGI component on page 8. Some exceptions to the rule of UI accomplished in the main IHtmlViewer are the authentication dialogs, page information dialogs, and unexpected error condition message boxes. These are also HTML pages, but they're shown in a separate, smaller IHtmlViewer on top of the main pane.

Navigation and History

BREW Browser keeps a singly linked history list, where the tail of the list is never freed (and is allocated as part of the Brewser object).

When a new page is requested by a forward link, a new history entry is allocated and held by pbhCur; this history entry is not added to the history list until the IWEB_GetResponse() callback fires. An IWeb request is formulated, and headers for the request, such as cookies, authentication information, and a User-Agent header, are collected in Brewser_NavStart(). Next, IWEB_GetReponse() is called. When the AEECallback passed to the IWeb_GetResponse() fires, pbhCur is added to the history list, and the ISource resulting from the request is given to the main IHtmlViewer to display. It's important that new history entries not be considered *history* until a successful GetResponse() fires. If the user cancels the navigation before the GetResponse() callback fires, the new history entry is discarded and the main IHtmlViewer's content is left intact.

New history entries are not allocated when the user performs a Back or Refresh; instead, pbhCur is set to a history entry already in the list, and the entry is marked as a *back* entry. After the GetResponse() callback fires, all history entries after the current target are discarded (so there's no Forward function in BREW Browser). If the user cancels the request before the GetResponse() callback fires, the history list is unaffected.

Error conditions are handled in the GetResponse() callback according to whether anything from the server can be shown. Internal—BREW or other non-protocol—errors show an error page loaded from resources. Protocol error pages are shown as is from the server.

CookieMgr component

The CookieMgr implementation lives in brewser.c.

The Brewser object hands all Set-Cookie headers to the CookieMgr. CookieMgr keeps track of expiration, and persists non-ethereal cookies to disk at BREW Browser exit, loads non-ethereal cookies at BREW Browser startup, and manages the cookies in RAM. CookieMgr frees cookies in response to BREW Low-RAM callbacks, but it won't free any data below the 2048 byte mark. This ensures that most well-behaved cookie-dependant sites will function properly.

CookieMgr writes to the Embedded File System (EFS) minimally, so until a new non-ethereal cookie is received or a non-expired cookie is deleted, the cookie database can contain stale cookies. Expired cookies do little harm at startup, and are discarded. CookieMgr is also responsible for formulating the correct Cookie header for web requests, given a URL.

AuthMgr component

The AuthMgr implementation lives in brewser.c.

AuthMgr keeps the authentication cache and shows prompts by means of callbacks into the Brewser object. Each web transaction that requires authentication goes through the BrewserAuth object, which groups the structures necessary to asynchronously request that AuthMgr prompt the user.

BrewserAuth takes the URL and realm from an authentication-required web response that shows the server-supplied “auth failed” page in the main pane, and asks AuthMgr to prompt the user. If the user clicks **OK** in the authentication dialog, AuthMgr's cache is updated with the new credentials and the page is refetched. If the user clicks **Cancel**, AuthMgr's cache is cleared of any information for the relevant URL and the dialog is dismissed, revealing the already-loaded “auth failed” page.

LCGI component

The native UI screens of the browser—that is, screens that allow the user to select bookmarks, enter a URL, or modify settings—are accomplished with HTML forms presented in the main IHtmlViewer. BREW Browser includes an LCGI component that intercepts web requests and generates the appropriate HTML. One of the native UI screens can be invoked by simply navigating to a special URL. In this way, the browser itself is actually a web application. Indeed, BREW Browser's UI and config could reasonably live on a remote server.

The LCGI component takes the form of an IWeb protocol engine. BREW Browser directs requests to IWeb, which examines its set of protocol engines to determine how to service the request. The LCGI module recognizes the lcgi: URL scheme, so each screen in the native UI is allocated from within this namespace. That is, URLs beginning with lcgi are assigned to each screen.

Some requests are handled directly by the LCGI component, and some are delegated to other relatively self-contained objects: Bookmarks and Config.

Special URLs

To overcome some web application limitations and enhance the UI, a few non-standard URLs have been defined for use by LCGI. These URLs are interpreted by the navigation engine in BREW Browser.

URL	Description
brewser:back	This URL causes BREW Browser to go back in its history list. This URL has optional arguments: • to=<URL> If this argument is present, BREW Browser skips back to a history entry whose URL is the named URL. If the URL is not found, BREW Browser goes back one entry, unless “past” is specified. • past=<URLprefix> If this argument is present, it causes BREW Browser to skip past any URLs that begin with URLprefix; for example, brewser:back?past=lcgi: would cause BREW Browser to skip back past any history entries whose URLs begin with lcgi:. **NOTE:** If both the “to” and “past” arguments are present, “to” takes precedence.
brewser:refresh	This URL causes BREW Browser to reload the current history entry.

UI design concerns

Because LCGI URLs end up in BREW Browser's history list just like any other URL, and because some UI screens may be entered by more than one avenue, LCGIEng prepends an *invocation context* to allow brewser:back URLs to be unequivocal, and to allow for a degree of modular web programming (you can use your “referrer” to return from a multipage subroutine).

Security concerns

BREW Browser ensures that a correct referer URL is sent with each web request so the LCGI engine can validate that a POST originated from a valid LCGI form submitted by the user.

Some LCGI form submissions modify browser settings that could impair proper operation of the browser. Without this validation of the referrer, malicious content could include links that change these settings. For example, a link or button in a malicious page could mimic the proxy configuration screen in the native UI, and direct all requests through a nonexistent server.

All extensions to the existing LCGI must take these security considerations into account.

IWebCache component

IWebCache is a BREW-style interface you can use much like the BREW IWeb interface. For example, BREW Browser can pass IWebCache to an instance of IHtmlViewer. The IWebCache implementation is actually a thin wrapper around an instance of BREW's IWeb. All of BREW Browser's IWeb transactions pass through IWebCache and, therefore, can potentially be cached.

IWebCache is a "RAM only" cache; that is, when IWebCache is running, all of its data resides in RAM. On creation, IWebCache reconstitutes itself from the webcache.dat file. On deletion, IWebCache writes an image of itself out to disk in the same file.

IWebCache tries to maintain as much cached data as RAM on the device will allow, including expired documents, which are useful for the Back function. IWebCache may cull entries in the cache when its registered BREW "Low RAM" callback is invoked. When this occurs, IWebCache culls expired entries first, then it discards entries in a least recently used (LRU) fashion.

IWebCache does not cache responses that have non-protocol error codes, such as SSL trust errors, nor does it cache responses that are marked as Local, including those generated by LCGI.

Resources

BREW Browser has built up a considerable library of utility code to deal with BREW resources as byte-wise strings; BREW Browser converts UTF-16 string resources to UTF-8, but it does not convert HTML resources. Much of BREW Browser's UI is HTML, which it loads from BREW resources and then modifies appropriately before displaying any pages. For example, the authentication dialogs are written in HTML. Before they are shown, the strings for "URL" and "realm" are written into them.

For this reason, most of BREW Browser's resource strings and resource HTML files must be in the same character set, but they must not be in UTF-16. By default, the HTML viewer accepts UTF-8, and can map UTF-8 to all of BREW's supported handset character sets.

BREW Browser's resource source files live in the <res> directory. BREW Browser's resources are grouped by type: images (no text allowed) and text. The text resources are either strings or HTML files, which live in a language-keyed directory under <res>, and the image resources are grouped by color or monochrome (1-bit and 8-bit directories, respectively).

BREW Browser does not use the BREW Resource Script format (.bri extension). The scale of BREW Browser's localization requirements and the BREW Browser development team's use of a text-based version control system led to the abandonment of the binary BREW Resource Intermediate (BRI) file format. BREW Browser uses its own resource script file format, and includes a tool (barc.exe in the tools directory) to compile the language-specific brewser.brc files into a brewser.bar (BREW Applet Resource) file.

Since Module Information Files (MIF) are a special kind of BAR file, BREW Browser also uses barc to compile its MIF.

The xmod directory

The xmod directory contains a trivial IModule implementation (essentially the same as AEEModGen.c) and some helper makefiles that allow BREW Browser to be compiled in Thumb mode for handsets. BREW Browser does not use AEEAppGen.c; instead, the Brewser component implements IApplet.