QUALCOMM®

Binary Runtime Environment for Wireless®

BREW SDK® 3.1.5 Release Notes

brew™

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

BREW SDK 3.1.5 Release Notes

70-D4737-7 Version 3.1.5

April 7, 2006 4:29:25 PM

Contents

Introduction

The Binary Runtime Environment for Wireless® (BREW®) Software Development Kit (SDK) 3.1.5 Release Notes describe the new features added in the 3.1.5 release; changes made to the BREW SDK® since the release of version 3.1.4; and installation and uninstallation instructions for version 3.1.5.

This document is organized into the following sections:

Section title	**Description**
What's New Overview	This section describes new features added in BREW 3.1.5, since 3.1.4 was released.
BREW API Changes (3.1.4 to 3.1.5)	This section discusses the BREW API changes in this release, since 3.1.4 was released.
BREW API Changes (3.1.3 to 3.1.4)	This section discusses the BREW API changes in this release, since 3.1.3 was released.
BREW API Changes (3.1.1 to 3.1.2)	This section discusses the BREW API changes in this release, since 3.1.2 was released.
BREW API Changes (3.1.1 to 3.1.2)	This section discusses the BREW API changes in this release, since 3.1.1 was released.
BREW API Changes (3.1.0 to 3.1.1)	This section discusses the BREW API changes in this release, since 3.1.0 was released.
BREW API Changes (3.0.1 to 3.1.0)	This section discusses the BREW API changes in this release, since 3.0.1 was released.
BREW API Changes (3.0.0 to 3.0.1)	This section discusses the BREW API changes in this release, since 3.0.0 was released.
BREW API Changes (2.1.0 to 3.0.0)	This section discusses the BREW API changes in this release, since 2.1.0 was released.
BREW SDK changes (3.1.4 to 3.1.5)	This section discusses the BREW SDK changes in this release, since 3.1.4 was released.
BREW SDK changes (3.1.3 to 3.1.4)	This section discusses the BREW SDK changes in this release, since 3.1.3 was released.

Section title	Description
BREW SDK changes (3.1.2 to 3.1.3)	This section discusses the BREW SDK changes in this release, since 3.1.2 was released.
BREW SDK changes (3.1.0 to 3.1.2)	This section discusses the BREW SDK changes in this release, since 3.1.0 was released.
BREW SDK changes (3.0.1 to 3.1.0)	This section discusses the BREW SDK changes in this release, since 3.0.1 was released.
BREW SDK changes (3.0.0 to 3.0.1)	This section discusses the BREW SDK changes in this release, since 3.0.0 was released.
BREW SDK Changes (2.1.0 to 3.0.0)	This section discusses the BREW SDK changes in this release, since 2.1.0 was released.
Known Issues	This section explains where you can find information about known issues, if any, in this release.
Installation Instructions	This section describes installation and uninstallation instructions for the 3.1.5 release.

What’s New Overview

BREW 3.1.5 incorporates new features and important enhancements to enable complete device user interface development on BREW. All references to BREW imply BREW 3.1.5 unless otherwise stated.

What’s new in BREW 3.1.5

BREW application-level (API) features

QoS - Quality of Service

- Support has been added for enhanced functionality and extended features including a new interface, IQoSBundle.
- Commercial QoS support for 1XDOrA has been added, including basic QoS, QoS bundle, and QoS aware and unaware status/events.
- For UMTS, multiple secondary contexts have been added for IPv4.

Multicast

- Support has been added for generic multicast and BCMCS functionality, including a new interface (IBCMCSDB) and support for new information codes.

Camera enhancements

- New parameters have been added to support real-time de-blurring, WLED and strobe flash, and auto-focus options.
- Luma Adaptation, macro mode, new effect types

Camcorder enhancements

- Support has been included for timestamps on movie clips.
- New parameters for MPEG4 audio postcard have been added.
- AAC encoding of audio in MPEG4 movies.

Removable Media Card (RMC)

- Support has been added for applications to register or de-register for notifications associated with the inserting or removing of memory cards (Requires hot-plug manager support in AMSS).

SMS enhancements

- Support has been included in ISMS for absolute time deferred delivery, relative time deferred delivery, absolute time validity, and relative time deferred delivery.
- Delivery cancellation is now supported for CDMA networks.
- ISMS supports the inquiry of submitted message status on GSM networks.
- CPHS delivery of Voice Mail Notification is now supported.

Media enhancements

- AMR-WB playback is supported via the IMedia interface .
- IMedia is enhanced to enable applications to specify the number of simultaneously voiced notes for media formats with synthesized playback.
- IMedia supports delivery of PMD frames to an application via callback, allowing user interface applications better screen control with the ability to overlay frames.
- IMedia now supports simultaneous playback of multiple PCM media.

Data

- IWeb now supports SSL tunneling over an HTTP proxy.
- Two new WebOpts (options) have been added to provide the capability for applications to manage their own tunneled connections.

What's new in BREW 3.1.4

BREW application-level (API) features

SMS enhancements

BREW Client 3.1.4 added additional ISMS capabilities enabling:

- Support events indicating cell change
- Events indicating duplicate broadcast messages

Camera enhancements

BREW Client 3.1.4 added additional ICamera capabilities enabling:

- Exposure Control for controlling the auto exposure mode
- Exposure Value (EV) for adjusting the exposure value in camera preview
- Camera Sensor information for adjusting the frame rate in night shot mode

Stereo Bluetooth Headset support

BREW Client 3.1.4 added a "stereo bluetooth headset device type" to the ISound interface. This support enables an application to leverage the availability of a stereo bluetooth headset to play specific audio content.

Clamtype Flip interface

BREW Client 3.1.4 added a new interface called "IFlip". IFlip provides various information regarding the clamtype device (clam shell, slider, etc.) flip support and capabilities. This API functionality includes:

- Function to get the number of flips on the device and their unique IDs
- Function to get the current flip position (angles along x,y,z axes)

- Function to get the minimum and maximum angles possible along the x,y,z axes
- Function to get the number of displays on the flip, and the display class IDs
- Function to get the number of keys on the flip, and their key codes

Additional codec support

BREW Client 3.1.4 expanded IVocoder to support the 4GV codec.

"Beta" IPv6 support

BREW Client 3.1.4 added "beta" IPv6 capabilities to support initial IPv6 trials. This includes:

- ISockPort support for IPv6 socket addresses
- New network interface (INetwork) that supports both IPv4 and IPv6 addresses
- IDNS support for AAAA queries
- IWeb support for IPv6 host names

Commercial support is planned for a future release and may involve changes to the new IPv6 API. For more details, see BREW API Changes (3.1.3 to 3.1.4) on page 33.

Commercial QoS support

BREW Client 3.1.4 added commercial QoS support.

What's New in BREW 3.1.3

W-LAN support

BREW 3.1.3 enhanced ITelephone in order to support W-LAN. Using this enhanced ITelephone API, an application can query for W-LAN capability and receive notifications about coverage. Additionally, this API provides the ability to specify the transport channel preferences for system selection between 1x and WiFi service. Additionally, BREW 3.1.3 added the IWIFI API which is used to configure the W-LAN parameters and to get information on the current W-LAN configuration and status.

HTTP 1.1 support

BREW 3.1.3 added support for sending HTTP requests without predetermined length, using chunked encoding transfer format and it also added support for decoding of HTTP 1.1 server responses from chunked encoding format. The decoded responses are passed to the client as a peekable stream.

Backlight support

BREW 3.1.3 added the IBacklight interface, which enables applications to control the various backlights that may be available on a device, for example, primary display, secondary display and keypad. IBacklight enables an application to control the various parameters of a specific backlight, such as on/off status, brightness and contrast.

SMS enhancements

BREW 3.1.3 added additional SMS capabilities enabling:

- Notification of storage changes by ISMS storage
- ISMS access to raw TPDU for MT-SMS (limited to applications with system privilege level)
- ISMS to provide a means of letting its clients find the message reference number and Alpha ID corresponding to a MO-SMS.

HTMLViewer enhancements

BREW 3.1.3 provides several IHTMViewer fixes and enhancements. Refer to the CR section of the release notes for a complete listing.

Additional codec support

BREW 3.1.3 expanded IVocoder to support the following codecs:

- PCM
- G.711

What’s New in BREW 3.1.2

IMedia real-time PCM streaming support

BREW 3.1.2 provides applications with the following real time PCM streaming capabilities:

- Real time streaming and playback of PCM and QCP formats
- Playback of raw media data (without headers) such as raw QCP (13K/EVRC) or raw PCM data. The source of raw data can be file, buffer or stream.

SMS enhancements

BREW 3.1.2 added additional SMS capabilities enabling EMS, Broadcast Configuration, WAP push and message template storage.

Networking enhancements

BREW 3.1.2 added the following additional BREW networking capabilities:

- Mobile terminated PDP context support
- EV-DO broadcast air interface support
- W-TCP support

Additional codec support

BREW 3.1.2 expanded Ivocoder to support the following codec:

- AMR

What's New in BREW 3.1.1

On-target debugging support

BREW 3.1.1 added support for on-target debugging of dynamic BREW applications. Developers can use the BREW Debugger in conjunction with an IDebugger-enabled device to debug any BREW applications without the necessity of special hardware. The Debugger uses a standard serial connection for communication.

What's new in BREW 3.1.0

BREW 3.1.0 incorporates new features and important enhancements to enable complete device user interface development on BREW. Although there are additional APIs included in this release, these APIs are focused solely towards use by handset manufacturers as well as potentially by the small number of application developers that contribute to a device's primary UI and feature set (such as browser client providers, mail client providers, etc.). The release is not intended to affect typical third party BREW developers or existing third party BREW applications that BREW Operators may provide to end consumers.

The BREW 3.1.0 release is most effectively employed as a full user interface development platform when it is combined with the BREW UI Toolkit (also know as UI Widgets) product release. As described in more detail in that product's documentation, the UI Toolkit provides an extensible development framework and toolset for the UI controls and elements used within BREW applications. The BREW 3.1.0 client provides the core functionality needed for a system of applications to operate as an entire UI on a BREW device while the UI Toolkit provides the presentation layer system to achieve a consistent and highly functional "look and feel" across all of the applications.

Application history

Application stacking has been expanded to provide a complete application history. This application history exists as a separate element from the traditional BREW application stack although the Application History and Application Stack should be used together to most effectively manage the behavior of the device UI on BREW. In essence, the application history feature allows applications to write information to the history data that BREW maintains and then retrieve that data when needed, for example, at application start-up or resumption. Applications can appear and make entries in the history multiple times, even when an application is in the background. They can therefore have context-specific state information stored with each history entry to allow for seamless resumption of applications by the end user when navigating through or otherwise interacting with the multiple UI applications that comprise the device’s entire UI Resource Management.

Resource management

Application pre-emption of resources can now be controlled based on the priority or category of an application, as well as the state of that application. This allows higher priority applications to have access to necessary device resources. (This priority is not to be confused with the concept of tasking priorities in an operating system. All of the BREW applications in this release will continue to run in a single task as in prior BREW releases.)

Application interruption

The behavior of BREW application interruption can now be easily defined and controlled based on the priority or category of an application, as well as the state of that application. This allows only the appropriate applications the ability to interrupt the top visible application and only for specified reasons. For example, a manufacturer can implement a UI that allows a higher priority incoming call application to interrupt a game application but that does not allow a game application to interrupt an incoming call application.

Key event capture

Key event handling in BREW has been enhanced to allow privileged applications to receive key events even if they are not the top visible application. This feature is useful in creating macros or in assigning hot-keys that immediately start or resume specific applications. Prior BREW releases had a limitation that only the current top-visible application can receive keypad events.

Low memory notification

BREW now supports low memory notifications. Now, the low memory state behavior of BREW devices can be customized. An example might be that the handset determines the order in which BREW applications are closed in a low memory state.

Removal of test enable bit

The test enable bit functionality was removed in BREW 3.1.0. The associated changes provide several benefits to speed the development and commercialization of both BREW devices and BREW applications.

SMS enhancements

SMS capabilities were enhanced to enable development of SMS clients on BREW that fully comply with SMS standards. New capabilities include enabling of applications to access and utilize additional SMS sub parameters and features of the SMS standards (for example, IS-637).

Pen event handling

BREW 3.1.0 expands support for PEN_UP, PEN_DOWN, and PEN_MOVE events and adds pen event queuing as well as pen event support in widgets.

Privacy check removal

The privacy check was removed from ITAPI_MakeVoiceCall.

What's New in BREW 3.0.1

What's new in the BREW 3.0.1 APIs

BREW 3.0.1 includes the following features for use within BREW applications:

Call Manager Airlink Mode Indicator

BREW 3.0.1 provides applications with an indicator for 1xEV-DO airlink mode. For example on (hybrid) devices that are capable of both 1x and 1xEV-DO, this indicator is useful to distinguish the current active airlink mode. An application can take advantage of the higher data rates offered by the 1xEV-DO airlink to download larger content. Similarly, the same application can limit access to this larger content download if the airlink mode is switched to 1x. This feature enables applications to take advantage of the airlink capabilities and ensures a smooth experience for the user.

Airplane Mode Indicator

BREW 3.0.1 provides applications with an indicator of when the device is operating with RF completely turned off.

MDN Access

BREW 3.0.1 provides applications with access to get the device Mobile Directory Number (MDN). With Local Number Portability (LNP), a device's MDN remains the same after the number is ported across carriers, but the International Mobile Subscriber Identity (IMSI) / Mobile Identification Number (MIN) may change after the number is ported. Some applications may require access to the MDN for functions such as SMS. It cannot be assumed that the MDN is identical to the IMSI/MIN since the number may have been ported.

What’s New in BREW 3.0.0

BREW 3.0.0 incorporates many new exciting multimedia features and provides important enhancements to existing BREW capabilities. All references to BREW in this section imply BREW 3.0.0 unless otherwise stated.

BREW 3.0.0 application-level (API) feature

BREW 3.0.0 incorporates many new features and provides important enhancements to existing BREW capabilities.

Secondary Display Access Capabilities

BREW 3.0.0 provides the ability for applications to access multiple OEM-defined display areas on a device. A single application can access multiple displays simultaneously. Alternatively, separate applications can simultaneously use the separate display areas independently.

MMC/SD Shared Directory Access by Applications

BREW 3.0.0 provides the ability for an application to store, access, and manage content and files on a removable media card using a BREW directory path macro. Applications interact with the SD/MMC memory space in a similar manner to the BREW shared directory on the device’s permanent flash part. BREW applications (MIF or MOD) cannot be stored on the media card. This directory path is protected by a BREW privilege to enable carriers to control the applications that allow content to be removed off the device using the media card.

Generic Serial Interface

A new interface (IPort) has been added to allow applications to access I/O ports on the device. This generic port can be used for serial cables, USB cables, infrared wireless connections, and Bluetooth wireless connections. The interface supports plug-and-play behavior for device discovery by applications. This interface is protected by a BREW privilege to enable carriers to control the applications that provide serial device-to-device communications.

Address Book category access enhancements

BREW 3.0.0 added the IAddrBook_GetCategoryName() and IAddrBook_GetFieldName() functions so applications can determine the text names of OEM-defined address book categories and fields.

What’s new in BREW SDK 3.1.5

The BREW SDK 3.1.5 has been updated to provide simulation for new BREW features and interfaces introduced in 3.1.5 and provide enhancements to existing functionality. In this release, the BREW 3.1.5 Simulator has enhancements for SMS and the features related to network data services. The *API Reference Online Help* has also been enhanced with a series of use case scenarios of selected interfaces including descriptions, call flows, and code snippets.

What’s new in BREW SDK 3.1.4

The BREW SDK 3.1.4 has been updated to provide simulation for new BREW features introduced in 3.1.4 and provide enhancements to existing functionality (e.g. camera simulation and simulation of joystick hardware). Additionally, the simulator is now capable of loading BREW extensions from a generic folder that is available to the Simulator even when the Applet or MIF directory is changed.

What's new in BREW SDK 3.1.3

The BREW SDK 3.1.3 introduces additional Simulator features targeted to increase the efficiency of developing applications and core phone software using the BREW SDK. New features include support for streaming PCM media files and simulation of RSSI levels. Additionally, notifications for headset plug-in and removal and simulation of IBacklight support have been added.

What's new in BREW SDK 3.1.2

The BREW SDK 3.1.2.has been enhanced to provide simulation for new and existing BREW features. Camera simulation is now provided, as is simulation for battery level, airplane mode and high data rate mode. SMS simulation has been expanded in BREW SDK 3.1.2 to include support for ISMSStorage and the ability to loop back SMS message to the simulated device. In addition to these simulation features, the BREW Simulator can now trap misaligned data exceptions.

What's new in BREW SDK 3.1.0

The BREW SDK 3.1.0 has been enhanced to enable more efficient application development and customization. These enhancements are intended to enable the development of an entire Phone UI using the BREW SDK. In addition to simulating new BREW 3.1.0 APIs, new features in the BREW SDK 3.1.0 include enhancements to SMS and TAPI simulation.

The contents of BREW SDK 3.1.0 have been aligned to concentrate on BREW 3.1.0 version-specific development. Keeping that in mind, the SDK now contains only version-specific tools. These include the simulation environment, AEE header files, help documents, sample applications, and device packs. The SDK does not include the version independent tools that were previously part of the SDK package, including the Resource Editor, MIF Editor, BCI Authoring Tool, and BREW SDK Utilities (NMEA Logger, PureVoice® Converter, 2Bit Tool). At the time of publication, the most updated version of these additional tools can be downloaded as part of the BREW SDK 3.0 and in the future may be posted as a separate package for download.

What's new in the BREW SDK 3.0.1

The BREW Simulator has been enhanced to support the updated format of the device packs that are shipped in 3.0.1 BREW SDK.

What’s new in the BREW SDK 3.0.0

The BREW SDK 3.0.0 has been enhanced to be more user-friendly and to enable more efficient application development, testing, localization and customization.

BREW Simulator

Enhancements to the BREW Simulator (previously called the Emulator) are described below.

Device specific simulation

The BREW Simulator now supports more accurate simulation of device specific features. This simulation is based on real device data that is packaged as device packs. As they become available, these packs can be accessed by authenticated BREW developers.

Display of device information

The same data that is used to simulate device specific behavior is now displayed in an optional window in the Simulator.

Updated user interface

The User Interface of the BREW Simulator has been updated to make often used features more accessible for increased usability. In addition there is the capability to modify device data fields through the UI, such as heap size and maximum number of sockets.

BREW API Reference Guide enhanced

The BREW API Reference Guide has been enhanced to describe the version of BREW when interfaces and functions are first introduced.

BREW API Changes (3.1.4 to 3.1.5)

IAddrInfo

IAddrInfo provides host name resolving functionality, allowing the retrieval of IP addresses associated with a specified host name. It attempts to follow the spirit of RFC 3493 section 6.1 "Protocol-Independent Nodename and Service Name Translation". The main exception is that instead of getaddrinfo() and freeaddrinfo() APIs, IAddrInfo encapsulates a single resolve transaction and thus exposes a somewhat different API.

IAddrInfoCache

IAddrInfoCache provides access to the address info results cache and clears a specific host name or the whole cache.

IBCMCSDB

A new IBCMCSDB API, protected by privilege AEECLSID_BCMCSDATABASE, was added. IBCMCSDB allows an application to add records to the device BCMCS database of flow mapping information, which enables BCMCS client applications to register to the new flows. Upon handset reboot, all BCMCSDB updates are lost, and the BCMCSDB is populated by default values.

ICamera

ICamera is enhanced with the following camera and camcorder features:

- New control parameters
  - CAM_PARM_FRAME_TIMESTAMP: Timestamp encoding in movie frames
  - CAM_PARM_DEBLUR: Deblur

- CAM_PARM_LUMA_ADAPTATION: Luma adaptation
- CAM_PARM_FLASH_CTL: Strobe and LED flash control
- CAM_PARM_AF_INFO: Auto focus macro and auto metering modes

ICAMERA_SetVideoEncode() is enhanced to include CAM_ENCODE_UUID_LIST_ATOM_INFO.

ICAMERA_SetAudioEncode() is enhanced to include CameraAACEncodingInfo.

ICAMERA_RecordMoviePostcard() enables movie postcard recoding.

New CAM_EFFECT_types were added: CAM_EFFECT_POSTERIZE, CAM_EFFECT_WHITEBOARD, CAM_EFFECT_BLACKBOARD, CAM_EFFECT_AQUA

New CAM_PARM_AF_INFO parameters were added: Macro mode, CAM_AF_METER_CENTER_SPOT, CAM_AF_METER_CENTER_WEIGHTED, CAM_AF_METER_AVERAGE

IBitmap

The IBITMAP_CreateCompatibleBitmap() implementation for bitmaps created with IDISPLAY_CreateDIBitmap() and IDISPLAY_CreateDIBitampEx() has been fixed so that the nColorScheme field of the IDIB structure is propagated to the new bitmap. (CR number 43759)

ICipher1

In BREW 3.1.5, the ICipher interface, which was introduced in BREW 3.1.3, was renamed ICipher1. It is the same as was introduced in BREW 3.1.3, but was renamed to prevent confusion with an earlier ICipher interface. The functions in ICipher1 enable BREW developers to perform symmetric cipher encryption or decryption in a flexible manner without having to know the internals of the encryption algorithm or implementation. ICipher1 supports both stream and block ciphers.

IImage

Enhancements were made to the IImage interface. The inline function IIMAGE_ExtFunc() was added to AEEImage.h. This is implemented in terms of IIMAGE_SetParm() and defines a standard way to extend IImage while avoiding conflicts.

IMedia

- IMedia is enhanced with a new parameter call MM_PARM_NOTES, which allows applications to specify the number of simultaneously voiced synthesized notes.
- IMedia implementation for AMR-WB (wide-band) is added.
- IMedia implementation for PMD is enhanced with a frame callback mechanism.
- There is enhanced IMedia implementation for PCM to allow simultaneous playback of multiple PCM media.

ILocalStorage

ILocalStorage is an interface for generic storage. This interface enables the store and retrieve of opaque data buffers to an object.

INetUtils

INetUtils provides network related services that are not associated with a specific network instance. INETUTILS_GetDefaultNetwork() returns the current default network type (CDMA, UMTS, etc.) of the device. INETUTILS_GetPDPCount() and INETUTILS_GetPDPInfo() return information regarding the configured Packet Data Profiles on the device.

INetwork

The INetwork interface deprecates INetMgr. It provides an API to control a BREW applications data network. INetwork's functions provide mechanisms to get and set the parameters associated with the network subsystem of the device.

NOTE: An application must have a privilege level of Network or All to invoke the functions in this interface.

IDNSConfig2

The IDNSConfig2 Interface deprecates IDNSConfig and provides generic methods to configure an IDNS. An instance of IDNSConfig2 may only be obtained via IDNS_QueryInterface(), not via ISHELL_CreateInstance(). IDNSConfig2_SelectNetworkEx() is used to select a network (specific network type and address family) other than the default. The IDNSConfig2_SetServers() and IDNSConfig2_GetServers() are used to specify and to query DNS servers other than the default.

IQoS family of interfaces

The IQoS family of interfaces was enhanced in the following ways:

- Supports the CDMA Profile ID flow specification option
- Supports auxiliary flows in a QoS specification
- Change in behavior of QoS events – Event AEE_QOS_AVAILABLE _ EV is no longer generated. AEE_QOS_AVAILABLE_MODIFIED_EV is generated instead.
- Supports QoS events to indicate QoS aware/unaware system
- Enables an application to query the network for the current system type – CDMA QoS aware/unaware.
- Supports additional info codes to convey information about QoS events, upon handoff to CDMA QoS aware/unaware system.
- Supports a new error code AEE_NET_EQOSUNAWARE which is returned for QoS session operations generated on a CDMA QoS unaware system.

IQoSBundle

IQoSBundle provides methods for executing QoS operations on a bundle of QoS sessions. The following header file is required: AEEQoSBundle.h.

IRegistry

IRegistry is not a new interface, but is documented for the first time in BREW 3.1.5. IRegistry is used to get, set, and enumerate handlers. Currently, the only class that implements the IRegistry interface is AEECLSID_REGISTRY.

Handlers are registered when BREW is initialized. There is no guaranteed order of how handlers are registered, except that MIFs will be registered after the built-in handlers. IREGISTRY_GetHandler will always return the currently active handler. If IREGISTRY_Enum is called with bAll set to FALSE, then only the active handlers are enumerated. If bAll is set to TRUE, then all the handlers present in MIFs are enumerated as well.

ISMS

ISMS was enhanced to enable applications to do the following:

- Cancel delivery of submitted SMS messages on CDMA as well as GSM networks.
- Enquire about the status of a submitted message on GSM networks. Message status is delivered as a status report.
- Deliver a status report on a submitted message on GSM networks. On successful enabling of a status report on a submitted message, a message status is delivered as a status report irrespective of whether it was requested or not in the original message.
- Disable a status report on a submitted message on GSM networks. On successful disabling of a status report on a submitted message, a message status is not delivered irrespective of whether a status report was requested or not in the original message.
- Specify an absolute time after which a submitted message becomes invalid and should not be delivered.
- Specify a relative time duration after which a submitted message becomes invalid and should not be delivered.
- Specify an absolute time by which delivery of a submitted message must be deferred.

- Specify a relative time duration by which delivery of a submitted message must be deferred.

Certain networks use TP-OA to specify voice mail notification (VMN) information. This mechanism is referred to as the CPHS mechanism of VMN. ISMS interfaces support delivery of CPHS VMN messages.

IWeb

A new feature was added to support SSL tunneling over an HTTP proxy from the IWEB library. To use this functionality, applications only need to set the proxy for their HTTPS server via the WEBOPT_PROXYSPEC option. IWEB will establish an SSL connection on behalf of the application to the given proxy using the CONNECT method, which allows the connection to tunnel through to the final HTTPS server.   The application can manage sending data as normal, and IWEB will ensure that all data will be sent to the HTTPS server by tunneling it through the proxy. An example proxy specification can be specified (using WEBOPT_PROXYSPEC) as:

https:///http://mysslserver.mydomain.com/mypath/myapp?abc

As a separate feature, two new WEBOPT options have also been added : WEBOPT_REQUESTHOST and WEBOPT_REQUESTURI to allow applications to manage their own tunneled connections, for example, to ftp to SSH servers.   The connection management needs to be done entirely by the application itself in these cases.

Updates to existing interfaces and helper functions

The following functions have been added to the existing interfaces in BREW 3.1.5 since the BREW 3.1.4 release. For more information on any of the following additions, see the *BREW API Reference Online Help*.

ICAMERA interface

- ICAMERA_GetFlashCtl()
- ICAMERA_RecordMoviePostcard()
- ICAMERA_SetFlashCtl()

ICORENOTIFIER

- NMASK_CORE_AUTO_INSTALL
- NMASK_CORE_REG_APPS

IIMAGE interface

- IIMAGE_ExtFunc()

INETMANAGER interface

- INETMGR_GetLastNetDownReason()
    - AEE_NET_DOWN_REASON_NOT_SPECIFIED - The network was closed for an unspecified reason.
    - AEE_NET_DOWN_REASON_UMTS_REATTACH_REQ - A network request for a PDP context reactivation after a GGSN restart.
    - AEE_NET_DOWN_REASON_CLOSE_IN_PROGRESS - The network connection request was rejected as the previous connection of the same network is currently being closed.
    - AEE_NET_DOWN_REASON_NW_INITIATED_TERMINATION - The session was terminated by the network.

IPARAMETERS interface

- IParametersRO

IQOSFLOW interface

- AEEQOSFLOWOPT_CDMA_PROFILE_ID – QoS flow option for CDMA profile ID.

IQOSSESSION interface

- AEE_QOS_INFO_CODE_HANDOFF_TO_QOS_AWARE_SYSTEM – Info code which indicates a handoff to a QoS aware system.
- AEE_QOS_INFO_CODE_HANDOFF_TO_QOS_UNAWARE_SYSTEM - Info code which indicates a handoff to a QoS unaware system.
- IQOSSESSION_Open(), IQOSSESSION_Modify(), IQOSSESSION_GoActive() – In case of a CDMA QoS unaware system, these functions will return AEE_NET_EQOSUNAWARE.
- IQOSSESSION_SelectNetworkEx()

Helper Functions

- uint64struct_to_uint64()
- uint64struct_from_uint64()
- int64struct_to_int64()
- int64struct_from_int64()
- doublestruct_to_double()
- doublestruct_from_double()
- AEE_IS_ADDR_UNSPECIFIED()
- AEE_IS_ADDR_LOOPBACK()
- AEE_IS_ADDR_BREW_LOOPBACK()
- AEE_IS_ADDR_V4MAPPED()
- AEE_IN6_IS_ADDR_UNSPECIFIED()
- AEE_IN6_IS_ADDR_LOOPBACK()
- AEE_IN6_IS_ADDR_BREW_LOOPBACK()
- AEE_IN6_IS_ADDR_V4MAPPED()

- GETALSCONTEXT()

BREW API Changes (3.1.3 to 3.1.4)

IAlarmMgr

IAlarmMgr is the new interface introduced in BREW 3.1.4, which augments the alarm functionality provided by IShell in previous releases of BREW. Besides setting and clearing alarms, this interface provides capability to specify the specific user code to be used for an alarm. Also provided is the capability to reserve a set of user codes without starting an alarm for them specifically. The list of currently reserved alarms codes can also be queried.

IAlarmMgr's functions enable an application to be notified when the current time reaches a specified value. Unlike timers, which can be active only while an application is running, you can receive notification that an alarm has expired even when your application is not running. For all IAlarmMgr API, when an application needs to do any alarm operation on another classID (start alarm, cancel alarm, reserve, list or query alarm user codes), the caller needs to have the AEEGROUPID_ALARM_ALL privilege level. For suspending/resuming the BREW alarm services, the application needs the AEEGROUPID_ALARM_MGR privilege level. For reserving more than 30 user codes, the application needs to have the AEEGROUPID_RESERVECODE_LARGE privilege level.

To set an alarm, IALARMMGR_SetAlarm() is called and the number of seconds is specified from the current time at which the alarm notification is to occur or the absolute number of seconds and the BREW ClassID of the application (yours or another) that receives notification when the alarm time is reached. The 16-bit alarm user code can be specified by the user, or it can be generated by BREW. If the specified alarm user code is already used by an active alarm, the alarm is canceled and a new one started. At notification time, the IAPPLET_HandleEvent() function of the notified application is called with an EVT_ALARM event and the 16-bit alarm code as parameters. The latter parameter allows an application to distinguish between more than one simultaneously active alarm. If the notified application is not currently running, AEEShell creates an instance of it to process the notification event, after which it is terminated. The application may choose to activate itself if necessary.

The alarms are stored in the BREW database and continuously checked for alarm expirations while the BREW-enabled device is turned on. If an alarm’s expiration time passes while the device is turned off, the alarm notification is generated the next time the device is turned on. IALARMMGR_CancelAlarm() cancels a currently active alarm.

IALARMMGR_AlarmsActive() checks whether any of the BREW built-in annunciators (alarm clock, countdown timer, or stopwatch) are currently active for the specified class ID.

IALARMMGR_ListAlarmCodes() lists all the user codes reserved for the application regardless of whether or not that alarm is active. The function can also be used to query if the specified alarm user code is reserved or not. IALARMMGR_ReserveUserCodes() reserves the specified number of alarm codes for the application class ID specified without needing the application to actually set alarms. The unused alarm codes for an application are released when it is deleted. For reserving more than 30 user codes the application needs to have a privilege level of AEEGROUPID_RESERVECODE_LARGE.

ICamera

ICamera interface provides a generic way for BREW applications to control a device camera and to record snapshots and movies in various formats like JPEG, MPEG4, etc. ICamera is a privileged class so the class ID must be specified in the MIF.

IDLS

IDLS interface represents one Downloadable Sounds (DLS) media. IDLS enables BREW applications to load a DLS content, which can be applied to one or more supported media content (like MIDI). IDLSLinker interface allows a DLS object to be associated with an IMedia object, which represents a media content. One DLS object can be associated with multiple IMedia objects. The header file, AEEMediaDLS.h, is required.

IDLSLinker

IDLSLinker interface enables BREW applications to link DLS content(s) to an IMedia object which is representing a media content (like MIDI). The relationship between IDLS and IMedia is many to many.

IDLSLinker object pointer can be obtained by calling IMEDIA_QueryInterface() with the class ID AEECLSID_MEDIADLSLINKER.

You can create an IDLS object, load a DLS, and link it to multiple IMedia objects using the corresponding IDLSLinker. You can link, using IDLSLinker, an IMedia object to multiple IDLS objects. The header file, AEEMediaDLS.h, is required.

The header file, AEEDNS.h, is required.

IFlip

The IFlip interface enables an application to find out the number of flips present on a device and their unique IDs. Each flip on the device has a unique flip ID that identifies the flip. A flip ID of zero always denotes the primary flip, for example, the primary LCD screen of a clamshell device.

An application can also find out information about flips such as current flip position, displays and keys accessible to the user based on the current flip positions of all flips, and minimum and maximum positions each flip can take along the x,y,z axes.

IJoystick

IJoystick interface controls the device joystick. When created, the IJoystick interface enables the generation of joystick (X,Y) coordinates, which are received by the application as BREW joystick events. The IJoystick interface further provides the capability for controlling the sample rate and inactivity timeout configuration parameters to the joystick device.   The header file AEEJoystick.h, is required.

“Beta” IPv6

The following are part of the “Beta” IPv6 support and are subject to change when IPv6 is supported commercially.

- IDNSConfig2. The IDNSConfig2 Interface deprecates IDNSConfig and provides generic methods to configure an IDNS. An instance of IDNSConfig2 may only be obtained via IDNS_QueryInterface(), not via ISHELL_CreateInstance(). IDNSConfig2_SelectNetworkEx() is used to select a network (specific network type and address family) other than the default. The IDNSConfig2_SetServers() and IDNSConfig2_GetServers() are used to specify and to query DNS servers other than the default.

- INetwork. The INetwork interface deprecates INetMgr. It’s functions provide mechanisms to get and set the parameters associated with the network subsystem of the device. NOTE: An application must have a privilege level of Network or All to invoke the functions in this interface.

- New functions:
  - ISOCKPORT_SelectNetworkEx() – selects a specific data network.
  - INET_PTON, INET_NTOP(), Helper functions – generic functions for converting between string representation and binary representation of an IP address.
- New data structures:
  - AEE_IPPROTO_UNSPEC – unspecified protocol (system default) used in INETWORK_GetAddrInfo().
  - AEE_AF_UNSPEC, AEE_AFINET6 – unspecified address family and IPv6 address family used in SelectNetworkEx() and INETWORK_GetAddrInfo().
  - IPAddr – denotes a generic address of a socket or endpoint.
  - AEEAddrInfo, AEEAddrInfoResult - Address Info Flags - used by INETWORK_GetAddrInfo().
  - AEEDNSTYPE_AAAA

QoS

The QoS feature was enhanced and is now tested in the BREW 3.1.4 release. Error reporting was reworked and detailed error status is now available. Documentation was enhanced. The QoS API was changed so that IQOSSESSION_GetError() replaces IQOSSESSION_GetErrorSpec(), which was removed. IQoSErrorSpec was removed. The AEEQoSError data structure was added.

Updates to existing interfaces and helper functions

The following functions have been added to the existing interfaces in BREW 3.1.4 since the BREW 3.1.3 release. For more information on any of the following additions, see the *BREW SDK API Reference online help*.

ICAMERA interface

- CAM_PARM_PREVIEW_FPS – FPS control in preview mode
- CAM_PARM_EXPOSURE_METERING – Exposure metering
- CAM_PARM_ EXPOSURE_COMP – Exposure compensation

- CAM_PARM_ISO: Sensor sensitivity
- CAM_PARAM_APERTURE – Set aperture
- CAM_PARAM_SHUTTER_SPEED – Shutter speed

IMEDIASVG interface

- IMEDIASVG_EnableClientURLImageHandling()
- IMEDIA_SetMediaParm()
    - MM_PARM_AUDIOSYNC – Sets the audio sync information. Useful to perform audio-video (AV) sync.

IQOSSESSION interface

- IQOSSESSION_GetError()

IRSA interface

- IRSA_Encrypt – supports OAEP mode for encrypting. Use AEE_RSA_PKCS1_OAEP_SHA1_MGF1_PADDING when specifying the padding type. Supports PSS mode for creating a signature but using the private key. The length of the data MUST be 20 bytes, the size of a SHA-1 hash. Use AEE_RSA_PKCS1_PSS_SHA1_MGFI_PADDING when specifying the padding type. NOTE: Creating an RSA signature on a mobile device is very slow.
- IRSA_Decrypt – Supports OAEP mode for encrypting. Use AEE_RSA_PKCS1_OAEP_SHA1_MGFI_PADDING when specifying the padding type.
- IRSA_VerifySig – Supports PSS mode for verifying a signature. Use AEE_RSA_PKCS1_PSS_SHA1_MGFI_PADDING when specifying the padding type.

ISHELL interface

- ISHELL_GetDeviceInfo()
    - AEE_DEVICEITEM_HWID: This returns the primary HW ID for the device. Depending on the device, the primary ID may be an MEID, IMEI or ESN.
    - AEE_DEVICEITEM_HWIDLIST: This returns a list of all of the HW IDs defined for the device.
    - AEE_DEVICEITEM_MEIDS: This returns the MEID for the device in ASCII format.

ISOUND interface

- ISOUND_SetDevice() has been enhanced to accept BT stereo headset device type called:
    - AEE_SOUND_DEVICE_BT_STEREO_HEADSET

ISSL interface

- ISSL_NegotiateV()
- ISSL_SetPort()

Helper Functions

- GETRAND
- INET_NTOP
- INET_PTON
- SET_NOTIFIER_MASK_ERROR
- WSTRTOUTF8

BREW API Changes (3.1.2 to 3.1.3)

IAppHistory

AppHistory change notify (Callback) on StartApplet

This feature will send notifications when the application history is moved to top-visible upon the application restart. Currently, when an application is suspending and does not set ResumeData, its history entry is moved to top-visible when it is started again. However, notification about this AppHistory change is not sent to listeners of IAppHistory-IModel.

Notification when SetData is called on an application history entry

This feature sends notifications when application history is changed via IAPPHISTORY_SetData() or IAPPHISTORY_SetReason(). When new application history is created, BREW sends notifications to parties registered to receive such notifications. Previously, when application history data changes via IAPPHISTORY_SetData() or IAPPHISTORY_SetReason(), no notifications were being sent.

IAPPHISTORY_Bottom - Set IAppHistory to point to bottom entry

This feature adds a new interface IAPPHISTORY_Bottom(), which is symmetrical to IAPPHISTORY_Top(). This new function makes the bottom entry in the application history (the farthest from ITOPVISIBLE) the current entry.

IBacklight

The IBacklight interface support in BREW 3.1.3 enables applications to control various back lights available on a device. For controlling a back light, applications now create the IBacklight interface by specifying the back light Class ID associated with the back light.

ICamera

ICamera is enhanced with multiple sensor support. Each sensor is identified by a Class ID. The default is AEECLSID_CAMERA (which is AEECLSID_CAMERA1). The remaining sensors are identified using AEECLSID_CAMERA2, etc.

ICipher

The functions in ICipher enable BREW developers to perform symmetric cipher encryption or decryption in a flexible manner without having to know the internals of the encryption algorithm or implementation. This interface supports both stream and block ciphers. ICipher is different from most BREW classes in that it is created via an instance of ICipherFactory instead of ISHELL_CreateInstance. ICipher now supports DES and Triple DES, seed and AES.

ICipherFactory

The ICipherFactory is used to instanciate ICiphers rather than using CreateInstance() because of the number of options when creating the correct algorithm and modes for an ICipher. The factory selects from the available implementations, the correct algorithm, mode and padding features.

Each ICipher algorithm, for example, DES, AES, may have multiple implementations, depending on the platform. Ciphers may be implemented in software, by a DSP or dedicated hardware. Implementations are not required to support all of the possible chaining modes or padding styles. The factory (in conjunction with the ICipherWrapper classes) determines how to best assemble the cipher components to provide the requested functionality.

For each cipher algorithm there is an abstract “best fit” class ID, which the factory uses to select the best available implementation for the current platform (generally this implies the fastest implementation). This allows for development of platform independent code.

ICipherWrapper

ICipherWrapper supports the chaining modes and padding for block ciphers which do not directly support the modes. It is derived from ICipher. The ICipherFactory will create an instance when the algorithm implementation doesn't directly support the mode, and it will use it to wrap the algorithm.

IMediaSVG

IMediaSVG interface provides specialized methods to control SVG/SVGz playback.

IParameters

IParameters defines a generic mechanism for setting and getting data using type, length value. The IParametersRO interface is the read-only version - it simply disallows the SetParams function.

IPosDet

Enhanced information in the Position Response Structure

- IPosDet now provides the time difference in seconds between GPS time and UTC in the position response structure.
- IPosDet now provides the probability that user's actual position is within the described ellipse in the position response structure.
- IPosDet now provides information in the position response structure about the positioning method used to calculate the location.

IRawBlockCipher

IRawBlockCipher provides the interface used by ICipherFactory and ICipherWrapper to create a complete cipher implementation, and is intended as an interface by which implementors of ciphers (block ciphers in particular) make their implementation available. A caller or user of ciphers should use ICipher and ICipherFactory and not call IRawBlockCipher directly.

By implementing the IRawBlockCipher, the implementor can leave the buffering and padding to the ICipherWrapper and can implement just a subset of the chaining modes. The

ICipherFactory will query the cipher and attach the necessary ICipherWrapper to complete the implementation.

ISMS

Notification of storage changes

ISMSStorage now notifies its clients when a message is stored or deleted because it supports an IModel interface that can be queried using ISMSSTORAGE_QueryInterface() and can be used to register for notifications. CR number: 33546

ISMSMsg support for TPDU

ISMSMsg defines new message options, which can be queried by an application with system privilege to access raw TPDU for mobile terminated messages.

ISMS Msg_Sent event

ISMS now provides a means of letting its clients find the message reference number and Alpha ID corresponding to a mobile originated message. ISMS Msg_Sent notifies the application that the MO SMS is sent to the network and the content and address of the message might have been changed by WMS and the card.

ISMSStorageUpdateMsg()

ISMSStorage now provides a method to update parameters of a stored message in memory.

ISMS support for the TPDU option for MT SMS

ISMS now allows access to raw TPDU, for mobile terminated messages, to applications with system privilege. For this purpose, ISMSMsg defines new message options, which can be queried by an application with system privilege to access raw TPDU for mobile terminated messages.

Support for MT broadcast messages

ISMS now allows mobile terminated broadcast SMS messages.

ISockPort

ISockPort provides standard socket networking services, both stream and datagram. It provides methods to open and close, connect, transmit and receive data and more, over TCP and UDP sockets. ISockPort supports multiple address families, which is specified once during ISOCKPORT_OpenEx() and later by using the appropriate address structure. Currently, only the AEE_AF_INET address family is supported, but there will likely be more added in the future (e.g. IPv6).

IVocoder

New vocoder types for PCM and G.711 are added.

IWIFI

IWIFI is used by applications to retrieve and store WiFi configurations. Each WiFi configuration consists of 25 options (AEEWIFI_OPT_*) and is controlled by its profile ID (AEEWIFI_OPT_PROFILE_ID).

IWIFIOPTS

IWIFIOpts interface is used by an application to examine and change individual WiFi options in a configuration identified by its profile ID. The typical usage of these interfaces is as follows:

- Create an IWIFI and IWIFIOpts objects.
- Call IWIF_LoadOptions() to populate IWIFIOpts object from permanent storage.
- Call various IWIFIOpts functions to manipulate options values locally.
- Call IWIFI_CommitOptions() to save the option values in IWIFIOpts object.

HTML viewer control

The HTML viewer control is now transparent so that the background screen shows through. This also allows the control to be wrapped inside a Widget and used within the BUIT framework. Previously, the background was filled with the default user background color (typically white). CR number: 33536

The following additional IHTML Viewer CR fixes were made:

- CR30579 textbox change to input-mode with key entry
- CR30580 after edit, textbox focus move to next component
- CR30581 Multiple line Anchor focus
- CR30583 pulldown menu item with wide length characters display
- CR30584 request: marquee tag support
- CR30585 input tag accesskey support
- CR30586 body tag bgcolor/background support
- CR30587 image automatic size change with image tag width/height
- CR30588 Off screen anchor can get focus too.
- CR30589 max number of tag nest extension
- CR30590 input-tag "type" attribute "image" support
- CR11999 hyperlink spanning multiple lines gets focus

Updates to existing interfaces and helper functions

The following functions have been added to the existing interfaces in BREW 3.1.3 since the BREW 3.1.2 release. For more information on any of the following additions, see the *BREW SDK API Reference online help*.

IAPPHISTORY interface

- IAPPHISTORY_Bottom()

IAPPLETCTL interface

- IAPPLETCTL_RunningApplet()

ICAMERA interface

- ICAMERA_SetBitRate()
- ICAMERA_GetParm/SetParm() now handles the following parameters:
  - CAM_PARM_FOCUS, CAM_PARM_FOCUS_RECT: Allows snapshot focus and setting focus rectangle
  - CAM_PARM_FADE: Allows fade in/out effect for movie recording
  - CAM_PARM_BITRATE: Sets the encoding bit-rate for video recording
  - CAM_PARM_SENSOR_INFO: Gives the camera sensor information

IFILEMGR interface

- IFILEMGR_GetFreeSpaceEx()

IHEAP interface

- IHEAP_GetModuleMemStats()

IMEDIA interface

- IMEDIA_SetMediaParm()/IMEDIA_GetMediaParm() now supports the following:
  - MM_PARM_PLAY_TYPE: Allows the user to specify the media play type which can be a ringer, alert, reminder, etc.

IMODEL interface

- IMODEL_Notify()

ISHELL interface

- ISHELL_GetDeviceInfo() is enhanced to support following device items:
  - AEE_DEVICEITEM_CAMERA_INFO: List of supported camera CLSIDs (Null-terminated)
  - AEE_DEVICEITEM_POSDET_PRIVACY_ALERT: boolean to return if Privacy Alert is enabled
- ISHELL_GetDeviceInfoEx() enhances support for AEE_DEVICEITEM_CAMERA_INFO.

ISMSSTORAGE interface

- ISMSSTORAGE_UpdateMsg()

ITEXTCTL interface

- ITEXTCTL_GetPropertiesEx()
- ITEXTCTL_GetSelection()
- ITEXTCTL_SetPropertiesEx()
- ITEXTCTL_SetSelection()

Helper Function

MACROS

- AEE_IS_PATCH_PRESENT

IAPPHISTORY interface

- IAPPHISTORY_Bottom()

BREW API Changes (3.1.1 to 3.1.2)

IBattery

The IBattery interface gives applications the ability to query battery and charger-related information.

IBatteryNotifier

IBatteryNotifier is the notifier class with which applications can register for battery notifications in the following ways:

1. Using the -application MIF
2. Using ISHELL_RegisterNotify()

Applications are not required and can't create an instance of IBatteryNotifier.

IForceFeed

This interface is used for feeding data to an object. The image decoders are examples of classes that use this interface. (See IImageDecoder.) The IForceFeed interface may be queried from an object with AEEIID_FORCEFEED. The following header file is required: IForceFeed.h

IImageDecoder

This interface is used to retrieve an IBitmap from an image decoder. Typically, classes that export the IImageDecoder interface will also export another interface that is used to feed encoded data to the decoder. Currently, the IForceFeed interface exists for this purpose.

The IImageDecoder interface may be queried from an object with AEEIID_IMAGEDECODER. The following header file is required: IImageDecoder.h

IDeviceNotifier

IDeviceNotifier is intended for background and suspended applications to receive device-level notifications, such as flip events and key guard events. It is inherited from INotifier and follows the same usage guidelines with the masks defined in AEEDeviceNotifier.h.

INetMTPDNotifier

INetMTPDNotifier is intended to allow registering for Mobile Terminated packet data notifications. It is inherited from INotifier and follows the same usage guidelines.

IQoSErrorSpec

This interface is part of the error reporting mechanism that allows the client to identify which options in a requested QoS Spec are invalid. Objects of IQoSErrorSpec cannot be created by a client but are rather obtained from the IQOSSESSION_GetErrorSpec() function. QoS Error specification is composed of four lists of specs:

- List of Error Rx Flow spec
- List of Error Tx Flow spec
- List of Error Rx Filter spec
- List of Error Tx Filter spec

IQoSErrorSpec provides Get accessors to each one of the above four lists.
The following header file is required: AEEQoSSession.h

IQoSFilter

IQoSFilter interface encapsulates a Quality of Service IP Filter specification.
The IP filter spec, when applied to a stream of IP packets, segregates a specific IP flow which is characterized by the filter spec parameters, such as IPaddress and TCP/UDP ports.
An IQoSFilter interface instance is obtained via the ISHELL_CreateInstance() mechanism.

IQoSFlow

IQoSFlow interface encapsulates a Quality of Service Flow specification where by the flow spec defines the treatment required for an IP flow, for example, QoS parameters such as required bit rate, etc. An IQoSFlow interface instance is obtained via the ISHELL_CreateInstance() mechanism.

IQoSList

This interface represents a generic single linked list. The following header file is required: AEEQoSList.h

IQoSSession

This interface represents a QoS link (secondary PDP context in UMTS). The following header file is required: AEEQoSSession.h

IQoSSpec

This interface represents a complete QoS specification. QoS specification is composed of four lists of specs:

- List of Rx Flow spec
- List of Tx Flow spec
- List of Rx Filter spec
- List of Tx Filter spec

The two lists of flow specs contain IQoSFlow objects and specify the treatment required for an IP flow for the Rx direction and for the Tx direction respectively. The order of IQoSFlow objects in these lists is significant. The first object is the most suitable flow for the client, (req_flow) while the last object is the minimum required flow by the client (min_req_flow). If only one object exists in the list, it is handled as the req_flow.

The two lists of filter specs contain IQoSFilter objects and specify the IP flow itself for the Rx direction and for the Tx direction respectively. The order of IQoSFilter objects in these lists is insignificant.

IQoSSpec provides Set and Get accessors to each one of the above four lists. In addition, IQOSSPEC_LoadFlowSpecByProfile () is used to populate the object with Rx and Tx flow specs from a pre-configured profile.

The IQoSSpec is created without any spec lists. When setting any of the spec lists (through the Set methods), IQoSSpec doesn't copy the given list but rather stores a reference to it. The given IQoSList object replaces the stored one and IQoSSpec takes responsibility for it. This means that the stored IQoSList is released while the given IQoSList is AddReferenced.

The following header file is required: AEEQoSSession.h

IResArbiter

The IResArbiter is an OEM/Carrier customizable extension, which allows custom arbitration decisions for transferring ownership of a resource.

For resources that implement Resource Management, the IRESARBITER_ConfirmAcquire function is called when there is contention for a resource, for example, one entity owns the resource and a second entity is requesting to use it. Examples for these resources include ICAMERA.

ITopVisibleCtl

The ITopVisibleCtl interface is used to control the ownership of a resource, which must be limited to a single owner at a time. Each managed resource implements a version of this interface.

Updates to existing interfaces and helper functions

The following functions have been added to the existing interfaces in BREW 3.1.2 since the BREW 3.1.1 release. For more information on any of the following additions, see the *BREW SDK API Reference online help*.

IAPPHISTORY interface

- IAPPHISTORY_GetReason()
- IAPPHISTORY_Move()
- IAPPHISTORY_SetReason()

IAPPLETCTL interface

- IAPPLETCTL_BrowseFile()
- IAPPLETCTL_BrowseURL()

ISHELL interface

- ISHELL_PostURL()
- ISHELL_SendURL()

ITELEPHONE interface

- ITELEPHONE_GetLineInfo()

IMEDIAUTIL interface

- IMEDIAUTIL_CreateMediaEx()

INETMGR interface

- INETMGR_SetDormancyTimeout()

IPHONECTL interface

- IPHONECTL_EnableLineSwitch()
- IPHONECTL_SelectLine()

BREW API Changes (3.1.0 to 3.1.1)

This section lists the API changes made since the last BREW feature release (BREW 3.1.0).

ITapi and ISMS interfaces

ITapi and ISMS are enhanced to support notification of mobile terminated WAP push messages.

Random number generator

Documentation has been added for AEECLSID_RANDOM, which can be used as a source of cryptographic quality random data from a FIPS186 compliant random number generator. This class ID was available in BREW releases starting with BREW 2.0.2, but was not documented. See AEERandomSource.h for more information.

IAPPLETCTL Changes

The condition in which starting a suspended applet destroyed the applet history forward link has been fixed. Previously, when StartApplet was called for suspending an applet, it broke the applet history forward link between the previous applet and the started applet. (CR 29173)

IAPPLECTL_Stop().The implementation of IAPPLECTL_Stop() was fixed so that it allows the closing of the currently running applet. (CR 28646,30537) The behavior now is:

- The applet must receive an EVT_APP_STOP (regardless of the number of entries this applet has on the history list).
- None of the history entries for the applet must be removed except the top-visible history entry if the top-visible entry matches the cis for which this function is called.

APPLETCTL_CanStart now returns an integer value of zero, which indicates SUCCESS, instead of a boolean. In release 3.1.0, it returns a boolean, which if TRUE (non-zero), indicates the application can start. This was published as a known issue in prior BREW releases, and is now fixed in BREW 3.1.1. (CR 28342)

BREW now cleans the AppHistory list when a request application fails to start. Previously, BREW did not clean the AppHistory list when encountering this condition, resulting in expected behavior and errors. (CR 28974)

New suspend reason code AEE_SUSPEND_CLOSE has been added to notify the application as to whether it is closing itself or starting another application. See the BREW API Reference entry for AEESuspendReason for more information. (CR 28978)

BREW now updates the start arguments in the history entry and sends the arguments when the application is resumed. Previously, arguments were not sent to an application if it was already running when ISHELL_StartAppletArgs was called. In BREW 3.1.0, this resulted in a condition in which the PEK OATTAPI ReceivedSMS.3 test did not behave as expected. This was published as a known issue in the BREW 3.1.0 release, and is now fixed in BREW 3.1.1. (CR 29405,28214)

IGRAPHICS changes

IGRAPHICS_StretchBlt now draws to the destination bitmap set by IGRAPHICS_SetDestination. Previously, IGRAPHICS_StretchBLT called IDISPLAY_BitBlt and drew to the screen instead of drawing to the destination bitmap set by IGRAPHICS_SetDestination. (CR16497)

Position location

A new failure code AEEGPS_ERR_STALE_BS_INFO has been added. Handsets failing to obtain position due to base station information being stale will be able to identify the condition with this failure status code. Whenever a handoff occurs while the mobile device is in a traffic call, it does not update the new BS information until the call is terminated. Under the scenario in which serving BS information is incorrect, a position request will fail. This condition is most common when a traffic call (such as a circuit switched data call or QNC) is up for an extended period of time before requesting a fix. (CR 29354)

The ability to provide a location server IP address to IPOSDET_SetGPSConfig() is now restricted. This operation is allowed for the applications that have access to

AEECLSID_LOCATIONSERVER in the dependencies list in their MIF. Applications that do not have access to AEECLSID_LOCATIONSERVER, configure a non-default server type to IPOSDET_SetGPSConfig() and will fail with EPRIVLEVEL. (CR 29267)

Resource Arbiter Changes

Resource arbiter return value propagation. ResArbiter errors and other errors are now propagated to the caller for each of the following (CR 28331):

- APPLETCTL_CanStart
- ISHELL_StartApplet()
- IAPPLETCTL_Start
- NMASK_SHELL_START_STATUS.

The condition in which Resource Arbiter always gets the system AEECLSID_SHELL as the requestor and owner has been fixed. Now, the current application is the requestor and the current owner is the owner. (CR 28693)

IRingerMgr enhancement

IRingerMgr is enhanced to allow the installation and setting as current, ringer multimedia content of any format. (CR 28756)

Critical fixes in IShell

Calls to supervisor code while in user mode in IShell_Reset() were fixed. BRIDLE enabled devices will no longer go into Prefetch abort upon a call to ISHELL_Reset(). (CR 29367)

A condition that could result in a data abort when AEE_Exit() is called if there were any supervisor callbacks scheduled was fixed. (CR 29751)

Critical low memory handler. BREW will not attempt to close active applets if one or more applications are registered for AEE_SCB_LOW_RAM_CRITICAL. Affected file is AEEShell.c. (CR 28518)

SMS changes

The following configuration items were added in BREW 3.1.1:

- CFGI_SMS_MO_ON_ACCESS_CHANNEL: boolean, if TRUE, and an SMS message is sent on the access channel.
- CFGI_SMS_MO_ON_TRAFFIC_CHANNEL: boolean, if TRUE, and an SMS message is sent on the traffic channel. If both CFGI_SMS_MO_ON_ACCESS_CHANNEL and CFGI_SMS_MO_ON_TRAFFIC_CHANNEL are TRUE, an SMS message is first sent on the access channel, and on being too large for the access channel, is sent on the traffic channel.

ITapi and ISMS are enhanced to support notification of mobile terminated WAP push messages. In BREW 3.0.3, the ITAPI implementation does not support WAP push messages. This was published as a known issue in the BREW 3.0.3 release, and is now fixed in BREW 3.1.1. (CR 29936, 31163)

Updates to existing interfaces and helper functions

The following functions have been added to the existing interfaces in BREW 3.1.1 since the BREW 3.1.0 release. No new helper functions were added in BREW 3.1.1. For more information on any of the following additions, see the *BREW SDK API Reference online help*.

ICALLHISTORY interface

- ICALLHISTORY_Notify()

ICAMERA interface

- ICAMERA_SetMaxFileSize()

IMEDIA interface

- IMEDIA_SetMediaDataEx()

ISMS interface

- ISMSBCCONFIG_GetAllServiceOpts()
- ISMSMSG_GetOptWithIndex()

IVIEWER interface

- IVIEWER_DrawOffscreen()
- IVIEWER_SetAnimationRate()
- IVIEWER_SetDisplay()

- IVIEWER_SetDrawSize()
- IVIEWER_SetFrameCount()
- IVIEWER_SetFrameSize()
- IVIEWER_SetOffset()

Documentation was added for the following interfaces in BREW 3.1.1.

IDIB interface

- IDIB_FlushPalette()
- IDIB_TO_IBITMAP()

IIMAGE interface

- IIMAGE_DrawOffscreen()

ILOGGER interface

- ILOGGER_PutRecord()

IPORT interface

- IPORT_AddRef()
- IPORT_QueryInterface()
- IPORT_Release

ISOURCE interface

- ISOURCE_QueryInterface()

BREW API Changes (3.0.1 to 3.1.0)

This section lists the API changes made in the current version (BREW 3.1.0).

IAPPLETCTL interface

The IAPPLETCTL interface is used to control the running applications. This includes getting the list of running applications, getting specific information on a running application, and starting an application and stopping an application.

IAppHistory interface

The functions in the IAppHistory interface allow the caller to interact with the application history list. The list consists of the visible and suspended applications. The APIs allow the caller to:

- Navigate through entries in the history list
- Obtain the ClassID of the history entry
- Set or retrieve data stored by the system when the application is suspended
- Close the application history entry
- Insert a new entry into the history

This interface is obtained by calling ISHELL_CreateInstance. A new instance of IAppHistory is initialized by pointing to the top-most history entry (that of the currently top-visible application), unless the history list is empty.

ICALL interface

This interface provides call-related services. These methods alter the state and resources of the device. These methods if used incorrectly may render the device unusable. Access to this interface goes through a privilege check. Applications using this interface must explicitly specify the use of this object in the access control list.

This interface also implements IModel. An instance of IModel can be obtained with Query for AEEIID_MODEL.

ICallMgr interface

This interface provides services that affect telephone calls in the system. These methods alter the state and resources of phone. These methods if used incorrectly may render the device unusable. Access to this interface goes through a privilege check. Applications using this interface must explicitly specify the use of this object in the access control list.

ICallOrigOpts interface

This interface provides a way of building origination options. These options can be passed to ICALLMGR_OriginateEx().

IFIFO interface

The IFIFO interface provides a lightweight form of InterProcess Communication (IPC).

A FIFO is simply a kernel buffer with a name, to which multiple readers and/or writers may simultaneously attach. The name space is a null terminated character string beginning with "fifo:/", which is typically "well known" to allow unrelated applications to attach to the same FIFO.

As its name implies, a FIFO provides first-in, first-out IPC. Write operations place bytes sequentially into the kernel buffer, until it is full. Read operations return the same bytes in the same order, until the buffer is empty.

When the buffer is empty, a read operation will return IPORT_END if there are no writers, else it will return IPORT_WAIT. To avoid processing an IPORT_END, a server may open a FIFO in read/write mode even when it is only interested in reading.

If there are multiple readers (writers), the read (write) operations will be interleaved as they are scheduled by the operating environment, so some other form of cooperative synchronization may be needed.

Access control lists

Modules may "publish" or export FIFOs in their home directories. As of BREW 3.0.1, BREW looks for the RESTYPE_BINARY resource IDB_MIF_FIFO_ACLS in each module's MIF to resolve which FIFOs are published with what permissions. See IFILEMGR_CheckPathAccess() for a complete description of ACL syntax.

Here are BREW's FIFO ACLs (which are considered relative to "fifo:/"): const char *cpszzFIFOACL =

"0x00000000 = / r:/sys" "\0" // all groups to read under fifo:/sys "0x00000000 = /:/sys/priv" "\0" // no access to fifo:/sys/priv

"0x00000000 = / rw:/shared" "\0" // all groups to read/write under fifo:/shared

IModel interface

IModel is the interface for a model in the Model-View-Controller triad. A model consists of state data along with a notification mechanism to inform its client of state changes.

The model interface provides object reference counting mechanisms that allow objects to manage their own memory instances. In addition, the interface provides APIs for registering a listener with a model, as well as interfaces for manipulating the data represented by a model.

IMultipartyCall interface

This interface supplements the services provided by ICall and offers making a multiparty call. This must be created as QueryInterface on the ICall object referred herein as primary ICall object. The designated primary ICall object plays the master for all the calls added. The calls added into a multiparty call are referred as secondary calls for clarity.

IPhoneCtl interface

This interface provides methods that alter the phone behavior. These methods if not used correctly may render the device unusable. Each method in this interface checks for the application's access privileges.

IResourceCtl interface

The IResourceCtl interface is used to control the ownership of a resource that must be limited to a single owner at a time. Each managed resource implements a version of this interface. From a managed resource, you can use IQI_QueryInterface.

ISMS, ISMSNotifier, ISMSMsg and ISMSStorage interface

ISMS, ISMSNotifier, ISMSMsg, and ISMSStorage interfaces are introduced in BREW 3.1.0.

- ISMS: Enables applications to send/receive SMS messages.
- ISMSNotifier: Enables applications to listen for various types of SMS messages.
- ISMSMsg: Provides an abstract representation of an SMS message.
- ISMSStorage: Enables applications to store SMS messages and templates in RUIM/SIM.

See the BREW SDK API Reference online help for more information.

ISuppsTrans interface

This interface provides services to send non-call related supplementary services. A new instance can be obtained from IPHONECTL_GetSuppTrans().

ITelephone interface

This interface provides services to read the phone's state and information. This interface does not require any special privileges.

This interface also supports IModel. To obtain IModel use ITELEPONE_QueryInterface() with AEEIID_MODEL. A listener added to this model will be notified of all the calls, phone, serving system, inband messages and events.

PhoneNotifier interface

This interface provides asynchronous notifications. Applications registered with one or more notification mask AEET_NMASK_* will be notified with EVT_ NOTIFY.

Updates to existing interfaces, helper functions, and datatypes

The following functions have been added to the existing interfaces in BREW 3.1.0 since the BREW 3.0.1 release. New helper functions and datatypes are also listed. For more information on any of the following additions, please refer to the BREW API Reference shipped with the BREW SDK 3.1.0.

APIs

IADDRBOOK interface

- IADDRBOOK_EnumRecInitEx()

ICALLHISTORY interface

- ICALLHISTORY_ClearEntry()
- ICALLHISTORY_EnumInitByCallType()
- ICALLHISTORY_QueryInterface()

IHASH interface

- IHASH_Restart()
- IHASH_SetKey()
- IHASHCTX_Update()

INETMGR interface

- INETMGR_GetDefaultNetwork()
- INETMGR_GetUMTSCount()
- INETMGR_GetUMTSInfo
- INETMGR_SelectNetwork()

ISHELL interface

- ISHELL_AppIsInGroup()

- ISHELL_CloseAppletClass()
- ISHELL_GetDeviceInfo() is enhanced to support following device items:
    - AEE_DEVICEITEM_SOFTKEY_COUNT: used to query how many softkeys the device has.
    - AEE_DEVICEITEM_IMEI: used to retrieve GSM device ID.
    - AEE_DEVICEITEM_ADS: used to retrieve the Application Download Server (ADS) hostname.
    - AEE_DEVICEITEM_RUIMID: returns the RUIM ID for this client.
    - AEE_DEVICEITEM_KEY_SUPPORT: used to query if a specified key is supported or not.
- ISHELL_OnLowRAMCritical()
- ISHELL_Reset
- ISHELL_SetAppPrefs()
- ISHELL_StartBackgroundApplet()
- ISHELL_SuspendAlarms()

Helper Functions

- FABS()
- FCOS()
- FGETFLT_MAX()
- FGETFLT_MIN()
- FGETHUGE_VAL()
- FLTTOINT()
- FSIN()
- FTAN()
- IDIB_FlushPalette()
- LISTENER_Cancel()
- LISTENER_Init()
- LISTENER_IsRegistered()
- TRUNC()
- UTRUNC()

BREW API Changes (3.0.0 to 3.0.1)

This section lists the API changes made in the current version (BREW 3.0.1).

Call Manager airlink mode indicator

Applications interested in using this feature should register for NMASK_TAPI_STATUS notifications. This is supported through enhancements to the ITAPI interface. An additional members bHDRMode has been added to the TAPIStatus structure. It will be set to 1 if the air link is in HDR(or 1xEv) mode.

Airplane mode Indicator

Applications interested in using this feature should register for NMASK_TAPI_STATUS notifications. This is supported through enhancements to the ITAPI interface. An additional member's bAirplaneMode has been added to the TAPIStatus structure. It will be set to 1 when the RF receiver and transmitter on the phone are turned off.

MDN access

For accessing the device MDN, an application must first specify AEECLSID_MDN as a dependency in its MIF using the dependencies tab of MIF Editor.
ISHELL_GetDeviceInfoEx() has been enhanced to support access to the MDN (AEE_DEVICEITEM_MDN).

Manner mode access

To access a user's manner mode settings, a BREW application can invoke SHELL_GetDeviceInfoEx() with AEE_DEVICEITEM_MANNER_MODE value.

Updates to existing interfaces, helper functions and datatypes

The following functions have been added to the existing interfaces in BREW 3.0.1 since the BREW 3.0.0 release. New helper functions and datatypes are also listed. For more information on any of the following additions, please refer to the BREW API Reference shipped with the BREW SDK 3.0.1.

APIs

- ICoreNotifier
- IHASH interface
  - IHASH_QueryInterface()
  - IHASH_SetKey()
- IHASHCTX interface
  - IHASHCTX_SetKey()
- ISHELL interface
  - ISHELL_GetDeviceInfoEx()

Helper functions

- AEECallback_Connect()
- AEECallback_Readable()
- AEECallback_SetTimer()
- AEECallback_SocketWait()
- AEECallback_Writeable()

Data types

- AEEAsyncError
- AEEDeviceItem
- AEEEvent
- AEESuspendInfo
- AEESysError
- HmacData
- AEE_MD5_HMAC_CTX
- AEE_MD2_HMAC_CTX
- AEE_SHA1_HMAC_CTX
- AEE_HASH_CTX
- MPROP_USESSTRING
- NMASK_CORE_APP_RECALL
- NMASK_CORE_DL_STATUS
- NMASK_SHELL_SYS_ERROR

BREW API Changes (2.1.0 to 3.0.0)

This section lists the API changes made in the 3.0.0 version of BREW.

Case sensitive file access

BREW 3.0.0 supports case-sensitive file access. Please refer to documentation for fs: and IFILEMGR in the BREW API Reference online help.

IAClockCtl interface

The IAClockCtl interface was added to allow the caller to create and display an analog clock on the display. In addition to the standard IControl functions, the clock can be:

- Set to display a specific time value.
- Set to automatically update with the current time.
- Sized to fit within a specified rectangle boundary.
- Placed on top of an image.
- Set to display any combination of hours, minutes, and second hands.

IBitmapDev interface

The IBitmapDev interface was added for use in various functions relevant to device bitmaps. It is only implemented for device bitmaps and is obtained by calling a bitmap's QueryInterface method with an interface ID of AEEIID_BITMAPDEV.

IPORT interface

The IPORT interface was added to provide an abstract bi-directional data stream providing Read/Readable, Write/Writeable, GetLastError and IOCtl interfaces.

ITHREAD interface

The IThread interface was added as a cooperatively-scheduled thread that layers over BREW's callback-based APIs. It exports methods to start and stop threads and the methods necessary to implement blocking APIs given an arbitrary callback-based API.

IThreads are not re-useable. For example, _Start() may only be called once. After a thread returns from its start function (after _Exit() or after _Stop() is called), _Start() may not called again.

IWEBENG interface

The IWebEng interface was added as an abstract base class that allows dynamic BREW modules to extend the functionality of IWeb by registering as a protocol handler either through ISHELL_RegisterHandler(), or by adding an entry in the module's MIF's list of extensions. It is an abstract base class.

Updates to existing interfaces, helper functions and datatypes

The following functions have been added to the existing interfaces in BREW 3.0.0 since the BREW 2.1.0 release. New helper functions and datatypes are also listed. For more information on any of the following additions, please refer to the BREW API Reference shipped with the BREW SDK 3.0.0.

IDISPLAY interface

- IDISPLAY_CreateDIBitmapEx()
- IDISPLAY_IsEnabled()
- IDISPLAY_MakeDefault()
- IDISPLAY_NotifyEnable()
- IDISPLAY_SetPrefs()

IFILEMGR interface

- IFILEMGR_CheckPathAccess()

- IFILEMGR_ResolvePath()

IMEDIA interface

- IMEDIA_EnableChannelShare()
- IMEDIA_EnableFrameCallback()
- IMEDIA_GetFrame()
- IMEDIA_IsChannelShare()
- IMEDIA_IsFrameCallback()
- IMEDIA_SeekFrame()

IMENUCTL interface

- IMENUCTL_AddOwnerDrawItem()
- IMENUCTL_GetColors()
- IMENUCTL_GetItemRect()
- IMENUCTL_GetStyle()
- IMENUCTL_SetOwnerDrawCB()

IPOSDET interface

- IPOSDET_ExtractPositionInfo()

ISHELL interface

- ISHELL_ClearShutdownErrors
- ISHELL_GetProperty
- ISHELL_RegisterEvent
- ISHELL_SetProperty

ISOCKET interface

- ISOCKET_GetOpt()
- ISOCKET_SetOpt()

Helper functions

- BASENAME()
- DBGEVENT()
- DBGEVENT_EX()
- GETCHTYPE()
- GETFSFREE()
- GETLASTFPERROR()
- GETUTCSECONDS()
- IDIB_TO_IBITMAP()
- ISBADREADPTR()
- ISBADWRITEPTR()
- LOG_TEXT()
- LOG_TEXT_EX()

- MAKEPATH()
- RESBLOB_DATA
- SPLITPATH()
- STRIBEGINS()

Datatypes etc

- AEEAppUsage
- AEEGROUPID_ANY_SID
- AEELinger
- AEEMIMETypes
- AEENetUsage
- AEEODCBData
- AEEPositionInfoEx
- AEESockOpt
- AEE_BREW_LOOPBACK
- AEE_INADDR_ANY
- AEE_INADDR_LOOPBACK
- MPROP_NETUSAGE
- MPROP_STATUS
- MPROP_USAGE
- NMASK_ALSEEP
- NMASK_CLOSED
- NMASK_IDLE
- NMASK_OPENED
- NMASK_SHELL_APP_EXPIRED
- NMASK_SHELL_APP_LICENSE_CHANGE
D
- NMASK_SHELL_INIT
- NMASK_SHELL_MOD_LIST_CHANGED
- NMASK_SHELL_START_STATUS
- NotifyStartStatus
- PFNTHREAD
- RESBLOB_DATA

BREW SDK changes (3.1.4 to 3.1.5)

The BREW SDK 3.1.5 has been enhanced to support the features described below.

BREW Simulator

- AppMgr v3.0.5 is built into the SDK 3.1.5 Simulator.
- Files with certain extensions are not to be used in calculating the file system space used by the Simulator. See the SDK User Docs for the exact extensions. Users can modify the entries with the filename extensions to be excluded from counting towards file system space usage on the Simulator. See the [FileSystem] section in BREW_Emu.dat to modify or configure the filename extensions.
- The Simulator is enhanced to provide the following SMS support:
  - Differentiation between store and update deferred delivery
  - Store/Read/Delete support for templates
  - Store/Read/Delete support for status reports
  - Support for multiple transactions (CR number 42127)

BREW Use Case Scenarios documentation

To enhance the clarity and details of BREW interfaces documentation, QUALCOMM is providing BREW Interfaces Use Case Scenarios documentation as part of the BREW API Reference Guide.

The use case scenarios documentation presents a series of sections covering use case scenarios of selected interfaces. The documentation includes description, call flows, and code snippets. The objective is to clearly show how APIs are used in these interfaces. The target audience is application developers who make use of the interface for their applications.

The interfaces covered are ICamera, IMedia, INetMgr, IPosDet, ISockPort, ITelephone.

Device Packs added

Two new device packs have been added to BREW SDK 3.1.5 - DevicePack3 and DevicePack4.

- DevicePack3 – A new device skin which supports 16-bit and 18-bit color depths, 176x220 pixel screen resolution and 4MB (4194304 bytes) default heap size was added.
- DevicePack4 – A new device skin which supports 16-bit and 18-bit color depths, QVGA (240x320) screen resolution and ~4MB (4096000 bytes) default heap size was added.

BREW SDK changes (3.1.3 to 3.1.4)

The BREW SDK 3.1.3 has been enhanced to support the features described below.

BREW Simulator

Following are enhancements to the BREW Simulator.

Enhancements to Camera simulation

Camera simulation has been enhanced to support:

- MPEG4 encoding of type ISO/IEC 14496-2
- Video encoding of the MPEG4 types mp4 and 3g2.
- Audio encoding of the types QCELP 13k Fixed Full and QCELP 13k Fixed Half

Joystick simulation

Simulator creates a pop up window with an image of a joystick to simulate joystick movements based on mouse movements. For example, when the mouse is moved within this pop up window, EVT_JOYSTICK_POS events are generated.

Generic extensions and applets folder

Simulator is capable of loading BREW extensions and applications from a generic folder. BREW extensions or applets installed in this folder will be available to the Simulator even when the Applet or MIF directory is changed.

BREW SDK changes (3.1.2 to 3.1.3)

The BREW SDK 3.1.2 has been enhanced to support the features described below.

BREW Simulator

Enhancements to the BREW Simulator are described below.

PCM Streaming

- Simulator is capable of streaming PCM media files (WAV format) using ISource as an input to IMedia

RSSI level simulation

- Simulator is now capable of displaying and simulating RSSI levels. It allows one to manipulate the RSSI settings as well.

Backlight simulation

- IBacklight interface is supported on Simulator. It allows one to control the backlight settings for primary display

Notifications for headset plug-in and removal

- Simulator sends appropriate notifications when user clicks on the headset icon. Headset icon toggles between headset plug-in and removed states.

BREW SDK changes (3.1.0 to 3.1.2)

The BREW SDK 3.1.2 has been enhanced to support the features described below.

BREW Simulator

Enhancements to the BREW Simulator are described below.

SMS simulation

SMS simulation enhancements include the following.

- Looping back an SMS message sent from the device back to the device -
  Prior to BREW 3.1.2, SMS messages could be sent from the device and sent to the device, but these were two separate operations. In this version of Simulator, a message can be sent from the Simulator to the Simulator's phone number, which causes the message to be looped back to the Simulator to provide an end-to-end SMS simulation feature.

- ISMSStorage simulation - Applications can use functions available in ISMSStorage to simulate storing messages on a device. The storage media is simulated through the windows file system. This feature is currently simulated only for WMS based CDMA and WMS based GSM systems.

- SMS simulation takes into account the underlying network that is set in the simulator through the Network Settings dialog.

Camera simulation

The BREW Simulator now simulates the ICamera interface when a DirectShow compliant camera is connected to the system. The main camera simulation features supported are

- Previewing frames

- Recording a snapshot

Battery simulation

The BREW Simulator now simulates the IBattery interface. It allows one to configure various parameters and test the application for low battery conditions.

Trapping Misaligned Data Exception

The BREW Simulator is now capable of trapping misaligned data exceptions in BREW applications. Just as RISC processors support data misalignment exception, BREW Simulator is now capable of generating the same exception (on the Windows platform).

EVT_FLIP and EVT_KEYGUARD simulation

An improved support for simulating flip (EVT_FLIP) and keyguard (EVT_KEYGUARD) has been added. It now support generating the appropriate IDeviceNotifier notifications.

Airplane Mode simulation

Support for Airplane mode simulation has been added to the BREW Simulator. In Airplane mode, the RF of the device is turned off and hence there is no network connectivity.

HDR Mode simulation

Checking the “Use 1x EV-DO HDR” in Network Settings dialog in Simulator enables high data rate simulation.

BREW SDK changes (3.0.1 to 3.1.0)

The BREW SDK 3.1.0 has enhanced support for simulation of TAPI and SMS. These enhancements in conjunction with the new APIs introduced in BREW 3.1.0, are intended to enable the development of an entire Phone UI using the BREW SDK.

BREW Simulator

Enhancements to the BREW Simulator are described below. All the interfaces mentioned in the BREW API Changes (3.0.1 to 3.1.0) section are simulated with the exception of IPhoneCtl, IMultiPartyCall, IsuppsTrans, and ISMSStorage.

SMS simulation

SMS simulation enhancements are as follows:

- Sending an SMS to a device:
  Prior to BREW 3.1.0, only BREW-directed SMS simulation was available. This has been significantly enhanced to allow one to configure various parameters of an SMS message format before simulating the sending of an SMS to the device (Simulator).

- Sending an SMS from the device (application)
  Applications can use functions available in ISMS or ITAPI to send an SMS. The BREW Simulator creates a *.sim file based on the current date and time and stores the contents of the message in this file.

Multiple SMS implementations are supported in the Simulator (WMS, UASMS). Implementations of WMS based GSM SMS simulation, WMS based CDMA SMS simulation, and UASMS based CDMA SMS simulation is supported.

TAPI simulation

The TAPI Emulation dialog has been enhanced to display the Call Descriptor, Calling Line ID, Call Duration, Call Status, and Call Type. TAPI emulation dynamically uses the underlying network that is set in the Simulator (Using Network Settings).

EVT_FLIP simulation

Clicking on the icon available in the Simulator’s toolbar sends an EVT_FLIP event to the active application. The wParam value specified in the EVT_FLIP event toggles between TRUE and FALSE on successive clicks to indicate flip open and flip close respectively.

BREW SDK changes (3.0.0 to 3.0.1)

Various problems that were found in 3.0.0 version of SDK components have been fixed in this release. Following are the most noticeable changes.

Device pack

The device pack format has been updated to be more flexible in accommodating device specific features. Device packs shipped with 3.0.1 SDK obsolete the device packs shipped with 3.0.0 version of SDK.

BREW Simulator

- The BREW Simulator has been modified to recognize the updated formats of the 3.0.1 version device packs. Once the BREW SDK 3.0.1 is installed, the BREW Simulator will no longer be able to load the device packs shipped with the BREW SDK 3.0.0.

- Enhancements to File System Simulation
  The device packs contain information about the maximum space and maximum number of files allowed for BREW apps.

  The BREW Simulator simulates the device file system space and number of files using the Applet Directory.

  File Exclusion:
  On the simulation platform, there are a number of files (generated or otherwise) that are never used on the actual device. For example, when you build an application in the MS Visual Studio environment, several files specific to Visual Studio (such as dll, .ncb, and .opt) get created but are not necessary on the device.

  To exclude unnecessary files, the BREW Simulator allows you to configure the list of extensions to be excluded by the BREW Simulator while performing file system simulation.

To Configure the exclusion list

1. Open ($BREWDIR)\bin\BREW_Emu.dat file in a text editor such as Notepad.
2. Add or delete your extensions to or from the ExcludeExts1, ExcludeExts2, or ExcludeExts3 lists. Make sure extensions are comma separated with no spaces.
3. Save the file.
4. Restart.

- The BREW Simulator now remembers the state, location, and size of the properties bar.

- A problem that made The Grinder® inoperable with BREW Simulator 3.0.0 has been fixed in 3.0.1 version of the BREW Simulator.

BREW SDK Changes (2.1.0 to 3.0.0)

This section lists the BREW SDK changes made in the 3.0.0 version of BREW.

Device pack

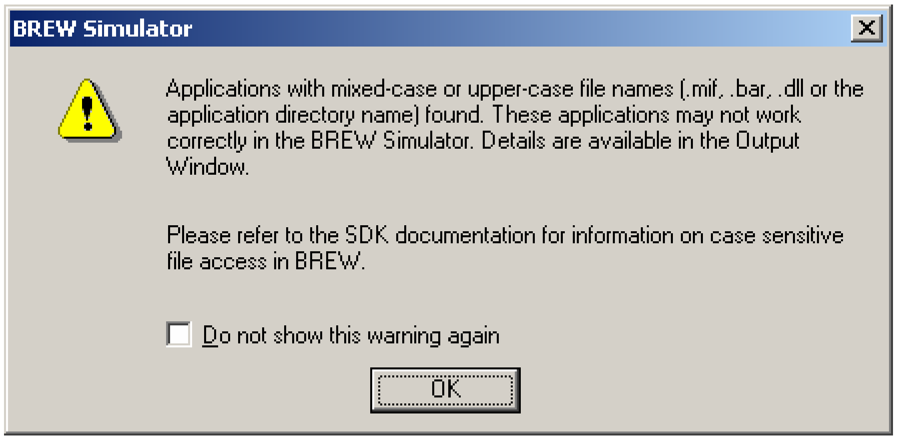
Device packs contain all the information required by the BREW Simulator to simulate device specific behavior in a set of files bundled together. A specification file included in the device pack contains all the details of a device that a BREW developer might be interested in. Refer to the BREW SDK User documentation for more details.

BREW Simulator

An updated BREW simulation environment capable of device specific simulation is now available with BREW SDK 3.0.0. Some of the updates are as follows.

- For the currently loaded device, the Simulator displays a list of device details in a new properties pane. Prior to 3.0.0 release, this information was made available in a Device Data Sheet (Excel spreadsheet) on the BREW developer extranet along with the skins.
- The Simulator allows modifying some of the device information such as heap size and number of sockets.
- MMC1 simulation is available in Simulator provided the device and device pack support the feature.
- Some of the commonly executed operations in the Simulator are available through a toolbar and properties pane.
- The Simulator allows you to specify the DNS servers to use when resolving host names.
- The following Simulator warning lets you know if the current applet or MIF directory contains upper-case or mixed-case file names (.dll, .mif, .bar or the application directory). This is done to simulate behavior available on all BREW

devices. Refer to Accessing Files or APIs in an EFS in the BREW Programming Concepts for more information.


BREW Simulator

Applications with mixed-case or upper-case file names (.mif, .bar, .dll or the application directory name) found. These applications may not work correctly in the BREW Simulator. Details are available in the Output Window.

Please refer to the SDK documentation for information on case sensitive file access in BREW.

☐ Do not show this warning again

OK

Known Issues

The list of known issues for the BREW SDK 3.1.5 can be accessed through the BREW Developer Extranet.

Installation Instructions

Minimum system requirements

To install the BREW SDK 3.1.5 software, you must have a PC with Windows 2000 or XP and Internet Explorer version 6.0 or later. The full software install uses about 24 megabytes of disk space.

NOTE: To run the installation on a Windows 2000 or Windows XP machine, you must have Administrator privileges because Windows 2000 and Windows XP do not permit installations to run without Administrator rights.

Installing the BREW SDK

The BREW SDK has a web-based installer. Content and disk requirements are determined during the installation process.

To install the BREW SDK

1. Download the BREW SDK from the BREW web site.
2. Accept the license agreement and follow the instructions on the screen.

   BREW SDK 3.1.5 software installs.

Uninstalling BREW SDK

To uninstall this version of the BREW SDK

1. Select **Start > Settings > Control Panel > Add/Remove Programs**.
2. Select **BREW SDK v3.1.5** installed program.
3. Select **Change/Remove**.

4. Select **Remove** and follow the instructions on the screen.

   BREW SDK uninstalls.