BeOS R4
Programmer's Cheatsheets

by David Orr

IDD, 209 Brom Bones Lane, Longwood, Florida 32750
<http://www.pobox.com/~idd/index.html>
idd@pobox.com

How to Print:

The Cheatsheets are intended to be printed and bound in a specific way for the easiest viewing. It is recommended that you place the pages in page protectors bound in a 3-ring binder. This page should be printed on the front of the first sheet (with the binder holes on the left side of the page). The next page should be printed on the back of the same sheet, with the next page on the front of the second sheet, etc. If you can't print on both sides of a sheet of paper, you can print out everything normally, and then place every two pages together (back to back) in the same sheet protector.

If your printer can't print close to the margins, you will see that the edges of the pages won't print correctly. To fix this problem you need to select the "Shrink to Fit" option from the print dialog box before printing.

Tips:

- Classes and structures do not include private or obsolete member objects.
- In most cases, for classes that were inherited from other Be classes, class member objects and functions that were originally defined in a parent class are not listed in the child class, even for functions that are redefined in a child class to modify the function's behavior.
- Brackets ("[" and "]") are used to show that some of a function's parameters may be omitted (because the function has been overloaded with another function that doesn't require certain parameters). Brackets may be nested. For example: "*foo(int a, [,int b [, int c]])*" means that you can call the function as *foo(a)*, *foo(a, b)*, or *foo(a,b,c)*
- Please submit corrections and comments to the author at <idd@pobox.com>.

License:

Contents:

Kernel Kit

<be/KernelKit.h> <libroot.so> ***BeOS R4 Programmer's Cheatsheets by David Orr***

Note: All Kernel functions and objects are defined as extern "C" type.

Typedefs (all are int32) <be/kernel/OS.h>	
area_id	port_id
sem_id	thread_id
team_id	image_id

THREADS

Thread Priorities <be/kernel/OS.h>
B_LOW_PRIORITY (5)
B_NORMAL_PRIORITY (10)
B_DISPLAY_PRIORITY (15)
B_URGENT_DISPLAY_PRIORITY (20)
B_REAL_TIME_DISPLAY_PRIORITY (100)
B_URGENT_PRIORITY (110)
B_REAL_TIME_PRIORITY (120)

Thread Hook Function Typdef <be/kernel/OS.h>

typedef int32 (***thread_func**) (void *);

typedef enum 'thread_state'	<be/kernel/OS.h>
B_THREAD_RUNNING	B_THREAD_ASLEEP
B_THREAD_READY	B_THREAD_SUSPENDED
B_THREAD_RECEIVING	B_THREAD_WAITING

typedef struct **thread_info** <be/kernel/OS.h>

thread_id **thread**;
team_id **team**;
char **name**[B_OS_NAME_LENGTH];
thread_state **state**;
int32 **priority**;
sem_id **sem**;
bigtime_t **user_time**, **kernel_time**;
void ***stack_base**, ***stack_end**;

Thread Functions <be/kernel/OS.H>

thread_id **spawn_thread**(thread_func function_name, const char *thread_name, int32 priority, void *arg);
thread_id **find_thread**(const char *name);
status_t **kill_thread**(thread_id thread);
status_t **resume_thread**(thread_id thread);
status_t **suspend_thread**(thread_id thread);
status_t **rename_thread**(thread_id thread, const char *new_name);
status_t **set_thread_priority**(thread_id thread, int32 new_priority);
void **exit_thread**(status_t status);
status_t **wait_for_thread** (thread_id thread, status_t *thread_return_value);
status_t **get_thread_info**(thread, info
status_t **get_next_thread_info**(tmid, cookie, info);
status_t **send_data**(thread_id thread, int32 code, const void *buf, size_t buffer_size);
status_t **receive_data**(thread_id *sender, void *buf, size_t buffer_size);
bool **has_data**(thread_id thread);

status_t **snooze**(bigtime_t microseconds);
status_t **snooze_until**(bigtime_t time, int timebase); *(the only currently defined timebase is **B_SYSTEM_TIMEBASE**)*

SCHEDULER

suggest_thread_priority() Bit Flags <be/kernel/scheduler.h>
B_DEFAULT_MEDIA_PRIORITY (0)
B_OFFLINE_PROCESSING
B_STATUS_RENDERING
B_USER_INPUT_HANDLING
B_LIVE_VIDEO_MANIPULATION
B_VIDEO_PLAYBACK
B_VIDEO_RECORDING
B_LIVE_AUDIO_MANIPULATION
B_AUDIO_PLAYBACK
B_AUDIO_RECORDING
B_LIVE_3D_RENDERING
B_NUMBER_CRUNCHING

Scheduling Functions <be/kernel/scheduler.h>

Note: Parameter defaults apply to C++ onlly.

int32 **suggest_thread_priority**(uint32 what = B_DEFAULT_MEDIA_PRIORITY, int32 period = 0, bigtime_t jitter = 0, bigtime_t length = 0);
bigtime_t **estimate_max_scheduling_latency**(thread_id th = -1); *(Note: Default is the current thread.)*

TEAMS

System Team ID <be/kernel/OS.h>
B_SYSTEM_TEAM (2)

typedef struct **team_info** <be/kernel/OS.h>

team_id **team**;
int32 **image_count**, **thread_count**, **area_count**;
thread_id **debugger_nub_thread**;
port_id **debugger_nub_port**;
int32 **argc**;
char **args**[64]; *(Note: Abreviated command line args.)*
uid_t **uid**;
gid_t **gid**;

Team Functions <be/kernel/OS.h>

status_t **kill_team**(team_id team); *(see also: send_signal())*
status_t **get_team_info**(team, info;
status_t **get_next_team_info**(cookie, info;

AREAS

Area Locking Codes <be/kernel/OS.h>	
B_NO_LOCK (0)	B_CONTIGUOUS
B_LAZY_LOCK	B_LOMEM
B_FULL_LOCK	

Area Codes <be/kernel/OS.h>	
B_ANY_ADDRESS (0)	B_CLONE_ADDRESS
B_EXACT_ADDRESS	B_ANY_KERNEL_ADDRESS
B_BASE_ADDRESS	

Area Permission Codes <be/kernel/OS.h>
B_READ_AREA
B_WRITE_AREA

typedef struct **area_info** <be/kernel/OS.h>

area_id **area**;
char **name**[B_OS_NAME_LENGTH];
size_t **size**;
uint32 **lock**;
uint32 **protection**;
uint32 **ram_size**;
uint32 **copy_count**;
uint32 **in_count**, **out_count**;
team_id **team**;
void ***address**;

Area Functions <be/kernel/OS.h>

area_id **create_area**(const char *name, void **start_addr, uint32 addr_spec, size_t size, uint32 lock, uint32 protection);
area_id **clone_area**(const char *name, void **dest_addr, uint32 addr_spec, uint32 protection, area_id source);
area_id **find_area**(const char *name);
area_id **area_for**(void *addr);
status_t **delete_area**(area_id id);
status_t **resize_area**(area_id id, size_t new_size);
status_t **set_area_protection**(area_id id, uint32 new_protection);
status_t **get_area_info**(id, ainfo)
status_t **get_next_area_info**(team, cookie, ainfo)

IMAGES

typedef enum 'image_type' <kernel/image.h>	
B_APP_IMAGE	B_ADD_ON_IMAGE
B_LIBRARY_IMAGE	B_SYSTEM_IMAGE

typedef struct **image_info** <be/kernel/image.h>

image_id **id**;
image_type **type**;
int32 **sequence**, **init_order**;
B_PFV **init_routine**, **term_routine**;
dev_t **device**;
ino_t **node**;
char **name**[MAXPATHLEN];
void ***text**, ***data**;
int32 **text_size**, **data_size**;

Image Functions <be/kernel/image.h>

thread_id **load_image**(int32 argc, const char **argv, const char **envp);
image_id **load_add_on**(const char *path);
status_t **unload_add_on**(image_id imid);
status_t **get_image_info**(image, info);
status_t **get_next_image_info**(team, cookie, info);

Add-on Symbol Type Codes <be/kernel/image.h>
B_SYMBOL_TYPE_DATA
B_SYMBOL_TYPE_TEXT
B_SYMBOL_TYPE_ANY

Add-on Symbol Functions <be/kernel/image.h>

status_t **get_image_symbol**(image_id imid, const char *name, int32 sclass, void **ptr);
status_t **get_nth_image_symbol**(image_id imid, int32 index, char *buf, int32 *bufsize, int32 *sclass, void **ptr);

Kernel Kit

<be/KernelKit.h> <libroot.so>

PORTS

typedef struct **port_info** <be/kernel/OS.h>

port_id **port**;
team_id **team**;
char **name**[B_OS_NAME_LENGTH];
int32 **capacity**, **queue_count**, **total_count**;

Port Functions <be/kernel/OS.h>

port_id **create_port**(int32 capacity, const char *name);
port_id **find_port**(const char *name);
status_t **write_port**(port_id port, int32 code, const void *buf, size_t buf_size);
status_t **write_port_etc**(port_id port, int32 code, const void *buf, size_t buf_size, uint32 flags, bigtime_t timeout);
status_t **read_port**(port_id port, int32 *code, void *buf, size_t buf_size);
status_t **read_port_etc**(port_id port, int32 *code, void *buf, size_t buf_size, uint32 flags, bigtime_t timeout);
ssize_t **port_buffer_size**(port_id port);
ssize_t **port_buffer_size_etc**(port_id port, uint32 flags, bigtime_t timeout);
ssize_t **port_count**(port_id port);
status_t **set_port_owner**(port_id port, team_id team);
status_t **delete_port**(port_id port);
status_t **get_port_info**(port, info);
status_t **get_next_port_info**(team, cookie, info);

SEMAPHORES

Semaphore Control Flags <be/kernel/OS.h>
B_CAN_INTERRUPT
B_DO_NOT_RESCHEDULE
B_CHECK_PERMISSION
B_TIMEOUT

typedef struct **sem_info** <be/kernel/OS.h>

sem_id **sem**;
team_id **team**;
char **name**[B_OS_NAME_LENGTH];
int32 **count**;
thread_id **latest_holder**;

Semaphore Functions <be/kernel/OS.h>

sem_id **create_sem**(int32 count, const char *name);
status_t **delete_sem**(sem_id sem);
status_t **acquire_sem**(sem_id sem);
status_t **acquire_sem_etc**(sem_id sem, int32 count, uint32 flags, bigtime_t microsecond_timeout);
status_t **release_sem**(sem_id sem);
status_t **release_sem_etc**(sem_id sem, int32 count, uint32 flags);
status_t **get_sem_count**(sem_id sem, int32 *count);
status_t **set_sem_owner**(sem_id sem, team_id team);
status_t **get_sem_info**(sem, info);
status_t **get_next_sem_info**(team, cookie, info);

Defined Lengths <be/kernel/OS.h>
B_OS_NAME_LENGTH (32)
B_PAGE_SIZE (4096)
B_INFINITE_TIMEOUT (92233720368547758O7LL)

CACHE MANIPULATION

Cache Manipulation Bit Flags <be/kernel/image.h>
B_FLUSH_DCACHE
B_FLUSH_ICACHE
B_INVALIDATE_DCACHE
B_INVALIDATE_ICACHE

Cache Manipulation Functions <be/kernel/image.h>

void **clear_caches**(void *addr, size_t len, uint32 flags);

SYSTEM INFORMATION

typedef enum 'cpu_types' <be/kernel/OS.h>

B_CPU_PPC_601	B_CPU_AMD_X86
B_CPU_PPC_603	B_CPU_AMD_K5_MODEL0
B_CPU_PPC_603e	B_CPU_AMD_K5_MODEL1
B_CPU_PPC_604	B_CPU_AMD_K5_MODEL2
B_CPU_PPC_604e	B_CPU_AMD_K5_MODEL3
B_CPU_PPC_750	B_CPU_AMD_K6_MODEL6
B_CPU_PPC_686	B_CPU_AMD_K6_MODEL7
B_CPU_AMD_29K	B_CPU_AMD_K6_MODEL8
B_CPU_X86	B_CPU_AMD_K6_MODEL9
B_CPU_MC6502	B_CPU_CYRIX_X86
B_CPU_Z80	B_CPU_CYRIX_GXm
B_CPU_ALPHA	B_CPU_CYRIX_6x86MX
B_CPU_MIPS	B_CPU_AMD_X86
B_CPU_HPPA	B_CPU_AMD_K5_MODEL0
B_CPU_M68K	B_CPU_AMD_K5_MODEL1
B_CPU_ARM	B_CPU_AMD_K5_MODEL2
B_CPU_SH	B_CPU_AMD_K5_MODEL3
B_CPU_SPARC	B_CPU_AMD_K6_MODEL6
B_CPU_CYRIX_X86	B_CPU_AMD_K6_MODEL7
B_CPU_CYRIX_GXm	B_CPU_AMD_K6_MODEL8
B_CPU_CYRIX_6x86MX	B_CPU_AMD_K6_MODEL9
B_CPU_INTEL_X86	B_CPU_INTEL_PENTIUM
B_CPU_INTEL_PENTIUM_MMX_MODEL_8	B_CPU_INTEL_PENTIUM75_486_OVERDRIVE
B_CPU_INTEL_PENTIUM_MMX	B_CPU_INTEL_PENTIUM_PRO
B_CPU_INTEL_PENTIUM75	B_CPU_INTEL_PENTIUM_II
B_CPU_INTEL_PENTIUM_486_OVERDRIVE	B_CPU_INTEL_PENTIUM_II_MODEL_3
B_CPU_INTEL_PENTIUM_MMX_MODEL_4	B_CPU_INTEL_PENTIUM_II_MODEL_5
B_CPU_INTEL_PENTIUM_MMX	B_CPU_INTEL_CELERON

CPU Vendor Mask <be/kernel/OS.h>
B_CPU_X86_VENDOR_MASK (0x1F00)

CPU Constants <be/kernel/OS.h>
B_MAX_CPU_COUNT (8)

typedef enum 'platform_type' <be/kernel/OS.h>
B_BEBOX_PLATFORM
B_MAC_PLATFORM
B_AT_CLONE_PLATFORM
B_ENIAC_PLATFORM
B_APPLE_II_PLATFORM
B_CRAY_PLATFORM
B_LISA_PLATFORM
B_TI_994A_PLATFORM
B_TIMEX_SINCLAIR_PLATFORM
B_ORAC_1_PLATFORM
B_HAL_PLATFORM
B_BESM_6_PLATFORM
B_MK_61_PLATFORM
B_NINTENDO_64_PLATFORM

typedef struct **cpu_info** <be/kernel/OS.h>

bigtime_t **active_time**; *(Note: In microseconds.)*

typedef int32 **machine_id**[2];

typedef struct **system_info** <be/kernel/OS.h>

machine_id **id**;
bigtime_t **boot_time**; *(number of usec since 1/1/70)*
int32 **cpu_count**;
enum cpu_types **cpu_type**;
int32 **cpu_revision**;
cpu_info **cpu_infos**[B_MAX_CPU_COUNT];
int64 **cpu_clock_speed**, **bus_clock_speed**
enum platform_types **platform_type**;
int32 **max_pages**, **used_pages**, **page_faults**, **max_sems**, **used_sems**, **max_ports**, **used_ports**, **max_threads**, **used_threads**, **max_teams**, **used_teams**
char **kernel_name** [B_FILE_NAME_LENGTH], **kernel_build_date**[B_OS_NAME_LENGTH, **kernel_build_time**[B_OS_NAME_LENGTH
int64 **kernel_version**;

System Info Functions <be/kernel/OS.h>

status_t **get_system_info**(info)
int 32 **is_computer_on**(void)
double **is_computer_on_fire**(void)

MANUFACTURER INFO (Intel Only)

typedef union **cpuid_info** <be/kernel/OS.h>

struct **eax_0**
    uint32 **max_eax**;
    char **vendorid**[12];
struct **eax_1**
    uint32 **stepping** : 4, **model** : 4, **family** : 4, **type** : 2;
    uint32 **features**;
struct **regs**
    uint32 **eax**, **ebx**, **edx**, **ecx**;

CPU ID Functions <be/kernel/OS.h>

status_t **get_cpuid**(cpuid_info* info, uint32 eax_register);

Support Kit Tools

<be/SupportKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

StopWatch Utility

Defines a handy code-timing debug tool.

class BStopWatch <be/support/StopWatch.h>

```
BStopWatch(const char *name, bool silent = false);
virtual ~BStopWatch();

void Suspend();
void Resume();
bigtime_t Lap();
bigtime_t ElapsedTime() const;
void Reset();
const char *Name() const;
```

BList

Provides storage for pointers. BList does not provide any thread-safe locking mechanisms.

class BList <be/support/List.h>

```
BList(int32 itemsPerBlock = 20);
BList(const BList&);
virtual ~BList();

BList &operator=(const BList &from);

Adding and removing items...

bool AddItem(void *item);
bool AddItem(void *item, int32 atIndex);
bool AddList(BList *newItems);
bool AddList(BList *newItems, int32 atIndex);
bool RemoveItem(void *item);
void *RemoveItem(int32 index);
bool RemoveItems(int32 index, int32 count);
bool ReplaceItem(int32 index, void *newItem);
void MakeEmpty();

Reordering items...

void SortItems(int (*cmp)(const void *, const void *));
bool SwapItems(int32 indexA, int32 indexB);
bool MoveItem(int32 fromIndex, int32 toIndex);

Retrieving items...

void *ItemAt(int32) const;
void *ItemAtFast(int32) const;
void *FirstItem() const;
void *LastItem() const;
void *Items() const;

Querying the list...

bool HasItem(void *item) const;
int32 IndexOf(void *item) const;
int32 CountItems() const;
bool IsEmpty() const;

Iterating over the list...

void DoForEach(bool (*func)(void *));
void DoForEach(bool (*func)(void *, void *), void *);
```

BBlockCache

A simple fixed-size block caching mechanism.

Block Allocation Codes <be/support/BlockCache.h>
B_OBJECT_CACHE
B_MALLOC_CACHE

class BBlockCache <be/support/BlockCache.h>

```
BBlockCache(size_t cache_size,size_t block_size, uint32
   type);
virtual ~BBlockCache();

void *Get(size_t block_size);
void Save(void *pointer, size_t block_size);
```

I/O Classes

Pure virtual BDataIO and BPositioIO classes provide the protocol for Read()/Write()/Seek(). Inherited by: BMallocIO, BMemoryIO, and BFile (Storage Kit).

class BDataIO *(pure virtual)* <be/support/DataIO.h>

```
BDataIO();
virtual ~BDataIO();

virtual ssize_t Read(void *buffer, size_t size) = 0;
virtual ssize_t Write(const void *buffer, size_t size) =0;
```

class BPositionIO *(pure virtual)* <be/support/DataIO.h>

```
   : public BDataIO

BPositionIO();
virtual ~BPositionIO();

virtual ssize_t Read(void *buffer, size_t size);
virtual ssize_t Write(const void *buffer, size_t size);

virtual ssize_t ReadAt(off_t pos, void *buffer, size_t size) = 0;
virtual ssize_t WriteAt(off_t pos, const void *buffer, size_t size)
   = 0;

virtual off_t Seek(off_t position, uint32 seek_mode) = 0;
virtual off_t Position() const = 0;
virtual status_t SetSize(off_t size);
```

class BBufferIO <be/support/BufferIO.h>

```
   : public BPositionIO

enum { DEFAULT_BUF_SIZE = 65536L };

BBufferIO(BPositionIO * stream, size_t buf_size =
   DEFAULT_BUF_SIZE, bool owns_stream = true);
virtual ~BBufferIO();

virtual ssize_t ReadAt(off_t pos, void *buffer, size_t size);
virtual ssize_t WriteAt(off_t pos, const void *buffer, size_t
   size);
virtual off_t Seek(off_t position, uint32 seek_mode);
virtual off_t Position() const;
virtual status_t SetSize(off_t size);
virtual status_t Flush();
BPositionIO * Stream() const;
size_t BufferSize() const;
bool OwnsStream() const;
void SetOwnsStream(bool owns_stream);
void PrintToStream() const;
```

class BMallocIO <be/support/DataIO.h>

```
   : public BPositionIO

BMallocIO();
virtual ~BMallocIO();

virtual ssize_t ReadAt(off_t pos, void *buffer, size_t size);
virtual ssize_t WriteAt(off_t pos, const void *buffer, size_t
   size);
virtual off_t Seek(off_t pos, uint32 seek_mode);
virtual off_t Position() const;
virtual status_t SetSize(off_t size);

void SetBlockSize(size_t blocksize);
const void *Buffer() const;
size_t BufferLength() const;
```

class BMemoryIO <be/support/DataIO.h>

```
   : public BPositionIO

BMemoryIO([const] void *p, size_t len);
virtual ~BMemoryIO();

virtual ssize_t ReadAt(off_t pos, void *buffer, size_t size);
virtual ssize_t WriteAt(off_t pos, const void *buffer, size_t
   size);
virtual off_t Seek(off_t pos, uint32 seek_mode);
virtual off_t Position() const;
virtual status_t SetSize(off_t size);
```

Disassembly

Functions to dissasseble code. Currently for Intel only. *These functions are currently undocumented and may be subject to change.*

Disasm Flags <be/devel/disasm.h>
DISASM_FLAG_OP_SIZE_16
DISASM_FLAG_ADDR_SIZE_16
DISASM_FLAG_INTEL_STYLE
DISASM_FLAG_RELATIVE_ADDRESSES

Disasm Global Function <be/devel/disasm.h>

```
status_t disasm(uchar *in, uint32 insize, char *out, uint32
   outsize, uint32 eip, uint32 flags,
      status_t (*lookup)(void *cookie, uint32 eip, uint32
   *sym_addr, char *sym_name, int max_name_len, int
   is_lower),
      void *cookie);
```

Demangle

Function to interpret the name of a C++ object.

Demangle Global Function
<be/devel/Unmangle.h>

```
Declared as extern "C" type...
int demangle(const char *mangled_name,char
   *unmangled_name, size_t buffersize);
```

Demangled Sizes <be/devel/Unmangle.h>
UNAME_SIZE (512)

Support Kit Tools

<be/SupportKit.h> <libbe.so>

BString

A string class supporting common string operations.

class **BString** <be/support/String.h>

BString();
BString(const char *);
BString(const BString &);
~**BString**();

Access...

const char ***String**() const;
int32 **Length**() const;
int32 **CountChars**() const; *(Note: UTF8 characters in string.)*

Comparative operators...

bool operator**<**(const BString &) const;
bool operator**>**(const BString &) const;
bool operator**<=**(const BString &) const;
bool operator**>=**(const BString &) const;
bool operator**==**(const BString &) const;
bool operator**!=**(const BString &) const;

bool operator**<**(const char *) const;
bool operator**>**(const char *) const;
bool operator**<=**(const char *) const;
bool operator**>=**(const char *) const;
bool operator**==**(const char *) const;
bool operator**!=**(const char *) const;

Assignment...

BString &operator**=(**const BString &);
BString &operator**=(**const char *);
BString &operator**=(**char);

BString &**SetTo**(char * [,int32 length]);
BString &**SetTo**(const BString &from [,int32 length]);
BString &**SetTo**(char, int32 count);

BString &**Adopt**(BString &from [,int32 length]);

Substring copying...

Note: Returns <into> ref as it's result, doesn't do anything if <into> is <this>.
BString &**CopyInto**(BString &into, int32 fromOffset,int32 length) const;

Note: Caller guarantees that <into> is large enough.
void **CopyInto**(char *into, int32 fromOffset,int32 length) const;

Appending...

BString &operator**+=(**const BString &);
BString &operator**+=(**const char *);
BString &operator**+=(**char);

BString &**Append**(const BString & [,int32 length]);
BString &**Append**(const char * [,int32 length]);
BString &**Append**(char, int32 count);

Prepending...

BString &**Prepend**(const char * [,int32]);
BString &**Prepend**(const BString & [,int32]);
BString &**Prepend**(char, int32 count);

Inserting...

BString &**Insert**(const char * [,int32 length] [,int32 pos]);
BString &**Insert**(const char *, int32 fromOffset, int32 length, int32 pos);
BString &**Insert**(const BString &, int32 fromOffset, int32 length, int32 pos);
BString &**Insert**(char, int32 count, int32 pos);

Removing...

BString &**Truncate**(int32 newLength, bool lazy = true);
BString &**Remove**(int32 from, int32 length);

BString &**RemoveFirst/Last/All**(const BString &);
BString &**RemoveFirst/Last/All**(const char *);

BString &**RemoveSet**(const char *setOfCharsToRemove);

Moving...
Note: Caller guarantees that <into> is large enough.
BString &**MoveInto**(BString &into, int32 from, int32 length);
void **MoveInto**(char *into, int32 from, int32 length);

strcmp-style compare functions...

int **Compare**(const BString & [,int32 n]) const;
int **Compare**(const char * [,int32 n]) const;

int **ICompare**(const BString & [,int32 n]) const;
int **ICompare**(const char * [,int32 n]) const;

Searching...

int32 **FindFirst/Last**(const BString & [,int32 fromOffset]) const;
int32 **FindFirst/Last**(const char * [,int32 fromOffset]) const;
int32 **FindFirst/Last**(char [,int32 fromOffset]) const;

int32 **IFindFirst/Last**(const BString & [,int32 fromOffset])const;
int32 **IFindFirst/Last**(const char * [,int32 fromOffset]) const;

Replacing...

BString &**ReplaceFirst/Last**(char replaceThis, char withThis);
BString &**ReplaceFirst/Last**(const char *replaceThis, const char *withThis);

BString &**ReplaceAll**(char replaceThis, char withThis, int32 fromOffset = 0);
BString &**ReplaceAll**(const char *replaceThis, const char *withThis, int32 fromOffset = 0);

BString &**Replace**(char replaceThis, char withThis, int32 maxReplaceCount, int32 fromOffset = 0);
BString &**Replace**(const char *replaceThis, const char *withThis, int32 maxReplaceCount, int32 fromOffset = 0);

BString &**IReplaceFirst/Last**(char replaceThis, char withThis);
BString &**IReplaceFirst/Last**(const char *replaceThis, const char *withThis);

BString &**IReplaceAll**(char replaceThis, char withThis, int32 fromOffset = 0);
BString &**IReplaceAll**(const char *replaceThis, const char *withThis, int32 fromOffset = 0);

BString &**IReplace**(char replaceThis, char withThis, int32 maxReplaceCount, int32 fromOffset = 0);
BString &**IReplace**(const char *replaceThis, const char *withThis, int32 maxReplaceCount, int32 fromOffset = 0);

BString &**IReplaceSet**(const char *setOfChars, char with);
BString &**IReplaceSet**(const char *setOfChars, const char *with);

Unchecked char access...

char operator**[ ]**(int32 index) const;
char &operator**[ ]**(int32 index);

Checked char access...

char **ByteAt**(int32 index) const;

Fast low-level manipulation...

Note: LockBuffer() returns the equivalent of String(). 'maxLength' includes room for trailing null character. It is illegal to call other BString routines that reply on data/length consistency after LockBuffer() is called, until UnlockBuffer() is called.

char ***LockBuffer**(int32 maxLength);
BString &**UnlockBuffer**(int32 length = -1);

Upercase<->Lowercase...

BString &**ToLower**();
BString &**ToUpper**();

BString &**Capitalize**();
BString &**CapitalizeEachWord**();

Simple sprintf replacement calls...
Note: Slower than sprintf but type and overflow safe.

BString &operator**<<(----)**;

*...where "**----**" is an argument of type BString &, const char *, char, uint32, int32, uint64, int64, or float. Float output only %.2f format.*

BString Global Functions <be/support/string.h>

Compare operators...

bool operator**<**(const char *, const BString &);
bool operator**>**(const char *, const BString &);
bool operator**<=**(const char *, const BString &);
bool operator**>=**(const char *, const BString &);
bool operator**==**(const char *, const BString &);
bool operator**!=**(const char *, const BString &);

Non-member compare for sorting, etc....

int **Compare**(const BString &, const BString &);
int **ICompare**(const BString &, const BString &);

Support & Mail Kits

<be/SupportKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

Error Base Defined Values <be/support/Errors.h>
B_GENERAL_ERROR_BASE (LONG_MIN)
B_OS_ERROR_BASE
B_APP_ERROR_BASE
B_INTERFACE_ERROR_BASE
B_MEDIA_ERROR_BASE
B_TRANSLATION_ERROR_BASE
B_MIDI_ERROR_BASE
B_STORAGE_ERROR_BASE
B_POSIX_ERROR_BASE
B_MAIL_ERROR_BASE
B_PRINT_ERROR_BASE
B_DEVICE_ERROR_BASE
Developer-defined errors start at (B_ERRORS_END+1).

System-wide Error Values <be/support/Errors.h>	
B_NO_MEMORY	B_NAME_IN_USE
B_IO_ERROR	B_TIMED_OUT
B_PERMISSION_DENIED	B_INTERRUPTED
B_BAD_INDEX	B_WOULD_BLOCK
B_BAD_TYPE	B_CANCELED
B_BAD_VALUE	B_NO_INIT
B_MISMATCHED_VALUES	B_BUSY
B_NAME_NOT_FOUND	B_NOT_ALLOWED
B_OK (0)	B_NO_ERROR (0)
B_ERROR (-1)	

Shortcut Typedefs <be/support/SupportDefs.h>			
int8	signed char	uint64	unsigned long long
uint8	unsigned char	vint64	volatile long long
vint8	volatile signed char	vuint64	volatile unsigned long long
vuint8	volatile unsigned char	vlong	volatile long
int16	short	vint	volatile int
uint16	unsigned short	vshort	volatile short
vint16	volatile short	vchar	volatile char
vuint16	volatile unsigned short	vulong	volatile unsigned long
int32	long	vuint	volatile unsigned int
uint32	unsigned long	vushort	volatile unsigned short
vint32	volatile long	vuchar	volatile unsigned char
vuint32	volatile unsigned long	uchar	unsigned char
int64	long long	unichar	unsigned short

Other Global Typdefs <be/support/SupportDefs.h>	
status_t	int32
bigtime_t	int64
type_code	uint32
perform_code	uint32

Empty String <be/support/SupportDefs.h>
extern const char ***B_EMPTY_STRING**;

System Beep Function <be/support/Beep.h>
status_t **beep**();

Other Functions <be/support/SupportsDefs.h>
uint32 **get_stack_frame**(); *(Note: extern "C" type.)*

Common Defines <be/support/SupportDefs.h>

C only, these won't work in C++...
min(a,b)
max(a,b)

C or C++...
min_c(a,b)
max_c(a,b)

For C compatibility with C++ notation...
typedef unsigned char **bool**;
#define **false** 0
#define **true** 1
#define **NULL** (0)

Atomic Functions <be/support/SupportDefs.h>
Note: extern "C" type functions. Old value is returned.

int32 **atomic_add**(int32 *value, int32 addvalue);
int32 **atomic_and**(int32 *value, int32 andvalue);
int32 **atomic_or**(int32 *value, int32 orvalue);

Obsolete or Discouraged API <be/support/SupportDefs.h>
***FALSE** and **TRUE** are discouraged, use **true** and **false** instead.*

Function pointer types; unused in the Be API...
typedef void (***B_PFV**)();
typedef int (***B_PFI**)();
typedef long (***B_PFL**)();

TYPE CONSTANTS

Constants that represent distinct data types, as used by BMessage and other classes.

Data Type Constants <be/support/TypeConstants.h>	
B_ANY_TYPE	B_PATTERN_TYPE
B_ASCII_TYPE	B_POINTER_TYPE
B_BOOL_TYPE	B_POINT_TYPE
B_CHAR_TYPE	B_RAW_TYPE
B_DOUBLE_TYPE	B_RECT_TYPE
B_FLOAT_TYPE	B_REF_TYPE
B_INT64_TYPE	B_SIZE_T_TYPE
B_INT32_TYPE	B_SSIZE_T_TYPE
B_INT16_TYPE	B_STRING_TYPE
B_INT8_TYPE	B_TIME_TYPE
B_MESSAGE_TYPE	B_UINT64_TYPE
B_MESSENGER_TYPE	B_UINT32_TYPE
B_MIME_TYPE	B_UINT16_TYPE
B_OBJECT_TYPE	B_UINT8_TYPE
B_OFF_T_TYPE	B_RGB_COLOR_TYPE
B_MONOCHROME_1_BIT_TYPE	B_MEDIA_PARAMETER_TYPE
B_GRAYSCALE_8_BIT_TYPE	B_MEDIA_PARAMETER_WEB_TYPE
B_COLOR_8_BIT_TYPE	B_MEDIA_PARAMETER_GROUP_TYPE
B_RGB_32_BIT_TYPE	

DEBUGGING

Debug Macros <be/support/Debug.h>
Note: These macros do nothing if DEBUG is not true.

BOOL **SET_DEBUG_ENABLED**(FLAG)
BOOL **IS_DEBUG_ENABLED**()
int **SERIAL_PRINT**(ARGS)
int **PRINT**(ARGS)
int **PRINT_OBJECT**(OBJ)
int **TRACE**()
int **SERIAL_TRACE**()
void **DEBUGGER**(MSG)
int **ASSERT_WITH_MESSAGE**(expr, msg)
void **TRESPASS**()
DEBUG_ONLY(arg)

Debugger Functions <kernel/OS.h>
void **debugger**(const char *message);
const int **disable_debugger**(int state); *(for this team only)*

BArchivable PROTOCOL

Mix-in class defining the archiving protocol to save an object's data in a BMessage.

class **BArchivable** <be/support/Archivable.h>

BArchivable([BMessage *from]);
virtual ~**BArchivable**();

virtual status_t **Archive**(BMessage *into, bool deep = true) const;
static BArchivable ***Instantiate**(BMessage *from);

BArchivable Function Typedef <be/support/Archivable.h>
typedef BArchivable *(***instantiation_func**) (BMessage *);

BArchivable Global Functions <be/support/Archivable.h>

BArchivable ***instantiate_object**(BMessage *from [,image_id *id]);
bool **validate_instantiation**(BMessage *from, const char *class_name);

instantiation_func **find_instantiation_func**(const char *class_name [,const char *sig]);
instantiation_func **find_instantiation_func**(BMessage *archive_data);

BFlattenable PROTOCOL

Pure virtual class that defines a protocol for flattening and unflattening an object's data so it can be transmitted as a stream of bytes.

class **BFlattenable** <be/support/Flattenable.h>

virtual bool **IsFixedSize**() const = 0;
virtual type_code **TypeCode**() const = 0;
virtual ssize_t **FlattenedSize**() const = 0;
virtual status_t **Flatten**(void *buffer, ssize_t size) const = 0;
virtual bool **AllowsTypeCode**(type_code code) const;
virtual status_t **Unflatten**(type_code c, const void *buf, ssize_t size) = 0;

Support & Mail Kits

<be/SupportKit.h> <libbe.so>

BAutolock

A stack-based locking mechanism.

class **BAutolock** <be/support/Autolock.h>

BAutolock(BLocker *lock);
BAutolock(BLocker &lock);
BAutolock(BLooper *looper);
~**BAutolock**();

bool **IsLocked**();

BLocker

Defines a nestable locking mechanism.

class **BLocker** <be/support/Locker.h>

BLocker([const char *name,] [bool benaphore_style]);
virtual ~**BLocker**();

bool **Lock**();
void **Unlock**();
bool **IsLocked**() const;
status_t **LockWithTimeout**(bigtime_t timeout);

For debugging (only!)...
thread_id **LockingThread**() const;
int32 **CountLocks**() const;
int32 **CountLockRequests**() const;
sem_id **Sem**() const;

UTF-8 CONVERSION

Text conversion using the UTF-8 standard.

Conversion Codes <be/support/UTF8.h>	
BB_ISO1_CONVERSION_ANY_TYPE	ISO 8859-1
B_ISO2_CONVERSION	ISO 8859-2
B_ISO3_CONVERSION	ISO 8859-3
B_ISO4_CONVERSION	ISO 8859-4
B_ISO5_CONVERSION	ISO 8859-5
B_ISO6_CONVERSION	ISO 8859-6
B_ISO7_CONVERSION	ISO 8859-7
B_ISO8_CONVERSION	ISO 8859-8
B_ISO9_CONVERSION	ISO 8859-9
B_ISO10_CONVERSION	ISO 8859-10
B_MAC_ROMAN_CONVERSION	Macintosh Roman
B_SJIS_CONVERSION	Shift-JIS
B_EUC_CONVERSION	EUC Packed Japanese
B_JIS_CONVERSION	JIS X 0208-1990
B_MS_WINDOWS_CONVERSION	Windows Codepage 1252
B_UNICODE_CONVERSION	Unicode 2.0
B_KOI8R_CONVERSION	KOI8-R
B_MS_WINDOWS_1251_CONVERSION	Windows Codepage 1251
B_MS_DOS_866_CONVERSION	MS-DOS Codepage 866

Conversion Functions <be/support/UTF8.h>

status_t **convert_to_utf8**(uint32 srcEncoding, const char *src, int32 *srcLen, char *dst, int32 *dstLen, int32 *state, char substitute = B_SUBSTITUTE);
status_t **convert_from_utf8**(uint32 dstEncoding, const char *src, int32 *srcLen, char *dst, int32 *dstLen, int32 *state, char substitute = B_SUBSTITUTE);

MAIL KIT

Use <be/MailKit.h> and link with libmail.so.

Mail Kit Error Values <be/support/Errors.h>
B_MAIL_NO_DAEMON
B_MAIL_UNKNOWN_USER
B_MAIL_WRONG_PASSWORD
B_MAIL_UNKNOWN_HOST
B_MAIL_ACCESS_ERROR
B_MAIL_UNKNOWN_FIELD
B_MAIL_NO_RECIPIENT
B_MAIL_INVALID_MAIL

Mail File Attributes <be/mail/email.h>		
#Define Name	**Attribute**	**Type**
B_MAIL_ATTR_NAME	'MAIL:name'	indexed string
B_MAIL_ATTR_STATUS	'MAIL:status'	indexed string
B_MAIL_ATTR_PRIORITY	'MAIL:priority'	indexed string
B_MAIL_ATTR_TO	'MAIL:to'	indexed string
B_MAIL_ATTR_FROM	'MAIL:from'	indexed string
B_MAIL_ATTR_SUBJECT	'MAIL:subject'	indexed string
B_MAIL_ATTR_REPLY	'MAIL:reply'	indexed string
B_MAIL_ATTR_WHEN	'MAIL:when'	indexed time
B_MAIL_ATTR_FLAGS	'MAIL:flags'	indexed int32
B_MAIL_ATTR_RECIPIENTS	'MAIL:recipients'	string
B_MAIL_ATTR_MIME	'MAIL:mime'	string
B_MAIL_ATTR_HEADER	'MAIL:header_length'	int32
B_MAIL_ATTR_CONTENT	'MAIL:content_length'	int32

enum 'mail_flags' <be/mail/email.h>
B_MAIL_PENDING
B_MAIL_SENT
B_MAIL_SAVE

E-Mail Mime Type <be/mail/email.h>
B_MAIL_TYPE ('text/x-email')

Schedule Days <be/mail/email.h>
B_CHECK_NEVER
B_CHECK_WEEKDAYS
B_CHECK_DAILY
B_CHECK_CONTINUOUSLY
B_CHECK_CONTINUOSLY

Mail Header Fields (rfc822) <be/mail/email.h>
B_MAIL_TO ("To: ")
B_MAIL_CC ("Cc: ")
B_MAIL_BCC ("Bcc: ")
B_MAIL_FROM ("From: ")
B_MAIL_DATE ("Date: ")
B_MAIL_REPLY ("Reply-To: ")
B_MAIL_SUBJECT ("Subject: ")
B_MAIL_PRIORITY ("Priority: ")

Max Lengths <be/mail/email.h>
B_MAX_USER_NAME_LENGTH (32)
B_MAX_HOST_NAME_LENGTH (64)

typedef struct **mail_pop_account** <be/mail/email.h>

char **pop_name**[B_MAX_USER_NAME_LENGTH];
char **pop_password**[B_MAX_USER_NAME_LENGTH];
char **pop_host**[B_MAX_HOST_NAME_LENGTH];
char **real_name**[128];
char **reply_to**[128];
int32 **days**; (Note: see Schedule Days table.)
int32 **interval**, **begin_time**, **end_time**; (Note: in seconds)

typedef struct **mail_notification** <be/mail/email.h>

bool **alert**,
bool **beep**;

Mail Global Functions <be/mail/email.h>

int32 **count_pop_accounts**(void);
status_t **get_pop_account**(mail_pop_account*, int32 index=0);
status_t **set_pop_account**(mail_pop_account*, int32 index=0, bool save = true);

status_t **get_smtp_host**(char*);
status_t **set_smtp_host**(char*, bool save = true);

status_t **get_mail_notification**(mail_notification*);
status_t **set_mail_notification**(mail_notification*, bool save = true);
status_t **check_for_mail**(int32 *incoming_count = NULL);
status_t **send_queued_mail**(void);
status_t **forward_mail**(entry_ref*, const char* recipients, bool now = true);

ssize_t **decode_base64**(char *out, char *in, off_t length, bool replace_cr = false);
ssize_t **encode_base64**(char *out, char *in, off_t length);

class **BMailMessage** <be/mail/email.h>

BMailMessage();
~**BMailMessage**();

status_t **AddContent**(const char *text, int32 length, uint32 encoding = B_ISO1_CONVERSION, bool clobber = false);
status_t **AddContent**(const char *text, int32 length, const char *encoding, bool clobber = false);

status_t **AddEnclosure**(entry_ref *ref, bool clobber = false);
status_t **AddEnclosure**(const char *path, bool clobber = false);
status_t **AddEnclosure**(const char *MIME_type, void *data, int32 len, bool clobber = false);

status_t **AddHeaderField**(uint32 encoding, const char *field_name, const char *str, bool clobber = false);
status_t **AddHeaderField**(const char *field_name, const char *str, bool clobber = false);

status_t **Send**(bool send_now = false, bool remove_after_send = false);

AppKit General

<be/AppKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

Application Kit Error Values <be/support/Errors.h>
B_BAD_REPLY
B_DUPLICATE_REPLY
B_MESSAGE_TO_SELF
B_BAD_HANDLER
B_ALREADY_RUNNING
B_LAUNCH_FAILED
B_AMBIGUOUS_APP_LAUNCH
B_UNKNOWN_MIME_TYPE
B_BAD_SCRIPT_SYNTAX
B_LAUNCH_FAILED_NO_RESOLVE_LINK
B_LAUNCH_FAILED_EXECUTABLE
B_LAUNCH_FAILED_APP_NOT_FOUND
B_LAUNCH_FAILED_APP_IN_TRASH
B_LAUNCH_FAILED_NO_PREFERRED_APP
B_LAUNCH_FAILED_FILES_APP_NOT_FOUND

System Message Codes <be/app/AppDefs.h>	
B_ABOUT_REQUESTED	B_OPEN_IN_WORKSPACE
B_WINDOW_ACTIVATED	B_PULSE
B_APP_ACTIVATED	B_READY_TO_RUN
B_ARGV_RECEIVED	B_REFS_RECEIVED
B_QUIT_REQUESTED	B_SCREEN_CHANGED
B_CANCEL	B_VALUE_CHANGED
B_KEY_DOWN	B_VIEW_MOVED
B_KEY_UP	B_VIEW_RESIZED
B_MODIFIERS_CHANGED	B_WINDOW_MOVED
B_MINIMIZE	B_WINDOW_RESIZED
B_MOUSE_DOWN	B_WORKSPACES_CHANGED
B_MOUSE_MOVED	B_WORKSPACE_ACTIVATED
B_MOUSE_ENTER_EXIT	B_ZOOM
B_MOUSE_UP	

Other Commands <be/app/AppDefs.h>	
B_SET_PROPERTY	B_REPLY
B_GET_PROPERTY	B_SIMPLE_DATA
B_CREATE_PROPERTY	B_MIME_DATA
B_DELETE_PROPERTY	B_ARCHIVED_OBJECT
B_COUNT_PROPERTIES	B_UPDATE_STATUS_BAR
B_EXECUTE_PROPERTY	B_RESET_STATUS_BAR
B_GET_SUPPORTED_SUITES	B_NODE_MONITOR
B_UNDO	B_QUERY_UPDATE
B_CUT	B_ENDORSABLE
B_COPY	B_COPY_TARGET
B_PASTE	B_MOVE_TARGET
B_SELECT_ALL	B_TRASH_TARGET
B_SAVE_REQUESTED	B_LINK_TARGET
B_MESSAGE_NOT_UNDERSTOO	B_INPUT_DEVICES_CHANGED
B_NO_REPLY	B_INPUT_METHOD_EVENT
Media Kit reserves all codes starting in 'TRI'.	

Global Cursors <be/app/AppDefs.h>

```
extern _IMPEXP_BE const unsigned char
  B_HAND_CURSOR[];

extern _IMPEXP_BE const unsigned char
  B_I_BEAM_CURSOR[];
```

BApplication

Basic functioning of all applications.

Application Global Declaration <be/apps/Application.h>

```
BApplication *b_app;
```

class **BApplication** <be/app/Application.h>

: public BLooper

```
BApplication(const char *signature);
virtual ~BApplication();

App control and System Message handling...
virtual thread_id Run();
virtual void Quit();
virtual bool QuitRequested();
virtual void Pulse();
virtual void ReadyToRun();
virtual void MessageReceived(BMessage *msg);
virtual void ArgvReceived(int32 argc, char **argv);
virtual void AppActivated(bool active);
virtual void RefsReceived(BMessage *a_message);
virtual void AboutRequested();

Cursor control, window list, and app info...
void ShowCursor();
void HideCursor();
void ObscureCursor();
bool IsCursorHidden() const;
void SetCursor(const void *cursor);
int32 CountWindows() const;
BWindow *WindowAt(int32 index) const;
bool IsLaunching() const;
status_t GetAppInfo(app_info *info) const;
static BResources *AppResources();

BLooper inherrited virtual functions...
virtual BHandler *ResolveSpecifier(BMessage *msg, int32
  index, BMessage *specifier, int32 form, const char
  *property);
virtual status_t GetSupportedSuites(BMessage *data);

virtual void DispatchMessage(BMessage *an_event,
  BHandler *handler);
void SetPulseRate(bigtime_t rate);
```

PROPERTY INFO

Utility class for maintaining scripting information.

struct **compound_type** <be/app/PropertyInfo.h>

```
struct field_pair {
    char *name;
    type_code type;
} pairs[5];
```

struct **property_info** <be/app/PropertyInfo.h>

```
char *name;
uint32 commands[10], specifiers[10];
char *usage;
uint32 extra_data, types[10];
compound_type ctypes[3];
```

enum 'value_kind' <be/app/PropertyInfo.h>
B_COMMAND_KIND
B_TYPE_CODE_KIND

struct **value_info** <be/app/PropertyInfo.h>

```
char *name;
uint32 value;
value_kind kind;
char *usage;
uint32 extra_data;
```

Property Info Datatype <be/app/PropertyInfo.h>
B_PROPERTY_INFO_TYPE ('SCTD')

class **BPropertyInfo** <be/app/PropertyInfo.h>

: public BFlattenable

```
BPropertyInfo(property_info *p = NULL, value_info *ci =
  NULL, bool free_on_delete = false);
virtual ~BPropertyInfo();

virtual int32 FindMatch(BMessage *msg, int32 index,
  BMessage *spec, int32 form, const char *prop, void *data =
  NULL) const;

const property_info *Properties() const;
const value_info *Values() const;
int32 CountProperties() const;
int32 CountValues() const;

BFlattenable inherrited virtual functions...
virtual bool IsFixedSize() const;
virtual type_code TypeCode() const;
virtual ssize_t FlattenedSize() const;
virtual status_t Flatten(void *buffer, ssize_t size) const;
virtual bool AllowsTypeCode(type_code code) const;
virtual status_t Unflatten(type_code c, const void *buf, ssize_t
  s);

void PrintToStream() const;
```

BClipboard

BClipboard class defines clipboard functionality.

Clipboard Global Declaration <be/app/Clipboard.h>:

```
extern BClipboard *be_clipboard;
```

class **BClipboard** <be/app/Clipboard.h>

```
BClipboard(const char *name, bool transient = false);
virtual ~BClipboard();

const char *Name() const;

bool Lock();
void Unlock();
bool IsLocked() const;

status_t Clear();
status_t Commit();
status_t Revert();

BMessenger DataSource() const;
BMessage *Data() const;
```

AppKit General

<be/AppKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

BRoster

BRoster Global Objects <be/app/Roster.h>

extern const BRoster ***be_roster**;

struct app_info <be/app/Roster.h>

app_info();
~**app_info**();

thread_id **thread**;
team_id **team**;
port_id **port**;
uint32 **flags**;
entry_ref **ref**;
char **signature**[B_MIME_TYPE_LENGTH];

Launch Codes <be/app/Roster.h>	
B_SINGLE_LAUNCH	B_BACKGROUND_APP
B_MULTIPLE_LAUNCH	B_ARGV_ONLY
B_EXCLUSIVE_LAUNCH	B_LAUNCH_MASK

Request Codes <be/app/Roster.h>	
B_REQUEST_LAUNCHED	B_REQUEST_ACTIVATED
B_REQUEST_QUIT	

Some_App Message Codes <be/app/Roster.h>
B_SOME_APP_LAUNCHED ('BRAS')
B_SOME_APP_QUIT ('BRAQ')
B_SOME_APP_ACTIVATED ('BRAW')

class BRoster <be/app/Roster.h>

BRoster();
~**BRoster**();

Querying for apps...

bool **IsRunning**(const char *mime_sig) const;
bool **IsRunning**(entry_ref *ref) const;

team_id **TeamFor**(const char *mime_sig) const;
team_id **TeamFor**(entry_ref *ref) const;

void **GetAppList**(BList *team_id_list) const;
void **GetAppList**(const char *sig, BList *team_id_list) const;

status_t **GetAppInfo**(const char *sig, app_info *info) const;
status_t **GetAppInfo**(entry_ref *ref, app_info *info) const;

status_t **GetRunningAppInfo**(team_id team, app_info *info) const;
status_t **GetActiveAppInfo**(app_info *info) const;

status_t **FindApp**(const char *mime_type, entry_ref *app) const;
status_t **FindApp**(entry_ref *ref, entry_ref *app) const;

Launching, activating, and broadcasting to apps...

status_t **Broadcast**(BMessage *msg [,BMessenger reply_to]) const;
status_t **StartWatching**(BMessenger target, uint32 event_mask = B_REQUEST_LAUNCHED | B_REQUEST_QUIT) const;
status_t **StopWatching**(BMessenger target) const;
status_t **ActivateApp**(team_id team) const;

status_t **Launch**(const char *mime_type, BMessage *initial_msgs = NULL, team_id *app_team = NULL) const;
status_t **Launch**(const char *mime_type, BList *message_list, team_id *app_team = NULL) const;
status_t **Launch**(const char *mime_type, int argc, char **args, team_id *app_team = NULL) const;
status_t **Launch**(entry_ref *ref, BMessage *initial_message=NULL, team_id *app_team=NULL) const;
status_t **Launch**(entry_ref *ref, BList *message_list, team_id *app_team = NULL) const;
status_t **Launch**(entry_ref *ref, int argc, char **args, team_id *app_team = NULL) const;

AppFileInfo

struct version_info <be/storage/AppFileInfo.h>

uint32 **major**, **middle**, **minor**, **variety**;
uint32 **internal**;
char **short_info**[64], **long_info**[256];

enum 'info_location' <be/storage/AppFileInfo.h>
B_USE_ATTRIBUTES
B_USE_RESOURCES
B_USE_BOTH_LOCATIONS

enum 'version_kind" <be/storage/AppFileInfo.h>
B_APP_VERSION_KIND
B_SYSTEM_VERSION_KIND

class BAppFileInfo <be/storage/AppFileInfo.h>

: public BNodeInfo *(Storage Kit)*

BAppFileInfo([BFile *file]);
virtual ~**BAppFileInfo**();

status_t **SetTo**(BFile *file);

virtual status_t **GetType**(char *type) const;
status_t **GetSignature**(char *sig) const;
status_t **GetAppFlags**(uint32 *flags) const;
status_t **GetSupportedTypes**(BMessage *types) const;
status_t **GetIcon**(BBitmap *icon, icon_size which) const;
status_t **GetVersionInfo**(version_info *vinfo, version_kind k) const;
status_t **GetIconForType**(const char *type, BBitmap *icon, icon_size which) const;
bool **IsSupportedType**(const char *type) const;
virtual status_t **SetType**(const char *type);
status_t **SetSignature**(const char *sig);
status_t **SetAppFlags**(uint32 flags);
status_t **SetSupportedTypes**(const BMessage *types, bool sync_all);
status_t **SetSupportedTypes**(const BMessage *types);
status_t **SetIcon**(const BBitmap *icon, icon_size which);
status_t **SetVersionInfo**(const version_info *vinfo, version_kind k);
status_t **SetIconForType**(const char *type, const BBitmap *icon, icon_size which);
void **SetInfoLocation**(info_location loc);
bool **IsUsingAttributes**() const;
bool **IsUsingResources**() const;
bool **Supports**(BMimeType *mt) const;

RESOURCES.

class BResources <be/storage/Resources.h>

BResources([const BFile *file, bool truncate = false]);
virtual ~**BResources**();

status_t **SetTo**(const BFile *file, bool truncate = false);
const BFile &**File**() const;

const void ***LoadResource**(type_code type, int32 id, size_t * out_size);
const void * **LoadResource**(type_code type, const char * name, size_t * out_size);

status_t **PreloadResourceType**(type_code type = 0);
status_t **Sync**();
status_t **MergeFrom**(BFile * from_file);
status_t **WriteTo**(BFile * new_file);

status_t **AddResource**(type_code type, int32 id, const void *data, size_t data_size, const char *name = NULL);

bool **HasResource**(type_code type, int32 id);
bool **HasResource**(type_code type, const char *name);

bool **GetResourceInfo**(int32 resIndex, type_code* typeFound, int32* idFound, const char **nameFound, size_t* size);
bool **GetResourceInfo**(type_code type, int32 resIndex, int32* idFound, const char **nameFound, size_t* size);
bool **GetResourceInfo**(type_code type, int32 id, const char **nameFound, size_t* size);
bool **GetResourceInfo**(type_code type, const char *name, int32* idFound, size_t* size);
bool **GetResourceInfo**(const void * resource, type_code out_type, int32 * out_id, size_t * out_size, const char ** out_name);

status_t **RemoveResource**(const void * resource);
status_t **RemoveResource**(type_code type, int32 id);

class BResourceStrings <be/storage/ResourceStrings.h>

BResourceStrings([const entry_ref & ref]);
virtual ~**BResourceStrings**();

status_t **InitCheck**();
virtual BString ***NewString**(int32 id);
virtual const char ***FindString**(int32 id); *(Note: Returned pointer is valid until ~BResourceStrings() or SetStringFile() called)*
virtual status_t **SetStringFile**(const entry_ref * file);
status_t **GetStringFile**(entry_ref * out_ref);

enum { **RESOURCE_TYPE** = 'CSTR' };

TIME FUNCTIONS (KERNEL)

Link with libroot.so for these functions.

Time Functions <kernel/OS.h>

uint32 **real_time_clock**(void);
void **set_real_time_clock**(int32 secs_since_jan1_1970);
bigtime_t **real_time_clock_usecs**(void);
status_t **set_timezone**(char *str);
bigtime_t **system_time**(void);

AppKit Messaging

<be/AppKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

class BHandler <be/app/Handler.h>

: public BArchivable

BHandler(const char *name = NULL);
virtual ~**BHandler**();

virtual void **MessageReceived**(BMessage *message);
BLooper ***Looper**() const;
void **SetName**(const char *name);
const char ***Name**() const;
virtual void **SetNextHandler**(BHandler *handler);
BHandler ***NextHandler**() const;

virtual void **AddFilter**(BMessageFilter *filter);
virtual bool **RemoveFilter**(BMessageFilter *filter);
virtual void **SetFilterList**(BList *filters);
BList ***FilterList**();

bool **LockLooper**();
status_t **LockLooperWithTimeout**(bigtime_t timeout);
void **UnlockLooper**();

Scripting...
virtual BHandler ***ResolveSpecifier**(BMessage *msg, int32 index, BMessage *specifier, int32 form, const char *property);
virtual status_t **GetSupportedSuites**(BMessage *data);

class BMessageQueue <be/app/MessageQueue.h>

BMessageQueue();
virtual ~**BMessageQueue**();

void **AddMessage**(BMessage *an_event);
bool **RemoveMessage**(BMessage *an_event);
BMessage ***NextMessage**();

BMessage ***FindMessage**(int32 index) const;
BMessage ***FindMessage**(uint32 what, int32 index = 0) const;

int32 **CountMessages**() const;
bool **IsEmpty**() const;

bool **Lock**();
void **Unlock**();

BMessage

Defined Lengths <be/app/Message.h>
B_FIELD_NAME_LENGTH (255)
B_PROPERTY_NAME_LENGTH (255)

Message Specifiers <be/app/Message.h>
B_NO_SPECIFIER (0)
B_DIRECT_SPECIFIER
B_INDEX_SPECIFIER
B_REVERSE_INDEX_SPECIFIER
B_RANGE_SPECIFIER
B_REVERSE_RANGE_SPECIFIER
B_NAME_SPECIFIER
B_ID_SPECIFIER
B_SPECIFIERS_END (128)
App-defined specifiers start at B_SPECIFIERS_END+1.

class BMessage <be/app/Message.h>

BMessage([uint32 what]);
BMessage(const BMessage &a_message);
BMessage(BMessage *a_message);
virtual ~**BMessage**();

BMessage &operator**=(**const BMessage &msg);
void *operator **new**(size_t size);
void operator **delete**(void *ptr, size_t size);

uint32 **what**;

status_t **GetInfo**(type_code typeRequested, int32 which, char **name, type_code *typeReturned, int32 *count = NULL) const;
status_t **GetInfo**(const char *name, type_code *type, int32 *c = 0) const;
status_t **GetInfo**(const char *name, type_code *type, bool *fixed_size) const;

int32 **CountNames**(type_code type) const;
bool **IsEmpty**() const;
bool **IsSystem**() const;
bool **IsReply**() const;

void **PrintToStream**() const;

Delivery info...
bool **WasDelivered**() const;
bool **IsSourceWaiting**() const;
bool **IsSourceRemote**() const;
BMessenger **ReturnAddress**() const;
const BMessage ***Previous**() const;
bool **WasDropped**() const;
BPoint **DropPoint**(BPoint *offset = NULL) const;

Replying...
status_t **SendReply**(uint32 command [,BHandler *reply_to]);
status_t **SendReply**(uint32 command, BMessage *reply_to_reply);
status_t **SendReply**(BMessage *the_reply, BMessage *reply_to_reply [,bigtime_t sendTimeout [,bigtime_t replyTimeout]]);
status_t **SendReply**(BMessage *the_reply [,BHandler *reply_to [,bigtime_t timeout]]);

Flattening data...
ssize_t **FlattenedSize**() const;

status_t **Flatten**(char *buffer, ssize_t size) const;
status_t **Flatten**(BDataIO *stream, ssize_t *size = NULL) const;

status_t **Unflatten**(const char *flat_buffer);
status_t **Unflatten**(BDataIO *stream);

Specifiers (scripting)...
status_t **AddSpecifier**(const char *property [,int32 index [,int32 range]]);
status_t **AddSpecifier**(const char *property, const char *name);
status_t **AddSpecifier**(const BMessage *specifier);

status_t **SetCurrentSpecifier**(int32 index);
status_t **GetCurrentSpecifier**(int32 *index, BMessage *specifier = NULL, int32 *form = NULL, const char **property = NULL) const;
bool **HasSpecifiers**() const;
status_t **PopSpecifier**();

Adding data...
status_t **AddRect**(const char *name, BRect a_rect);
status_t **AddPoint**(const char *name, BPoint a_point);
status_t **AddString**(const char *name, const char *a_string);
status_t **AddInt8/16/32/64**(const char *name, int8/16/32/64 val);
status_t **AddBool/Float/Double**(const char *name, bool/float/double);
status_t **AddPointer**(const char *name, const void *ptr);
status_t **AddMessenger**(const char *name, BMessenger messenger);
status_t **AddRef**(const char *name, const entry_ref *ref);
status_t **AddMessage**(const char *name, const BMessage *msg);
status_t **AddFlat**(const char *name, BFlattenable *obj, int32 count = 1);
status_t **AddData**(const char *name, type_code type, const void *data, ssize_t numBytes, bool is_fixed_size = true, int32 count = 1);

Finding data...
The same methods as adding data are repeated for finding data using this format...

status_t **Find---**(const char *name [, int32 index] , **---** *found_data) const;

*Where "**---**" is any of the types of objects as listed for Adding. For "Data" data type also pass* ssize_t *numBytes.

Replacing data...
The same methods as adding data are repeated for replacing data using this format...

status_t **Replace---**(const char *name [,int32 index] , **---** new_data) const;

*Where "**---**" is any of the types of objects as listed for Adding. For "Data" data type also pass* ssize_t numBytes.

Removing data...
status_t **RemoveData**(const char *name, int32 index = 0);
status_t **RemoveName**(const char *name);
status_t **MakeEmpty**();

AppKit Messaging

<be/AppKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

BLooper Constants <be/app/Looper.h>

B_LOOPER_PORT_DEFAULT_CAPACITY (100)

class BLooper
<be/app/Looper.h>

: public BHandler

BLooper(const char *name = NULL, int32 priority = B_NORMAL_PRIORITY, int32 port_capacity = B_LOOPER_PORT_DEFAULT_CAPACITY);
virtual ~**BLooper**();

Message transmission..
status_t **PostMessage**(uint32 command [,BHandler *handler [,BHandler *reply_to]]);
status_t **PostMessage**(BMessage *message [,BHandler *handler [,BHandler *reply_to]]);
virtual void **DispatchMessage**(BMessage *message, BHandler *handler);
virtual void **MessageReceived**(BMessage *msg);
BMessage ***CurrentMessage**() const;
BMessage ***DetachCurrentMessage**();
BMessageQueue ***MessageQueue**() const;
bool **IsMessageWaiting**() const;

Message handlers...
void **AddHandler**(BHandler *handler);
bool **RemoveHandler**(BHandler *handler);
int32 **CountHandlers**() const;
BHandler ***HandlerAt**(int32 index) const;
int32 **IndexOf**(BHandler *handler) const;
BHandler ***PreferredHandler**() const;
void **SetPreferredHandler**(BHandler *handler);

Loop control...
virtual thread_id **Run**();
virtual void **Quit**();
virtual bool **QuitRequested**();
bool **Lock**();
void **Unlock**();
bool **IsLocked**() const;
status_t **LockWithTimeout**(bigtime_t timeout);
thread_id **Thread**() const;
team_id **Team**() const;
static BLooper ***LooperForThread**(thread_id tid);

Loop debugging (for debugging only!)...
thread_id **LockingThread**() const;
int32 **CountLocks**() const;
int32 **CountLockRequests**() const;
sem_id **Sem**() const;

Scripting... (BHandler inherrited)
virtual BHandler ***ResolveSpecifier**(BMessage *msg, int32 index, BMessage *specifier, int32 form, const char *property);
virtual status_t **GetSupportedSuites**(BMessage *data);

Message Filters (also see BHandler)...
virtual void **AddCommonFilter**(BMessageFilter *filter);
virtual bool **RemoveCommonFilter**(BMessageFilter *filter);
virtual void **SetCommonFilterList**(BList *filters);
BList ***CommonFilterList**() const;

Global Messenger Declaration <be/apps/Application.h>
BMessenger **be_app_messenger**;

class BMessenger
<be/app/Messenger.h>

BMessenger();
BMessenger(const char *mime_sig, team_id team = -1,status_t *perr = NULL);
BMessenger(const BHandler *handler, const BLooper *looper = NULL, status_t *perr = NULL);
BMessenger(const BMessenger &from);
~**BMessenger**();

BMessenger &operator**=(**const BMessenger &from);
bool operator**==**(const BMessenger &other) const;

bool **IsTargetLocal**() const;
BHandler ***Target**(BLooper **looper) const;
bool **LockTarget**() const;
status_t **LockTargetWithTimeout**(bigtime_t timeout) const;

status_t **SendMessage**(uint32 command,BHandler *reply_to = NULL) const;
status_t **SendMessage**(BMessage *a_message, BHandler *reply_to = NULL, bigtime_t timeout = B_INFINITE_TIMEOUT) const;
status_t **SendMessage**(BMessage *a_message, BMessenger reply_to, bigtime_t timeout = B_INFINITE_TIMEOUT) const;
status_t **SendMessage**(uint32 command, BMessage *reply) const;
status_t **SendMessage**(BMessage *a_message, BMessage *reply, bigtime_t send_timeout = B_INFINITE_TIMEOUT, bigtime_t reply_timeout = B_INFINITE_TIMEOUT) const;

bool **IsValid**() const;
team_id **Team**() const;

Global Messenger Operators <be/app/Messenger.h>

bool operator**<**(const BMessenger & a, const BMessenger &b);
bool operator**!=**(const BMessenger & a,const BMessenger &b);

class BInvoker
<be/app/Invoker.h>

BInvoker();
BInvoker(BMessage *message, const BHandler *handler, const BLooper *looper = NULL);
BInvoker(BMessage *message, BMessenger target);
virtual ~**BInvoker**();

virtual status_t **SetMessage**(BMessage *message);
BMessage ***Message**() const;
uint32 **Command**() const;

virtual status_t **SetTarget**(const BHandler *h, const BLooper *loop = NULL);
virtual status_t **SetTarget**(BMessenger messenger);
bool **IsTargetLocal**() const;
BHandler ***Target**(BLooper **looper = NULL) const;

BMessenger **Messenger**() const;
virtual status_t **SetHandlerForReply**(BHandler *handler);
BHandler ***HandlerForReply**() const;

virtual status_t **Invoke**(BMessage *msg = NULL);

status_t **SetTimeout**(bigtime_t timeout);
bigtime_t **Timeout**() const;

class BMessageRunner
<be/app/MessageRunner.h>

BMessageRunner(BMessenger target, const BMessage *msg, bigtime_t interval, int32 count = -1);
BMessageRunner(BMessenger target, const BMessage *msg, bigtime_t interval, int32 count, BMessenger reply_to);
virtual ~**BMessageRunner**();

status_t **InitCheck**() const;

status_t **SetInterval**(bigtime_t interval);
status_t **SetCount**(int32 count);
status_t **GetInfo**(bigtime_t *interval, int32 *count) const;

MESSAGE FILTER

Designates a function that is called when a BMessage arrives at a BLooper.

enum ' filter_result' <be/app/MessageFilter.h>

B_SKIP_MESSAGE
B_DISPATCH_MESSAGE

enum 'message_delivery' <be/app/MessageFilter.h>

B_ANY_DELIVERY
B_DROPPED_DELIVERY
B_PROGRAMMED_DELIVERY

enum 'message_source' <be/app/MessageFilter.h>

B_ANY_SOURCE
B_REMOTE_SOURCE
B_LOCAL_SOURCE

Filter Result Hook Typedef <be/app/MessageFilter.h>

typedef filter_result (***filter_hook**) (BMessage *message, BHandler **target, BMessageFilter *filter);

class BMessageFilter
<be/app/MessageFilter.h>

BMessageFilter(uint32 what [,filter_hook func]);
BMessageFilter(message_delivery delivery, message_source source [,filter_hook func]);
BMessageFilter(message_delivery delivery, message_source source, uint32 what [,filter_hook func]);
BMessageFilter(const BMessageFilter &filter);
BMessageFilter(const BMessageFilter *filter);
virtual ~**BMessageFilter**();

BMessageFilter &operator**=(**const BMessageFilter &from);

Note: Ignored if filter_hook is non-NULL.
virtual filter_result **Filter**(BMessage *message, BHandler **target);message_delivery **MessageDelivery**() const;

message_source **MessageSource**() const;

uint32 **Command**() const;

bool **FiltersAnyCommand**() const;

BLooper ***Looper**() const;

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

COLOR

typedef struct **rgb_color** <be/interface/GraphicsDefs.h>

uint8 **red**, **green**, **blue**, **alpha**;

Transparency Values <be/interface/GraphicsDefs.h>
B_TRANSPARENT_COLOR (rgb_color value)
B_TRANSPARENT_MAGIC_CMAP8 (uint value)
B_TRANSPARENT_MAGIC_RGBA15 (uint16)
B_TRANSPARENT_MAGIC_RGBA15_BIG (uint16)
B_TRANSPARENT_MAGIC_RGBA32 (uint32)
B_TRANSPARENT_MAGIC_RGBA32_BIG (uint32)

typedef struct **color_map** <be/interface/GraphicsDefs.h>

int32 **id**;
rgb_color **color_list**[256];
uint8 **inversion_map**[256], **index_map**[32768];

typedef enum 'color_space' <be/interface/GraphicsDefs.h>
B_NO_COLOR_SPACE
Bitmap Formats (little-endian order)
Note: Append '_BIG' to any of these six names for big-endian format bitmaps. Names appended with '_LITTLE' are also provided for completeness, but are duplicates of these same values.
B_RGB32 (xRGB 8:8:8:8)
B_RGBA32 (ARGB 8:8:8:8)
B_RGB24 (currently unused)
B_RGB16 (RGB 5:6:5)
B_RGB15 (xRGB 1:5:5:5)
B_RGBA15 (ARGB 1:5:5:5)
Indexed Color and Grayscale Bitmap Formats (endian independent)
B_CMAP8 (256 color indexed)
B_GRAY8 (256 shade gray value)
B_GRAY1 (1 bit/pixel, black or white)

Non-linear Color Spaces <be/interface/GraphicsDefs.h>	
Note: These may not be supported for BBitmaps.	
B_YCbCr422	B_HSI24
B_YCbCr411	B_HSI32
B_YCbCr444	B_HSIA32
B_YCbCr420	B_HSV24
B_HLS24	B_HSV32
B_HLS32	B_HSVA32
B_HLSA32	B_CMY24
B_YUV9	B_CMY32
B_YUV12	B_CMYA32
B_UVL24	B_CMYK32
B_UVL32	B_YUV422
B_UVLA32	B_YUV411
B_LAB24	B_YUV444
B_LAB32	B_YUV420
B_LABA32	

Color Space Supported Codes <be/interface/GraphicsDefs
B_VIEWS_SUPPORT_DRAW_BITMAP
B_BITMAPS_SUPPORT_ATTACHED_VIEWS

Graphics Global Functions <be/interface/GraphicsDefs.h>

bool **bitmaps_support_space**(color_space space, uint32 *support_flags);
status_t **get_pixel_size_for**(color_space space, size_t *pixel_chunk, size_t * row_alignment, size_t *pixels_per_chunk);

USER INTERFACE GLOBAL COLORS

enum color_which <be/interface/InterfaceDefs.h>
B_PANEL_BACKGROUND_COLOR
B_MENU_BACKGROUND_COLOR
B_WINDOW_TAB_COLOR
B_KEYBOARD_NAVIGATION_COLOR
B_DESKTOP_COLOR

tint_color() Codes <be/interface/InterfaceDefs.h>
B_LIGHTEN_MAX_TINT (0.0)
B_LIGHTEN_2_TINT (0.385)
B_LIGHTEN_1_TINT (0.590)
B_NO_TINT (1.0)
B_DARKEN_1_TINT (1.147)
B_DARKEN_2_TINT (1.295)
B_DARKEN_3_TINT (1.407)
B_DARKEN_4_TINT (1.555)
B_DARKEN_MAX_TINT (2.0)
B_DISABLED_LABEL_TINT (B_DARKEN_3_TINT)
B_HIGHLIGHT_BACKGROUND_TINT (B_DARKEN_2_TINT)
B_DISABLED_MARK_TINT (B_LIGHTEN_2_TINT)

UI Color Global Functions <be/interface/InterfaceDefs.h>
rgb_color **ui_color**(color_which which);
rgb_color **tint_color**(rgb_color color, float tint);

INTERFACE CODES AND TYPES

enum 'border_style' <be/interface/InterfaceDefs.h>	
B_PLAIN_BORDER	B_NO_BORDER
B_FANCY_BORDER	

enum 'orientation' <be/interface/InterfaceDefs.h>	
B_HORIZONTAL	B_VERTICAL

enum 'button_width' <be/interface/InterfaceDefs.h>	
B_WIDTH_AS_USUAL	B_WIDTH_FROM_LABEL
B_WIDTH_FROM_WIDEST	

enum 'bitmap_tiling' <be/interface/Input.h>	
B_TILE_BITMAP_X	B_TILE_BITMAP
B_TILE_BITMAP_Y	

enum 'alignment' <be/interface/InterfaceDefs.h>	
B_ALIGN_LEFT	B_ALIGN_CENTER
B_ALIGN_RIGHT	

enum 'vertical_alignment' <be/interface/InterfaceDefs.h>	
B_ALIGN_TOP	B_ALIGN_BOTTOM
B_ALIGN_MIDDLE	B_ALIGN_NO_VERTICAL

Font Property Bit Codes <be/interface/View.h>	
B_FONT_FAMILY_AND_STYLE	B_FONT_ENCODING
B_FONT_SIZE	B_FONT_FACE
B_FONT_SHEAR	B_FONT_FLAGS
B_FONT_ROTATION	B_FONT_ALL
B_FONT_SPACING	

Interface Kit

<be/InterfaceKit.h> <libbe.so>

INTERFACE GLOBAL SETTINGS

Screen Size Bit Codes <be/interface/GraphicsDefs.h>	
Note: For set_screen_space( ) and get_screen_info( ).	
B_8_BIT_640x400	B_16_BIT_640x480
B_8_BIT_640x480	B_16_BIT_800x600
B_8_BIT_800x600	B_16_BIT_1024x768
B_8_BIT_1024x768	B_16_BIT_1152x900
B_8_BIT_1152x900	B_16_BIT_1280x1024
B_8_BIT_1280x1024	B_16_BIT_1600x1200
B_8_BIT_1600x1200	B_32_BIT_640x480
B_15_BIT_640x480	B_32_BIT_800x600
B_15_BIT_800x600	B_32_BIT_1024x768
B_15_BIT_1024x768	B_32_BIT_1152x900
B_15_BIT_1280x1024	B_32_BIT_1280x1024
B_15_BIT_1600x1200	B_32_BIT_1600x1200
B_15_BIT_1152x900	

struct **scroll_bar_info** <be/interface/InterfaceDefs.h>

bool **proportional**, **double_arrows**;
int32 **knob**, **min_knob_size**;

Interface Kit Global Functions be/interface/InterfaceDefs.h>

status_t **get_deskbar_frame**(BRect *frame);
const color_map ***system_colors**();
status_t **set_screen_space**(int32 index, uint32 res, bool stick = true);

status_t **get_scroll_bar_info**(scroll_bar_info *info);
status_t **set_scroll_bar_info**(scroll_bar_info *info);

int32 **count_workspaces**();
void **set_workspace_count**(int32 count);
int32 **current_workspace**();
void **activate_workspace**(int32 workspace);

bigtime_t **idle_time**();

void **run_select_printer_panel**();
void **run_add_printer_panel**();

void **set_focus_follows_mouse**(bool follow);
bool **focus_follows_mouse**();

MOUSE GLOBAL SETTINGS

struct **mouse_map** <be/interface/InterfaceDefs.h>

uint32 **left**, **right**, **middle**;

Global Mouse Functions <be/interface/InterfaceDefs.h>
status_t **get_mouse_type**(int32 *type);
status_t **set_mouse_type**(int32 type);
status_t **get_mouse_map**(mouse_map *map);
status_t **set_mouse_map**(mouse_map *map);
status_t **get_click_speed**(bigtime_t *speed);
status_t **set_click_speed**(bigtime_t speed);
status_t **get_mouse_speed**(int32 *speed);
status_t **set_mouse_speed**(int32 speed);

KEYBOARD AND KEYS

Key Tables <be/interface/InterfaceDefs.h>	
B_CONTROL_TABLE	B_CAPS_SHIFT_TABLE
B_OPTION_CAPS_SHIFT_TABLE	B_CAPS_TABLE
B_OPTION_CAPS_TABLE	B_SHIFT_TABLE
B_OPTION_SHIFT_TABLE	B_NORMAL_TABLE
B_OPTION_TABLE	

Key Modifier Codes <be/interface/InterfaceDefs.h>	
B_SHIFT_KEY	B_LEFT_SHIFT_KEY
B_COMMAND_KEY	B_RIGHT_SHIFT_KEY
B_CONTROL_KEY	B_LEFT_COMMAND_KEY
B_CAPS_LOCK	B_RIGHT_COMMAND_KEY
B_SCROLL_LOCK	B_LEFT_CONTROL_KEY
B_NUM_LOCK	B_RIGHT_CONTROL_KEY
B_OPTION_KEY	B_LEFT_OPTION_KEY
B_MENU_KEY	B_RIGHT_OPTION_KEY

Common Character Code Strings <be/interface/InterfaceDe	
Note: Each of these are three-byte long strings.	
B_UTF8_ELLIPSIS	B_UTF8_REGISTERED
B_UTF8_OPEN_QUOTE	B_UTF8_TRADEMARK
B_UTF8_CLOSE_QUOTE	B_UTF8_SMILING_FACE
B_UTF8_COPYRIGHT	B_UTF8_HIROSHI

Key Character Constants <be/interface/InterfaceDefs.h>	
B_BACKSPACE (0x08)	B_LEFT_ARROW (0x1c)
B_RETURN (0x0a)	B_RIGHT_ARROW (0x1d)
B_ENTER (0x0a)	B_UP_ARROW (0x1e)
B_SPACE (0x20)	B_DOWN_ARROW (0x1f)
B_TAB (0x09)	B_INSERT (0x05)
B_ESCAPE (0x1b)	B_DELETE (0x7f)
B_SUBSTITUTE (0x1a)	B_HOME (0x01)
B_PAGE_UP (0x0b)	B_END (0x04)
B_PAGE_DOWN (0x0c)	B_FUNCTION_KEY (0x10)

Function Key Codes <be/interface/InterfaceDefs.h>	
B_F1_KEY (0x02)	B_F9_KEY (0x0a)
B_F2_KEY (0x03)	B_F10_KEY (0x0b)
B_F3_KEY (0x04)	B_F11_KEY (0x0c)
B_F4_KEY (0x05)	B_F12_KEY (0x0d)
B_F5_KEY (0x06)	B_PRINT_KEY (0x0e)
B_F6_KEY (0x07)	B_SCROLL_KEY (0x0f)
B_F7_KEY (0x08)	B_PAUSE_KEY (0x10)
B_F8_KEY (0x09)	

struct **key_info** <be/interface/InterfaceDefs.h>

uint32 **modifiers**;
uint8 **key_states**[16];

struct **key_map** <be/interface/InterfaceDefs.h>

uint32 **version**;
uint32 **caps_key**, **scroll_key**, **num_key**, **left_shift_key**, **right_shift_key**, **left_command_key**, **right_command_key**, **left_control_key**, **right_control_key**, **left_option_key**, **right_option_key**, **menu_key**, **lock_settings**;
int32 **control_map**[128], **option_caps_shift_map**[128], **option_caps_map**[128], **option_shift_map**[128], **option_map**[128], **caps_shift_map**[128], **caps_map**[128], **shift_map**[128], **normal_map**[128], **acute_dead_key**[32], **grave_dead_key**[32], **circumflex_dead_key**[32], **dieresis_dead_key**[32], **tilde_dead_key**[32];
uint32 **acute_tables**, **grave_tables**, **circumflex_tables**, **dieresis_tables**, **tilde_tables**;

Keyboard Global Functions <be/interface/InterfaceDefs.h>
status_t **get_key_repeat_rate**(int32 *rate);
status_t **set_key_repeat_rate**(int32 rate);
status_t **get_key_repeat_delay**(bigtime_t *delay);
status_t **set_key_repeat_delay**(bigtime_t delay);
uint32 **modifiers**();
status_t **get_key_info**(key_info *info);
void **get_key_map**(key_map **map, char **key_buffer);
status_t **get_keyboard_id**(uint16 *id);
void **set_modifier_key**(uint32 modifier, uint32 key);
void **set_keyboard_locks**(uint32 modifiers);
rgb_color **keyboard_navigation_color**();

INPUT

Functions and classes to manage input devices.

enum 'input_method_op' <be/interface/Input.h>
B_INPUT_METHOD_STARTED
B_INPUT_METHOD_STOPPED
B_INPUT_METHOD_CHANGED
B_INPUT_METHOD_LOCATION_REQUEST

enum 'input_device_type' <be/interface/Input.h>
B_POINTING_DEVICE
B_KEYBOARD_DEVICE
B_UNDEFINED_DEVICE

enum 'input_device_notification' Bit Codes <be/interface/I
B_INPUT_DEVICE_ADDED
B_INPUT_DEVICE_STARTED
B_INPUT_DEVICE_STOPPED
B_INPUT_DEVICE_REMOVED

Input Device Global Functions <be/interface/Input.h>
BInputDevice* **find_input_device**(const char *name);
status_t **get_input_devices**(BList *list);
status_t **watch_input_devices**(BMessenger target, bool start);

class **BInputDevice** <be/interface/Input.h>

~**BInputDevice**();

const char* **Name**() const;
input_device_type **Type**() const;
bool **IsRunning**() const;

status_t **Start**();
status_t **Stop**();
status_t **Control**(uint32 code, BMessage *message);

static status_t **Start**(input_device_type type);
static status_t **Stop**(input_device_type type);
static status_t **Control**(input_device_type type, uint32 code, BMessage *message);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

DISPLAY AND DRAWING

class **BBitmap** <be/interface/Bitmap.h>

: public BArchivable

BBitmap(BRect bounds, color_space depth, bool accepts_views = false, bool need_contiguous = false);
virtual ~**BBitmap**();

bool **IsValid**() const;
void **SetBits**(const void *data, int32 length, int32 offset, color_space cs);
void ***Bits**() const;
int32 **BitsLength**() const;
int32 **BytesPerRow**() const;
color_space **ColorSpace**() const;
BRect **Bounds**() const;

virtual void **AddChild**(BView *view);
virtual bool **RemoveChild**(BView *view);
int32 **CountChildren**() const;
BView ***ChildAt**(int32 index) const;

BView ***FindView**(const char *view_name) const;
BView ***FindView**(BPoint point) const;

bool **Lock**();
void **Unlock**();
bool **IsLocked**() const;

class **BPicture** <be/interface/Picture.h>

: public BArchivable

BPicture();
BPicture(const BPicture &original);
virtual ~**BPicture**();

virtual status_t **Perform**(perform_code d, void *arg);
status_t **Play**(void **callBackTable, int32 tableEntries, void *userData);

status_t **Flatten**(BDataIO *stream);
status_t **Unflatten**(BDataIO *stream);

class **BPoint** <be/interface/Point.h>

BPoint();
BPoint(float X, float Y);
BPoint(const BPoint& pt);

float **x**, **y**;

BPoint operator**+**(const BPoint&) const;
BPoint operator**-**(const BPoint&) const;
BPoint& operator**+=**(const BPoint&);
BPoint& operator**-=**(const BPoint&);
bool operator**!=**(const BPoint&) const;
bool operator**==**(const BPoint&) const;
BPoint &operator**=**(const BPoint &from);

void **Set**(float X, float Y);
void **ConstrainTo**(BRect rect);
void **PrintToStream**() const;

BPoint Global Constants <be/interface/Point.h>

const BPoint **B_ORIGIN**;

class **BRect** <be/interface/Rect.h>

BRect();
BRect(const BRect &);
BRect(float l, float t, float r, float b);
BRect(BPoint leftTop, BPoint rightBottom);

float **left**, **top**, **right**, **bottom**;

BRect &operator**=**(const BRect &from);
void **Set**(float l, float t, float r, float b);

void **PrintToStream**() const;

BPoint selectors...
BPoint **LeftTop**() const;
BPoint **RightBottom**() const;
BPoint **LeftBottom**() const;
BPoint **RightTop**() const;

BPoint setters...
void **SetLeftTop**(const BPoint);
void **SetRightBottom**(const BPoint);
void **SetLeftBottom**(const BPoint);
void **SetRightTop**(const BPoint);

Transformation...
void **InsetBy**(BPoint);
void **InsetBy**(float dx, float dy);
void **OffsetBy**(BPoint);
void **OffsetBy**(float dx, float dy);
void **OffsetTo**(BPoint);
void **OffsetTo**(float x, float y);

Expression transformations...
BRect & **InsetBySelf**(BPoint);
BRect & **InsetBySelf**(float dx, float dy);
BRect **InsetByCopy**(BPoint);
BRect **InsetByCopy**(float dx, float dy);
BRect &**OffsetBySelf**(BPoint);
BRect &**OffsetBySelf**(float dx, float dy);
BRect **OffsetByCopy**(BPoint);
BRect **OffsetByCopy**(float dx, float dy);
BRect &**OffsetToSelf**(BPoint);
BRect &**OffsetToSelf**(float dx, float dy);
BRect **OffsetToCopy**(BPoint);
BRect **OffsetToCopy**(float dx, float dy);

Comparison...
bool operator**==**(BRect) const;
bool operator**!=**(BRect) const;

Intersection and union...
BRect operator**&**(BRect) const;
BRect operator**|**(BRect) const;

Utilities...
bool **Intersects**(BRect r) const;
bool **IsValid**() const;
float **Width**() const;
int32 **IntegerWidth**() const;
float **Height**() const;
int32 **IntegerHeight**() const;
bool **Contains**(BPoint) const;
bool **Contains**(BRect) const;

class **BRegion** <be/interface/Region.h>

BRegion([const BRegion &region]);
virtual ~**BRegion**();

BRegion &operator**=**(const BRegion &from);

BRect **Frame**() const;
BRect **RectAt**(int32 index);
int32 **CountRects**();
void **Set**(BRect newBounds);
bool **Intersects**(BRect r) const;
bool **Contains**(BPoint pt) const;
void **PrintToStream**() const;
void **OffsetBy**(int32 dh, int32 dv);
void **MakeEmpty**();

void **Include**(BRect r);
void **Include**(const BRegion*);

void **Exclude**(BRect r);
void **Exclude**(const BRegion*);

void **IntersectWith**(const BRegion*);

class **BPolygon** <be/interface/Polygon.h>

BPolygon();
BPolygon(const BPoint *ptArray, int32 numPoints);
BPolygon(const BPolygon *poly);
virtual ~**BPolygon**();

BPolygon &operator**=**(const BPolygon &from);

BRect **Frame**() const;
void **AddPoints**(const BPoint *ptArray, int32 numPoints);
int32 **CountPoints**() const;
void **MapTo**(BRect srcRect, BRect dstRect);
void **PrintToStream**() const;

class **BShapeIterator** <be/interface/Shape.h>

BShapeIterator();
virtual ~**BShapeIterator**();

virtual status_t **IterateMoveTo**(BPoint *point);
virtual status_t **IterateLineTo**(int32 lineCount, BPoint *linePts);
virtual status_t **IterateBezierTo**(int32 bezierCount, BPoint *bezierPts);
virtual status_t **IterateClose**();
status_t **Iterate**(BShape *shape);

class **BShape** <be/interface/Shape.h>

: BArchivable

BShape([BShape &copyFrom]);
virtual ~**BShape**();

void **Clear**();
BRect **Bounds**();
status_t **AddShape**(BShape *other);
status_t **MoveTo**(BPoint point);
status_t **LineTo**(BPoint linePoint);
status_t **BezierTo**(BPoint controlPoints[3]);
status_t **Close**();

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

FONTS

Font Sizes <be/interface/Font.h>
B_FONT_FAMILY_LENGTH (63)
B_FONT_SYLE_LENGTH (63)

Font Typedefs <be/interface/Font.h>

typedef char **font_family**[B_FONT_FAMILY_LENGTH + 1];
typedef char **font_style**[B_FONT_STYLE_LENGTH + 1];

Spacing Codes <be/interface/Font.h>	
B_CHAR_SPACING	B_STRING_SPACING
B_BITMAP_SPACING	B_FIXED_SPACING

enum 'font_direction' <be/interface/Font.h>	
B_FONT_LEFT_TO_RIGHT	B_FONT_RIGHT_TO_LEFT

Antialiasing Bit Codes <be/interface/Font.h>	
B_DISABLE_ANTIALIASING	B_FORCE_ANTIALIASING

Truncate Codes <be/interface/Font.h>	
B_TRUNCATE_END	B_TRUNCATE_BEGINNING
B_TRUNCATE_MIDDLE	B_TRUNCATE_SMART

Encoding Codes <be/interface/Font.h>	
B_UNICODE_UTF8	B_ISO_8859_6
B_ISO_8859_1	B_ISO_8859_7
B_ISO_8859_2	B_ISO_8859_8
B_ISO_8859_3	B_ISO_8859_9
B_ISO_8859_4	B_ISO_8859_10
B_ISO_8859_5	B_MACINTOSH_ROMAN

Cache Bit Codes <be/interface/Font.h>
B_SCREEN_FONT_CACHE
B_PRINTING_FONT_CACHE
B_DEFAULT_CACHE_SETTING
B_APP_CACHE_SETTING

Screen Display Tuning Codes <be/interface/Font.h>	
B_HAS_TUNED_FONT	B_IS_FIXED

Style Bit Codes <be/interface/Font.h>	
B_ITALIC_FACE	B_UNDERSCORE_FACE
B_NEGATIVE_FACE	B_OUTLINED_FACE
B_STRIKEOUT_FACE	B_BOLD_FACE
B_REGULAR_FACE	

enum 'font_metric_mode' <be/interface/Font.h>	
B_SCREEN_METRIC	B_PRINTING_METRIC

enum 'font_file_format' <be/interface/Font.h>
B_TRUETYPE_WINDOWS
B_POSTSCRIPT_TYPE1_WINDOWS

struct **edge_info** <be/interface/Font.h>

float **left**, **right**;

struct **font_height** <be/interface/Font.h>

float **ascent**, **descent**, **leading**;

struct **escapement_delta** <be/interface/Font.h>

float **nonspace**, **space**;

struct **font_cache_info** <be/interface/Font.h>

int32 **sheared_font_penalty**, **rotated_font_penalty**;
float **oversize_threshold**;
int32 **oversize_penalty**;
int32 **cache_size**;
float **spacing_size_threshold**;

struct **tuned_font_info** <be/interface/Font.h>

float **size**, **shear**, **rotation**;
uint32 **flags**;
uint16 **face**;

class **BFont** <be/interface/Font.h>

BFont();
BFont(const BFont &font);
BFont(const BFont *font);

BFont& operator**=**(const BFont &font);
bool operator**==**(const BFont &font) const;
bool operator**!=**(const BFont &font) const;

void **SetFamilyAndStyle**(const font_family family, const font_style style);
void **SetFamilyAndStyle**(uint32 code);
void **SetFamilyAndFace**(const font_family family, uint16 face);
void **SetSize**(float size);
void **SetShear**(float shear);
void **SetRotation**(float rotation);
void **SetSpacing**(uint8 spacing);
void **SetEncoding**(uint8 encoding);
void **SetFace**(uint16 face);
void **SetFlags**(uint32 flags);

void GetFamilyAndStyle(font_family *family, font_style *style) const;
uint32 **FamilyAndStyle**() const;
float **Size**() const;
float **Shear**() const;
float **Rotation**() const;
uint8 **Spacing**() const;
uint8 **Encoding**() const;
uint16 **Face**() const;
uint32 **Flags**() const;

font_direction **Direction**() const;
bool **IsFixed**() const;
bool **IsFullAndHalfFixed**() const;
BRect **BoundingBox**() const;
unicode_block **Blocks**() const;
font_file_format **FileFormat**() const;

int32 **CountTuned**() const;
void **GetTunedInfo**(int32 index, tuned_font_info *info) const;

void **GetTruncatedStrings**(const char *stringArray[], int32 numStrings, uint32 mode, float width, char *resultArray[]) const;

float **StringWidth**(const char *string) const;
float **StringWidth**(const char *string, int32 length) const;
void **GetStringWidths**(const char *stringArray[], const int32 lengthArray[], int32 numStrings, float widthArray[]) const;

void **GetEscapements**(const char charArray[], int32 numChars, float escapementArray[]) const;
void **GetEscapements**(const char charArray[], int32 numChars, escapement_delta *delta, float escapementArray[]) const;
void **GetEscapements**(const char charArray[], int32 numChars, escapement_delta *delta, BPoint escapementArray[]) const;
void **GetEscapements**(const char charArray[], int32 numChars, escapement_delta *delta, BPoint escapementArray[], BPoint offsetArray[]) const;

void **GetEdges**(const char charArray[], int32 numBytes, edge_info edgeArray[]) const;
void **GetHeight**(font_height *height) const;

void **GetBoundingBoxesAsGlyphs**(const char charArray[], int32 numChars, font_metric_mode mode, BRect boundingBoxArray[]) const;
void **GetBoundingBoxesAsString**(const char charArray[], int32 numChars, font_metric_mode mode, escapement_delta *delta, BRect boundingBoxArray[]) const;
void **GetBoundingBoxesForStrings**(const char *stringArray[], int32 numStrings,font_metric_mode mode, escapement_delta deltas[], BRect boundingBoxArray[]) const;

void **GetGlyphShapes**(const char charArray[], int32 numChars, BShape *glyphShapeArray[]) const;
void **GetHasGlyphs**(const char charArray[], int32 numChars, bool hasArray[]) const;

void **PrintToStream**() const;

Font Global Objects <be/interface/Font.h>

const BFont* **be_plain_font**, ***be_bold_font**, ***be_fixed_font**;

Font Global Functions <be/interface/Font.h>

int32 **count_font_families**();
status_t **get_font_family**(int32 index, font_family *name, uint32 *flags = NULL);

int32 **count_font_styles**(font_family name);
status_t **get_font_style**(font_family family, int32 index, font_style *name, uint32 *flags = NULL);
status_t **get_font_style**(font_family family, int32 index, font_style *name, uint16 *face, uint32 *flags = NULL);

bool **update_font_families**(bool check_only);

status_t **get_font_cache_info**(uint32 id, void *set);
status_t **set_font_cache_info**(uint32 id, void *set);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

UNICODE

class **unicode_block** <be/interface/Font.h>

unicode_block([uint64 block2, uint64 block1]);

bool **Includes**(const unicode_block &block) const;

unicode_block operator**&**(const unicode_block &block) const;
unicode_block operator**|**(const unicode_block &block) const;
unicode_block &operator**=**(const unicode_block &block);
bool operator**==**(const unicode_block &block) const;
bool operator**!=**(const unicode_block &block) const;

Unicode Block Object <be/interface/UnicodeBlockObjects.h>

Note: These are all objects of type unicode_block, initialized with the listed values range.

B_BASIC_LATIN_BLOCK (0000-007F)
B_LATIN1_SUPPLEMENT_BLOCK (0080-00FF
B_LATIN_EXTENDED_A_BLOCK (0100-017F)
B_LATIN_EXTENDED_B_BLOCK (0180-024F)
B_IPA_EXTENSIONS_BLOCK (0250-02AF)
B_SPACING_MODIFIER_LETTERS_BLOCK (02B0-02FF)
B_COMBINING_DIACRITICAL_MARKS_BLOCK (0300-036F)
B_BASIC_GREEK_BLOCK (0370-03CF)
B_GREEK_SYMBOLS_AND_COPTIC_BLOCK (03D0-03FF)
B_CYRILLIC_BLOCK (0400-04FF)
B_ARMENIAN_BLOCK (0530-058F)
B_BASIC_HEBREW_BLOCK (0590-05CF)
B_HEBREW_EXTENDED_BLOCK (05D0-05FF)
B_BASIC_ARABIC_BLOCK (0600-0670)
B_ARABIC_EXTENDED_BLOCK (0671-06FF)
B_DEVANAGARI_BLOCK (0900-097F)
B_BENGALI_BLOCK (0980-09FF)
B_GURMUKHI_BLOCK (0A00-0A7F)
B_GUJARATI_BLOCK (0A80-0AFF)
B_ORIYA_BLOCK (0B00-0B7F)
B_TAMIL_BLOCK (0B80-0BFF)
B_TELUGU_BLOCK (0C00-0C7F)
B_KANNADA_BLOCK (0C80-0CFF)
B_MALAYALAM_BLOCK (0D00-0D7F)
B_THAI_BLOCK (0E00-0E7F)
B_LAO_BLOCK (0E80-0EFF)
B_BASIC_GEORGIAN_BLOCK (10A0-10CF)
B_GEORGIAN_EXTENDED_BLOCK (10D0-10FF)
B_HANGUL_JAMO_BLOCK (1100-11FF)
B_LATIN_EXTENDED_ADDITIONAL_BLOCK (1E00-1EFF)
B_GREEK_EXTENDED_BLOCK (1F00-1FFF)
B_GENERAL_PUNCTUATION_BLOCK (2000-206F)
B_SUPERSCRIPTS_AND_SUBSCRIPTS_BLOCK (2070-209F)
B_CURRENCY_SYMBOLS_BLOCK (20A0-20CF)
B_COMBINING_MARKS_FOR_SYMBOLS_BLOCK (20D0-20FF)
B_LETTERLIKE_SYMBOLS_BLOCK (2100-214F)
B_NUMBER_FORMS_BLOCK (2150-218F)
B_ARROWS_BLOCK (2190-21FF)
B_MATHEMATICAL_OPERATORS_BLOCK (2200-22FF)
B_MISCELLANEOUS_TECHNICAL_BLOCK (2300-23FF)
B_CONTROL_PICTURES_BLOCK (2400-243F)
B_OPTICAL_CHARACTER_RECOGNITION_BLOCK (2440-245F)
B_ENCLOSED_ALPHANUMERICS_BLOCK (2460-24FF)
B_BOX_DRAWING_BLOCK (2500-257F)
B_BLOCK_ELEMENTS_BLOCK (2580-259F)
B_GEOMETRIC_SHAPES_BLOCK (25A0-25FF)
B_MISCELLANEOUS_SYMBOLS_BLOCK (2600-26FF)
B_DINGBATS_BLOCK (2700-27BF)
B_CJK_SYMBOLS_AND_PUNCTUATION_BLOCK (3000-303F)
B_HIRAGANA_BLOCK (3040-309F)
B_KATAKANA_BLOCK (30A0-30FF)
B_BOPOMOFO_BLOCK (3100-312F)
B_HANGUL_COMPATIBILITY_JAMO_BLOCK (3130-318F)
B_CJK_MISCELLANEOUS_BLOCK (3190-319F)
B_ENCLOSED_CJK_LETTERS_AND_MONTHS_BLOCK (3200-32FF)
B_CJK_COMPATIBILITY_BLOCK (3300-33FF)
B_HANGUL_BLOCK (AC00-D7AF)
B_HIGH_SURROGATES_BLOCK (D800-DBFF)
B_LOW_SURROGATES_BLOCK (DC00-DFFF)
B_CJK_UNIFIED_IDEOGRAPHS_BLOCK (4E00-9FFF)
B_PRIVATE_USE_AREA_BLOCK (E000-F8FF)
B_CJK_COMPATIBILITY_IDEOGRAPHS_BLOCK (F900-FAFF)
B_ALPHABETIC_PRESENTATION_FORMS_BLOCK (FB00-FB4F)
B_ARABIC_PRESENTATION_FORMS_A_BLOCK (FB50-FDFF)
B_COMBINING_HALF_MARKS_BLOCK (FE20-FE2F)
B_CJK_COMPATIBILITY_FORMS_BLOCK (FE30-FE4F)
B_SMALL_FORM_VARIANTS_BLOCK (FE50-FE6F)
B_ARABIC_PRESENTATION_FORMS_B_BLOCK (FE70-FEFE)
B_HALFWIDTH_AND_FULLWIDTH_FORMS_BLOCK (FF00-FFEF)
B_SPECIALS_BLOCK (FEFF and FFF0-FFFF)
B_TIBETAN_BLOCK (0F00-0FBF)

PRINTING

Printing Error Values <be/support/Errors.h>

B_NO_PRINT_SERVER

struct **print_file_header** <be/interface/PrintJob.h>

int32 **version**;
int32 **page_count**;
off_t **first_page**;

class **BPrintJob** <be/interface/PrintJob.h>

BPrintJob(const char *job_name);
virtual ~**BPrintJob**();

int32 **ConfigPage**();
int32 **ConfigJob**();
virtual void **DrawView**(BView *a_view, BRect a_rect, BPoint where);
void **CommitJob**();
int32 **FirstPage**();
int32 **LastPage**();
BRect **PaperRect**();
BRect **PrintableRect**();
bool **CanContinue**();
void **BeginJob**();
void **SpoolPage**();
BMessage ***Settings**();
void **SetSettings**(BMessage *a_msg);
void **CancelJob**();

SCREENS

struct **screen_id** <be/interface/Screen.h>

int32 **id**;

const screen_id **B_MAIN_SCREEN_ID**;

class **BScreen** <be/interface/Screen.h>

BScreen(screen_id id=B_MAIN_SCREEN_ID);
BScreen(BWindow *win);
~**BScreen**();

bool **IsValid**();
status_t **SetToNext**();

color_space **ColorSpace**();
BRect **Frame**();
screen_id **ID**();

status_t **WaitForRetrace**();
status_t **WaitForRetrace**(bigtime_t timeout);

uint8 **IndexForColor**(rgb_color rgb);
uint8 **IndexForColor**(uint8 r, uint8 g, uint8 b, uint8 a=255);

rgb_color **ColorForIndex**(const uint8 index);
uint8 **InvertIndex**(uint8 index);

const color_map *__ColorMap__();

status_t **GetBitmap**(BBitmap **screen_shot, bool draw_cursor = true, BRect *bound = NULL);
status_t **ReadBitmap**(BBitmap *buffer, bool draw_cursor = true, BRect *bound = NULL);

rgb_color **DesktopColor**();
void **SetDesktopColor**(rgb_color rgb, bool stick=true);

status_t **GetModeList**(display_mode **mode_list, uint32 *count);
status_t **GetMode**(display_mode *mode);
status_t **SetMode**(display_mode *mode, bool makeDefault = false);
status_t **GetDeviceInfo**(accelerant_device_info *adi);
status_t **GetPixelClockLimits**(display_mode *mode, uint32 *low, uint32 *high);
status_t **GetTimingConstraints**(display_timing_constraints *dtc);
status_t **SetDPMS**(uint32 dpms_state);
uint32 **DPMSState**(void);
uint32 **DPMSCapabilites**(void);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

BWindow

enum 'window_type' <be/interface/Window.h>	
B_UNTYPED_WINDOW	B_DOCUMENT_WINDOW
B_TITLED_WINDOW	B_BORDERED_WINDOW
B_MODAL_WINDOW	B_FLOATING_WINDOW

enum 'window_look' <be/interface/Window.h>
B_BORDERED_WINDOW_LOOK
B_NO_BORDER_WINDOW_LOOK
B_TITLED_WINDOW_LOOK
B_DOCUMENT_WINDOW_LOOK
B_MODAL_WINDOW_LOOK
B_FLOATING_WINDOW_LOOK

enum 'window_feel' <be/interface/Window.h>
B_NORMAL_WINDOW_FEEL
B_MODAL_SUBSET_WINDOW_FEEL
B_MODAL_APP_WINDOW_FEEL
B_MODAL_ALL_WINDOW_FEEL
B_FLOATING_SUBSET_WINDOW_FEEL
B_FLOATING_APP_WINDOW_FEEL
B_FLOATING_ALL_WINDOW_FEEL

enum 'window_alignment' <be/interface/Window.h>	
B_BYTE_ALIGNMENT	B_PIXEL_ALIGNMENT

Window Flag Bit Codes <be/interface/Window.h>
B_NOT_MOVABLE
B_NOT_CLOSABLE
B_NOT_ZOOMABLE
B_NOT_MINIMIZABLE
B_NOT_RESIZABLE
B_NOT_H_RESIZABLE
B_NOT_V_RESIZABLE
B_AVOID_FRONT
B_AVOID_FOCUS
B_WILL_ACCEPT_FIRST_CLICK
B_OUTLINE_RESIZE
B_NO_WORKSPACE_ACTIVATION
B_NOT_ANCHORED_ON_ACTIVATE
B_ASYNCHRONOUS_CONTROLS

Workspace Constants <be/interface/Window.h>
B_CURRENT_WORKSPACE (0)
B_ALL_WORKSPACES (0xffffffff)

class **BWindow** <be/interface/Window.h>

: public BLooper

BWindow(BRect frame, const char *title, window_type type, uint32 flags, uint32 workspace = B_CURRENT_WORKSPACE);
BWindow(BRect frame, const char *title, window_look look, window_feel feel, uint32 flags, uint32 workspace = B_CURRENT_WORKSPACE);
virtual ~**BWindow**();

void **AddChild**(BView *child, BView *before = NULL);
bool **RemoveChild**(BView *child);
int32 **CountChildren**() const;
BView ***ChildAt**(int32 index) const;

virtual void **FrameMoved**(BPoint new_position);
virtual void **WorkspacesChanged**(uint32 old_ws, uint32 new_ws);
virtual void **WorkspaceActivated**(int32 ws, bool state);
virtual void **FrameResized**(float new_width, float new_height);
virtual void **Minimize**(bool minimize);
virtual void **Zoom**([BPoint rec_position, float rec_width, float rec_height]);
void **SetZoomLimits**(float max_h, float max_v);
virtual void **ScreenChanged**(BRect screen_size, color_space depth);
void **SetPulseRate**(bigtime_t rate);
bigtime_t **PulseRate**() const;
void **Close**(); *(Synonym of BLooper's Quit())*

void **AddShortcut**(uint32 key, uint32 modifiers, BMessage msg);
void **AddShortcut**(uint32 key, uint32 modifiers, BMessage *msg, BHandler *target);

void **RemoveShortcut**(uint32 key, uint32 modifiers);
void **SetDefaultButton**(BButton *button);
BButton ***DefaultButton**() const;
virtual void **MenusBeginning**();
virtual void **MenusEnded**();
bool **NeedsUpdate**() const;
void **UpdateIfNeeded**();
BView ***FindView**(const char *view_name) const;
BView ***FindView**(BPoint) const;
BView ***CurrentFocus**() const;
void **Activate**(bool = true);
virtual void **WindowActivated**(bool state);

void **ConvertToScreen**(BPoint *pt) const;
BPoint **ConvertToScreen**(BPoint pt) const;
void **ConvertToScreen**(BRect *rect) const;
BRect **ConvertToScreen**(BRect rect) const;

void **ConvertFromScreen**(BPoint *pt) const;
BPoint **ConvertFromScreen**(BPoint pt) const;
void **ConvertFromScreen**(BRect *rect) const;
BRect **ConvertFromScreen**(BRect rect) const;

void **MoveBy**(float dx, float dy);

void **MoveTo**(BPoint);
void **MoveTo**(float x, float y);

void **ResizeBy**(float dx, float dy);
void **ResizeTo**(float width, float height);
virtual void **Show**();
virtual void **Hide**();
bool **IsHidden**() const;
bool **IsMinimized**() const;
status_t **SendBehind**(const BWindow *window);

void **Flush**() const;
void **Sync**() const;

void **Disable/EnableUpdates**();
void **Begin/EndViewTransaction**();
BRect **Bounds**() const;
BRect **Frame**() const;
const char ***Title**() const;
void **SetTitle**(const char *title);
bool **IsFront**() const;
bool **IsActive**() const;
void **SetKeyMenuBar**(BMenuBar *bar);
BMenuBar ***KeyMenuBar**() const;
void **SetSizeLimits**(float min_h, float max_h, float min_v, float max_v);
void **GetSizeLimits**(float *min_h, float *max_h, float *min_v, float *max_v);
uint32 **Workspaces**() const;
void **SetWorkspaces**(uint32);
BView ***LastMouseMovedView**() const;
status_t **AddToSubset**(BWindow *window);
status_t **RemoveFromSubset**(BWindow *window);
virtual status_t **Perform**(perform_code d, void *arg);
status_t **SetType**(window_type type);
window_type **Type**() const;
bool **IsModal**() const;
bool **IsFloating**() const;
status_t **SetWindowAlignment**(window_alignment mode, int32 h, int32 hOffset = 0, int32 width = 0, int32 widthOffset = 0, int32 v = 0, int32 vOffset = 0, int32 height = 0, int32 heightOffset = 0);
status_t **GetWindowAlignment**(window_alignment *mode = NULL, int32 *h = NULL, int32 *hOffset = NULL, int32 *width = NULL, int32 *widthOffset = NULL, int32 *v = NULL, int32 *vOffset = NULL, int32 *height = NULL, int32 *heightOffset = NULL) const;

status_t **SetLook**(window_look look);
window_look **Look**() const;
status_t **SetFeel**(window_feel feel);
window_feel **Feel**() const;
status_t **SetFlags**(uint32);
uint32 **Flags**() const;

BAlert

enum 'alert_type' <be/interface/Alert.h>	
B_EMPTY_ALERT	B_WARNING_ALERT
B_INFO_ALERT	B_STOP_ALERT
B_IDEA_ALERT	

enum 'button_spacing" <be/interface/Alert.h>	
B_EVEN_SPACING	B_OFFSET_SPACING

class **BAlert** <be/interface/Alert.h>

: public BWindow

BAlert(const char *title, const char *text, const char *button1, const char *button2 = NULL, const char *button3 = NULL, button_width width = B_WIDTH_AS_USUAL, alert_type type = B_INFO_ALERT);
BAlert(const char *title, const char *text, const char *button1, const char *button2, const char *button3, button_width width, button_spacing spacing, alert_type type = B_INFO_ALERT);
virtual ~**BAlert**();

void **SetShortcut**(int32 button_index, char key);
char **Shortcut**(int32 button_index) const;
status_t **Go**([BInvoker *invoker]);
BButton ***ButtonAt**(int32 index) const;
BTextView ***TextView**() const;
static BPoint **AlertPosition**(float width, float height);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

VIEWS

Mouse Button Bit Codes <be/interface/View.h>
B_PRIMARY_MOUSE_BUTTON
B_SECONDARY_MOUSE_BUTTON
B_TERTIARY_MOUSE_BUTTON

Cursor Transit Codes <be/interface/View.h>	
B_ENTERED_VIEW	B_EXITED_VIEW
B_INSIDE_VIEW	B_OUTSIDE_VIEW

SetMouseEventMask() Bit Codes <be/interface/View.h>	
B_POINTER_EVENTS	B_KEYBOARD_EVENTS

Event Mask Bit Flags <be/interface/View.h>
B_LOCK_WINDOW_FOCUS
B_SUSPEND_VIEW_FOCUS
B_NO_POINTER_HISTORY

Tracking Codes <be/interface/View.h>	
B_TRACK_WHOLE_RECT	B_TRACK_RECT_CORNER

View Flags Bit Codes <be/interface/View.h>
B_FULL_UPDATE_ON_RESIZE
B_WILL_DRAW
B_PULSE_NEEDED
B_NAVIGABLE_JUMP
B_FRAME_EVENTS
B_NAVIGABLE
B_SUBPIXEL_PRECISE
B_DRAW_ON_CHILDREN
B_INPUT_METHOD_AWARE

Resizing Mode Bit Codes <be/interface/View.h>	
B_FOLLOW_LEFT	B_FOLLOW_TOP
B_FOLLOW_RIGHT	B_FOLLOW_BOTTOM
B_FOLLOW_LEFT_RIGHT	B_FOLLOW_TOP_BOTTOM
B_FOLLOW_H_CENTER	B_FOLLOW_V_CENTER
B_FOLLOW_NONE	B_FOLLOW_ALL

typedef struct **pattern** <be/interface/GraphicsDefs.h>

uint8 data[8];

Pattern Objects <be/interface/GraphicsDefs.h>	
B_SOLID_HIGH	B_MIXED_COLORS
B_SOLID_LOW	

enum 'drawing_mode' <be/interface/GraphicsDefs.h>	
B_OP_COPY	B_OP_OVER
B_OP_ERASE	B_OP_INVERT
B_OP_ADD	B_OP_SUBTRACT
B_OP_BLEND	B_OP_MIN
B_OP_MAX	B_OP_SELECT
B_OP_ALPHA	

enum 'source_alpha' <be/interface/GraphicsDefs.h>	
B_PIXEL_ALPHA	B_CONSTANT_ALPHA

enum 'alpha_function' <be/interface/GraphicsDefs.h>	
B_ALPHA_OVERLAY	B_ALPHA_COMPOSITE

enum 'join_mode' <be/interface/InterfaceDefs.h>	
B_ROUND_JOIN	B_BUTT_JOIN
B_MITER_JOIN	B_SQUARE_JOIN
B_BEVEL_JOIN	

enum 'cap_mode' <be/interface/InterfaceDefs.h>	
B_ROUND_CAP	B_SQUARE_CAP
B_BUTT_CAP	

Limits <be/interface/InterfaceDefs.h>
B_DEFAULT_MITER_LIMIT (10.0)

class **BView** <be/interface/View.h>

: public BHandler

BView(BRect frame, const char *name, uint32 resizeMask, uint32 flags);
virtual ~**BView**();

virtual void **AttachedToWindow**();
virtual void **AllAttached**();
virtual void **DetachedFromWindow**();
virtual void **AllDetached**();

void **AddChild**(BView *child, BView *before = NULL);
bool **RemoveChild**(BView *child);
int32 **CountChildren**() const;
BView ***ChildAt**(int32 index) const;
BView ***NextSibling**() const;
BView ***PreviousSibling**() const;
bool **RemoveSelf**();

BWindow ***Window**() const;

virtual void **Draw**(BRect updateRect);
virtual void **MouseDown**(BPoint where);
virtual void **MouseUp**(BPoint where);
virtual void **MouseMoved**(BPoint where, uint32 code, const BMessage *a_message);
virtual void **WindowActivated**(bool state);
virtual void **KeyDown**(const char *bytes, int32 numBytes);
virtual void **KeyUp**(const char *bytes, int32 numBytes);
virtual void **Pulse**();
virtual void **FrameMoved**(BPoint new_position);
virtual void **FrameResized**(float new_width, float new_height);

virtual void **TargetedByScrollView**(BScrollView *scroll_view);
void **BeginRectTracking**(BRect startRect, uint32 style = B_TRACK_WHOLE_RECT);
void **EndRectTracking**();

void **GetMouse**(BPoint* location, uint32 *buttons, bool checkMessageQueue = true);

void **DragMessage**(BMessage *aMessage, BRect dragRect, BHandler *reply_to = NULL);
void **DragMessage**(BMessage *aMessage, BBitmap *anImage, BPoint offset, BHandler *reply_to = NULL);
void **DragMessage**(BMessage *aMessage, BBitmap *anImage, drawing_mode dragMode, BPoint offset, BHandler *reply_to = NULL);

BView ***FindView**(const char *name) const;
BView ***Parent**() const;
BRect **Bounds**() const;
BRect **Frame**() const;

status_t **SetEventMask**(uint32 mask, uint32 options = 0);
uint32 **EventMask**();
status_t **SetMouseEventMask**(uint32 mask, uint32 options = 0);

virtual void **SetFlags**(uint32 flags);
uint32 **Flags**() const;
virtual void **SetResizingMode**(uint32 mode);
uint32 **ResizingMode**() const;
void **MoveBy**(float dh, float dv);

void **MoveTo**(BPoint where);
void **MoveTo**(float x, float y);

void **ResizeBy**(float dh, float dv);
void **ResizeTo**(float width, float height);
void **ScrollBy**(float dh, float dv);

void **ScrollTo**(float x, float y);
virtual void **ScrollTo**(BPoint where);

virtual void **MakeFocus**(bool focusState = true);
bool **IsFocus**() const;

virtual void **Show**();
virtual void **Hide**();
bool **IsHidden**() const;

void **Flush**() const;
void **Sync**() const;

virtual void **GetPreferredSize**(float *width, float *height);
virtual void **ResizeToPreferred**();

BScrollBar ***ScrollBar**(orientation posture) const;

bool **IsPrinting**() const;
void **SetScale**(float scale) const;
void **Invalidate**([BRect invalRect]);

DRAWING

void **ConvertToScreen**(BPoint* pt) const;
BPoint **ConvertToScreen**(BPoint pt) const;
void **ConvertToScreen**(BRect *r) const;
BRect **ConvertToScreen**(BRect r) const;

void **ConvertFromScreen**(BPoint* pt) const;
BPoint **ConvertFromScreen**(BPoint pt) const;
void **ConvertFromScreen**(BRect *r) const;
BRect **ConvertFromScreen**(BRect r) const;

void **ConvertToParent**(BPoint *pt) const;
BPoint **ConvertToParent**(BPoint pt) const;
void **ConvertToParent**(BRect *r) const;
BRect **ConvertToParent**(BRect r) const;

void **ConvertFromParent**(BPoint *pt) const;
BPoint **ConvertFromParent**(BPoint pt) const;
void **ConvertFromParent**(BRect *r) const;
BRect **ConvertFromParent**(BRect r) const;

BPoint **LeftTop**() const;

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

void **GetClippingRegion**(BRegion *region) const;
virtual void **ConstrainClippingRegion**(BRegion *region);
void **ClipToPicture**(BPicture *picture, BPoint where = B_ORIGIN, bool sync = true);
void **ClipToInversePicture**(BPicture *picture, BPoint where = B_ORIGIN, bool sync = true);

virtual void **SetDrawingMode**(drawing_mode mode);
drawing_mode **DrawingMode**() const;

void **SetBlendingMode**(source_alpha srcAlpha, alpha_function alphaFunc);
void **GetBlendingMode**(source_alpha *srcAlpha, alpha_function *alphaFunc) const;

virtual void **SetPenSize**(float size);
float **PenSize**() const;

virtual void **SetViewColor**(rgb_color c);
void **SetViewColor**(uchar r, uchar g, uchar b, uchar a = 255);
rgb_color **ViewColor**() const;

void **SetViewBitmap**(const BBitmap *bitmap, BRect srcRect, BRect dstRect, uint32 followFlags = B_FOLLOW_TOP|B_FOLLOW_LEFT, uint32 options = B_TILE_BITMAP);
void **SetViewBitmap**(const BBitmap *bitmap, uint32 followFlags = B_FOLLOW_TOP | B_FOLLOW_LEFT, uint32 options = B_TILE_BITMAP);
void **ClearViewBitmap**();

virtual void **SetHighColor**(rgb_color a_color);
void **SetHighColor**(uchar r, uchar g, uchar b, uchar a = 255);
rgb_color **HighColor**() const;

virtual void **SetLowColor**(rgb_color a_color);
void **SetLowColor**(uchar r, uchar g, uchar b, uchar a = 255);
rgb_color **LowColor**() const;

void **SetLineMode**(cap_mode lineCap, join_mode lineJoin, float miterLimit = B_DEFAULT_MITER_LIMIT);
join_mode **LineJoinMode**() const;
cap_mode **LineCapMode**() const;
float **LineMiterLimit**() const;

void **SetOrigin**(BPoint pt);
void **SetOrigin**(float x, float y);
BPoint **Origin**() const;

void **PushState**();
void **PopState**();

void **MovePenTo**(BPoint pt);
void **MovePenTo**(float x, float y);

void **MovePenBy**(float x, float y);
BPoint **PenLocation**() const;

void **StrokeLine**(BPoint toPt, pattern p = B_SOLID_HIGH);
void **StrokeLine**(BPoint pt0, BPoint pt1, pattern p = B_SOLID_HIGH);

void **BeginLineArray**(int32 count);
void **AddLine**(BPoint pt0, BPoint pt1, rgb_color col);
void **EndLineArray**();

void **StrokePolygon**(const BPolygon *aPolygon, bool closed = true, pattern p = B_SOLID_HIGH);
void **StrokePolygon**(const BPoint *ptArray, int32 numPts [,BRect bounds], bool closed = true, pattern p = B_SOLID_HIGH);

void **FillPolygon**(const BPolygon *aPolygon, pattern p = B_SOLID_HIGH);
void **FillPolygon**(const BPoint *ptArray, int32 numPts [,BRect bounds], pattern p = B_SOLID_HIGH);

void **StrokeTriangle**(BPoint pt1, BPoint pt2, BPoint pt3 [,BRect bounds], pattern p = B_SOLID_HIGH);
void **FillTriangle**(BPoint pt1, BPoint pt2, BPoint pt3 [,BRect bounds], pattern p = B_SOLID_HIGH);

void **StrokeRect**(BRect r, pattern p = B_SOLID_HIGH);
void **FillRect**(BRect r, pattern p = B_SOLID_HIGH);
void **FillRegion**(BRegion *a_region, pattern p = B_SOLID_HIGH);
void **InvertRect**(BRect r);

void **StrokeRoundRect**(BRect r, float xRadius, float yRadius, pattern p = B_SOLID_HIGH);
void **FillRoundRect**(BRect r, float xRadius, float yRadius, pattern p = B_SOLID_HIGH);

void **StrokeEllipse**(BPoint center, float xRadius, float yRadius, pattern p = B_SOLID_HIGH);
void **StrokeEllipse**(BRect r, pattern p = B_SOLID_HIGH);

void **FillEllipse**(BPoint center, float xRadius, float yRadius, pattern p = B_SOLID_HIGH);
void **FillEllipse**(BRect r, pattern p = B_SOLID_HIGH);

void **StrokeArc**(BPoint center, float xRadius, float yRadius, float start_angle, float arc_angle, pattern p = B_SOLID_HIGH);
void **StrokeArc**(BRect r, float start_angle, float arc_angle, pattern p = B_SOLID_HIGH);

void **FillArc**(BPoint center, float xRadius, float yRadius, float start_angle, float arc_angle, pattern p = B_SOLID_HIGH);
void **FillArc**(BRect r, float start_angle, float arc_angle, pattern p = B_SOLID_HIGH);

void **StrokeBezier**(BPoint *controlPoints, pattern p = B_SOLID_HIGH);
void **FillBezier**(BPoint *controlPoints, pattern p = B_SOLID_HIGH);

void **StrokeShape**(BShape *shape, pattern p = B_SOLID_HIGH);
void **FillShape**(BShape *shape, pattern p = B_SOLID_HIGH);

void **CopyBits**(BRect src, BRect dst);

void **DrawBitmapAsync**(const BBitmap *aBitmap);
void **DrawBitmapAsync**(const BBitmap *aBitmap, BRect srcRect, BRect dstRect);
void **DrawBitmapAsync**(const BBitmap *aBitmap, BPoint where);
void **DrawBitmapAsync**(const BBitmap *aBitmap, BRect dstRect);

void **DrawBitmap**(const BBitmap *aBitmap);
void **DrawBitmap**(const BBitmap *aBitmap, BRect srcRect, BRect dstRect);
void **DrawBitmap**(const BBitmap *aBitmap, BPoint where);
void **DrawBitmap**(const BBitmap *aBitmap, BRect dstRect);

void **DrawChar**(char aChar [,BPoint location]);

void **DrawString**(const char *aString [,BPoint location], escapement_delta *delta = NULL);
void **DrawString**(const char *aString, int32 length [,BPoint location], escapement_delta *delta = 0L);

virtual void **SetFont**(const BFont *font, uint32 mask = B_FONT_ALL);
void **GetFont**(BFont *font) const;

float **StringWidth**(const char *string [,int32 length]) const;
void **GetStringWidths**(char *stringArray[], int32 lengthArray[], int32 numStrings, float widthArray[]) const;
void **SetFontSize**(float size);
void **ForceFontAliasing**(bool enable);
void **GetFontHeight**(font_height *height) const;

void **SetDiskMode**(char *filename, long offset);
void **BeginPicture**(BPicture *a_picture);
void **AppendToPicture**(BPicture *a_picture);
BPicture ***EndPicture**();

void **DrawPicture**(const BPicture *a_picture [,BPoint where]);
void **DrawPicture**(const char *filename, long offset, BPoint where);

void **DrawPictureAsync**(const BPicture *a_picture [,BPoint where]);
void **DrawPictureAsync**(char *filename, long offset, BPoint where);

BHandler inherited virtual functions...
virtual void **MessageReceived**(BMessage *msg);
virtual BHandler ***ResolveSpecifier**(BMessage *msg, int32 index, BMessage *specifier, int32 form, const char *property);
virtual status_t **GetSupportedSuites**(BMessage *data);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

List Views

class **BListItem** <be/interface/ListItem.h>

: public BArchivable

BListItem(uint32 outlineLevel = 0, bool expanded = true);
virtual ~**BListItem**();

float **Height**() const;
float **Width**() const;
bool **IsSelected**() const;
void **Select**()
void **Deselect**();

virtual void **SetEnabled**(bool on);
bool **IsEnabled**() const;

void **SetHeight**(float height);
void **SetWidth**(float width);
virtual void **DrawItem**(BView *owner, BRect bounds, bool complete = false) = 0;
virtual void **Update**(BView *owner, const BFont *font);

bool **IsExpanded**() const;
void **SetExpanded**(bool expanded);
uint32 **OutlineLevel**() const;

class **BStringItem** <be/interface/ListItem.h>

: public BListItem

BStringItem(const char *text, uint32 outlineLevel = 0, bool expanded = true);
virtual ~**BStringItem**();

These two functions override functions inherited from BListItem...
virtual void **DrawItem**(BView *owner, BRect frame, bool complete = false);
virtual void **Update**(BView *owner, const BFont *font);

virtual void **SetText**(const char *text);
const char ***Text**() const;

enum 'list_view_type' <be/interface/ListView.h>

B_SINGLE_SELECTION_LIST
B_MULTIPLE_SELECTION_LIST

class **BListView** <be/interface/ListView.h>

: public BView, public BInvoker

BListView(BRect frame, const char *name, list_view_type type = B_SINGLE_SELECTION_LIST, uint32 resizeMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_FRAME_EVENTS | B_NAVIGABLE);
virtual ~**BListView**();

virtual void **TargetedByScrollView**(BScrollView *scroller);
virtual bool **AddItem**(BListItem *item);
virtual bool **AddItem**(BListItem *item, int32 atIndex);
virtual bool **AddList**(BList *newItems);
virtual bool **AddList**(BList *newItems, int32 atIndex);
virtual bool **RemoveItem**(BListItem *item);
virtual BListItem ***RemoveItem**(int32 index);
virtual bool **RemoveItems**(int32 index, int32 count);

virtual void **SetSelectionMessage**(BMessage *message);
virtual void **SetInvocationMessage**(BMessage *message);

BMessage ***SelectionMessage**() const;
uint32 **SelectionCommand**() const;
BMessage ***InvocationMessage**() const;
uint32 **InvocationCommand**() const;

virtual void **SetListType**(list_view_type type);
list_view_type **ListType**() const;

BListItem ***ItemAt**(int32 index) const;

int32 **IndexOf**(BPoint point) const;
int32 **IndexOf**(BListItem *item) const;

BListItem ***FirstItem**() const;
BListItem ***LastItem**() const;
bool **HasItem**(BListItem *item) const;
int32 **CountItems**() const;
virtual void **MakeEmpty**();
bool **IsEmpty**() const;

void **DoForEach**(bool (*func)(BListItem *));
void **DoForEach**(bool (*func)(BListItem *, void *), void *);

const BListItem ****Items**() const;
void **InvalidateItem**(int32 index);
void **ScrollToSelection**();

void **Select**(int32 index, bool extend = false);
void **Select**(int32 from, int32 to, bool extend = false);

bool **IsItemSelected**(int32 index) const;
int32 **CurrentSelection**(int32 index = 0) const;

void **DeselectAll**();
void **DeselectExcept**(int32 except_from, int32 except_to);
void **Deselect**(int32 index);

virtual void **SelectionChanged**();

void **SortItems**(int (*cmp)(const void *, const void *));

These functions bottleneck through DoMiscellaneous()...
bool **SwapItems**(int32 a, int32 b);
bool **MoveItem**(int32 from, int32 to);
bool **ReplaceItem**(int32 index, BListItem * item);

BRect **ItemFrame**(int32 index);

virtual bool **InitiateDrag**(BPoint pt, int32 itemIndex, bool initialySelected);

Protected:

enum **MiscCode** { B_NO_OP, B_REPLACE_OP, B_MOVE_OP, B_SWAP_OP };
union **MiscData** {
    struct Replace { int32 index; BListItem * item; } **replace**;
    struct Move { int32 from; int32 to; } **move**;
    struct Swap { int32 a; int32 b; } **swap**;
};
virtual bool **DoMiscellaneous**(MiscCode code, MiscData * data);

Outline List View

class **BOutlineListView** <be/interface/OutlineListView.h>

: public BListView

BOutlineListView(BRect frame, const char * name, list_view_type type = B_SINGLE_SELECTION_LIST, uint32 resizeMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_FRAME_EVENTS | B_NAVIGABLE);
virtual ~**BOutlineListView**();

virtual bool **AddUnder**(BListItem *item, BListItem *underItem);

The following calls operator on the full outlinelist...

BListItem ***FullListItemAt**(int32 fullListIndex) const;

int32 **FullListIndexOf**(BPoint point) const;
int32 **FullListIndexOf**(BListItem *item) const;

BListItem ***FullListFirstItem**() const;
BListItem ***FullListLastItem**() const;
bool **FullListHasItem**(BListItem *item) const;
int32 **FullListCountItems**() const;
int32 **FullListCurrentSelection**(int32 index = 0) const;
virtual void MakeEmpty();
bool **FullListIsEmpty**() const;

void **FullListDoForEach**(bool (*func)(BListItem *));
void **FullListDoForEach**(bool (*func)(BListItem *, void *), void*);

BListItem ***Superitem**(const BListItem *item);

void **Expand**(BListItem *item);
void **Collapse**(BListItem *item);

bool **IsExpanded**(int32 fullListIndex);

void **FullListSortItems**(int (*compareFunc)(const BListItem *, const BListItem *));
void **SortItemsUnder**(BListItem *underItem, bool oneLevelOnly, int (*compareFunc)(const BListItem *, const BListItem*));
int32 **CountItemsUnder**(BListItem *under, bool oneLevelOnly) const;
BListItem ***EachItemUnder**(BListItem *underItem, bool oneLevelOnly, BListItem *(*eachFunc)(BListItem *, void *), void *);
BListItem ***ItemUnderAt**(BListItem *underItem, bool oneLevelOnly, int32 index) const;

Protected:

virtual void **ExpandOrCollapse**(BListItem *underItem, bool expand);

Menus

enum 'menu_layout' <be/interface/Menu.h>	
B_ITEMS_IN_ROW	B_ITEMS_IN_MATRIX
B_ITEMS_IN_COLUMN	

struct **menu_info** <be/interface/Menu.h>

float **font_size**;
font_family **f_family**;
font_style **f_style**;
rgb_color **background_color**;
int32 **separator**;
bool **click_to_open**, **triggers_always_shown**;

Menu Global Functions <be/interface/Menu.h>

status_t **set_menu_info**(menu_info *info);
status_t **get_menu_info**(menu_info *info);

class **BMenu** <be/interface/Menu.h>

: public BView

BMenu(const char *title, menu_layout layout = B_ITEMS_IN_COLUMN);
BMenu(const char *title, float width, float height);
virtual ~**BMenu**();

bool **AddItem**(BMenuItem *item);
bool **AddItem**(BMenuItem *item, int32 index);
bool **AddItem**(BMenuItem *item, BRect frame);
bool **AddItem**(BMenu *menu);
bool **AddItem**(BMenu *menu, int32 index);
bool **AddItem**(BMenu *menu, BRect frame);

bool **AddList**(BList *list, int32 index);
bool **AddSeparatorItem**();

bool **RemoveItem**(BMenuItem *item);
BMenuItem ***RemoveItem**(int32 index);
bool **RemoveItem**(BMenu *menu);

bool **RemoveItems**(int32 index, int32 count, bool del = false);

BMenuItem ***ItemAt**(int32 index) const;
BMenu ***SubmenuAt**(int32 index) const;
int32 **CountItems**() const;

int32 **IndexOf**(BMenuItem *item) const;
int32 **IndexOf**(BMenu *menu) const;

BMenuItem ***FindItem**(uint32 command) const;
BMenuItem ***FindItem**(const char *name) const;

virtual status_t **SetTargetForItems**(BHandler *target);
virtual status_t **SetTargetForItems**(BMessenger messenger);
virtual void **SetEnabled**(bool state);
virtual void **SetRadioMode**(bool state);
virtual void **SetTriggersEnabled**(bool state);
virtual void **SetMaxContentWidth**(float max);
void **SetLabelFromMarked**(bool on);
bool **IsLabelFromMarked**();
bool **IsEnabled**() const;
bool **IsRadioMode**() const;
bool **AreTriggersEnabled**() const;
bool **IsRedrawAfterSticky**() const;
float **MaxContentWidth**() const;

BMenuItem ***FindMarked**();
BMenu ***Supermenu**() const;
BMenuItem ***Superitem**() const;

void **InvalidateLayout**();

enum 'add_state'	
B_INITIAL_ADD	B_PROCESSING
B_ABORT	

virtual bool **AddDynamicItem**(add_state s);

virtual void **DrawBackground**(BRect update);

Protected:

BMenu(BRect frame, const char *viewName, uint32 resizeMask, uint32 flags, menu_layout layout, bool resizeToFit);

virtual BPoint **ScreenLocation**();
void **SetItemMargins**(float left, float top, float right, float bottom);
void **GetItemMargins**(float *left, float *top, float *right, float *bottom) const;
menu_layout **Layout**() const;

Not to be confused with Show() with no arguments, as defined by the BView class that this class is derived from...
void **Show**(bool selectFirstItem);
BMenuItem ***Track**(bool start_opened = false, BRect *special_rect = NULL);

enum 'menu_bar_border' <be/interface/MenuBar.h>	
B_BORDER_FRAME	B_BORDER_EACH_ITEM
B_BORDER_CONTENTS	

class **BMenuBar** <be/interface/Menu.h>

: public BMenu

BMenuBar(BRect frame, const char *title, uint32 resizeMask = B_FOLLOW_LEFT_RIGHT | B_FOLLOW_TOP, menu_layout layout = B_ITEMS_IN_ROW, bool resizeToFit = true);
virtual ~**BMenuBar**();

virtual void **SetBorder**(menu_bar_border border);
menu_bar_border **Border**() const;

class **BMenuField** <be/interface/MenuField.h>

: public BView

BMenuField(BRect frame, const char *name, const char *label, BMenu *menu [,bool fixed_size], uint32 resize = B_FOLLOW_LEFT|B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_NAVIGABLE);
virtual ~**BMenuField**();

BMenu ***Menu**() const;
BMenuBar ***MenuBar**() const;
BMenuItem ***MenuItem**() const;
virtual void **SetLabel**(const char *label);
const char ***Label**() const;
virtual void **SetEnabled**(bool on);
bool **IsEnabled**() const;
virtual void **SetAlignment**(alignment label);
alignment **Alignment**() const;
virtual void **SetDivider**(float dividing_line);
float **Divider**() const;

void **ShowPopUpMarker**();
void **HidePopUpMarker**();

class **BMenuItem** <be/interface/MenuItem.h>

: public BArchivable, public BInvoker

BMenuItem(const char *label, BMessage *message, char shortcut = 0, uint32 modifiers = 0);
BMenuItem(BMenu *menu, BMessage *message = NULL);
virtual ~**BMenuItem**();

virtual void **SetLabel**(const char *name);
virtual void **SetEnabled**(bool state);
virtual void **SetMarked**(bool state);
virtual void **SetTrigger**(char ch);
virtual void **SetShortcut**(char ch, uint32 modifiers);

const char ***Label**() const;
bool **IsEnabled**() const;
bool **IsMarked**() const;
char **Trigger**() const;
char **Shortcut**(uint32 *modifiers = NULL) const;

BMenu ***Submenu**() const;
BMenu ***Menu**() const;

Protected:

virtual void **GetContentSize**(float *width, float *height);
virtual void **TruncateLabel**(float max, char *new_label);
virtual void **DrawContent**();
virtual void **Draw**(); *// originally inherited from BView*
virtual void **Highlight**(bool on);
bool **IsSelected**() const;
BPoint **ContentLocation**() const;

Note: Inherited from BInvoker, protected in this class...
virtual status_t **Invoke**(BMessage *msg = NULL);

class **BSeparatorItem** <be/Interface/MenuItem.h>

: public BMenuItem

BSeparatorItem();
virtual ~**BSeparatorItem**();

class **BPopUpMenu** <be/interface/PopUpMenu.h>

: public BMenu

BPopUpMenu(const char *title, bool radioMode = true, bool autoRename = true, menu_layout layout = B_ITEMS_IN_COLUMN);
virtual ~**BPopUpMenu**();

BMenuItem ***Go**(BPoint where, bool delivers_message = false, bool open_anyway = false, bool async = false);
BMenuItem ***Go**(BPoint where, bool delivers_message, bool open_anyway, BRect click_to_open, bool async = false);

Protected:

virtual BPoint **ScreenLocation**();

BPopUpMenu &operator=(const BPopUpMenu &);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

Tab View

enum 'tab_position' <be/interface/TabView.h>	
B_TAB_FIRST (999)	B_TAB_ANY
B_TAB_FRONT	

class **BTab** <be/interface/TabView.h>

: public BArchivable

BTab(BView* contents=NULL);
virtual ~**BTab**();

const char* **Label**() const;
virtual void **SetLabel**(const char* label);

bool **IsSelected**() const;
virtual void **Select**(BView* owner);
virtual void **Deselect**();

virtual void **SetEnabled**(bool on);
bool **IsEnabled**() const;

void **MakeFocus**(bool infocus = true);
book **IsFocus**() const;

virtual void **SetView**(BView* contents);
BView* **View**() const;

virtual void **DrawFocusMark**(BView* owner, BRect tabFrame);
virtual void **DrawLabel**(BView* owner, BRect tabFrame);
virtual void **DrawTab**(BView* owner, BRect tabFrame, tab_position, bool full=true);

class **BTabView** <be/interface/TabView.h>

: public BView

BTabView(BRect frame, const char *name, button_width width=B_WIDTH_AS_USUAL, uint32 resizingMode = B_FOLLOW_ALL, uint32 flags = B_FULL_UPDATE_ON_RESIZE | B_WILL_DRAW | B_NAVIGABLE_JUMP | B_FRAME_EVENTS | B_NAVIGABLE);
virtual ~**BTabView**();

virtual void **Select**(int32 tabIndex);
int32 **Selection**() const;

virtual void **MakeFocus**(bool focusState = true);
virtual void **SetFocusTab**(int32 tabIndex, bool focusState);
int32 **FocusTab**() const;

virtual BRect **DrawTabs**();
virtual void **DrawBox**(BRect selectedTabFrame);
virtual BRect **TabFrame**(int32 tabIndex) const;
virtual void **AddTab**(BView* tabContents, BTab* tab=NULL);
virtual BTab* **RemoveTab**(int32 tabIndex);
virtual BTab* **TabAt**(int32 tabIndex) const;
BView* **ContainerView**() const;
int32 **CountTabs**() const;
BView* **ViewForTab**(int32 tabIndex) const;

virtual void **SetTabWidth**(button_width s);
button_width **TabWidth**() const;

virtual void **SetTabHeight**(float height);
float **TabHeight**() const;

Text View

struct **text_run** <be/interface/TextView.h>

int32 offset;
BFont font;
rgb_color color;

struct **text_run_array** <be/interface/TextView.h>

int32 **count**;
text_run **runs**[1];

enum 'undo_state' <be/interface/TextView.h>	
B_UNDO_UNAVAILABLE	B_UNDO_PASTE
B_UNDO_TYPING	B_UNDO_CLEAR
B_UNDO_CUT	B_UNDO_DROP

class **BTextView** <be/interface/TextView.h>

: public BView

BTextView(BRect frame, const char *name, BRect textRect [,const BFont *initialFont, const rgb_color *initialColor], uint32 resizeMask, uint32 flags);
virtual ~**BTextView**();

void **SetText**(const char *inText [,int32 inLength], const text_run_array *inRuns = NULL);
void **SetText**(BFile *inFile, int32 startOffset, int32 inLength, const text_run_array *inRuns = NULL);

void **Insert**(const char *inText [,int32 inLength], const text_run_array *inRuns = NULL);
void **Insert**(int32 startOffset, const char *inText, int32 inLength, const text_run_array *inRuns = NULL);

void **Delete**([int32 startOffset, int32 endOffset]);
const char* **Text**() const;
int32 **TextLength**() const;
void **GetText**(int32 offset, int32 length, char *buffer) const;
uchar **ByteAt**(int32 offset) const;

int32 **CountLines**() const;
int32 **CurrentLine**() const;
void **GoToLine**(int32 lineNum);

virtual void **Cut**(BClipboard *clipboard);
virtual void **Copy**(BClipboard *clipboard);
virtual void **Paste**(BClipboard *clipboard);
void **Clear**();
virtual bool **AcceptsPaste**(BClipboard *clipboard);
virtual bool **AcceptsDrop**(const BMessage *inMessage);
virtual void **Select**(int32 startOffset, int32 endOffset);
void **SelectAll**();
void **GetSelection**(int32 *outStart, int32 *outEnd) const;

void **SetFontAndColor**(const BFont *inFont, uint32 inMode = B_FONT_ALL, const rgb_color *inColor = NULL);
void **SetFontAndColor**(int32 startOffset, int32 endOffset, const BFont *inFont, uint32 inMode = B_FONT_ALL, const rgb_color *inColor = NULL);

void **GetFontAndColor**(int32 inOffset, BFont *outFont, rgb_color *outColor = NULL) const;
void **GetFontAndColor**(BFont *outFont, uint32 *outMode, rgb_color *outColor = NULL, bool *outEqColor = NULL) const;
void **SetRunArray**(int32 startOffset, int32 endOffset, const text_run_array *inRuns);
text_run_array* **RunArray**(int32 startOffset, int32 endOffset, int32 *outSize = NULL) const;

int32 **LineAt**(int32 offset) const;
int32 **LineAt**(BPoint point) const;

BPoint **PointAt**(int32 inOffset, float *outHeight = NULL) const;

int32 **OffsetAt**(BPoint point) const;
int32 **OffsetAt**(int32 line) const;

virtual void **FindWord**(int32 inOffset, int32 *outFromOffset, int32 *outToOffset);
virtual bool **CanEndLine**(int32 offset);
float **LineWidth**(int32 lineNum = 0) const;
float **LineHeight**(int32 lineNum = 0) const;
float **TextHeight**(int32 startLine, int32 endLine) const;
void **GetTextRegion**(int32 startOffset, int32 endOffset, BRegion *outRegion) const;
virtual void **ScrollToOffset**(int32 inOffset);
void **ScrollToSelection**();
void **Highlight**(int32 startOffset, int32 endOffset);
void **SetTextRect**(BRect rect);
BRect **TextRect**() const;
void **SetStylable**(bool stylable);
bool **IsStylable**() const;
void **SetTabWidth**(float width);
float **TabWidth**() const;
void **MakeSelectable**(bool selectable = true);
bool **IsSelectable**() const;
void **MakeEditable**(bool editable = true);
bool **IsEditable**() const;
void **SetWordWrap**(bool wrap);
bool **DoesWordWrap**() const;
void **SetMaxBytes**(int32 max);
int32 **MaxBytes**() const;
void **DisallowChar**(uint32 aChar);
void **AllowChar**(uint32 aChar);
void **SetAlignment**(alignment flag);
alignment **Alignment**() const;
void **SetAutoindent**(bool state);
bool **DoesAutoindent**() const;
void **SetColorSpace**(color_space colors);
color_space **ColorSpace**() const;
void **MakeResizable**(bool resize, BView *resizeView = NULL);
bool **IsResizable**() const;
void **SetDoesUndo**(bool undo);
bool **DoesUndo**() const;

static void* **FlattenRunArray**(const text_run_array *inArray, int32 *outSize = NULL);
static text_run_array* **UnflattenRunArray**(const void *data, int32 *outSize = NULL);

virtual void **Undo**(BClipboard *clipboard);
undo_state **UndoState**(bool *isRedo) const;

Protected:

virtual void **InsertText**(const char *inText, int32 inLength, int32 inOffset, const text_run_array *inRuns);
virtual void **DeleteText**(int32 fromOffset, int32 toOffset);
virtual void **GetDragParameters**(BMessage *drag, BBitmap **bitmap, BPoint *point, BHandler **handler);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

OTHER BVIEW-INHERITED CLASSES

class **BBox** <be/interface/Box.h>

: public BView

BBox(BRect bounds, const char *name = NULL, uint32 resizeFlags = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_FRAME_EVENTS | B_NAVIGABLE_JUMP, border_style border = B_FANCY_BORDER);
virtual ~**BBox**();

virtual void **SetBorder**(border_style style);
border_style **Border**() const;

void **SetLabel**(const char *label);
status_t **SetLabel**(BView *view_label);

const char ***Label**() const;
BView ***LabelView**() const;

Scrollbar Size Constants <be/interface/ScrollBar.h>
B_V_SCROLL_BAR_WIDTH (14.0)
B_H_SCROLL_BAR_HEIGHT (14.0)

class **BScrollBar** <be/interface/ScrollBar.h>

: public BView

BScrollBar(BRect frame, const char *name, BView *target, float min, float max, orientation direction);
virtual ~**BScrollBar**();

void **SetValue**(float value);
float **Value**() const;
void **SetProportion**(float);
float **Proportion**() const;
virtual void **ValueChanged**(float newValue);

void **SetRange**(float min, float max);
void **GetRange**(float *min, float *max) const;
void **SetSteps**(float smallStep, float largeStep);
void **GetSteps**(float *smallStep, float *largeStep) const;
void **SetTarget**(BView *target);
void **SetTarget**(const char *targetName);
BView ***Target**() const;
orientation **Orientation**() const;

class **BScrollView** <be/interface/ScrollView.h>

: public BView

BScrollView(const char *name, BView *target, uint32 resizeMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = 0, bool horizontal = false, bool vertical = false, border_style border = B_FANCY_BORDER);
virtual ~**BScrollView**();

BScrollBar ***ScrollBar**(orientation flag) const;
virtual void **SetBorder**(border_style border);
border_style **Border**() const;
virtual status_t **SetBorderHighlighted**(bool state);
bool **IsBorderHighlighted**() const;

void **SetTarget**(BView *new_target);
BView ***Target**() const;

class **BStatusBar** <be/interface/StatusBar.h>

: public BView

BStatusBar(BRect frame, const char *name, const char *label = NULL, const char *trailing_label = NULL);
virtual ~**BStatusBar**();

virtual void **Update**(float delta, const char *main_text = NULL, const char *trailing_text = NULL);
virtual void **Reset**(const char *label = NULL, const char *trailing_label = NULL);
float **CurrentValue**() const;

virtual void **SetMaxValue**(float max);
float **MaxValue**() const;

virtual void **SetBarColor**(rgb_color color);
rgb_color **BarColor**() const;

virtual void **SetBarHeight**(float height);
float **BarHeight**() const;

virtual void **SetText**(const char *str)
const char ***Text**() const;

virtual void **SetTrailingText**(const char *str);
const char ***TrailingText**() const;

const char ***Label**() const;
const char ***TrailingLabel**() const;

class **BStringView** <be/interface/StringView.h>

: public BView

BStringView(BRect bounds, const char *name, const char *text, uint32 resizeFlags = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW);
virtual ~**BStringView**();

void **SetText**(const char *text);
const char ***Text**() const;

void **SetAlignment**(alignment flag);
alignment **Alignment**() const;

REPLICANTS

class **BDragger** <be/interface/Dragger.h>

: public BView

BDragger(BRect bounds, BView *target, uint32 rmask = B_FOLLOW_NONE, uint32 flags = B_WILL_DRAW);
virtual ~**BDragger**();

static status_t **ShowAllDraggers**(); *(Note: system wide!)*
static status_t **HideAllDraggers**(); *(Note: system wide!)*
static bool **AreDraggersDrawn**();

status_t **SetPopUp**(BPopUpMenu *context_menu);
BPopUpMenu ***PopUp**() const;

Protected:

bool **IsVisibilityChanging**() const;

class **BShelf** <be/interface/Shelf.h>

: public BHandler

BShelf(BView *view, bool allow_drags = true, const char *shelf_type = NULL);
BShelf(const entry_ref *ref, BView *view, bool allow_drags = true, const char *shelf_type = NULL);
BShelf(BDataIO *stream, BView *view, bool allow_drags = true, const char *shelf_type = NULL);
virtual ~**BShelf**();

status_t **Save**();
virtual void **SetDirty**(bool state);
bool **IsDirty**() const;

virtual status_t **Perform**(perform_code d, void *arg);

bool **AllowsDragging**() const;
void **SetAllowsDragging**(bool state);

bool **AllowsZombies**() const;
void **SetAllowsZombies**(bool state);
bool **DisplaysZombies**() const;
void **SetDisplaysZombies**(bool state);

bool **IsTypeEnforced**() const;
void **SetTypeEnforced**(bool state);

status_t **SetSaveLocation**(BDataIO *data_io);
status_t **SetSaveLocation**(const entry_ref *ref);
BDataIO ***SaveLocation**(entry_ref *ref) const;

status_t **AddReplicant**(BMessage *data, BPoint location);

status_t **DeleteReplicant**(BView *replicant);
status_t **DeleteReplicant**(BMessage *data);
status_t **DeleteReplicant**(int32 index);

int32 **CountReplicants**() const;
BMessage ***ReplicantAt**(int32 index, BView **view = NULL, uint32 *uid = NULL, status_t *perr = NULL) const;

int32 **IndexOf**(const BView *replicant_view) const;
int32 **IndexOf**(const BMessage *archive) const;
int32 **IndexOf**(uint32 id) const;

Protected:

virtual bool **CanAcceptReplicantMessage**(BMessage*) const;
virtual bool **CanAcceptReplicantView**(BRect, BView*, BMessage*) const;
virtual BPoint **AdjustReplicantBy**(BRect, BMessage*) const;

virtual void **ReplicantDeleted**(int32 index, const BMessage *archive, const BView *replicant);

Interface Kit

<be/InterfaceKit.h> <libbe.so>

BeOS R4 Programmer's Cheatsheets by David Orr

BControl and BControl-Inherited Classes

SetValue() Values for Boolean Controls <be/interface/Cont	
B_CONTROL_OFF (0)	B_CONTROL_ON (1)

class **BControl** <be/interface/Control.h>

: public BView, public BInvoker

BControl(BRect frame, const char *name, const char *label, BMessage *message, uint32 resizeMask, uint32 flags);
virtual ~**BControl**();

virtual void **SetLabel**(const char *text);
const char ***Label**() const;

virtual void **SetValue**(int32 value);
int32 **Value**() const;

virtual void **SetEnabled**(bool on);
bool **IsEnabled**() const;

Protected:

bool **IsFocusChanging**() const;
bool **IsTracking**() const;
void **SetTracking**(bool state);

class **BButton** <be/interface/Button.h>

: public BControl

BButton(BRect frame, const char *name, const char *label, BMessage *message, uint32 resizeMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_NAVIGABLE);
virtual ~**BButton**();

virtual void **MakeDefault**(bool state);
bool **IsDefault**() const;

class **BCheckBox** <be/interface/CheckBox.h>

: public BControl

BCheckBox(BRect frame, const char *name, const char *label, BMessage *message, uint32 resizeMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_NAVIGABLE);
virtual ~**BCheckBox**();

enum 'control_color_layout' <be/interface/ColorControl.h>	
B_CELLS_4x64	B_CELLS_32x8
B_CELLS_8x32	B_CELLS_64x4
B_CELLS_16x16	

class **BColorControl** <be/interface/ColorControl.h>

: public BControl

BColorControl(BPoint start, color_control_layout layout, float cell_size, const char *name, BMessage *message = NULL, bool use_offscreen = false);
virtual ~**BColorControl**();

virtual void **SetValue**(int32 color_value);
void **SetValue**(rgb_color color);
rgb_color **ValueAsColor**();
virtual void **SetCellSize**(float size);
float **CellSize**() const;
virtual void **SetLayout**(color_control_layout layout);
color_control_layout **Layout**() const;

Behavior Codes <be/interface/PictureButton.h>	
B_ONE_STATE_BUTTON	B_TWO_STATE_BUTTON

class **BPictureButton** <be/interface/PictureButton.h>

: public BControl

BPictureButton(BRect frame, const char* name, BPicture *off, BPicture *on, BMessage *message, uint32 behavior = B_ONE_STATE_BUTTON, uint32 resizeMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flgs = B_WILL_DRAW | B_NAVIGABLE);
virtual ~**BPictureButton**();

virtual void **SetEnabledOn**(BPicture *on);
virtual void **SetEnabledOff**(BPicture *off);
virtual void **SetDisabledOn**(BPicture *on);
virtual void **SetDisabledOff**(BPicture *off);

BPicture ***EnabledOn**() const;
BPicture ***EnabledOff**() const;
BPicture ***DisabledOn**() const;
BPicture ***DisabledOff**() const;

virtual void **SetBehavior**(uint32 behavior);
uint32 **Behavior**() const;

class **BRadioButton** <be/interface/RadioButton.h>

: public BControl

BRadioButton(BRect frame, const char *name, const char *label, BMessage *message, uint32 resizMask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_NAVIGABLE);
virtual ~**BRadioButton**();

enum 'hash_mark_location' <be/interface/Slider.h>	
B_HASH_MARKS_NONE	B_HASH_MARKS_BOTTOM
B_HASH_MARKS_TOP	B_HASH_MARKS_BOTH

enum 'thumb_style' <be/interface/Slider.h>	
B_BLOCK_THUMB	B_TRIANGLE_THUMB

class **BSlider** <be/interface/Slider.h>

: public BControl

BSlider(BRect frame, const char *name, const char *label, BMessage *message, int32 minValue, int32 maxValue, thumb_style thumbType = B_BLOCK_THUMB, uint32 resizingMode = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_NAVIGABLE | B_WILL_DRAW | B_FRAME_EVENTS);
virtual ~**BSlider**();

virtual void **SetLimitLabels**(const char *minLabel, const char *maxLabel);
const char* **MinLimitLabel**() const;
const char* **MaxLimitLabel**() const;

virtual int32 **ValueForPoint**(BPoint) const;
virtual void **SetPosition**(float);
float **Position**() const;

virtual void **DrawSlider**();
virtual void **DrawBar**();
virtual void **DrawHashMarks**();
virtual void **DrawThumb**();
virtual void **DrawFocusMark**();
virtual void **DrawText**();
virtual char* **UpdateText**() const;

virtual BRect **BarFrame**() const;
virtual BRect **HashMarksFrame**() const;
virtual BRect **ThumbFrame**() const;

virtual void **SetModificationMessage**(BMessage *message);
BMessage ***ModificationMessage**() const;

virtual void **SetSnoozeAmount**(int32);
int32 **SnoozeAmount**() const;

virtual void **SetKeyIncrementValue**(int32 value);
int32 **KeyIncrementValue**() const;

virtual void **SetHashMarkCount**(int32 count);
int32 **HashMarkCount**() const;

virtual void **SetHashMarks**(hash_mark_location where);
hash_mark_location **HashMarks**() const;

virtual void **SetStyle**(thumb_style s);
thumb_style **Style**() const;

virtual void **SetBarColor**(rgb_color);
rgb_color **BarColor**() const;
virtual void **UseFillColor**(bool, const rgb_color* c=NULL);
bool **FillColor**(rgb_color*) const;
BView ***OffscreenView**() const;

class **BTextControl** <be/interface/TextControl.h>

: public BControl

BTextControl(BRect frame, const char *name, const char *label, const char *initial_text, BMessage *message, uint32 rmask = B_FOLLOW_LEFT | B_FOLLOW_TOP, uint32 flags = B_WILL_DRAW | B_NAVIGABLE);
virtual ~**BTextControl**();

virtual void **SetText**(const char *text);
const char ***Text**() const;

virtual void **SetValue**(int32 value);
virtual status_t **Invoke**(BMessage *msg = NULL);

BTextView ***TextView**() const;

virtual void **SetModificationMessage**(BMessage *message);
BMessage ***ModificationMessage**() const;

virtual void **SetAlignment**(alignment label, alignment text);
void **GetAlignment**(alignment *label, alignment *text) const;
virtual void **SetDivider**(float dividing_line);
float **Divider**() const;

This page is intentionally left blank.

Storage Kit

<be/StorageKit.h> <libbe.so>

StorageKit Error Values <be/support/Errors.h>
B_FILE_ERROR
B_FILE_EXISTS
B_FILE_NOT_FOUND
B_NAME_TOO_LONG
B_NOT_A_DIRECTORY
B_DIRECTORY_NOT_EMPTY
B_DEVICE_FULL
B_READ_ONLY_DEVICE
B_IS_A_DIRECTORY
B_NO_MORE_FDS
B_CROSS_DEVICE_LINK
B_LINK_LIMIT
B_BUSTED_PIPEB_UNSUPPORTED
B_PARTITION_TOO_SMALL

Defined Limits <be/storage/StorageDefs.h>
B_FILE_NAME_LENGTH
B_PATH_NAME_LENGTH
B_ATTR_NAME_LENGTH
B_MIME_TYPE_LENGTH
B_MAX_SYMLINKS

File Open Mode Codes <be/storage/StorageDefs.h>
B_READ_ONLY (O_RDONLY)
B_WRITE_ONLY (O_WRONLY)
B_READ_WRITE (O_RDWR)
B_FAIL_IF_EXISTS (O_EXCL)
B_CREATE_FILE (O_CREAT)
B_ERASE_FILE (O_TRUNC)
B_OPEN_AT_END (O_APPEND)

class **BStatable** *(pure abstract)* <be/storage/Statable.h>

virtual status_t **GetStat**(struct stat *st) const = 0;

bool **IsFile**() const;
bool **IsDirectory**() const;
bool **IsSymLink**() const;

status_t **GetNodeRef**(node_ref *ref) const;
status_t **GetSize**(off_t *size) const;
status_t **GetVolume**(BVolume *vol) const;

status_t **GetOwner**(uid_t *owner) const;
status_t **SetOwner**(uid_t owner);

status_t **GetGroup**(gid_t *group) const;
status_t **SetGroup**(gid_t group);

status_t **GetPermissions**(mode_t *perms) const;
status_t **SetPermissions**(mode_t perms);

status_t **GetModificationTime**(time_t *mtime) const;
status_t **SetModificationTime**(time_t mtime);

status_t **GetCreationTime**(time_t *ctime) const;
status_t **SetCreationTime**(time_t ctime);

status_t **GetAccessTime**(time_t *atime) const;
status_t **SetAccessTime**(time_t atime);

NODES

enum 'node_flavor' <be/storage/StorageDefs.h>
B_FILE_NODE
B_SYMLINK_NODE
B_DIRECTORY_NODE
B_ANY_NODE

struct **node_ref** <be/storage/Node.h>

node_ref([const node_ref &ref]);

bool operator**==**(const node_ref &ref) const;
bool operator**!=**(const node_ref &ref) const;
node_ref & operator**=**(const node_ref &ref);

dev_t **device**;
ino_t **node**;

class **BNode** <be/storage/Node.h>

: public BStatable

BNode();
BNode(const entry_ref *ref);
BNode(const BEntry *entry);
BNode(const char *path);
BNode(const BDirectory *dir, const char *path);
BNode(const BNode &node);
virtual ~**BNode**();

status_t **InitCheck**() const;

(BStatable inherited virtual function...)
virtual status_t **GetStat**(struct stat *st) const;

BNode & operator**=**(const BNode &node);
bool operator**==**(const BNode &node) const;
bool operator**!=**(const BNode &node) const;

status_t **SetTo**(const entry_ref *ref);
status_t **SetTo**(const BEntry *entry);
status_t **SetTo**(const char *path);
status_t **SetTo**(const BDirectory *dir, const char *path);

void **Unset**();

status_t **Lock**();
status_t **Unlock**();

status_t **Sync**();

ssize_t **WriteAttr**(const char *attr, type_code type, off_t off, const void *buf, size_t l);
ssize_t **ReadAttr**(const char *attr, type_code type, off_t off, void *buf, size_t l) const;
status_t **RemoveAttr**(const char *attr);
status_t **RenameAttr**(const char *oldname, const char *newname);
status_t **GetAttrInfo**(const char *attr, struct attr_info *buf) const;
status_t **GetNextAttrName**(char *buf) ;
status_t **RewindAttrs**() ;

int **Dup**();

class **BNodeInfo** <be/storage/NodeInfo.h>

BNodeInfo([BNode *node]);
virtual ~**BNodeInfo**();

status_t **SetTo**(BNode *node);
status_t **InitCheck**() const;

virtual status_t **Get/SetType**(char *type);
virtual status_t **Get/SetIcon**(BBitmap *icon, icon_size k = B_LARGE_ICON);
status_t **Get/SetPreferredApp**(char *signature, app_verb verb = B_OPEN);
status_t **Get/SetAppHint**(entry_ref *ref);

status_t **GetTrackerIcon**(BBitmap *icon, icon_size k = B_LARGE_ICON) const;
static status_t **GetTrackerIcon**(entry_ref *ref, BBitmap *icon, icon_size k = B_LARGE_ICON);

Node Monitoring Bit Codes <be/storage/NodeMonitoring.h>
B_STOP_WATCHING
B_WATCH_NAME
B_WATCH_STAT
B_WATCH_ATTR
B_WATCH_DIRECTORY
B_WATCH_ALL (does not include B_WATCH_MOUNT)
B_WATCH_MOUNT (see also BVolumeRoster)

Node Watch Result 'opcode' <be/storage/NodeMonitoring.h>
B_ENTRY_CREATED
B_ENTRY_REMOVED
B_ENTRY_MOVED
B_STAT_CHANGED
B_ATTR_CHANGED
B_DEVICE_MOUNTED
B_DEVICE_UNMOUNTED

Node Monitoring Functions<be/storage/NodeMonitor.h>

status_t **watch_node**(const node_ref *node, uint32 flags, BMessenger target);
status_t **watch_node**(const node_ref *node, uint32 flags, const BHandler *handler, const BLooper *looper = NULL);

status_t **stop_watching**(BMessenger target);
status_t **stop_watching**(const BHandler *handler, const BLooper *looper=NULL);

Storage Kit

<be/StorageKit.h> <libbe.so>

Entry Global Functions <be/storage/Entry.h>

status_t **get_ref_for_path**(const char *path, entry_ref *ref);
bool operator**<**(const entry_ref & a, const entry_ref & b);

struct **entry_ref** <be/storage/Entry.h>

entry_ref();
entry_ref(dev_t dev, ino_t dir, const char *name);
entry_ref(const entry_ref &ref);
~**entry_ref**();

status_t **set_name**(const char *name);

bool operator**==**(const entry_ref &ref) const;
bool operator**!=**(const entry_ref &ref) const;
entry_ref & operator**=**(const entry_ref &ref);

dev_t **device**;
ino_t **directory**;
char ***name**;

class **BEntry** <be/storage/Entry.h>

: public BStatable

BEntry();
BEntry(const BDirectory *dir, const char *path, bool traverse = false); *(Note: BEntry(dir, NULL) gets the entry for dir.)*
BEntry(const entry_ref *ref, bool traverse = false);
BEntry(const char *path, bool traverse = false);
BEntry(const BEntry &entry);
virtual ~**BEntry**();

status_t **InitCheck**() const;

bool operator**==**(const BEntry &item) const;
bool operator**!=**(const BEntry &item) const;
BEntry & operator**=**(const BEntry &item);

status_t **SetTo**(const BDirectory *dir, const char *path, bool traverse = false);
status_t **SetTo**(const entry_ref *ref, bool traverse = false);
status_t **SetTo**(const char *path, bool traverse = false);
void **Unset**();

status_t **GetParent**(BEntry *entry) const;
status_t **GetParent**(BDirectory *dir) const;

bool **Exists**() const;
virtual status_t **GetStat**(struct stat *st) const;
status_t **GetRef**(entry_ref *ref) const;
status_t **GetPath**(BPath *path) const;
status_t **GetName**(char *buffer) const;
status_t **Rename**(const char *path, bool clobber = false);
status_t **MoveTo**(BDirectory *dir, const char *path = NULL, bool clobber = false);
status_t **Remove**();

class **BEntryList** *(pure virtual)* <be/storage/EntryList.h>

virtual status_t **GetNextEntry**(BEntry *entry, bool traverse=false) = 0;
virtual status_t **GetNextRef**(entry_ref *ref) = 0;
virtual int32 **GetNextDirents**(struct dirent *buf, size_t length, int32 count = INT_MAX) = 0;
virtual status_t **Rewind**() = 0;
virtual int32 **CountEntries**() = 0;

DIRECTORIES

Directory Global Functions <be/storage/Directory.h>

status_t **create_directory**(const char *path, mode_t mode);

class **BDirectory** <be/storage/Directory.h>

: public BNode, public BEntryList

BDirectory();
BDirectory(const BEntry *entry);
BDirectory(const entry_ref *ref);
BDirectory(const char *path);
BDirectory(const BDirectory *dir, const char *path);
BDirectory(const node_ref *ref);
BDirectory(const BDirectory &dir);
virtual ~**BDirectory**();

BDirectory & operator**=**(const BDirectory &dir);

status_t **GetEntry**(BEntry *entry) const; bool IsRootDirectory() const;
status_t **FindEntry**(const char *path, BEntry *entry, bool traverse = false) const;

bool **Contains**(const char *path, int32 node_flags = B_ANY_NODE) const;
bool **Contains**(const BEntry *entry, int32 node_flags = B_ANY_NODE) const;

status_t **GetStatFor**(const char *path, struct stat *st) const;

status_t **CreateDirectory**(const char *path, BDirectory *dir);
status_t **CreateFile**(const char *path, BFile *file, bool failIfExists = false);
status_t **CreateSymLink**(const char *path, const char *content, BSymLink *link);

Redefinitions of inherited BNode functions...
status_t **SetTo**(const entry_ref *ref);
status_t **SetTo**(const BEntry *entry);
status_t **SetTo**(const char *path);
status_t **SetTo**(const BDirectory *dir, const char *path);
status_t **SetTo**(const node_ref *ref);

BEntryList inhertied virtual functions...
virtual status_t **GetNextEntry**(BEntry *entry, bool traverse = false);
virtual status_t **GetNextRef**(entry_ref *ref);
virtual int32 **GetNextDirents**(struct dirent *buf, size_t length, int32 count = INT_MAX);
virtual status_t **Rewind**();
virtual int32 **CountEntries**();

FIND DIRECTORY

Find Directory Global Functions <be/storage/FindDirectory.h>

For C programs...
status_t **find_directory** (directory_which which, dev_t device, bool create_it, char *returned_path, int32 path_length);

For C++ programs only...
status_t **find_directory** (directory_which which, BPath *path, bool and_create_it = false, BVolume *vol = NULL);

enum 'directory which' <be/storage/FindDirectory.h>

Specify a volume for these (default is boot volume)...
B_DESKTOP_DIRECTORY
B_TRASH_DIRECTORY
BeOS Directories (mostly read-only)...
B_BEOS_DIRECTORY
B_BEOS_SYSTEM_DIRECTORY
B_BEOS_ADDONS_DIRECTORY
B_BEOS_BOOT_DIRECTORY
B_BEOS_FONTS_DIRECTORY
B_BEOS_LIB_DIRECTORY
B_BEOS_SERVERS_DIRECTORY
B_BEOS_APPS_DIRECTORY
B_BEOS_BIN_DIRECTORY
B_BEOS_ETC_DIRECTORY
B_BEOS_DOCUMENTATION_DIRECTORY
B_BEOS_PREFERENCES_DIRECTORY
B_BEOS_TRANSLATORS_DIRECTORY
B_BEOS_MEDIA_NODES_DIRECTORY
B_BEOS_SOUNDS_DIRECTORY
Common Directories (shared by all users)...
B_COMMON_DIRECTORY
B_COMMON_SYSTEM_DIRECTORY
B_COMMON_ADDONS_DIRECTORY
B_COMMON_BOOT_DIRECTORY
B_COMMON_FONTS_DIRECTORY
B_COMMON_LIB_DIRECTORY
B_COMMON_SERVERS_DIRECTORY
B_COMMON_BIN_DIRECTORY
B_COMMON_ETC_DIRECTORY
B_COMMON_DOCUMENTATION_DIRECTORY
B_COMMON_SETTINGS_DIRECTORY
B_COMMON_DEVELOP_DIRECTORY
B_COMMON_LOG_DIRECTORY
B_COMMON_SPOOL_DIRECTORY
B_COMMON_TEMP_DIRECTORY
B_COMMON_VAR_DIRECTORY
B_COMMON_TRANSLATORS_DIRECTORY
B_COMMON_MEDIA_NODES_DIRECTORY
B_COMMON_SOUNDS_DIRECTORY
User Directories (depends on the current user)...
B_USER_DIRECTORY
B_USER_CONFIG_DIRECTORY
B_USER_ADDONS_DIRECTORY
B_USER_BOOT_DIRECTORY
B_USER_FONTS_DIRECTORY
B_USER_LIB_DIRECTORY
B_USER_SETTINGS_DIRECTORY
B_USER_DESKBAR_DIRECTORY
B_USER_PRINTERS_DIRECTORY
B_USER_TRANSLATORS_DIRECTORY
B_USER_MEDIA_NODES_DIRECTORY
B_USER_SOUNDS_DIRECTORY
Global Directories...
B_APPS_DIRECTORY
B_PREFERENCES_DIRECTORY
B_UTILITIES_DIRECTORY

Storage Kit

<be/StorageKit.h> <libbe.so>

PATHS, FILES, VOLUMES

class **BFile** <be/storage/File.h>

: public BNode, public BPositionIO

BFile([const entry_ref *ref, uint32 open_mode]);
BFile(const BEntry *entry, uint32 open_mode);
BFile(const char *path, uint32 open_mode);
BFile(const BDirectory *dir, const char *path, uint32 open_mode);
BFile(const BFile &file);
virtual ~**BFile**();

BFile & operator**=**(const BFile &file);

bool **IsReadable**() const;
bool **IsWritable**() const;

Redefinitions of BNode inherited functions...
status_t **SetTo**(const entry_ref *ref, uint32 open_mode);
status_t **SetTo**(const BEntry *entry, uint32 open_mode);
status_t **SetTo**(const char *path, uint32 open_mode);
status_t **SetTo**(const BDirectory *dir, const char *path, uint32 open_mode);

BPositionIO inherited virtual functions...
virtual ssize_t **Read**(void *buffer, size_t size);
virtual ssize_t **ReadAt**(off_t pos, void *buffer, size_t size);
virtual ssize_t **Write**(const void *buffer, size_t size);
virtual ssize_t **WriteAt**(off_t pos, const void *buffer, size_t size);
virtual off_t **Seek**(off_t position, uint32 seek_mode);
virtual off_t **Position**() const;
virtual status_t **SetSize**(off_t size);

class **BPath** <be/storage/Path.h>

: public BFlattenable

BPath([const char *dir, const char *leaf = NULL, bool normalize = false]);
BPath(const BDirectory *dir, const char *leaf, bool normalize = false);
BPath(const BPath &path);
BPath(const BEntry *entry);
virtual ~**BPath**();

status_t **InitCheck**() const;

bool operator**==**(const BPath &item) const;
bool operator**==**(const char *path) const;
bool operator**!=**(const BPath &item) const;
bool operator**!=**(const char *path) const;
BPath & operator**=**(const BPath &item);
BPath & operator**=**(const char *path);

status_t **SetTo**(const char *path, const char *leaf = NULL, bool normalize = false);
status_t **SetTo**(const BDirectory *dir, const char *path, bool normalize = false);
status_t **SetTo**(const BEntry *entry);

status_t **Append**(const char *path, bool normalize = false);
void **Unset**();
const char ***Path**() const;
const char ***Leaf**() const;
status_t **GetParent**(BPath *) const;

class **BSymLink** <be/storage/SymLink.h>

: public BNode

BSymLink();
BSymLink(const entry_ref *ref);
BSymLink(const BEntry *entry);
BSymLink(const char *path);
BSymLink(const BDirectory *dir, const char *path);
BSymLink(const BSymLink &link);
virtual ~**BSymLink**();

Note: ReadLink() doesn't traverse to the end of the "link chain. 'path' might be relative.
ssize_t **ReadLink**(char *path, size_t length);

Note: It's up to the caller to pass the correct dir. If the linked-to path is absolute, then the dir is ignored.
ssize_t **MakeLinkedPath**(const char *dir, BPath *path);
ssize_t **MakeLinkedPath**(const BDirectory *dir, BPath *path);

bool **IsAbsolute**();

class **BVolume** <be/storage/Volume.h>

BVolume();
BVolume(dev_t dev);
BVolume(const BVolume &vol);
virtual ~**BVolume**();

status_t **InitCheck**() const;
status_t **SetTo**(dev_t dev);
void **Unset**(void);

bool operator**==**(const BVolume &vol) const;
bool operator**!=**(const BVolume &vol) const;
BVolume & operator**=**(const BVolume &vol);

dev_t **Device**() const;
status_t **GetRootDirectory**(BDirectory *dir) const;
off_t **Capacity**() const;
off_t **FreeBytes**() const;
status_t **GetName**(char *name) const;
status_t **SetName**(const char *name);
status_t **GetIcon**(BBitmap *icon, icon_size which) const;
bool **IsRemovable**() const;
bool **IsReadOnly**() const;
bool **IsPersistent**() const;
bool **IsShared**() const;
bool **KnowsMime**() const;
bool **KnowsAttr**() const;
bool **KnowsQuery**() const;

class **BVolumeRoster** <be/storage/VolumeRoster.h>

BVolumeRoster();
virtual ~**BVolumeRoster**();

status_t **GetNextVolume**(BVolume *vol);
void **Rewind**();
status_t **GetBootVolume**(BVolume *vol);
status_t **StartWatching**(BMessenger msngr = be_app_messenger);
void **StopWatching**(void);

BMessenger **Messenger**(void) const;

FILE PANEL

A file dialog box for opening, saving, and working with files.

Panel Global Functions <be/storage/FilePanel.h>

void **run_open_panel**();
void **run_save_panel**();

class **BRefFilter** <be/storage/FilePanel.h>

virtual bool **Filter**(const entry_ref *, BNode *, struct stat *, const char *mimetype) = 0;

enum 'file_panel_mode' <be/storage/FilePanel.h>

B_OPEN_PANEL
B_SAVE_PANEL

enum 'file_panel_button' <be/storage/FilePanel.h>

B_CANCEL_BUTTON
B_DEFAULT_BUTTON

class **BFilePanel** <be/storage/FilePanel.h>

*Note: Any of these parameters may also be set by function calls, except: **mode**, **node_flavors**, and **modal**.*
BFilePanel(file_panel_mode mode = B_OPEN_PANEL, BMessenger *target = 0, entry_ref *start_directory = 0, uint32 node_flavors = 0, bool allow_multiple_selection = true, BMessage *message = 0, BRefFilter * = 0, bool modal = false, bool hide_when_done = true);
virtual ~**BFilePanel**();

void **Show**();
void **Hide**();
bool **IsShowing**() const;

virtual void **WasHidden**();
virtual void **SelectionChanged**();
virtual void **SendMessage**(const BMessenger*, BMessage*);

BWindow* **Window**() const;
BMessenger **Messenger**() const;
BRefFilter* **RefFilter**() const;
void **GetPanelDirectory**(entry_ref*) const;
file_panel_mode **PanelMode**() const;
void **SetTarget**(BMessenger);
void **SetMessage**(BMessage *msg);

void **SetRefFilter**(BRefFilter* filter);
void **SetSaveText**(const char* text);
void **SetButtonLabel**(file_panel_button, const char* label);

void **SetPanelDirectory**(BEntry* new_directory);
void **SetPanelDirectory**(BDirectory* new_directory);
void **SetPanelDirectory**(entry_ref* new_directory);
void **SetPanelDirectory**(const char *new_directory);

void **SetHideWhenDone**(bool);
bool **HidesWhenDone**(void) const;

void **Refresh**();
void **Rewind**();
status_t **GetNextSelectedRef**(entry_ref*);

Storage Kit

<be/StorageKit.h> <libbe.so>

MIME

A system of associating a set of standard filetype strings with files.

Mime Global Functions <be/storage/Mime.h>

(defined as Extern "C")

int **update_mime_info**(const char *path, int recursive, int synchronous, int force);
status_t **create_app_meta_mime**(const char *path, int recursive, int synchronous, int force);
status_t **get_device_icon**(const char *dev, void *icon, int32 size);

static const uint32 **B_MIME_STRING_TYPE** = 'MIMS';

enum 'icon_size' <be/storage/Mime.h>

B_LARGE_ICON
B_MINI_ICON

enum 'app_verb' <be/storage/Mime.h>

B_OPEN

Mime Types (const char *) <be/storage/Mime.h>

B_APP_MIME_TYPE	*(Platform Dependent)*
B_PEF_APP_MIME_TYPE	"application/x-be-executable"
B_PE_APP_MIME_TYPE	"application/x-vnd.be-peexecutable"
B_ELF_APP_MIME_TYPE	"application/x-vnd.be-elfexecutable"
B_RESOURCE_MIME_TYPE	"application/x-be-resource"
B_FILE_MIME_TYPE	application/octet-stream"

MIME Related Message Codes <be/storage/Mime.h>

B_META_MIME_CHANGED ('MMCH')

MIME Watching Bit Codes <be/storage/Mime.h>

B_ICON_CHANGED
B_PREFERRED_APP_CHANGED
B_ATTR_INFO_CHANGED
B_FILE_EXTENSIONS_CHANGED
B_SHORT_DESCRIPTION_CHANGED
B_LONG_DESCRIPTION_CHANGED
B_ICON_FOR_TYPE_CHANGED
B_APP_HINT_CHANGED
B_EVERYTHING_CHANGED (0xFFFFFFFF)

class **BMimeType** <be/storage/Mime.h>

BMimeType([const char *MIME_type]);
virtual ~**BMimeType**();

status_t **SetTo**(const char *MIME_type);
void **Unset**();
status_t **InitCheck**() const;

String manipulation functions...
const char ***Type**() const;
bool **IsValid**() const;
bool **IsSupertypeOnly**() const;
bool **IsInstalled**() const;
status_t **GetSupertype**(BMimeType *super_type) const;
bool **Contains**(const BMimeType *type) const;
bool operator**==**(const BMimeType &type) const;
bool operator**==**(const char *type) const;

Managing the mime type database...
status_t **Install**();
status_t **Delete**();

status_t **Get/SetIcon**(BBitmap *icon, icon_size size) const;
status_t **Get/SetPreferredApp**(char *signature, app_verb verb = B_OPEN);
status_t **Get/SetAttrInfo**(BMessage *info) const;
status_t **Get/SetFileExtensions**(BMessage *extensions);
status_t **Get/SetShortDescription**(char *description);
status_t **Get/SetLongDescription**(char *description);

status_t **GetSupportingApps**(BMessage *signatures) const;
static status_t **GetInstalledSupertypes**(BMessage *super_types);
static status_t **GetInstalledTypes**(BMessage *types);
static status_t **GetInstalledTypes**(const char *super_type, BMessage *subtypes);
static status_t **GetWildcardApps**(BMessage *wild_ones);
static bool IsValid(const char *string);

status_t **GetAppHint**(entry_ref *ref) const;
status_t **SetAppHint**(const entry_ref *ref);

For mime types...
status_t **GetIconForType**(const char *type, BBitmap *icon, icon_size which) const;
status_t **SetIconForType**(const char *type, const BBitmap *icon, icon_size which);

static status_t **StartWatching**(BMessenger target);
static status_t **StopWatching**(BMessenger target);

QUERIES

A method of asking the file system about files' attributes.

enum 'query_op' <be/storage/Query.h>

B_EQ	B_GT
B_GE	B_LT
B_LE	B_NE
B_CONTAINS	B_BEGINS_WITH
B_ENDS_WITH	B_AND
B_OR	B_NOT
B_INVALID_OP	

class **BQuery** <be/storage/Query.h>

: public BEntryList

BQuery();
virtual ~**BQuery**();

status_t **Clear**();

void **PushAttr**(const char *);
void **PushOp**(query_op op);

void **PushInt32**/**UInt32**/**Int64**/**UInt64**(int32/uint32/int64/uint64);
void **PushFloat**(float c);
void **PushDouble**(double c);
void **PushString**(const char *c, bool case_insensitive = false);

status_t **SetVolume**(const BVolume *vol);
status_t **SetPredicate**(const char *expr);
status_t **SetTarget**(BMessenger msngr);

bool **IsLive**(void) const;

status_t **GetPredicate**(char *buf, size_t length);
size_t **PredicateLength**();

dev_t **TargetDevice**() const;

status_t **Fetch**();

BEntryList inherited functions...
virtual status_t **GetNextEntry**(BEntry *entry, bool traverse = FALSE);
virtual status_t **GetNextRef**(entry_ref *ref);
virtual int32 **GetNextDirents**(struct dirent *buf, size_t length, int32 num = INT_MAX);
*Note: **Rewind()** and **CountEntries()** (inherited from BEntryList) are cannot be used with this class.*

Storage Kit

<be/StorageKit.h> <libbe.so>

C LANGUAGE STORAGE KIT API

C interface to the BeOS file system, link to libroot.so.

Query C API

fs_open_query() Flags <be/kernel/fs_query.h>
B_LIVE_QUERY

fs_query Functions <be/kernel/fs_query.h>

DIR ***fs_open_query**(dev_t device, const char *query, uint32 flags);
DIR ***fs_open_live_query**(dev_t device, const char *query, uint32 flags, port_id port, int32 token);
int **fs_close_query**(DIR *d);
struct dirent ***fs_read_query**(DIR *d);
status_t **get_path_for_dirent**(struct dirent *dent, char *buf, size_t len);

Attributes C API

typedef struct **attr_info** <be/kernel/fs_attr.h>

uint32 **type**;
off_t **size**;

fs_attr Functions <be/kernel/fs_attr.h>

ssize_t **fs_read_attr**(int fd, const char *attribute, uint32 type, off_t pos, void *buf, size_t count);
ssize_t **fs_write_attr**(int fd, const char *attribute, uint32 type, off_t pos, const void *buf, size_t count);
int **fs_remove_attr**(int fd, const char *attr);

DIR * **fs_open_attr_dir**(const char *path);
DIR * **fs_fopen_attr_dir**(int fd);
int **fs_close_attr_dir**(DIR *dirp);
struct dirent ***fs_read_attr_dir**(DIR *dirp);
void **fs_rewind_attr_dir**(DIR *dirp);

int **fs_stat_attr**(int fd, const char *name, struct attr_info *ai);

typedef struct **index_info** <be/kernel/fs_index.h>

uint32 **type**;
off_t **size**;
time_t **modification_time**, **creation_time**;
uid_t **uid**;
gid_t **gid**;

fs_index Functions <be/kernel/fs_index.h>

DIR ***fs_open_index_dir**(dev_t device);
int **fs_close_index_dir**(DIR *d);
struct dirent ***fs_read_index_dir**(DIR *d);
void **fs_rewind_index_dir**(DIR *d);

int **fs_create_index**(dev_t device, const char *name, int type, uint flags);
int **fs_remove_index**(dev_t device, const char *name);
int **fs_stat_index**(dev_t device, const char *name, struct index_info *buf);

File System C API

File System Flags <be/kernel/fs_info.h>
B_FS_IS_READONLY
B_FS_IS_REMOVABLE
B_FS_IS_PERSISTENT
B_FS_IS_SHARED
B_FS_HAS_MIME
B_FS_HAS_ATTR
B_FS_HAS_QUERY

struct **fs_info** <be/kernel/fs_info.h>

dev_t **dev**;
ino_t **root**;
uint32 **flags**;
off_t **block_size**, **io_size**, **total_blocks**, **free_blocks**, **total_nodes**, **free_nodes**;
char **device_name**[128], **volume_name**[B_FILE_NAME_LENGTH], **fsh_name**[B_OS_NAME_LENGTH];

fs_info Functions <be/kernel/fs_info.h>

dev_t **dev_for_path**(const char *path);
dev_t **next_dev**(int32 *pos);
int **fs_stat_dev**(dev_t dev, fs_info *info);

This page is intentionally left blank.

Media Kit

<be/MediaKit.h> <libmedia.so> ***BeOS R4 Programmer's Cheatsheets by David Orr***

MEDIA KIT High Level Interface

PlaySound

Interface for a simple beep sound.

PlaySound Typedefs <be/media/PlaySound.h>

typedef sem_id **sound_handle**;

PlaySound Global Functions <be/media/PlaySound.h>

sound_handle **play_sound**(const entry_ref *soundRef, bool ix, bool queue, bool background);
status_t **stop_sound**(sound_handle handle);
status_t **wait_for_sound**(sound_handle handle);

Sound File Classes

class **BMediaFiles** <be/media/MediaFiles.h>

BMediaFiles();
virtual ~**BMediaFiles**();

virtual status_t **RewindTypes**();
virtual status_t **GetNextType**(char *out_type);
virtual status_t **RewindRefs**(const char *type);
virtual status_t **GetNextRef**(char *out_type, entry_ref *out_ref = NULL);

virtual status_t **GetRefFor**(const char *type, const char *item, entry_ref *out_ref);
virtual status_t **SetRefFor**(const char *type, const char *item, const entry_ref &ref);
virtual status_t **RemoveRefFor**(const char *type, const char *item, const entry_ref &ref);

static const char **B_SOUNDS**[];

Protected:

BMediaFiles(bool start_automatically);

class **BSound** <be/media/Sound.h>

BSound(void *data, size_t size, const media_raw_audio_format &format, bool free_when_done = false);
BSound(const entry_ref *sound_file, bool load_into_memory = false);

status_t **InitCheck**();
BSound ***AcquireRef**();
bool **ReleaseRef**();
int32 **RefCount**() const; *(Note: unreliable!)*

virtual bigtime_t **Duration**() const;
virtual const media_raw_audio_format &**Format**() const;
virtual const void ***Data**() const;
virtual off_t **Size**() const;

virtual bool **GetDataAt**(off_t offset, void *into_buffer, size_t buffer_size, size_t *out_used);

Protected:

BSound(const media_raw_audio_format &format);

virtual status_t **Perform**(int32 code, ...);

struct **media_raw_audio_format** <be/media/MediaDefs.h>

float **frame_rate**;
uint32 **channel_count**;

"format" Codes	
B_AUDIO_UCHAR	B_AUDIO_FLOAT
B_AUDIO_SHORT	B_AUDIO_INT

uint32 **format**;

uint32 **byte_order**;
size_t **buffer_size**;

static media_raw_audio_format wildcard;

Sound Format Codes <be/media/SoundFile.h>	
B_UNKNOWN_FILE	B_AIFF_FILE
B_WAVE_FILE	B_UNIX_FILE

class **BSoundFile** <be/me/SoundFile.h>

BSoundFile([const entry_ref *ref, uint32 open_mode]);
virtual ~**BSoundFile**();

status_t **InitCheck**() const;

status_t **SetTo**(const entry_ref *ref, uint32 open_mode);

bool **IsCompressed**() const;
int32 **CompressionType**() const;
char ***CompressionName**() const;

virtual int32 **SetCompressionType**(int32 type);
virtual char ***SetCompressionName**(char *name);
virtual bool **SetIsCompressed**(bool tf);

int32 **FileFormat**() const;
int32 **SamplingRate**() const;
int32 **CountChannels**() const;
int32 **SampleSize**() const;
int32 **ByteOrder**() const;
int32 **SampleFormat**() const;
int32 **FrameSize**() const;
off_t **CountFrames**() const;

virtual int32 **SetFileFormat**(int32 format);
virtual int32 **SetSamplingRate**(int32 fps);
virtual int32 **SetChannelCount**(int32 spf);
virtual int32 **SetSampleSize**(int32 bps);
virtual int32 **SetByteOrder**(int32 bord);
virtual int32 **SetSampleFormat**(int32 fmt);
virtual off_t **SetDataLocation**(off_t offset);
virtual off_t **SetFrameCount**(off_t count);

size_t **ReadFrames**(char *buf, size_t count);
size_t **WriteFrames**(char *buf, size_t count);
virtual off_t **SeekToFrame**(off_t n);
off_t **FrameIndex**() const;
off_t **FramesRemaining**() const;

BFile ***fSoundFile**;

SoundPlayer

class **sound_error** <be/media/SoundPlayer.h>

: public exception

sound_error(const char *str);
const char ***what**() const;

class **BSoundPlayer** <be/media/SoundPlayer.h>

enum 'sound_player_notification' <be/media/SoundPlayer.	
B_STARTED	B_STOPPED
B_SOUND_DONE	

BSoundPlayer(const char *name = NULL, void (*PlayBuffer)(void *, void *buffer, size_t size, const media_raw_audio_format &format) = NULL, void (*Notifier)(void *, sound_player_notification what, ...) = NULL, void *cookie = NULL);
BSoundPlayer(const media_raw_audio_format *format, const char *name = NULL, void (*PlayBuffer)(void *, void *buffer, size_t size, const media_raw_audio_format &format) = NULL, void (*Notifier)(void *, sound_player_notification what, ...) = NULL, void *cookie = NULL);
virtual ~**BSoundPlayer**();

status_t **Start**();
void **Stop**(bool block = true, bool flush = true);

typedef void (***BufferPlayerFunc**)(void *, void *, size_t, const media_raw_audio_format &);
BufferPlayerFunc **BufferPlayer**() const;

void **SetBufferPlayer**(void (*PlayBuffer)(void *, void *buffer, size_t size, const media_raw_audio_format &format));

typedef void (***EventNotifierFunc**)(void *, sound_player_notification what, ...);
EventNotifierFunc **EventNotifier**() const;

void **SetNotifier**(void (*Notifier)(void *, sound_player_notification what, ...));
void ***Cookie**() const;
void **SetCookie**(void *cookie);
void **SetCallbacks**(void (*PlayBuffer)(void *, void *buffer, size_t size, const media_raw_audio_format &format) = NULL, void (*Notifier)(void *, sound_player_notification what, ...) = NULL, void *cookie = NULL);

typedef int32 play_id;

bigtime_t **CurrentTime**();
play_id **StartPlaying**(BSound *sound, bigtime_t at_time = 0);
bool **IsPlaying**(play_id id);
status_t **StopPlaying**(play_id id);
status_t **WaitForSound**(play_id id);

float **Volume**();
void **SetVolume**(float new_volume);

virtual bool **HasData**();
void **SetHasData**(bool has_data);

Media Kit & Device Kit

<be/DeviceKit..h> <libdevice.so>

Media Kit Error Codes

Media Kit Error Codes <be/media/MediaDefs.h>
B_MEDIA_SYSTEM_FAILURE
B_MEDIA_BAD_NODE
B_MEDIA_NODE_BUSY
B_MEDIA_BAD_FORMAT
B_MEDIA_BAD_BUFFER
B_MEDIA_TOO_MANY_NODES
B_MEDIA_TOO_MANY_BUFFERS
B_MEDIA_NODE_ALREADY_EXISTS
B_MEDIA_BUFFER_ALREADY_EXISTS
B_MEDIA_CANNOT_SEEK
B_MEDIA_CANNOT_CHANGE_RUN_MODE
B_MEDIA_APP_ALREADY_REGISTERED
B_MEDIA_APP_NOT_REGISTERED
B_MEDIA_CANNOT_RECLAIM_BUFFERS
B_MEDIA_BUFFERS_NOT_RECLAIMED
B_MEDIA_TIME_SOURCE_STOPPED
B_MEDIA_TIME_SOURCE_BUSY
B_MEDIA_BAD_SOURCE
B_MEDIA_BAD_DESTINATION
B_MEDIA_ALREADY_CONNECTED
B_MEDIA_NOT_CONNECTED
B_MEDIA_BAD_CLIP_FORMAT
B_MEDIA_ADDON_FAILED
B_MEDIA_ADDON_DISABLED
B_MEDIA_CHANGE_IN_PROGRESS
B_MEDIA_STALE_CHANGE_COUNT
B_MEDIA_ADDON_RESTRICTED
B_MEDIA_NO_HANDLER
B_MEDIA_DUPLICATE_FORMAT

DEVICE KIT

Joystick

class **BJoystick** <be/device/Joystick.h>

BJoystick();
virtual ~**BJoystick**();

status_t **Open**(const char *portName, bool enter_enhanced = TRUE);
void **Close**(void);

status_t **Update**(void);
status_t **SetMaxLatency**(bigtime_t max_latency);

bigtime_t **timestamp**;
int16 **horizontal**, **vertical**;
bool **button1**; *(Note: true == off)*
bool **button2**;

int32 **CountDevices**();
status_t **GetDeviceName**(int32 n, char * name, size_t bufSize = B_OS_NAME_LENGTH);

bool **EnterEnhancedMode**(const entry_ref * ref = NULL);
int32 **CountSticks**();
int32 **CountAxes**();
int32 **CountHats**();
int32 **CountButtons**();
status_t **GetAxisValues**(int16 * out_values, int32 for_stick = 0);
status_t **GetHatValues**(uint8 * out_hats, int32 for_stick = 0);
uint32 **ButtonValues**(int32 for_stick = 0);
status_t **GetAxisNameAt**(int32 index, BString * out_name);
status_t **GetHatNameAt**(int32 index, BString * out_name);
status_t **GetButtonNameAt**(int32 index, BString * out_name);
status_t **GetControllerModule**(BString * out_name);
status_t **GetControllerName**(BString * out_name);

bool **IsCalibrationEnabled**();
status_t **EnableCalibration**(bool calibrates = true);

Protected:
virtual void **Calibrate**(struct _extended_joystick * reading);

Serial Port

enum 'data_rate' <be/devices/SerialPort.h>	
B_0_BPS	B_1800_BPS
B_50_BPS	B_2400_BPS
B_75_BPS	B_4800_BPS
B_110_BPS	B_9600_BPS
B_134_BPS	B_19200_BPS
B_150_BPS	B_38400_BPS
B_200_BPS	B_57600_BPS
B_300_BPS	B_115200_BPS
B_600_BPS	B_230400_BPS
B_1200_BPS	B_31250_BPS

enum 'data_bits' <be/devices/SerialPort.h>	
B_DATA_BITS_7	B_DATA_BITS_8

enum 'stop_bits' <be/devices/SerialPort.h>	
B_STOP_BITS_1	B_STOP_BITS_2

enum 'parity_mode' <be/devices/SerialPort.h>	
B_ODD_PARITY	B_NO_PARITY
B_EVEN_PARITY	

Flow Control <be/devices/SerialPort.h>	
B_NOFLOW_CONTROL	B_HARDWARE_CONTROL
B_SOFTWARE_CONTROL	

class **BSerialPort** <be/device/SerialPort.h>

BSerialPort();
virtual ~**BSerialPort**();

status_t **Open**(const char *portName);
void **Close**(void);

ssize_t **Read**(void *buf, size_t count);
ssize_t **Write**(const void *buf, size_t count);
void **SetBlocking**(bool Blocking);
status_t **SetTimeout**(bigtime_t microSeconds);

status_t **SetDataRate**(data_rate bitsPerSecond);
data_rate **DataRate**();

void **SetDataBits**(data_bits numBits);
data_bits **DataBits**();
void **SetStopBits**(stop_bits numBits);
stop_bits **StopBits**();

void **SetParityMode**(parity_mode which);
parity_mode **ParityMode**();

void **ClearInput**();
void **ClearOutput**();

void **SetFlowControl**(uint32 method);
uint32 **FlowControl**();

status_t **SetDTR**(bool asserted);
status_t **SetRTS**(bool asserted);
status_t **NumCharsAvailable**(int32 *wait_until_this_many);

bool **IsCTS**(void);
bool **IsDSR**(void);
bool **IsRI**(void);
bool **IsDCD**(void);
ssize_t **WaitForInput**(void);

int32 **CountDevices**();
status_t **GetDeviceName**(int32 n, char * name, size_t bufSize = B_OS_NAME_LENGTH);