Buzztrax Bt-Core Reference Manual

COLLABORATORS

	TITLE : Buzztrax Bt-Core Reference Manual	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 18, 2015	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Introduction

Buzztrax aims to be a successor of the freeware music tracker called Buzz with a focus on Linux. The development of Buzz for windows had been discontinued as the author lost all his source-codes. Buzztrax is only related to Buzz in the concepts, Buzztraxs source code is written from scratch.

The homepage of the buzztrax project can be found at www.buzztrax.org. It is a social site containing forums, a wiki and bug tracker and many other resoures.

Part I

Overview

Chapter 1

Conventions

During the API docs some conventions are used to avoid duplication and improve precision.

1. It is never valid to supply NULL for an object reference, unless it is explicitly said so.
2. Methods do checking of preconditions using `g_return_if_fail()` or `g_return_val_if_fail()`. Therefore watch for the glib log messages indicating wrong API usage.
3. All methods that return a pointer to an object, increase the reference counter. Therefore you should do release this reference by calling `g_object_unref()` when you are done with the object.
4. When a return value is marked as const e.g. const gchar * or const BtObjectName * then this is a static reference. You must not call `g_free()` or `g_object_unref()` on it.

Part II

API Reference

Abstract

libbuzztrax-core is the main library for the buzztrax software system. This library contains all classes needed for the components of a song like machines, patterns, wires and so on. It further defines bases classes for common operations such as song input/output. The synthesis engine and the low-level parts of the sequencer are built into GStreamer. Please make sure you have read section conventions before reading further.

All data structures are encapsulated in objects based on GObject. Central starting points are BtApplication and BtSong. All the objects that belong to a song have their song-instance as their member. Likewise all objects that belong to the applications have their application instance as a member.

Chapter 2

Core Class Reference

2.1 libbtcore

libbtcore — core library of the buzztrax application framework

Functions

#define	BT_IS_GVALUE()
#define	BT_IS_STRING()
guint	bt_cpu_load_get_current ()
void	bt_init ()
void	bt_init_add_option_groups ()
gboolean	bt_init_check ()
GOptionGroup *	bt_init_get_option_group ()
void	bt_deinit ()
#define	return_if_disposed
#define	return_val_if_disposed()
#define	safe_string()
#define	G_OBJECT_REF_COUNT()
#define	G_OBJECT_LOG_REF_COUNT()
GType	bt_g_type_get_base_type ()
guint	bt_g_object_idle_add ()
gulong	bt_g_signal_connect ()
#define	g_object_try_ref()
#define	g_object_try_unref()
#define	g_object_try_weak_ref()
#define	g_object_try_weak_unref()
gboolean	bt_bin_activate_tee_chain ()
gboolean	bt_bin_deactivate_tee_chain ()
GstClockTime	bt_gst_analyzer_get_waittime ()
GList *	bt_gst_check_core_elements ()
GList *	bt_gst_check_elements ()
const gchar *	bt_gst_debug_pad_link_return ()
GstPadTemplate *	bt_gst_element_factory_get_pad_template ()
gboolean	bt_gst_element_factory_can_sink_media_type ()
GList *	bt_gst_registry_get_element_factories_matching_all_categories ()
GList *	bt_gst_registry_get_element_names_matching_all_categories ()
gboolean	bt_gst_try_element ()
gdouble	bt_gst_level_message_get_aggregated_field ()

const gchar *	bt_str_format_double ()
const gchar *	bt_str_format_enum ()
const gchar *	bt_str_format_long ()
const gchar *	bt_str_format_uchar ()
const gchar *	bt_str_format_ulong ()
gint	bt_str_parse_enum ()
gchar *	bt_str_format_gvalue ()
gboolean	bt_str_parse_gvalue ()

Types and Values

extern const guint	bt_major_version
extern const guint	bt_micro_version
extern const guint	bt_minor_version
#define	G_OBJECT_REF_COUNT_FMT

Includes

```
#include <libbtcore/core.h>
```

Description

The library offers base objects such as BtApplication and BtSong.

Functions

BT_IS_GVALUE()

```
#define BT_IS_GVALUE(v) (G_VALUE_TYPE(v)!=G_TYPE_INVALID)
```

checks if the supplied gvalue is initialized (not all fields zero).

Parameters

v	pointer to a GValue

BT_IS_STRING()

```
#define BT_IS_STRING(a) (a && *a)
```

Checks if the supplied string pointer is not NULL and contains not just '\0'

Parameters

a	string pointer

bt_cpu_load_get_current ()

```
guint
```

```
bt_cpu_load_get_current (void);
```

Determines the current CPU load. Run this from a timeout handler (with e.g. a 1 second interval).

Returns

CPU usage as integer ranging from 0% to 100%

bt_init ()

```
void
bt_init (gint *argc,
         gchar **argv[]);
```

Initializes the Buzztrax Core library.

Note
This function will terminate your program if it was unable to initialize the core for some reason. If you want your program to fall back, use bt_init_check() instead.

WARNING: This function does not work in the same way as corresponding functions in other glib-style libraries, such as gtk_init(). In particular, unknown command line options cause this function to abort program execution.

Parameters

argc	pointer to application's argc.	*[inout][allow-none]*
argv	pointer to application's argv.	*[inout][array length=argc][allow-none]*

bt_init_add_option_groups ()

```
void
bt_init_add_option_groups (GOptionContext * const ctx);
```

Adds all option groups to the main context the core library will pull in.

Parameters

ctx	main option context	

bt_init_check ()

```
gboolean
bt_init_check (gint *argc,
               gchar **argv[],
               GError **err);
```

Initializes the Buzztrax core library.

This function will return FALSE if Buzztrax core could not be initialized for some reason. If you want your program to fail fatally, use bt_init() instead.

Parameters

argc	pointer to application's argc.	*[inout][allow-none]*
argv	pointer to application's argv.	*[inout][array length=argc][allow-none]*
err	pointer to a GError to which a message will be posted on error	

Returns

TRUE if Buzztrax core could be initialized.

bt_init_get_option_group ()

```
GOptionGroup~*
bt_init_get_option_group (void);
```

Returns a GOptionGroup with libbtcore's argument specifications. The group is set up to use standard GOption callbacks, so when using this group in combination with GOption parsing methods, all argument parsing and initialization is automated.

This function is useful if you want to integrate libbtcore with other libraries that use GOption (see g_option_context_add_group()).

[skip]

Returns

a pointer to a GOption group. Should be dereferenced after use.

[transfer full]

bt_deinit ()

```
void
bt_deinit (void);
```

It is normally not needed to call this function in a normal application as the resources will automatically be freed when the program terminates. This function is therefore mostly used by testsuites and other memory profiling tools.

return_if_disposed

```
#define return_if_disposed() if(self->priv->dispose_has_run) return
```

Checks `self->priv->dispose_has_run` and if TRUE returns. This macro is handy to use at the start of all class routines such as GObjectClass.get_property(), GObjectClass.set_property(), GObjectClass.dispose().

return_val_if_disposed()

```
#define return_val_if_disposed(a) if(self->priv->dispose_has_run) return(a)
```

Checks `self->priv->dispose_has_run` and if TRUE returns with the supplied arg `a` . This macro is handy to use at the start of e.g. idle handlers.

Parameters

a	return value

safe_string()

```
#define safe_string(a)  ((gchar *)(a)?(gchar *)(a):"")
```

Pass the supplied string through or return an empty string when it is NULL.

Parameters

a	string pointer

Returns

the given string or an empty string in the case of a NULL argument

G_OBJECT_REF_COUNT()

```
#define G_OBJECT_REF_COUNT(obj)  ((obj)?((G_OBJECT(obj))->ref_count):0)
```

Read the objects reference counter. Implemented as a macro, so don't use expressions for *obj* .

Parameters

obj	the object (may be NULL)

Returns

the reference counter.

G_OBJECT_LOG_REF_COUNT()

```
#define             G_OBJECT_LOG_REF_COUNT(o)
```

Logs an object pointer together with its refcount value and the floating flag. Use with G_OBJECT_REF_COUNT_FMT.

Parameters

o	the object (may be NULL)

bt_g_type_get_base_type ()

```
GType
bt_g_type_get_base_type (const GType type);
```

Call g_type_parent() as long as it returns a parent.

Parameters

type	a GType	

Returns

the super parent type, aka base type.

bt_g_object_idle_add ()

```
guint
bt_g_object_idle_add (GObject *obj,
                      gint pri,
                      GSourceFunc func);
```

A g_idle_add_full() variant, that passes `obj` as user_data and detaches the handler when `obj` gets destroyed.

Parameters

obj	the old GObject	
pri	the priority of the idle source, e.g. G_PRIORITY_DEFAULT_IDLE	
func	the callback.	*[scope async]*

Returns

the handler id

bt_g_signal_connect ()

```
gulong
bt_g_signal_connect (gpointer instance,
                     const gchar *detailed_signal,
                     GCallback c_handler,
                     gpointer data);
```

Like g_signal_connect(), but checks first if the handler is already connected.

Parameters

instance	the instance to connect to.	*[type GObject.Object]*
detailed_signal	a string of the form "signal-name::detail".	
c_handler	the GCallback to connect.	*[scope async]*
data	data to pass to c_handler calls.	

Returns

the handler id

g_object_try_ref()

```
#define             g_object_try_ref(obj)
```

If the supplied object is not NULL then reference it via g_object_ref().

Parameters

obj	the object to reference

Returns

the referenced object or NULL

g_object_try_unref()

```
#define             g_object_try_unref(obj)
```

If the supplied object is not NULL then release the reference via g_object_unref().

Parameters

obj	the object to release the reference

g_object_try_weak_ref()

```
#define             g_object_try_weak_ref(obj)
```

If the supplied object is not NULL then reference it via g_object_add_weak_pointer().

Parameters

obj	the object to reference

g_object_try_weak_unref()

```
#define             g_object_try_weak_unref(obj)
```

If the supplied object is not NULL then release the reference via g_object_remove_weak_pointer().

Parameters

obj	the object to release the reference

bt_bin_activate_tee_chain ()

```
gboolean
bt_bin_activate_tee_chain (GstBin *bin,
                           GstElement *tee,
                           GList *elements,
                           gboolean is_playing);
```

Add the *elements* to the *bin* and link them. Handle pad blocking in playing mode.

Return: TRUE for success

Parameters

bin	the bin	
tee	the tee to connect the chain to	
elements	the list of elements to activate.	*[element-type Gst.Element]*
is_playing	whether the pipeline is streaming data	

bt_bin_deactivate_tee_chain ()

```
gboolean
bt_bin_deactivate_tee_chain (GstBin *bin,
                             GstElement *tee,
                             GList *elements,
                             gboolean is_playing);
```

Add the *elements* to the *bin* and link them. Handle pad blocking in playing mode.

Return: TRUE for success

Parameters

bin	the bin	
tee	the tee to connect the chain to	
elements	the list of elements to deactivate.	*[element-type Gst.Element]*
is_playing	wheter the pipeline is streaming data	

bt_gst_analyzer_get_waittime ()

```
GstClockTime
bt_gst_analyzer_get_waittime (GstElement *analyzer,
                              const GstStructure *structure,
                              gboolean endtime_is_running_time);
```

Get the time to wait for audio corresponding to the analyzed data to be rendered.

Parameters

analyzer	the analyzer	
structure	the message data	

endtime_is_running_time	some elements (level) report endtime as running time and therefore need segment correction	

Returns

the wait time in ns.

bt_gst_check_core_elements ()

```
GList~*
bt_gst_check_core_elements (void);
```

Check if all core elements exist.

Returns

a list of elements that does not exist, NULL if all elements exist. The list is static, don't free or modify.

[element-type utf8][transfer none]

bt_gst_check_elements ()

```
GList~*
bt_gst_check_elements (GList *list);
```

Check if the given elements exist.

Parameters

list	a GList with element names.	*[element-type utf8]*

Returns

a list of element-names which do not exist, NULL if all elements exist, g_list_free after use.

[element-type utf8][transfer full]

bt_gst_debug_pad_link_return ()

```
const gchar~*
bt_gst_debug_pad_link_return (GstPadLinkReturn link_res,
                              GstPad *src_pad,
                              GstPad *sink_pad);
```

Format a nice debug message from failed pad links.

Parameters

link_res	pad link result	
src_pad	the source pad	
sink_pad	the sink pad	

Returns

the message. The returned string has to be used before the can be called again, otherwise the previous reult will be overwritten.

[transfer none]

bt_gst_element_factory_get_pad_template ()

```
GstPadTemplate~*
bt_gst_element_factory_get_pad_template
                                    (GstElementFactory *factory,
                                     const gchar *name);
```

Lookup a pad template by name.

Parameters

factory	element factory	
name	name of the pad-template, e.g. "src" or "sink_u"	

Returns

the pad template or NULL if not found.

[transfer full]

bt_gst_element_factory_can_sink_media_type ()

```
gboolean
bt_gst_element_factory_can_sink_media_type
                                    (GstElementFactory *factory,
                                     const gchar *name);
```

Check if the sink pads of the given `factory` are compatible with the given `name` . The `name` can e.g. be "audio/x-raw".

Parameters

factory	element factory to check	
name	caps type name	

Returns

TRUE if the pads are compatible.

bt_gst_registry_get_element_factories_matching_all_categories ()

```
GList~*
bt_gst_registry_get_element_factories_matching_all_categories
                                    (const gchar *class_filter);
```

Iterates over all available elements and filters by categories given in `class_filter` .

Parameters

class_filter	path for filtering (e.g. "Sink/Audio")

Returns

list of element factories, use gst_plugin_feature_list_free() after use.

[element-type Gst.PluginFeature][transfer full]

Since: 0.6

bt_gst_registry_get_element_names_matching_all_categories ()

```
GList~*
bt_gst_registry_get_element_names_matching_all_categories
                               (const gchar *class_filter);
```

Iterates over all available elements and filters by categories given in `class_filter` .

Parameters

class_filter	path for filtering (e.g. "Sink/Audio")

Returns

list of read-only element names, g_list_free after use.

[element-type utf8][transfer container]

bt_gst_try_element ()

```
gboolean
bt_gst_try_element (GstElementFactory *factory,
                    const gchar *format);
```

Create an instance of the element and try to set it to GST_STATE_READY.

Parameters

factory	plugin feature to try
format	required media format

Returns

TRUE, if the element is usable

bt_gst_level_message_get_aggregated_field ()

```
gdouble
bt_gst_level_message_get_aggregated_field
```

```
                                    (const GstStructure *structure,
                                     const gchar *field_name,
                                     gdouble default_value);
```

Aggregate the levels per channel and return the averaged level.

Parameters

structure	the message structure	
field_name	the field, such as 'decay' or 'peak'	
default_value	a default, in the case of inf/nan levels	

Returns

the average level field for all channels

bt_str_format_double ()

```
const gchar~*
bt_str_format_double (const gdouble val);
```

Convenience methods, that formats a value to be serialized as a string.

Parameters

val	a value	

Returns

a reference to static memory containing the formatted value.

[transfer none]

bt_str_format_enum ()

```
const gchar~*
bt_str_format_enum (GType enum_type,
                    gint value);
```

Convenience methods, that formats a value to be serialized as a string.

Parameters

enum_type	the GType for the enum	
value	the integer value for the enum	

Returns

a reference to static memory containing the formatted value.

[transfer none]

bt_str_format_long ()

```
const gchar~*
bt_str_format_long (const glong val);
```

Convenience methods, that formats a value to be serialized as a string.

Parameters

val	a value

Returns

a reference to static memory containing the formatted value.

[transfer none]

bt_str_format_uchar ()

```
const gchar~*
bt_str_format_uchar (const guchar val);
```

Convenience methods, that formats a value to be serialized as a string.

Parameters

val	a value

Returns

a reference to static memory containing the formatted value.

[transfer none]

bt_str_format_ulong ()

```
const gchar~*
bt_str_format_ulong (const gulong val);
```

Convenience methods, that formats a value to be serialized as a string.

Parameters

val	a value

Returns

a reference to static memory containing the formatted value.

[transfer none]

bt_str_parse_enum ()

```
gint
bt_str_parse_enum (GType enum_type,
                   const gchar *str);
```

Convenience methods, that parses a enum value nick and to get the corresponding integer value.

Parameters

enum_type	the GType for the enum	
str	the enum value name	

Returns

the integer value for the enum, or -1 for invalid strings.

bt_str_format_gvalue ()

```
gchar~*
bt_str_format_gvalue (GValue * const gvalue);
```

Returns the string representation of the given `gvalue` . Free it when done.

Parameters

gvalue	the event cell	

Returns

a newly allocated string with the data or NULL on error.

[transfer full]

bt_str_parse_gvalue ()

```
gboolean
bt_str_parse_gvalue (GValue * const gvalue,
                     const gchar * const svalue);
```

Stores the supplied value into the given `gvalue` .

Parameters

gvalue	a GValue	
svalue	the string representation of the value to store	

Returns

TRUE for success

Types and Values

bt_major_version

```
extern const guint bt_major_version;
```

buzztrax version stamp, major part

bt_micro_version

```
extern const guint bt_micro_version;
```

buzztrax version stamp, micro part

bt_minor_version

```
extern const guint bt_minor_version;
```

buzztrax version stamp, minor part

G_OBJECT_REF_COUNT_FMT

```
#define G_OBJECT_REF_COUNT_FMT "p,ref_ct=%d,floating=%d"
```

Printf format string for G_OBJECT_LOG_REF_COUNT.

2.2 BtApplication

BtApplication — base class for a buzztrax based application

Properties

GstBin *	bin	Read
BtSettings *	settings	Read

Types and Values

struct	BtApplication

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtApplication
```

Includes

```
#include <libbtcore/core.h>
```

Description

Every application using the libbtcore library should inherit from this class. Implementations should implement the singleton pattern.

The base class automatically creates a GstBin element as a container for the song. This can be retrieved via the “bin” property. When creating BtSong instances, the BtApplication instance needs to be passed to the bt_song_new() constructor, so that it can retrieve the GstBin element.

```
BtApplication *app;
BtSong *song;
...
song=bt_song_new(app);
```

Another module the application base class maintains is a settings instance (see BtSettings), that manages application preferences.

Functions

Types and Values

struct BtApplication

```
struct BtApplication;
```

base object for a buzztrax based application

Property Details

The “bin” property

```
  “bin”                        GstBin~*
```

The top-level gstreamer element for the song, e.g. a GstPipeline or GstBin.

Flags: Read

The “settings” property

```
  “settings”                   BtSettings~*
```

applications configuration settings.

Flags: Read

2.3 BtAudioSession

BtAudioSession — bin to be used by BtSinkMachine

Functions

BtAudioSession *	bt_audio_session_new ()

Properties

gboolean	audio-locked	Read / Write
GstElement *	audio-sink	Read
gchar *	audio-sink-device	Read / Write
gchar *	audio-sink-name	Read / Write

Types and Values

struct	BtAudioSession

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtAudioSession
```

Includes

```
#include <libbtcore/core.h>
```

Description

The audio-session provides a persistent audio-sink for some classes. This e.g. ensures a persistent presence in qjackctrl if jackaudiosink is used.

The top BtApplication should create one and dispose it at the end of the lifecycle. The audio-session is a singleton, parts in the code can just call bt_audio_session_new() to get the instance.

Functions

bt_audio_session_new ()

```
BtAudioSession~*
bt_audio_session_new (void);
```

Create a new audio-session or return the existing one. The audio session keeps the audio setup alive across songs. An application can only have one audio-session. This method can be called several times though.

Returns

the audio-session, unref when done.

Types and Values

struct BtAudioSession

```
struct BtAudioSession;
```

Maintains the audio connection for the life time of the application.

Property Details

The “audio-locked” property

```
“audio-locked”             gboolean
```

locked state for the audio-sink.

Flags: Read / Write

Default value: FALSE

The “audio-sink” property

```
“audio-sink”               GstElement~*
```

the audio-sink for the session.

Flags: Read

The “audio-sink-device” property

```
“audio-sink-device”        gchar~*
```

The name of the audio sink device.

Flags: Read / Write

Default value: NULL

The “audio-sink-name” property

```
“audio-sink-name”          gchar~*
```

The name of the audio sink factory.

Flags: Read / Write

Default value: NULL

2.4 BtSettings

BtSettings — class for buzztrax settings handling

Functions

gboolean	bt_settings_determine_audiosink_name ()
GHashTable *	bt_settings_parse_ic_playback_spec ()
gchar *	bt_settings_format_ic_playback_spec ()
BtSettings *	bt_settings_make ()

Properties

gchar *	audiosink	Read / Write
gchar *	audiosink-device	Read / Write
guint	channels	Read / Write
gboolean	coherence-upnp-active	Read / Write
guint	coherence-upnp-port	Read / Write
gboolean	compact-theme	Read / Write
gboolean	dark-theme	Read / Write
gchar *	grid-density	Read / Write
gboolean	ic-playback-active	Read / Write
gchar *	ic-playback-spec	Read / Write
gboolean	jack-transport-master	Read / Write
gboolean	jack-transport-slave	Read / Write
guint	latency	Read / Write
gchar *	missing-machines	Read / Write
guint	news-seen	Read / Write
gchar *	presented-tips	Read / Write
gchar *	record-folder	Read / Write
gchar *	sample-folder	Read / Write
guint	sample-rate	Read / Write
gboolean	show-tips	Read / Write
gchar *	song-folder	Read / Write
gboolean	statusbar-hide	Read / Write
gchar *	system-audiosink	Read
gboolean	tabs-hide	Read / Write
gboolean	toolbar-hide	Read / Write
gchar *	toolbar-style	Read
gint	window-height	Read / Write
gint	window-width	Read / Write
gint	window-xpos	Read / Write
gint	window-ypos	Read / Write

Types and Values

struct	BtSettings

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSettings
```

Includes

```
#include <libbtcore/core.h>
```

Description

Wraps the settings a GObject. Single settings are accessed via normal g_object_get() and g_object_set() calls. Changes in the settings will be notified to the application by the GObject::notify signal.

Functions

bt_settings_determine_audiosink_name ()

```
gboolean
bt_settings_determine_audiosink_name (const BtSettings * const self,
                                      gchar **element_name,
                                      gchar **device_name);
```

Check the settings for the configured audio sink. Pick a fallback if none has been chosen. Verify that the sink works.

Free the strings in the output variables, when done.

Parameters

self	the settings	
element_name	out variable for the element name	
device_name	out variable for the device property, if any	

Returns

TRUE if a audiosink has been found.

bt_settings_parse_ic_playback_spec ()

```
GHashTable~*
bt_settings_parse_ic_playback_spec (const gchar *spec);
```

Parses the string.

Parameters

spec	the spec string from the settings	

Returns

a hashtable with strings as keys and values.

[element-type utf8 utf8][transfer full]

bt_settings_format_ic_playback_spec ()

```
gchar~*
bt_settings_format_ic_playback_spec (GHashTable *ht);
```

Format the settings as a string.

Parameters

ht	the ht settings	

Returns

a string for storage.

bt_settings_make ()

```
BtSettings~*
bt_settings_make (void);
```

Create a new instance. The type of the settings depends on the subsystem found during configuration run.

Settings are implemented as a singleton. Thus the first invocation will create the object and further calls will just give back a reference.

Returns

the instance or NULL in case of an error.

[transfer full]

Types and Values

struct BtSettings

```
struct BtSettings;
```

base object for a buzztrax based settings

Property Details

The “audiosink” property

```
  “audiosink”                gchar~*
```

audio output gstreamer element.

Flags: Read / Write

Default value: NULL

The “audiosink-device” property

```
  “audiosink-device”         gchar~*
```

audio output device name.

Flags: Read / Write

Default value: NULL

The “channels” property

```
  “channels”                 guint
```

number of audio output channels.

Flags: Read / Write

Allowed values: [1,2]

Default value: 2

The “coherence-upnp-active” property

```
“coherence-upnp-active”    gboolean
```

activate Coherence UPnP based playback controller.

Flags: Read / Write

Default value: FALSE

The “coherence-upnp-port” property

```
“coherence-upnp-port”      guint
```

the port number for the communication with the coherence backend.

Flags: Read / Write

Default value: 7654

The “compact-theme” property

```
“compact-theme”            gboolean
```

use dense theme variant for small screens.

Flags: Read / Write

Default value: FALSE

The “dark-theme” property

```
“dark-theme”               gboolean
```

use dark theme variant.

Flags: Read / Write

Default value: FALSE

The “grid-density” property

```
“grid-density”             gchar~*
```

machine view grid detail level.

Flags: Read / Write

Default value: "low"

The “ic-playback-active” property

```
“ic-playback-active”       gboolean
```

activate interaction controller library based playback controller.

Flags: Read / Write

Default value: FALSE

The “ic-playback-spec” property

```
“ic-playback-spec”         gchar~*
```

list of device and control names.

Flags: Read / Write

Default value: NULL

The “jack-transport-master” property

```
“jack-transport-master”    gboolean
```

sync other jack clients to buzztrax playback state.

Flags: Read / Write

Default value: FALSE

The “jack-transport-slave” property

```
“jack-transport-slave”     gboolean
```

sync buzztrax to the playback state other jack clients.

Flags: Read / Write

Default value: FALSE

The “latency” property

```
“latency”                  guint
```

target audio latency in ms.

Flags: Read / Write

Allowed values: [1,200]

Default value: 30

The “missing-machines” property

```
“missing-machines”         gchar~*
```

list of tip-numbers that were shown already.

Flags: Read / Write

Default value: NULL

The “news-seen” property

```
“news-seen”                guint
```

version number for that the user has seen the news.

Flags: Read / Write

Default value: 0

The “`presented-tips`” property

```
“presented-tips”          gchar~*
```

list of missing machines to ignore.

Flags: Read / Write

Default value: NULL

The “`record-folder`” property

```
“record-folder”           gchar~*
```

default directory for recordings.

Flags: Read / Write

Default value: "/home/ensonic"

The “`sample-folder`” property

```
“sample-folder”           gchar~*
```

default directory for sample-waveforms.

Flags: Read / Write

Default value: "/home/ensonic"

The “`sample-rate`” property

```
“sample-rate”             guint
```

audio output sample-rate.

Flags: Read / Write

Allowed values: [1,96000]

Default value: 44100

The “`show-tips`” property

```
“show-tips”               gboolean
```

show tips on startup.

Flags: Read / Write

Default value: TRUE

The “`song-folder`” property

```
“song-folder”             gchar~*
```

default directory for songs.

Flags: Read / Write

Default value: "/home/ensonic"

The “statusbar-hide” property

```
“statusbar-hide”           gboolean
```

hide bottom statusbar.

Flags: Read / Write

Default value: FALSE

The “system-audiosink” property

```
“system-audiosink”         gchar~*
```

system audio output gstreamer element.

Flags: Read

Default value: NULL

The “tabs-hide” property

```
“tabs-hide”                gboolean
```

hide main page tabs.

Flags: Read / Write

Default value: FALSE

The “toolbar-hide” property

```
“toolbar-hide”             gboolean
```

hide main toolbar.

Flags: Read / Write

Default value: FALSE

The “toolbar-style” property

```
“toolbar-style”            gchar~*
```

system tolbar style.

Flags: Read

Default value: "both"

The “window-height” property

```
“window-height”            gint
```

last application window height.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “window-width” property

```
“window-width”              gint
```

last application window width.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The “window-xpos” property

```
“window-xpos”               gint
```

last application window x-position.

Flags: Read / Write

Default value: 0

The “window-ypos” property

```
“window-ypos”               gint
```

last application window y-position.

Flags: Read / Write

Default value: 0

2.5 BtChildProxy

BtChildProxy — Interface for multi child elements.

Functions

void	bt_child_proxy_get ()
GObject *	bt_child_proxy_get_child_by_index ()
GObject *	bt_child_proxy_get_child_by_name ()
guint	bt_child_proxy_get_children_count ()
void	bt_child_proxy_get_property ()
void	bt_child_proxy_get_valist ()
gboolean	bt_child_proxy_lookup ()
void	bt_child_proxy_set ()
void	bt_child_proxy_set_property ()
void	bt_child_proxy_set_valist ()

Types and Values

	BtChildProxy
struct	BtChildProxyInterface

Object Hierarchy

```
    GInterface
    &#x2570;&#x2500;&#x2500; BtChildProxy
```

Includes

```
#include <libbtcore/core.h>
```

Description

This interface abstracts handling of property sets for elements with children. Imagine elements such as mixers or polyphonic generators. They all have multiple GstPad or some kind of voice objects. Another use case are container elements like GstBin. The element implementing the interface acts as a parent for those child objects.

By implementing this interface the child properties can be accessed from the parent element by using bt_child_proxy_get() and bt_child_proxy_set().

Property names are written as "child-name::property-name". The whole naming scheme is recursive. Thus "child1::child2::property" is valid too, if "child1" and "child2" are objects that implement the interface or are properties that return a GObject. The later is a convenient way to set or get properties a few hops down the hierarchy in one go (without being able to forget the unrefs of the intermediate objects).

Functions

bt_child_proxy_get ()

```
void
bt_child_proxy_get (gpointer object,
                    const gchar *first_property_name,
                    ...);
```

Gets properties of the parent object and its children.

Parameters

object	the parent object	
first_property_name	name of the first property to get	
...	return location for the first property, followed optionally by more name/return location pairs, followed by NULL	

bt_child_proxy_get_child_by_index ()

```
GObject~*
bt_child_proxy_get_child_by_index (BtChildProxy *parent,
                                   guint index);
```

Fetches a child by its number.

Parameters

parent	the parent object to get the child from
index	the childs position in the child list

Returns

the child object or NULL if not found (index too high). Unref after usage.

[transfer full]

bt_child_proxy_get_child_by_name ()

```
GObject~*
bt_child_proxy_get_child_by_name (BtChildProxy *parent,
                                  const gchar *name);
```

Looks up a child element by the given name.

Parameters

parent	the parent object to get the child from
name	the childs name

Returns

the child object or NULL if not found. Unref after usage.

[transfer full]

bt_child_proxy_get_children_count ()

```
guint
bt_child_proxy_get_children_count (BtChildProxy *parent);
```

Gets the number of child objects this parent contains.

Parameters

parent	the parent object

Returns

the number of child objects

bt_child_proxy_get_property ()

```
void
bt_child_proxy_get_property (GObject *object,
                             const gchar *name,
                             GValue *value);
```

Gets a single property using the BtChildProxy mechanism. You are responsible for for freeing it by calling g_value_unset()

Parameters

object	object to query	
name	name of the property	
value	a GValue that should take the result.	

bt_child_proxy_get_valist ()

```
void
bt_child_proxy_get_valist (GObject *object,
                           const gchar *first_property_name,
                           va_list var_args);
```

Gets properties of the parent object and its children.

Parameters

object	the object to query	
first_property_name	name of the first property to get	
var_args	return location for the first property, followed optionally by more name/return location pairs, followed by NULL	

bt_child_proxy_lookup ()

```
gboolean
bt_child_proxy_lookup (GObject *object,
                       const gchar *name,
                       GObject **target,
                       GParamSpec **pspec);
```

Looks up which object and GParamSpec would be effected by the given `name` .

Parameters

object	object to lookup the property in	
name	name of the property to look up	
target	pointer to a GObject that takes the real object to set property on	
pspec	pointer to take the GParamSpec describing the property	

Returns

TRUE if `target` and `pspec` could be found. FALSE otherwise. In that case the values for `pspec` and `target` are not modified. Unref `target` after usage.

bt_child_proxy_set ()

```
void
bt_child_proxy_set (gpointer object,
                    const gchar *first_property_name,
                    ...);
```

Sets properties of the parent object and its children.

Parameters

object	the parent object	
first_property_name	name of the first property to set	
...	value for the first property, followed optionally by more name/value pairs, followed by NULL	

bt_child_proxy_set_property ()

```
void
bt_child_proxy_set_property (GObject *object,
                             const gchar *name,
                             const GValue *value);
```

Sets a single property using the BtChildProxy mechanism.

Parameters

object	the parent object	
name	name of the property to set	
value	new GValue for the property	

bt_child_proxy_set_valist ()

```
void
bt_child_proxy_set_valist (GObject *object,
                           const gchar *first_property_name,
                           va_list var_args);
```

Sets properties of the parent object and its children.

Parameters

object	the parent object	

first_property_name	name of the first property to set
var_args	value for the first property, followed optionally by more name/value pairs, followed by NULL

Types and Values

BtChildProxy

```
typedef struct _BtChildProxy BtChildProxy;
```

Opaque interface handle.

struct BtChildProxyInterface

```
struct BtChildProxyInterface {
  /* virtual methods */
  GObject *(*get_child_by_name)(BtChildProxy *parent,const gchar *name);
  GObject *(*get_child_by_index) (BtChildProxy *parent,guint index);
  guint (*get_children_count) (BtChildProxy *parent);
};
```

BtChildProxy interface.

Members

get_child_by_name ()	virtual method to fetch the child by name
get_child_by_index ()	virtual method to fetch the child by in-dex
get_children_count ()	virtual method to get the chil-dren count

2.6 BtPersistence

BtPersistence — object persistence interface

Functions

void	bt_persistence_collect_hashtable_entries ()
BtPersistence *	bt_persistence_load ()
gboolean	bt_persistence_load_hashtable ()
xmlNodePtr	bt_persistence_save ()
gboolean	bt_persistence_save_hashtable ()
gboolean	bt_persistence_save_list ()

Types and Values

	BtPersistence
struct	BtPersistenceInterface

Object Hierarchy

```
    GInterface
    &#x2570;&#x2500;&#x2500; BtPersistence
```

Known Implementations

BtPersistence is implemented by BtMachine, BtPattern, BtProcessorMachine, BtSequence, BtSetup, BtSinkMachine, BtSong, BtSongInfo, BtSourceMachine, BtWave, BtWavelevel, BtWavetable and BtWire.

Includes

```
#include <libbtcore/core.h>
```

Description

Classes can implement this interface to store their data as xml and restore them from xml. They should call the interface methods on their children objects (which also implement the interface) to serialize/ deserialize a whole object hierarchy.

Functions

bt_persistence_collect_hashtable_entries ()

```
void
bt_persistence_collect_hashtable_entries
                               (gpointer const key,
                                gpointer const value,
                                gpointer const user_data);
```

Gather GHashTable entries in a list. Should be used with g_hash_table_foreach().

Parameters

key	hashtable key	
value	hashtable value	
user_data	GList **list	

bt_persistence_load ()

```
BtPersistence~*
bt_persistence_load (const GType type,
                     const BtPersistence * const self,
                     xmlNodePtr node,
                     GError **err,
                     ...);
```

Deserializes the given object from the `node` . If `self` is NULL and a `type` is given it constructs a new object.

Parameters

type	a GObject type	
self	a deserialiable object	
node	the xml node	
err	a GError for deserialisation errors	
...	extra parameters NULL terminated name/value pairs.	

Returns

the deserialized object or NULL.

[transfer none]

bt_persistence_load_hashtable ()

```
gboolean
bt_persistence_load_hashtable (GHashTable *hashtable,
                               xmlNodePtr node);
```

Iterates over the xml-node and deserializes elements into the hashtable.

Parameters

hashtable	a GHashTable.	*[element-type utf8 utf8]*
node	the list xml node	

Returns

TRUE if all elements have been deserialized.

bt_persistence_save ()

```
xmlNodePtr
bt_persistence_save (const BtPersistence * const self,
                        xmlNodePtr const parent_node);
```

Serializes the given object into `node` .

Parameters

self	a serialiable object	
parent_node	the parent xml node	

Returns

the new node if the object has been serialized, else NULL.

bt_persistence_save_hashtable ()

```
gboolean
bt_persistence_save_hashtable (GHashTable *hashtable,
                               xmlNodePtr const node);
```

Iterates over a hashtable with strings and serializes them.

Parameters

hashtable	a GHashTable with strings.	*[element-type utf8 utf8]*
node	the list xml node	

Returns

TRUE if all elements have been serialized.

bt_persistence_save_list ()

```
gboolean
bt_persistence_save_list (const GList *list,
                          xmlNodePtr const node);
```

Iterates over a list of objects, which must implement the BtPersistence interface and calls bt_persistence_save() on each item.

Parameters

list	a GList `doc` ; the xml-document.	*[element-type BuzztraxCore.Persistence]*
node	the list xml node	

Returns

TRUE if all elements have been serialized.

Types and Values

BtPersistence

```
typedef struct _BtPersistence BtPersistence;
```

Opaque interface handle.

struct BtPersistenceInterface

```
struct BtPersistenceInterface {
  /* virtual methods */
  xmlNodePtr (*save)(const BtPersistence * const self, xmlNodePtr const node);
  BtPersistence* (*load)(const GType type, const BtPersistence * const self, xmlNodePtr ↩
      node, GError **err, va_list var_args);
};
```

BtPersistence interface

Members

`save` ()	virtual method to serialize an object to an xml node
`load` ()	virtual method to de-serialze an object from an xml node

Chapter 3

Song Class Reference

3.1 BtCmdPattern

BtCmdPattern — class for an command pattern of a BtMachine instance

Functions

BtCmdPattern *	bt_cmd_pattern_new ()

Properties

BtPatternCmd	command	Read / Write /
BtMachine *	machine	Read / Write /
gchar *	name	Read / Write /
BtSong *	song	Read / Write /

Types and Values

struct	BtCmdPattern
enum	BtPatternCmd

Object Hierarchy

```
    GEnum
    &#x2570;&#x2500;&#x2500; BtPatternCmd
    GObject
    &#x2570;&#x2500;&#x2500; BtCmdPattern
        &#x2570;&#x2500;&#x2500; BtPattern
```

Includes

```
#include <libbtcore/core.h>
```

Description

A command pattern is used in the BtSequence to trigger certain actions. The actions are defined as the BtPatternCmd enum.

Functions

bt_cmd_pattern_new ()

```
BtCmdPattern~*
bt_cmd_pattern_new (const BtSong * const song,
                    const BtMachine * const machine,
                    const BtPatternCmd cmd);
```

Create a new default pattern instance containg the given *cmd* event. It will be automatically added to the machines pattern list. If *cmd* is BT_PATTERN_CMD_NORMAL use bt_pattern_new() instead.

Don't call this from applications.

Parameters

song	the song the new instance belongs to	
machine	the machine the pattern belongs to	
cmd	a BtPatternCmd	

Returns

the new instance or NULL in case of an error

Types and Values

struct BtCmdPattern

```
struct BtCmdPattern;
```

Holds a sequence of events for a BtMachine.

enum BtPatternCmd

The commands are only used in static internal patterns. Regular patterns use *BT_PATTERN_CMD_NORMAL* .

Members

BT_PATTERN_CMD_NORMAL	no command
BT_PATTERN_CMD_MUTE	be quiet immediately

BT_PATTERN_CMD_SOLO	be the only one play-ing
BT_PATTERN_CMD_BYPASS	be un-ef-fec-tive (pass through)
BT_PATTERN_CMD_BREAK	no more notes

Property Details

The "command" property

```
"command"                  BtPatternCmd
```

the command of this pattern.

Flags: Read / Write / Construct Only

Default value: BT_PATTERN_CMD_NORMAL

The "machine" property

```
"machine"                  BtMachine~*
```

Machine object, the pattern belongs to.

Flags: Read / Write / Construct Only

The "name" property

```
"name"                     gchar~*
```

the display-name of the pattern.

Flags: Read / Write / Construct

Default value: "unnamed"

The "song" property

```
"song"                     BtSong~*
```

Song object, the pattern belongs to.

Flags: Read / Write / Construct Only

3.2 BtCmdPatternControlSource

BtCmdPatternControlSource — Custom controlsource based on repeated event blocks (BtCmdPatterns).

Functions

BtCmdPatternControlSource *	bt_cmd_pattern_control_source_new ()

Properties

gpointer	default-value	Write
BtMachine *	machine	Read / Write /
BtSequence *	sequence	Read / Write /
BtSongInfo *	song-info	Read / Write /

Types and Values

struct	BtCmdPatternControlSource

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstControlBinding
                &#x2570;&#x2500;&#x2500; BtCmdPatternControlSource
```

Includes

```
#include <libbtcore/core.h>
```

Description

The control source will update machine parameters over time, based on the events from the sequences and the patterns. One control-source will handle one single parameter. It implements the logic of computing the parameter value for a given time, taking multiple tracks and overlapping patterns into account.

At the begin of the timeline (ts==0) all parameters that have no value in the sequence will be initialized from “default-value”. For trigger parameter this usualy is the no-value. For other parameters it is the last value one has set in the ui or via interaction controller.

Functions

bt_cmd_pattern_control_source_new ()

```
BtCmdPatternControlSource~*
bt_cmd_pattern_control_source_new (GstObject *object,
                                   const gchar *property_name,
                                   BtSequence *sequence,
                                   const BtSongInfo *song_info,
                                   const BtMachine *machine);
```

Create a cmd-pattern control source for the given `machine` . Use gst_control_source_bind() to attach it to the related parameter of the machine.

Parameters

object	the object of the property	
property_name	the property-name to attach the control source	
sequence	the songs sequence	
song_info	the song info	
machine	the machine	

Returns

the new cmd-pattern control source

Types and Values

struct BtCmdPatternControlSource

```
struct BtCmdPatternControlSource;
```

A pattern based control source

Property Details

The “default-value” property

```
  “default-value”              gpointer
```

pointer to value to use if no other found.

Flags: Write

The “machine” property

```
  “machine”                    BtMachine~*
```

the machine object, the controlsource belongs to.

Flags: Read / Write / Construct Only

The “sequence” property

```
  “sequence”                   BtSequence~*
```

the sequence object.

Flags: Read / Write / Construct Only

The “song-info” property

```
“song-info”                BtSongInfo~*
```

the song-info object.

Flags: Read / Write / Construct Only

3.3 BtMachine

BtMachine — base class for signal processing machines

Functions

gboolean	bt_machine_activate_adder ()
gboolean	bt_machine_activate_spreader ()
void	bt_machine_add_pattern ()
void	bt_machine_bind_parameter_control ()
void	bt_machine_bind_poly_parameter_control ()
gboolean	bt_machine_enable_input_gain ()
gboolean	bt_machine_enable_input_post_level ()
gboolean	bt_machine_enable_input_pre_level ()
gboolean	bt_machine_enable_output_gain ()
gboolean	bt_machine_enable_output_post_level ()
gboolean	bt_machine_enable_output_pre_level ()
BtParameterGroup *	bt_machine_get_global_param_group ()
BtCmdPattern *	bt_machine_get_pattern_by_index ()
BtCmdPattern *	bt_machine_get_pattern_by_name ()
BtParameterGroup *	bt_machine_get_prefs_param_group ()
gchar *	bt_machine_get_unique_pattern_name ()
BtParameterGroup *	bt_machine_get_voice_param_group ()
BtWire *	bt_machine_get_wire_by_dst_machine ()
gboolean	bt_machine_handles_waves ()
gboolean	bt_machine_has_active_adder ()
gboolean	bt_machine_has_active_spreader ()
gboolean	bt_machine_has_patterns ()
gboolean	bt_machine_is_polyphonic ()
void	bt_machine_randomize_parameters ()
void	bt_machine_remove_pattern ()
void	bt_machine_reset_parameters ()
void	bt_machine_set_param_defaults ()
void	bt_machine_unbind_parameter_control ()
void	bt_machine_unbind_parameter_controls ()

Properties

GstElement *	adder-convert	Read
gpointer	construction-error	Read / Write /
gulong	global-params	Read
gchar *	id	Read / Write /
GstElement *	input-gain	Read
GstElement *	input-post-level	Read
GstElement *	input-pre-level	Read

GstElement *	machine	Read
GstElement *	output-gain	Read
GstElement *	output-post-level	Read
GstElement *	output-pre-level	Read
gpointer	patterns	Read
gchar *	plugin-name	Read / Write /
gulong	prefs-params	Read
gchar *	pretty-name	Read
gpointer	properties	Read
BtSong *	song	Read / Write /
BtMachineState	state	Read / Write
gulong	voice-params	Read
gulong	voices	Read / Write /

Signals

void	pattern-added	No Hooks
void	pattern-removed	No Hooks

Types and Values

struct	BtMachine
struct	BtMachineClass
enum	BtMachineState

Object Hierarchy

```
    GEnum
    &#x2570;&#x2500;&#x2500; BtMachineState
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstElement
                &#x2570;&#x2500;&#x2500; GstBin
                    &#x2570;&#x2500;&#x2500; BtMachine
                        &#x251c;&#x2500;&#x2500; BtProcessorMachine
                        &#x251c;&#x2500;&#x2500; BtSinkMachine
                        &#x2570;&#x2500;&#x2500; BtSourceMachine
```

Implemented Interfaces

BtMachine implements GstChildProxy and BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Machines are pieces in a BtSong that generate, process or play media.

The machine class takes care of inserting additional low-level elemnts to do signal conversion etc.. Further it provides general facillities like input/output level monitoring. The resulting machine instance is a box containing several processing elements.

A machine can have several GstElements:

adder: mixes all incoming signals

input volume: gain for incoming signals

input pre/post-gain level: level meter for incoming signal

machine: the real machine

output volume: gain for outgoing signal

output pre/post-gain level: level meter for outgoing signal

spreader: distibutes signal to outgoing connections

The adder and spreader elements are activated depending on element type. The volume controls and level meters are activated as requested via the API. It is recommended to only activate them, when needed. The instances are cached after deactivation (so that they can be easily reactivated) and destroyed with the BtMachine object.

Furthermore the machine handles a list of BtCmdPattern instances. These contain event patterns that form a BtSequence.

Functions

bt_machine_activate_adder ()

```
gboolean
bt_machine_activate_adder (BtMachine * const self);
```

Machines use an adder to allow multiple incoming wires. This method is used by the BtWire class to activate the adder when needed.

Parameters

self	the machine to activate the adder in

Returns

TRUE for success

bt_machine_activate_spreader ()

```
gboolean
bt_machine_activate_spreader (BtMachine * const self);
```

Machines use a spreader to allow multiple outgoing wires. This method is used by the BtWire class to activate the spreader when needed.

Parameters

self	the machine to activate the spreader in

Returns

TRUE for success

bt_machine_add_pattern ()

```
void
bt_machine_add_pattern (const BtMachine *self,
                        const BtCmdPattern *pattern);
```

Add the supplied pattern to the machine. This is automatically done by bt_pattern_new().

Parameters

self	the machine to add the pattern to
pattern	the new pattern instance

bt_machine_bind_parameter_control ()

```
void
bt_machine_bind_parameter_control (const BtMachine * const self,
                                   GstObject *object,
                                   const gchar *property_name,
                                   BtIcControl *control,
                                   BtParameterGroup *pg);
```

Connect the interaction control object to the give parameter. Changes of the control-value are mapped into a change of the parameter.

Parameters

self	machine
object	child object (global or voice child)
property_name	name of the parameter
control	interaction control object
pg	the parameter group of the property

bt_machine_bind_poly_parameter_control ()

```
void
bt_machine_bind_poly_parameter_control
                               (const BtMachine * const self,
                                const gchar *property_name,
                                BtIcControl *control,
                                BtParameterGroup *pg);
```

Connect the interaction control object to the give parameter. Changes of the control-value are mapped into a change of the parameter.

Parameters

self	machine	
property_name	name of the parameter	
control	interaction control object	
pg	the parameter group of the property	

bt_machine_enable_input_gain ()

```
gboolean
bt_machine_enable_input_gain (BtMachine * const self);
```

Creates the input-gain element of the machine and activates it.

Parameters

self	the machine to enable the input-gain element in	

Returns

TRUE for success, FALSE otherwise

bt_machine_enable_input_post_level ()

```
gboolean
bt_machine_enable_input_post_level (BtMachine * const self);
```

Creates the post-gain input-level analyser of the machine and activates it.

Parameters

self	the machine to enable the post-gain input-level analyser in	

Returns

TRUE for success, FALSE otherwise

bt_machine_enable_input_pre_level ()

```
gboolean
bt_machine_enable_input_pre_level (BtMachine * const self);
```

Creates the pre-gain input-level analyser of the machine and activates it.

Parameters

self	the machine to enable the pre-gain input-level analyser in

Returns

TRUE for success, FALSE otherwise

bt_machine_enable_output_gain ()

```
gboolean
bt_machine_enable_output_gain (BtMachine * const self);
```

Creates the output-gain element of the machine and activates it.

Parameters

self	the machine to enable the output-gain element in

Returns

TRUE for success, FALSE otherwise

bt_machine_enable_output_post_level ()

```
gboolean
bt_machine_enable_output_post_level (BtMachine * const self);
```

Creates the post-gain output-level analyser of the machine and activates it.

Parameters

self	the machine to enable the post-gain output-level analyser in

Returns

TRUE for success, FALSE otherwise

bt_machine_enable_output_pre_level ()

```
gboolean
bt_machine_enable_output_pre_level (BtMachine * const self);
```

Creates the pre-gain output-level analyser of the machine and activates it.

Parameters

self	the machine to enable the pre-gain output-level analyser in

Returns

TRUE for success, FALSE otherwise

bt_machine_get_global_param_group ()

```
BtParameterGroup~*
bt_machine_get_global_param_group (const BtMachine * const self);
```

Get the parameter group of global parameters.

Parameters

self	the machine

Returns

the BtParameterGroup or NULL.

[transfer none]

bt_machine_get_pattern_by_index ()

```
BtCmdPattern~*
bt_machine_get_pattern_by_index (const BtMachine * const self,
                                 const gulong index);
```

Fetches the pattern from the given position of the machines pattern list. The pattern must have been added previously to this setup with bt_machine_add_pattern().

Parameters

self	the machine to search for the pattern
index	the index of the pattern in the machines pattern list

Returns

BtCmdPattern instance or NULL if not found. Unref the pattern, when done with it.

[transfer full]

bt_machine_get_pattern_by_name ()

```
BtCmdPattern~*
bt_machine_get_pattern_by_name (const BtMachine * const self,
                                const gchar * const name);
```

Search the machine for a pattern by the supplied name. The pattern must have been added previously to this setup with bt_machine_add_pattern().

Parameters

self	the machine to search for the pattern
name	the name of the pattern

Returns

BtCmdPattern instance or NULL if not found. Unref the pattern, when done with it.

[transfer full]

bt_machine_get_prefs_param_group ()

```
BtParameterGroup~*
bt_machine_get_prefs_param_group (const BtMachine * const self);
```

Get the parameter group of machine properties. Properties are settings that cannot be changed during playback.

Parameters

self	the machine

Returns

the BtParameterGroup or NULL.

[transfer none]

bt_machine_get_unique_pattern_name ()

```
gchar~*
bt_machine_get_unique_pattern_name (const BtMachine * const self);
```

The function generates a unique pattern name for this machine by eventually adding a number postfix. This method should be used when adding new patterns.

Parameters

self	the machine for which the name should be unique

Returns

the newly allocated unique name.

[transfer full]

bt_machine_get_voice_param_group ()

```
BtParameterGroup~*
bt_machine_get_voice_param_group (const BtMachine * const self,
                                  const gulong voice);
```

Get the parameter group of voice parameters for the given `voice` .

Parameters

self	the machine
voice	the voice number

Returns

the BtParameterGroup or NULL.

[transfer none]

bt_machine_get_wire_by_dst_machine ()

```
BtWire~*
bt_machine_get_wire_by_dst_machine (const BtMachine * const self,
                                    const BtMachine * const dst);
```

Searches for a wire in the wires originating from this machine that uses the given BtMachine instances as a target.

Parameters

self	the machine that is at src of a wire
dst	the machine that is at the dst end of the wire

Returns

the BtWire or NULL. Unref the wire, when done with it.

[transfer full]

Since: 0.6

bt_machine_handles_waves ()

```
gboolean
bt_machine_handles_waves (const BtMachine * const self);
```

Tells if the machine is using the wave-table.

Parameters

self	the machine to check

Returns

TRUE for wavetable machines, FALSE otherwise

Since: 0.7

bt_machine_has_active_adder ()

```
gboolean
bt_machine_has_active_adder (const BtMachine * const self);
```

Checks if the machine currently uses an adder. This method is used by the BtWire class to activate the adder when needed.

Parameters

self	the machine to check

Returns

TRUE for success

bt_machine_has_active_spreader ()

```
gboolean
bt_machine_has_active_spreader (const BtMachine * const self);
```

Checks if the machine currently uses an spreader. This method is used by the BtWire class to activate the spreader when needed.

Parameters

self	the machine to check

Returns

TRUE for success

bt_machine_has_patterns ()

```
gboolean
bt_machine_has_patterns (const BtMachine * const self);
```

Check if the machine has BtPattern entries appart from the standart private ones.

Parameters

self	the machine for which to check the patterns

Returns

TRUE if it has patterns

bt_machine_is_polyphonic ()

```
gboolean
bt_machine_is_polyphonic (const BtMachine * const self);
```

Tells if the machine can produce (multiple) voices. Monophonic machines have their (one) voice params as part of the global params.

Parameters

self	the machine to check

Returns

TRUE for polyphic machines, FALSE for monophonic ones

bt_machine_randomize_parameters ()

```
void
bt_machine_randomize_parameters (const BtMachine * const self);
```

Randomizes machine parameters.

Parameters

self	machine

bt_machine_remove_pattern ()

```
void
bt_machine_remove_pattern (const BtMachine *self,
                           const BtCmdPattern *pattern);
```

Remove the given pattern from the machine.

Parameters

self	the machine to remove the pattern from
pattern	the existing pattern instance

bt_machine_reset_parameters ()

```
void
bt_machine_reset_parameters (const BtMachine * const self);
```

Resets machine parameters back to defaults.

Parameters

self	machine

bt_machine_set_param_defaults ()

```
void
bt_machine_set_param_defaults (const BtMachine *const self);
```

Sets default values that should be used before the first control-point. Should be called, if all parameters are changed (like after switching presets).

Parameters

self	the machine

bt_machine_unbind_parameter_control ()

```
void
bt_machine_unbind_parameter_control (const BtMachine * const self,
                                     GstObject *object,
                                     const gchar *property_name);
```

Disconnect the interaction control object from the give parameter.

Parameters

self	machine
object	child object (global or voice child)
property_name	name of the parameter

bt_machine_unbind_parameter_controls ()

```
void
bt_machine_unbind_parameter_controls (const BtMachine * const self);
```

Disconnect all interaction controls.

Parameters

self	machine

Types and Values

struct BtMachine

```
struct BtMachine {
  /*< read-only >*/
  GList *src_wires;
  GList *dst_wires;
};
```

Base object for a virtual piece of hardware (generator, effect, ...).

Members

GList **src_wires*;	read-only list of outgoing BtWire objects
GList **dst_wires*;	read-only list of incoming BtWire objects

struct BtMachineClass

```
struct BtMachineClass {
  /* virtual methods for subclasses */
  gboolean (*check_type)(const BtMachine * const machine, const gulong pad_src_ct, const ↩
      gulong pad_sink_ct);
};
```

Base class for machines.

Members

check_type ()	sanity check that the given input/output characteristics are okay for the implementation

enum BtMachineState

A machine is always in one of the 4 states. Use the "state" property of the BtMachine to change or query the current state.

Members

BT_MACHINE_STATE_NORMAL	just working
BT_MACHINE_STATE_MUTE	be quiet
BT_MACHINE_STATE_SOLO	be the only one playing
BT_MACHINE_STATE_BYPASS	be uneffective (pass through)

Property Details

The “adder-convert” property

```
  “adder-convert”              GstElement~*
```

the after mixing format converter element, if any.

Flags: Read

The “construction-error” property

```
“construction-error”       gpointer
```

signal failed instance creation.

Flags: Read / Write / Construct Only

The “global-params” property

```
“global-params”            gulong
```

number of dynamic params for the machine.

Flags: Read

The “id” property

```
“id”                       gchar~*
```

machine identifier.

Flags: Read / Write / Construct

Default value: "unamed machine"

The “input-gain” property

```
“input-gain”               GstElement~*
```

the input-gain element, if any.

Flags: Read

The “input-post-level” property

```
“input-post-level”         GstElement~*
```

the post-gain input-level element, if any.

Flags: Read

The “input-pre-level” property

```
“input-pre-level”          GstElement~*
```

the pre-gain input-level element, if any.

Flags: Read

The “machine” property

```
“machine”                  GstElement~*
```

the machine element, if any.

Flags: Read

The “output-gain” property

```
“output-gain”              GstElement~*
```

the output-gain element, if any.

Flags: Read

The “output-post-level” property

```
“output-post-level”        GstElement~*
```

the post-gain output-level element, if any.

Flags: Read

The “output-pre-level” property

```
“output-pre-level”         GstElement~*
```

the pre-gain output-level element, if any.

Flags: Read

The “patterns” property

```
“patterns”                 gpointer
```

a copy of the list of patterns.

Flags: Read

The “plugin-name” property

```
“plugin-name”              gchar~*
```

the name of the gst plugin for the machine.

Flags: Read / Write / Construct

Default value: "unamed plugin"

The “prefs-params” property

```
“prefs-params”             gulong
```

number of static params for the machine.

Flags: Read

The “pretty-name” property

```
“pretty-name”              gchar~*
```

pretty-printed name for display purposes.

Flags: Read

Default value: NULL

The “properties” property

```
“properties”               gpointer
```

hashtable of machine properties.

Flags: Read

The “song” property

```
“song”                     BtSong~*
```

song object, the machine belongs to.

Flags: Read / Write / Construct Only

The “state” property

```
“state”                    BtMachineState
```

the current state of this machine.

Flags: Read / Write

Default value: BT_MACHINE_STATE_NORMAL

The “voice-params” property

```
“voice-params”             gulong
```

number of dynamic params for each machine voice.

Flags: Read

The “voices” property

```
“voices”                   gulong
```

number of voices in the machine.

Flags: Read / Write / Construct

Signal Details

The "pattern-added" signal

```
void
user_function (BtMachine *self,
               BtPattern *pattern,
               gpointer   user_data)
```

A new pattern item has been added to the machine

Parameters

self	the machine object that emitted the signal	
pattern	the new pattern	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The "pattern-removed" signal

```
void
user_function (BtMachine *self,
               BtPattern *pattern,
               gpointer   user_data)
```

A pattern item has been removed from the machine

Parameters

self	the machine object that emitted the signal	
pattern	the old pattern	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

3.4 BtParameterGroup

BtParameterGroup — a group of parameter

Functions

gchar *	bt_parameter_group_describe_param_value ()
void	bt_parameter_group_get_param_details ()

glong	bt_parameter_group_get_param_index ()
const gchar *	bt_parameter_group_get_param_name ()
GValue *	bt_parameter_group_get_param_no_value ()
GObject *	bt_parameter_group_get_param_parent ()
GParamSpec *	bt_parameter_group_get_param_spec ()
GType	bt_parameter_group_get_param_type ()
glong	bt_parameter_group_get_trigger_param_index ()
glong	bt_parameter_group_get_wave_param_index ()
gboolean	bt_parameter_group_is_param_no_value ()
gboolean	bt_parameter_group_is_param_trigger ()
BtParameterGroup *	bt_parameter_group_new ()
void	bt_parameter_group_randomize_values ()
void	bt_parameter_group_reset_values ()
void	bt_parameter_group_set_param_default ()
void	bt_parameter_group_set_param_defaults ()
void	bt_parameter_group_set_param_value ()

Properties

BtMachine *	machine	Read / Write /
gulong	num-params	Read / Write /
gpointer	params	Read / Write /
gpointer	parents	Read / Write /
BtSong *	song	Read / Write /

Types and Values

struct	BtParameterGroup

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtParameterGroup
```

Includes

```
#include <libbtcore/core.h>
```

Description

A group of parameters, such as used in machines or wires. Once created the group will not change.

Functions

bt_parameter_group_describe_param_value ()

```
gchar~*
bt_parameter_group_describe_param_value
                                (const BtParameterGroup * const self,
                                 const gulong index,
                                 GValue * const event);
```

Described a param value in human readable form. The type of the given `value` must match the type of the paramspec of the param referenced by `index` .

Parameters

self	the parameter group to get a param description from	
index	the offset in the list of params	
event	the value to describe	

Returns

the description as newly allocated string

bt_parameter_group_get_param_details ()

```
void
bt_parameter_group_get_param_details (const BtParameterGroup * const self,
                                      const gulong index,
                                      GParamSpec **pspec,
                                      GValue **min_val,
                                      GValue **max_val);
```

Retrieves the details of a param. Any detail can be NULL if its not wanted.

Parameters

self	the parameter group to search for the param details	
index	the offset in the list of params	
pspec	place for the param spec.	*[out]*
min_val	place to hold new GValue with minimum.	*[out]*
max_val	place to hold new GValue with maximum.	*[out]*

bt_parameter_group_get_param_index ()

```
glong
bt_parameter_group_get_param_index (const BtParameterGroup * const self,
                                    const gchar * const name);
```

Searches the list of registered param of a machine for a param of the given name and returns the index if found.

Parameters

self	the parameter group to search for the param	
name	the name of the param	

Returns

the index if found or returns -1.

bt_parameter_group_get_param_name ()

```
const gchar~*
bt_parameter_group_get_param_name (const BtParameterGroup * const self,
                                   const gulong index);
```

Gets the param name. Do not modify returned content.

Parameters

self	the parameter group to get the param name from
index	the offset in the list of params

Returns

the requested name

bt_parameter_group_get_param_no_value ()

```
GValue~*
bt_parameter_group_get_param_no_value (const BtParameterGroup * const self,
                                       const gulong index);
```

Get the neutral value for the machines parameter.

Parameters

self	the parameter group to get params from
index	the offset in the list of params

Returns

the value. Don't modify.

Since: 0.6

bt_parameter_group_get_param_parent ()

```
GObject~*
bt_parameter_group_get_param_parent (const BtParameterGroup * const self,
                                     const gulong index);
```

Retrieves the owner instance for the param

Parameters

self	the parameter group to search for the param	
index	the offset in the list of params	

Returns

the GParamSpec for the requested param.

[transfer none]

bt_parameter_group_get_param_spec ()

```
GParamSpec~*
bt_parameter_group_get_param_spec (const BtParameterGroup * const self,
                                   const gulong index);
```

Retrieves the parameter specification for the param

Parameters

self	the parameter group to search for the param	
index	the offset in the list of params	

Returns

the GParamSpec for the requested param.

[transfer none]

bt_parameter_group_get_param_type ()

```
GType
bt_parameter_group_get_param_type (const BtParameterGroup * const self,
                                   const gulong index);
```

Retrieves the GType of a param

Parameters

self	the parameter group to search for the param type	
index	the offset in the list of params	

Returns

the requested GType

bt_parameter_group_get_trigger_param_index ()

```
glong
bt_parameter_group_get_trigger_param_index
                                        (const BtParameterGroup * const self);
```

Searches for the first trigger parameter (if any).

Parameters

self	the parameter group to lookup the param from

Returns

the index of the first trigger parameter or -1 if none.

bt_parameter_group_get_wave_param_index ()

```
glong
bt_parameter_group_get_wave_param_index
                                        (const BtParameterGroup * const self);
```

Searches for the wave-table index parameter (if any). This parameter should refer to a wavetable index that should be used to play a note.

Parameters

self	the parameter group to lookup the param from

Returns

the index of the wave-table parameter or -1 if none.

bt_parameter_group_is_param_no_value ()

```
gboolean
bt_parameter_group_is_param_no_value (const BtParameterGroup * const self,
                                      const gulong index,
                                      GValue * const value);
```

Tests if the given value is the no-value of the param

Parameters

self	the parameter group to check params from
index	the offset in the list of params
value	the value to compare against the no-value

Returns

TRUE if it is the no-value

bt_parameter_group_is_param_trigger ()

```
gboolean
bt_parameter_group_is_param_trigger (const BtParameterGroup * const self,
                                     const gulong index);
```

Tests if the param is a trigger param (like a key-note or a drum trigger).

Parameters

self	the parameter group to check params from	
index	the offset in the list of params	

Returns

TRUE if it is a trigger

bt_parameter_group_new ()

```
BtParameterGroup~*
bt_parameter_group_new (gulong num_params,
                        GObject **parents,
                        GParamSpec **params,
                        BtSong *song,
                        const BtMachine *machine);
```

Create a parameter group.

Parameters

num_params	the number of parameters	
parents	array of parent GObjects for each parameter	
params	array of GParamSpecs for each parameter	
song	the song	
machine	the machine that is owns the parameter-group, use the target machine for wires.	

Returns

the new parameter group.

[transfer full]

bt_parameter_group_randomize_values ()

```
void
bt_parameter_group_randomize_values (const BtParameterGroup * const self);
```

Randomize all parameter values.

Parameters

self	the parameter group

bt_parameter_group_reset_values ()

```
void
bt_parameter_group_reset_values (const BtParameterGroup * const self);
```

Reset all parameter values to their defaults.

Parameters

self	the parameter group

bt_parameter_group_set_param_default ()

```
void
bt_parameter_group_set_param_default (const BtParameterGroup * const self,
                                      const gulong index);
```

Set a default value that should be used before the first control-point.

Parameters

self	the parameter group
index	the offset in the list of params

bt_parameter_group_set_param_defaults ()

```
void
bt_parameter_group_set_param_defaults (const BtParameterGroup * const self);
```

Set a default value that should be used before the first control-point for each parameter.

Parameters

self	the parameter group

bt_parameter_group_set_param_value ()

```
void
bt_parameter_group_set_param_value (const BtParameterGroup * const self,
                                    const gulong index,
                                        GValue * const event);
```

Sets a the specified param to the give data value.

Parameters

self	the parameter group to set the param value	
index	the offset in the list of params	
event	the new value	

Types and Values

struct BtParameterGroup

```
struct BtParameterGroup;
```

A group of parameters, such as used in machines or wires.

Property Details

The "machine" property

```
  "machine"                    BtMachine~*
```

the respective machine object.

Flags: Read / Write / Construct Only

The "num-params" property

```
  "num-params"                 gulong
```

number of params.

Flags: Read / Write / Construct Only

The "params" property

```
  "params"                     gpointer
```

pointer to GParamSpec array, takes ownership.

Flags: Read / Write / Construct Only

The "parents" property

```
  "parents"                    gpointer
```

pointer to array containing the Objects that own the paramers, takes ownership.

Flags: Read / Write / Construct Only

The “song” property

```
“song”                    BtSong~*
```

song object the param group belongs to.

Flags: Read / Write / Construct Only

3.5 BtPattern

BtPattern — class for an event pattern of a BtMachine instance

Functions

void	bt_pattern_blend_columns ()
void	bt_pattern_clear_columns ()
BtPattern *	bt_pattern_copy ()
void	bt_pattern_delete_row ()
void	bt_pattern_flip_columns ()
gchar *	bt_pattern_get_global_event ()
GValue *	bt_pattern_get_global_event_data ()
BtValueGroup *	bt_pattern_get_global_group ()
BtValueGroup *	bt_pattern_get_group_by_parameter_group ()
gchar *	bt_pattern_get_voice_event ()
GValue *	bt_pattern_get_voice_event_data ()
BtValueGroup *	bt_pattern_get_voice_group ()
gchar *	bt_pattern_get_wire_event ()
GValue *	bt_pattern_get_wire_event_data ()
BtValueGroup *	bt_pattern_get_wire_group ()
void	bt_pattern_insert_row ()
BtPattern *	bt_pattern_new ()
void	bt_pattern_randomize_columns ()
void	bt_pattern_range_randomize_columns ()
void	bt_pattern_transpose_coarse_down_columns ()
void	bt_pattern_transpose_coarse_up_columns ()
void	bt_pattern_transpose_fine_down_columns ()
void	bt_pattern_transpose_fine_up_columns ()
void	bt_pattern_serialize_columns ()
gboolean	bt_pattern_set_global_event ()
gboolean	bt_pattern_set_voice_event ()
gboolean	bt_pattern_set_wire_event ()
gboolean	bt_pattern_test_global_event ()
gboolean	bt_pattern_test_tick ()
gboolean	bt_pattern_test_voice_event ()
gboolean	bt_pattern_test_wire_event ()

Properties

BtPattern *	copy-source	Write / Constr
gulong	length	Read / Write /
gulong	voices	Read

Signals

void	group-changed	No Hooks
void	param-changed	No Hooks
void	pattern-changed	No Hooks

Types and Values

struct	BtPattern

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtCmdPattern
        &#x2570;&#x2500;&#x2500; BtPattern
```

Implemented Interfaces

BtPattern implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

A pattern contains a grid of events. Events are parameter changes in BtMachine objects. The events are stored as GValues. Cells containing NULL have no event for the parameter at the time.

Patterns can have individual lengths. The length is measured in ticks. How much that is in e.g. milliseconds is determined by “bpm” and “tpb”.

A pattern might consist of several groups. These are mapped to the global parameters of a machine and the voice parameters for each machine voice (if any). The number of voices (tracks) is the same in all patterns of a machine. If the voices are changed on the machine patterns resize themselves.

The patterns are used in the BtSequence to form the score of a song.

Functions

bt_pattern_blend_columns ()

```
void
bt_pattern_blend_columns (const BtPattern * const self,
                          const gulong start_tick,
                          const gulong end_tick);
```

Fade values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.3

bt_pattern_clear_columns ()

```
void
bt_pattern_clear_columns (const BtPattern * const self,
                          const gulong start_tick,
                          const gulong end_tick);
```

Clear values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.6

bt_pattern_copy ()

```
BtPattern~*
bt_pattern_copy (const BtPattern * const self);
```

Create a new instance as a copy of the given instance.

Parameters

self	the pattern to create a copy from	

Returns

the new instance or NULL in case of an error.

[transfer full]

bt_pattern_delete_row ()

```
void
bt_pattern_delete_row (const BtPattern * const self,
                       const gulong tick);
```

Delete row for all parameters.

Parameters

self	the pattern	
tick	the position to delete	

Since: 0.3

bt_pattern_flip_columns ()

```
void
bt_pattern_flip_columns (const BtPattern * const self,
                         const gulong start_tick,
                         const gulong end_tick);
```

Flips values from *start_tick* to *end_tick* for all params up-side down.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.5

bt_pattern_get_global_event ()

```
gchar~*
bt_pattern_get_global_event (const BtPattern * const self,
                             const gulong tick,
                             const gulong param);
```

Returns the string representation of the specified cell. Free it when done.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
param	the number of the global parameter starting with 0	

Returns

a newly allocated string with the data or NULL on error

bt_pattern_get_global_event_data ()

```
GValue~*
```

```
bt_pattern_get_global_event_data (const BtPattern * const self,
                                  const gulong tick,
                                  const gulong param);
```

Fetches a cell from the given location in the pattern. If there is no event there, then the GValue is uninitialized. Test with BT_IS_GVALUE(event).

Do not modify the contents!

Parameters

self	the pattern to search for the global param	
tick	the tick (time) position starting with 0	
param	the number of the global parameter starting with 0	

Returns

the GValue or NULL if out of the pattern range

bt_pattern_get_global_group ()

```
BtValueGroup~*
bt_pattern_get_global_group (const BtPattern * const self);
```

Get the BtValueGroup for global parameters.

Parameters

self	the pattern	

Returns

the group owned by the pattern.

[transfer none]

bt_pattern_get_group_by_parameter_group ()

```
BtValueGroup~*
bt_pattern_get_group_by_parameter_group
                                (const BtPattern * const self,
                                 BtParameterGroup *param_group);
```

Get the BtValueGroup for the given `param_group` .

Parameters

self	the pattern	
param_group	the BtParameterGroup to get the group for	

Returns

the group owned by the pattern.

[transfer none]

bt_pattern_get_voice_event ()

```
gchar~*
bt_pattern_get_voice_event (const BtPattern * const self,
                            const gulong tick,
                            const gulong voice,
                            const gulong param);
```

Returns the string representation of the specified cell. Free it when done.

Parameters

self	the pattern the cell belongs to
tick	the tick (time) position starting with 0
voice	the voice number starting with 0
param	the number of the voice parameter starting with 0

Returns

a newly allocated string with the data or NULL on error

bt_pattern_get_voice_event_data ()

```
GValue~*
bt_pattern_get_voice_event_data (const BtPattern * const self,
                                 const gulong tick,
                                 const gulong voice,
                                 const gulong param);
```

Fetches a cell from the given location in the pattern. If there is no event there, then the GValue is uninitialized. Test with BT_IS_GVALUE(event).

Do not modify the contents!

Parameters

self	the pattern to search for the voice param
tick	the tick (time) position starting with 0
voice	the voice number starting with 0
param	the number of the voice parameter starting with 0

Returns

the GValue or NULL if out of the pattern range

bt_pattern_get_voice_group ()

```
BtValueGroup~*
bt_pattern_get_voice_group (const BtPattern * const self,
                            const gulong voice);
```

Get the BtValueGroup for voice parameters.

Parameters

self	the pattern	
voice	the voice to get the group for	

Returns

the group owned by the pattern.

[transfer none]

bt_pattern_get_wire_event ()

```
gchar~*
bt_pattern_get_wire_event (const BtPattern * const self,
                           const gulong tick,
                           const BtWire *wire,
                           const gulong param);
```

Returns the string representation of the specified cell. Free it when done.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
wire	the related wire object	
param	the number of the wire parameter starting with 0	

Returns

a newly allocated string with the data or NULL on error

bt_pattern_get_wire_event_data ()

```
GValue~*
bt_pattern_get_wire_event_data (const BtPattern * const self,
                                const gulong tick,
                                const BtWire *wire,
                                const gulong param);
```

Fetches a cell from the given location in the pattern. If there is no event there, then the GValue is uninitialized. Test with BT_IS_GVALUE(event).

Do not modify the contents!

Parameters

self	the pattern to search for the wire param	
tick	the tick (time) position starting with 0	
wire	the related wire object	
param	the number of the wire parameter starting with 0	

Returns

the GValue or NULL if out of the pattern range

bt_pattern_get_wire_group ()

```
BtValueGroup~*
bt_pattern_get_wire_group (const BtPattern * const self,
                           const BtWire *wire);
```

Get the BtValueGroup for wire parameters.

Parameters

self	the pattern	
wire	the BtWire to get the group for	

Returns

the group owned by the pattern.

[transfer none]

bt_pattern_insert_row ()

```
void
bt_pattern_insert_row (const BtPattern * const self,
                       const gulong tick);
```

Insert one empty row for all parameters.

Parameters

self	the pattern	
tick	the position to insert at	

Since: 0.3

bt_pattern_new ()

```
BtPattern~*
bt_pattern_new (const BtSong * const song,
                const gchar * const name,
                const gulong length,
                const BtMachine * const machine);
```

Create a new instance. It will be automatically added to the machines pattern list.

Parameters

song	the song the new instance belongs to	
name	the display name of the pattern	
length	the number of ticks	
machine	the machine the pattern belongs to	

Returns

the new instance or NULL in case of an error

bt_pattern_randomize_columns ()

```
void
bt_pattern_randomize_columns (const BtPattern * const self,
                              const gulong start_tick,
                              const gulong end_tick);
```

Randomize values from *start_tick* to *end_tick* for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.3

bt_pattern_range_randomize_columns ()

```
void
bt_pattern_range_randomize_columns (const BtPattern * const self,
                                    const gulong start_tick,
                                    const gulong end_tick);
```

Randomize values from *start_tick* to *end_tick* for all params using the first and last value as bounds for the random values.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.7

bt_pattern_transpose_coarse_down_columns ()

```
void
bt_pattern_transpose_coarse_down_columns
                               (const BtPattern * const self,
                                const gulong start_tick,
                                const gulong end_tick);
```

Transposes values from *start_tick* to *end_tick* for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_pattern_transpose_coarse_up_columns ()

```
void
bt_pattern_transpose_coarse_up_columns
                               (const BtPattern * const self,
                                const gulong start_tick,
                                const gulong end_tick);
```

Transposes values from *start_tick* to *end_tick* for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_pattern_transpose_fine_down_columns ()

```
void
bt_pattern_transpose_fine_down_columns
                               (const BtPattern * const self,
                                const gulong start_tick,
                                const gulong end_tick);
```

Transposes values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_pattern_transpose_fine_up_columns ()

```
void
bt_pattern_transpose_fine_up_columns (const BtPattern * const self,
                                      const gulong start_tick,
                                      const gulong end_tick);
```

Transposes values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_pattern_serialize_columns ()

```
void
bt_pattern_serialize_columns (const BtPattern * const self,
                              const gulong start_tick,
                              const gulong end_tick,
                              GString *data);
```

Serializes values from `start_tick` to `end_tick` for all params into `data` .

Parameters

self	the pattern	

start_tick	the start position for the range	
end_tick	the end position for the range	
data	the target	

Since: 0.6

bt_pattern_set_global_event ()

```
gboolean
bt_pattern_set_global_event (const BtPattern * const self,
                             const gulong tick,
                             const gulong param,
                             const gchar * const value);
```

Stores the supplied value into the specified pattern cell.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
param	the number of the global parameter starting with 0	
value	the string representation of the value to store	

Returns

TRUE for success

bt_pattern_set_voice_event ()

```
gboolean
bt_pattern_set_voice_event (const BtPattern * const self,
                            const gulong tick,
                            const gulong voice,
                            const gulong param,
                            const gchar * const value);
```

Stores the supplied value into the specified pattern cell.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
voice	the voice number starting with 0	

param	the number of the voice parameter starting with 0
value	the string representation of the value to store

Returns

TRUE for success

bt_pattern_set_wire_event ()

```
gboolean
bt_pattern_set_wire_event (const BtPattern * const self,
                           const gulong tick,
                           const BtWire *wire,
                           const gulong param,
                           const gchar * const value);
```

Stores the supplied value into the specified pattern cell.

Parameters

self	the pattern the cell belongs to
tick	the tick (time) position starting with 0
wire	the related wire object
param	the number of the wire parameter starting with 0
value	the string representation of the value to store

Returns

TRUE for success

bt_pattern_test_global_event ()

```
gboolean
bt_pattern_test_global_event (const BtPattern * const self,
                              const gulong tick,
                              const gulong param);
```

Tests if there is an event in the specified cell.

Parameters

self	the pattern the cell belongs to
tick	the tick (time) position starting with 0
param	the number of the global parameter starting with 0

Returns

TRUE if there is an event

bt_pattern_test_tick ()

```
gboolean
bt_pattern_test_tick (const BtPattern * const self,
                      const gulong tick);
```

Check if there are any event in the given pattern-row.

Parameters

self	the pattern to check	
tick	the tick index in the pattern	

Returns

TRUE is there are events, FALSE otherwise

bt_pattern_test_voice_event ()

```
gboolean
bt_pattern_test_voice_event (const BtPattern * const self,
                             const gulong tick,
                             const gulong voice,
                             const gulong param);
```

Tests if there is an event in the specified cell.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
voice	the voice number starting with 0	
param	the number of the voice parameter starting with 0	

Returns

TRUE if there is an event

bt_pattern_test_wire_event ()

```
gboolean
bt_pattern_test_wire_event (const BtPattern * const self,
                            const gulong tick,
                            const BtWire *wire,
                            const gulong param);
```

Tests if there is an event in the specified cell.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
wire	the related wire object	
param	the number of the wire parameter starting with 0	

Returns

TRUE if there is an event

Types and Values

struct BtPattern

```
struct BtPattern;
```

Holds a sequence of events for a BtMachine.

Property Details

The “copy-source” property

```
  “copy-source”                BtPattern~*
```

pattern to copy data from.

Flags: Write / Construct Only

The “length” property

```
  “length”                     gulong
```

length of the pattern in ticks.

Flags: Read / Write / Construct

The “voices” property

```
  “voices”                     gulong
```

number of voices in the pattern.

Flags: Read

Signal Details

The “group-changed” signal

```
void
user_function (BtPattern         *self,
               BtParameterGroup *param_group,
               gboolean          intermediate,
               gpointer          user_data)
```

Signals that a group of this pattern has been changed (more than in one place). When doing e.g. line inserts, one will receive two updates, one before and one after. The first will have `intermediate` =TRUE. Applications can use that to defer change-consolidation.

Parameters

self	the pattern object that emitted the signal	
param_group	the parameter group	
intermediate	flag that is TRUE to signal that more change are coming	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The “param-changed” signal

```
void
user_function (BtPattern         *self,
               BtParameterGroup *param_group,
               gulong            tick,
               gulong            param,
               gpointer          user_data)
```

Signals that a param of this pattern has been changed.

Parameters

self	the pattern object that emitted the signal	
param_group	the parameter group	
tick	the tick position inside the pattern	
param	the parameter index	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The “pattern-changed” signal

```
void
user_function (BtPattern *self,
               gboolean   intermediate,
               gpointer   user_data)
```

Signals that this pattern has been changed (more than in one place). When doing e.g. line inserts, one will receive two updates, one before and one after. The first will have `intermediate` =TRUE. Applications can use that to defer change-consolidation.

Parameters

self	the pattern object that emitted the signal	
intermediate	flag that is TRUE to signal that more change are coming	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

3.6 BtPatternControlSource

BtPatternControlSource — Custom controlsource based on repeated event blocks (BtPatterns).

Functions

BtPatternControlSource *	bt_pattern_control_source_new ()

Properties

gpointer	default-value	Write
BtMachine *	machine	Read / Write /
BtParameterGroup *	parameter-group	Read / Write /
BtSequence *	sequence	Read / Write /
BtSongInfo *	song-info	Read / Write /

Types and Values

struct	BtPatternControlSource

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstControlBinding
                &#x2570;&#x2500;&#x2500; BtPatternControlSource
```

Includes

```
#include <libbtcore/core.h>
```

Description

The control source will update machine parameters over time, based on the events from the sequences and the patterns. One control-source will handle one single parameter. It implements the logic of computing the parameter value for a given time, taking multiple tracks and overlapping patterns into account.

At the begin of the timeline (ts==0) all parameters that have no value in the sequence will be initialized from "default-value". For trigger parameter this usualy is the no-value. For other parameters it is the last value one has set in the ui or via interaction controller.

Functions

bt_pattern_control_source_new ()

```
BtPatternControlSource~*
bt_pattern_control_source_new (GstObject *object,
                               const gchar *property_name,
                               BtSequence *sequence,
                               const BtSongInfo *song_info,
                               const BtMachine *machine,
                               BtParameterGroup *param_group);
```

Create a pattern control source for the given `machine` and `param_group` . Use gst_control_source_bind() to attach it to one specific parameter of the `param_group` .

Parameters

object	the object of the property	
property_name	the property-name to attach the control source	
sequence	the songs sequence	
song_info	the song info	
machine	the machine	
param_group	the parameter group	

Returns

the new pattern control source

Types and Values

struct BtPatternControlSource

```
struct BtPatternControlSource;
```

A pattern based control source

Property Details

The “default-value” property

```
“default-value”            gpointer
```

pointer to value to use if no other found.

Flags: Write

The “machine” property

```
“machine”                  BtMachine~*
```

the machine object, the controlsource belongs to.

Flags: Read / Write / Construct Only

The “parameter-group” property

```
“parameter-group”          BtParameterGroup~*
```

the parameter group.

Flags: Read / Write / Construct Only

The “sequence” property

```
“sequence”                 BtSequence~*
```

the sequence object.

Flags: Read / Write / Construct Only

The “song-info” property

```
“song-info”                BtSongInfo~*
```

the song-info object.

Flags: Read / Write / Construct Only

3.7 BtProcessorMachine

BtProcessorMachine — class for signal processing machines with inputs and outputs

Functions

BtProcessorMachine *	bt_processor_machine_new ()

Types and Values

struct	BtProcessorMachine
enum	BtProcessorMachinePatternIndex

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstElement
                &#x2570;&#x2500;&#x2500; GstBin
                    &#x2570;&#x2500;&#x2500; BtMachine
                        &#x2570;&#x2500;&#x2500; BtProcessorMachine
```

Implemented Interfaces

BtProcessorMachine implements GstChildProxy and BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Processors are machines that alter incomming audio.

Functions

bt_processor_machine_new ()

```
BtProcessorMachine~*
bt_processor_machine_new (const BtSong * const song,
                          const gchar * const id,
                          const gchar * const plugin_name,
                          const glong voices,
                          GError **err);
```

Create a new instance. The machine is automaticly added to the setup of the given song. You don't need to call `bt_setup_add_machine(setup,BT_MACHINE(machine));`.

Parameters

song	the song the new instance belongs to	
id	the id, we can use to lookup the machine	
plugin_name	the name of the gst-plugin the machine is using	
voices	the number of voices the machine should initially have	
err	inform about failed instance creation	

Returns

the new instance or NULL in case of an error

Types and Values

struct BtProcessorMachine

```
struct BtProcessorMachine;
```

Sub-class of a BtMachine that implements an effect-processor (a machine with in and outputs).

enum BtProcessorMachinePatternIndex

Use this with bt_machine_get_pattern_by_index() to get the command patterns.

Members

BT_PROCESSOR_MACHINE_PATTERN_INDEX_BREAK	stop the pattern
BT_PROCESSOR_MACHINE_PATTERN_INDEX_MUTE	mute the machine
BT_PROCESSOR_MACHINE_PATTERN_INDEX_BYPASS	bypass the machine
BT_PROCESSOR_MACHINE_PATTERN_INDEX_OFFSET	offset for real pattern ids

3.8 BtSequence

BtSequence — class for the event timeline of a BtSong instance

Functions

gboolean	bt_sequence_add_track ()
void	bt_sequence_delete_full_rows ()
void	bt_sequence_delete_rows ()
gchar *	bt_sequence_get_label ()
gulong	bt_sequence_get_loop_length ()
BtMachine *	bt_sequence_get_machine ()
BtCmdPattern *	bt_sequence_get_pattern ()
glong	bt_sequence_get_tick_by_pattern ()

glong	bt_sequence_get_track_by_machine ()
void	bt_sequence_insert_full_rows ()
void	bt_sequence_insert_rows ()
gboolean	bt_sequence_is_pattern_used ()
gulong	bt_sequence_limit_play_pos ()
gboolean	bt_sequence_move_track_left ()
gboolean	bt_sequence_move_track_right ()
BtSequence *	bt_sequence_new ()
gboolean	bt_sequence_remove_track_by_ix ()
gboolean	bt_sequence_remove_track_by_machine ()
void	bt_sequence_set_label ()
void	bt_sequence_set_pattern ()
gboolean	bt_sequence_set_pattern_quick ()

Properties

gulong	length	Read / Write
gboolean	loop	Read / Write
glong	loop-end	Read / Write
glong	loop-start	Read / Write
gpointer	properties	Read
BtSong *	song	Read / Write /
gpointer	toc	Read
gulong	tracks	Read / Write

Signals

void	pattern-added	No Hooks
void	pattern-removed	No Hooks
void	rows-changed	No Hooks
void	track-added	No Hooks
void	track-removed	No Hooks

Types and Values

struct	BtSequence

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSequence
```

Implemented Interfaces

BtSequence implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

A sequence holds grid of BtCmdPatterns, with labels on the time axis and BtMachine instances on the track axis. It tracks first and last use of patterns and provides two signals for notification - “pattern-added” and “pattern-removed”. The labels are exported as a GstToc, readable through the “toc” property.

It supports looping a section of the sequence (see “loop”, “loop-start”, “loop-end”).

The sequence is not aware of timing related information; for this take a look at BtSongInfo.

Functions

bt_sequence_add_track ()

```
gboolean
bt_sequence_add_track (const BtSequence * const self,
                       const BtMachine * const machine,
                       const glong ix);
```

Adds a new track with the *machine* at *ix* or the end.

Parameters

self	the BtSequence that holds the tracks	
machine	the BtMachine	
ix	position to add the track at, use -1 to append	

Returns

TRUE for success

bt_sequence_delete_full_rows ()

```
void
bt_sequence_delete_full_rows (const BtSequence * const self,
                              const gulong time,
                              const gulong rows);
```

Delete row for all tracks.

Parameters

self	the sequence	
time	the postion to delete	
rows	the number of rows to remove	

Since: 0.3

bt_sequence_delete_rows ()

```
void
bt_sequence_delete_rows (const BtSequence * const self,
                         const gulong time,
                         const glong track,
                         const gulong rows);
```

Delete row for given *track* .

Parameters

self	the sequence
time	the postion to delete
track	the track
rows	the number of rows to remove

Since: 0.3

bt_sequence_get_label ()

```
gchar~*
bt_sequence_get_label (const BtSequence * const self,
                       const gulong time);
```

Fetches the label for the given *time* position. Free when done.

Parameters

self	the BtSequence that holds the labels
time	the requested time position

Returns

a copy of the label or NULL in case of an error

bt_sequence_get_loop_length ()

```
gulong
bt_sequence_get_loop_length (const BtSequence * const self);
```

Calculates the length of the song loop in ticks.

Parameters

self	the BtSequence of the song

Returns

the length of the song loop in ticks

bt_sequence_get_machine ()

```
BtMachine~*
bt_sequence_get_machine (const BtSequence * const self,
                         const gulong track);
```

Fetches the BtMachine for the given `track` .

Parameters

self	the BtSequence that holds the tracks
track	the requested track index

Returns

a reference to the BtMachine pointer or NULL in case of an error. Unref when done.

[transfer full]

bt_sequence_get_pattern ()

```
BtCmdPattern~*
bt_sequence_get_pattern (const BtSequence * const self,
                         const gulong time,
                         const gulong track);
```

Fetches the pattern for the given `time` and `track` position.

Parameters

self	the BtSequence that holds the patterns
time	the requested time position
track	the requested track index

Returns

a reference to the BtCmdPattern or NULL when empty. Unref when done.

[transfer full]

bt_sequence_get_tick_by_pattern ()

```
glong
bt_sequence_get_tick_by_pattern (const BtSequence * const self,
                                 gulong track,
                                 const BtCmdPattern * const pattern,
                                 gulong tick);
```

Gets the next tick position >= `tick` this `pattern` is on.

Parameters

self	the sequence to search in	
track	the track to search in	
pattern	the pattern to find the next track for	
tick	the tick position to start the search from	

Returns

the tick position or -1 if there is no further tick for this *pattern* .

Since: 0.6

bt_sequence_get_track_by_machine ()

```
glong
bt_sequence_get_track_by_machine (const BtSequence * const self,
                                  const BtMachine * const machine,
                                  gulong track);
```

Gets the next track >= *track* this *machine* is on.

Parameters

self	the sequence to search in	
machine	the machine to find the next track for	
track	the track to start the search from	

Returns

the track-index or -1 if there is no further track for this *machine* .

Since: 0.6

bt_sequence_insert_full_rows ()

```
void
bt_sequence_insert_full_rows (const BtSequence * const self,
                              const gulong time,
                              const gulong rows);
```

Insert one empty row for all tracks.

Parameters

self	the sequence	
time	the postion to insert at	
rows	the number of rows to insert	

Since: 0.3

bt_sequence_insert_rows ()

```
void
bt_sequence_insert_rows (const BtSequence * const self,
                         const gulong time,
                         const glong track,
                         const gulong rows);
```

Insert one empty row for given `track` .

Parameters

self	the sequence	
time	the postion to insert at	
track	the track	
rows	the number of rows to insert	

Since: 0.3

bt_sequence_is_pattern_used ()

```
gboolean
bt_sequence_is_pattern_used (const BtSequence * const self,
                             const BtPattern * const pattern);
```

Checks if the `pattern` is used in the sequence.

Parameters

self	the sequence to check for pattern use	
pattern	the pattern to check for	

Returns

TRUE if `pattern` is used.

bt_sequence_limit_play_pos ()

```
gulong
bt_sequence_limit_play_pos (const BtSequence * const self,
                            const gulong play_pos);
```

Enforce the playback position to be within loop start and end or the song bounds if there is no loop.

Parameters

self	the sequence to trim the play position of	
play_pos	the time position to lock inbetween loop-boundaries	

Returns

the new *play_pos*

bt_sequence_move_track_left ()

```
gboolean
bt_sequence_move_track_left (const BtSequence * const self,
                             const gulong track);
```

Move the selected track on column left.

Parameters

self	the BtSequence that holds the tracks	
track	the track to move	

Returns

TRUE for success

bt_sequence_move_track_right ()

```
gboolean
bt_sequence_move_track_right (const BtSequence * const self,
                              const gulong track);
```

Move the selected track on column left.

Parameters

self	the BtSequence that holds the tracks	
track	the track to move	

Returns

TRUE for success

bt_sequence_new ()

```
BtSequence~*
bt_sequence_new (const BtSong * const song);
```

Create a new instance. One would not call this directly, but rather get this from a BtSong instance.

Parameters

song	the song the new instance belongs to	

Returns

the new instance or NULL in case of an error

bt_sequence_remove_track_by_ix ()

```
gboolean
bt_sequence_remove_track_by_ix (const BtSequence * const self,
                                const gulong ix);
```

Removes the specified *track* .

Parameters

self	the BtSequence that holds the tracks
ix	the requested track index

Returns

TRUE for success

bt_sequence_remove_track_by_machine ()

```
gboolean
bt_sequence_remove_track_by_machine (const BtSequence * const self,
                                     const BtMachine * const machine);
```

Removes all tracks that belong the the given *machine* .

Parameters

self	the BtSequence that holds the tracks
machine	the BtMachine

Returns

TRUE for success

bt_sequence_set_label ()

```
void
bt_sequence_set_label (const BtSequence * const self,
                       const gulong time,
                       const gchar * const label);
```

Sets a new label for the respective *time* position.

Parameters

self	the BtSequence that holds the labels	
time	the requested time position	
label	the new label	

bt_sequence_set_pattern ()

```
void
bt_sequence_set_pattern (const BtSequence * const self,
                         const gulong time,
                         const gulong track,
                         const BtCmdPattern * const pattern);
```

Sets the BtCmdPattern for the respective `time` and `track` position.

Parameters

self	the BtSequence that holds the patterns	
time	the requested time position	
track	the requested track index	
pattern	the BtCmdPattern or NULL to unset	

bt_sequence_set_pattern_quick ()

```
gboolean
bt_sequence_set_pattern_quick (const BtSequence * const self,
                               const gulong time,
                               const gulong track,
                               const BtCmdPattern * const pattern);
```

A quick version of bt_sequence_set_pattern() that does not check parameters. Useful when doing mass updates.

Parameters

self	the BtSequence that holds the patterns	
time	the requested time position	
track	the requested track index	
pattern	the BtCmdPattern or NULL to unset	

Returns

TRUE if a change has been made.

Since: 0.5

Types and Values

struct BtSequence

```
struct BtSequence;
```

Starting point for the BtSong timeline data-structures. Holds a series of array of BtCmdPatterns for time and tracks, which define the events that are sent to a BtMachine at a time.

Property Details

The "length" property

```
  "length"                    gulong
```

length of the sequence in timeline bars.

Flags: Read / Write

Allowed values: <= G_MAXINT64

The "loop" property

```
  "loop"                      gboolean
```

is loop activated.

Flags: Read / Write

Default value: FALSE

The "loop-end" property

```
  "loop-end"                  glong
```

end of the repeat sequence on the timeline.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The "loop-start" property

```
  "loop-start"                glong
```

start of the repeat sequence on the timeline.

Flags: Read / Write

Allowed values: >= -1

Default value: -1

The "properties" property

```
  "properties"                gpointer
```

hashtable of sequence properties.

Flags: Read

The “`song`” property

```
“song”                    BtSong~*
```

Set song object, the sequence belongs to.

Flags: Read / Write / Construct Only

The “`toc`” property

```
“toc”                     gpointer
```

TOC containing the labels.

Flags: Read

The “`tracks`” property

```
“tracks”                  gulong
```

number of tracks in the sequence.

Flags: Read / Write

Signal Details

The “`pattern-added`” signal

```
void
user_function (BtSequence *self,
               BtPattern  *pattern,
               gpointer    user_data)
```

A pattern has been used in the sequence for the first time.

Parameters

self	the sequence object that emitted the signal
pattern	the new pattern
user_data	user data set when the signal handler was connected.

Flags: No Hooks

The “`pattern-removed`” signal

```
void
user_function (BtSequence *self,
               BtPattern  *pattern,
               gpointer    user_data)
```

The last occurance of pattern has been removed from the sequence.

Parameters

self	the sequence object that emitted the signal	
pattern	the old pattern	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The "rows-changed" signal

```
void
user_function (BtSequence *self,
               gulong      begin,
               gulong      end,
               gpointer    user_data)
```

The content of the given rows in the sequence has changed.

Parameters

self	the sequence object that emitted the signal	
begin	start row that changed	
end	last row that changed	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

Since: 0.6

The "track-added" signal

```
void
user_function (BtSequence *self,
               BtMachine  *machine,
               gulong      track,
               gpointer    user_data)
```

A new track for `machine` has been added with the `track` index.

Parameters

self	the sequence object that emitted the signal	
machine	the machine for the track	
track	the track index	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

Since: 0.6

The "track-removed" signal

```
void
user_function (BtSequence *self,
               BtMachine  *machine,
               gulong      track,
               gpointer    user_data)
```

A track for *machine* has been removed at the *track* index.

Parameters

self	the sequence object that emitted the signal	
machine	the machine for the track	
track	the track index	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

Since: 0.6

3.9 BtSetup

BtSetup — class with all machines and wires (BtMachine and BtWire) for a BtSong instance

Functions

gboolean	bt_setup_add_machine ()
gboolean	bt_setup_add_wire ()
BtMachine *	bt_setup_get_machine_by_id ()
BtMachine *	bt_setup_get_machine_by_type ()
GList *	bt_setup_get_machines_by_type ()
gchar *	bt_setup_get_unique_machine_id ()
BtWire *	bt_setup_get_wire_by_dst_machine ()
BtWire *	bt_setup_get_wire_by_machines ()
BtWire *	bt_setup_get_wire_by_src_machine ()
GList *	bt_setup_get_wires_by_dst_machine ()
GList *	bt_setup_get_wires_by_src_machine ()
BtSetup *	bt_setup_new ()
void	bt_setup_remember_missing_machine ()
void	bt_setup_remove_machine ()
void	bt_setup_remove_wire ()

Properties

gpointer	machines	Read
gpointer	missing-machines	Read
gpointer	properties	Read
BtSong *	song	Read / Write /
gpointer	wires	Read

Signals

void	machine-added	No Hooks
void	machine-removed	No Hooks
void	wire-added	No Hooks
void	wire-removed	No Hooks

Types and Values

struct	BtSetup

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSetup
```

Implemented Interfaces

BtSetup implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

The setup manages virtual gear in a BtSong. This is a list of BtMachines that are used and the BtWires that connect them.

The setup manages the actual GStreamer GstPipeline content. Freshly created machines are not yet added to the pipeline. Only once a subgraph from a source is fully connected to the sink, that subgraph is added o the pipeline. Changing the machines and wires also works while the song is playing.

Applications can watch the GstObject:parent property to see whether a machine is physically inserted into the processing pipeline.

The setup takes ownership of the machines and wires. They are automatically added when they are created and destroyed together with the setup.

Functions

bt_setup_add_machine ()

```
gboolean
bt_setup_add_machine (const BtSetup * const self,
                      const BtMachine * const machine);
```

Let the setup know that the supplied machine is now part of the song.

Parameters

self	the setup to add the machine to
machine	the new machine instance

Returns

return TRUE, if the machine can be added. Returns FALSE if the machine is currently added to the setup.

bt_setup_add_wire ()

```
gboolean
bt_setup_add_wire (const BtSetup * const self,
                   const BtWire * const wire);
```

Let the setup know that the supplied wire is now part of the song.

Parameters

self	the setup to add the wire to
wire	the new wire instance

Returns

returns TRUE, if the wire is added. Returns FALSE, if the setup contains a wire which is linked between the same src and dst machines (cycle check).

bt_setup_get_machine_by_id ()

```
BtMachine~*
bt_setup_get_machine_by_id (const BtSetup * const self,
                            const gchar * const id);
```

Search the setup for a machine by the supplied id. The machine must have been added previously to this setup with bt_setup_add_machine

Parameters

self	the setup to search for the machine
id	the identifier of the machine

Returns

BtMachine instance or NULL if not found. Unref the machine, when done with it.

[transfer full]

bt_setup_get_machine_by_type ()

```
BtMachine~*
bt_setup_get_machine_by_type (const BtSetup * const self,
                                  const GType type);
```

Search the setup for the first machine with the given type. The machine must have been added previously to this setup with bt_setup_add_machine().

Parameters

self	the setup to search for the machine	
type	the gobject type (sink,processor,source)	

Returns

BtMachine instance or NULL if not found. Unref the machine, when done with it.

[transfer full]

bt_setup_get_machines_by_type ()

```
GList~*
bt_setup_get_machines_by_type (const BtSetup * const self,
                               const GType type);
```

Gathers all machines of the given type from the setup.

Parameters

self	the setup to search for the machine	
type	the gobject type (sink,processor,source)	

Returns

the list instance or NULL if not found. Free the list (and unref the machines), when done with it.

[element-type BuzztraxCore.Machine][transfer full]

bt_setup_get_unique_machine_id ()

```
gchar~*
bt_setup_get_unique_machine_id (const BtSetup * const self,
                                const gchar * const base_name);
```

The function makes the supplied base_name unique in this setup by eventually adding a number postfix. This method should be used when adding new machines.

Parameters

self	the setup for which the name should be unique	
base_name	the leading name part	

Returns

the newly allocated unique name.

[transfer full]

bt_setup_get_wire_by_dst_machine ()

```
BtWire~*
bt_setup_get_wire_by_dst_machine (const BtSetup * const self,
                                  const BtMachine * const dst);
```

Searches for the first wire in setup that uses the given BtMachine as a target. In other words - it returns the first wire that ends at the given BtMachine.

Parameters

self	the setup to search for the wire	
dst	the machine that is at the dst end of the wire	

Returns

the BtWire or NULL. Unref the wire, when done with it.

[transfer full]

bt_setup_get_wire_by_machines ()

```
BtWire~*
bt_setup_get_wire_by_machines (const BtSetup * const self,
                               const BtMachine * const src,
                               const BtMachine * const dst);
```

Searches for a wire in setup that uses the given BtMachine instances as a source and dest.

Parameters

self	the setup to search for the wire	
src	the machine that is at the src end of the wire	
dst	the machine that is at the dst end of the wire	

Returns

the BtWire or NULL. Unref the wire, when done with it.

[transfer full]

bt_setup_get_wire_by_src_machine ()

```
BtWire~*
bt_setup_get_wire_by_src_machine (const BtSetup * const self,
                                  const BtMachine * const src);
```

Searches for the first wire in setup that uses the given BtMachine as a source. In other words - it returns the first wire that starts at the given BtMachine.

Parameters

self	the setup to search for the wire	
src	the machine that is at the src end of the wire	

Returns

the BtWire or NULL. Unref the wire, when done with it.

[transfer full]

bt_setup_get_wires_by_dst_machine ()

```
GList~*
bt_setup_get_wires_by_dst_machine (const BtSetup * const self,
                                   const BtMachine * const dst);
```

Searches for all wires in setup that use the given BtMachine as a target.

Parameters

self	the setup to search for the wire	
dst	the machine that is at the dst end of the wire	

Returns

a GList with the BtWires or NULL. Free the list (and unref the wires), when done with it.

[element-type BuzztraxCore.Wire][transfer full]

bt_setup_get_wires_by_src_machine ()

```
GList~*
bt_setup_get_wires_by_src_machine (const BtSetup * const self,
                                   const BtMachine * const src);
```

Searches for all wires in setup that use the given BtMachine as a source.

Parameters

self	the setup to search for the wire
src	the machine that is at the src end of the wire

Returns

a GList with the BtWires or NULL. Free the list (and unref the wires), when done with it.

[element-type BuzztraxCore.Wire][transfer full]

bt_setup_new ()

```
BtSetup~*
bt_setup_new (const BtSong * const song);
```

Create a new instance

Parameters

song	the song the new instance belongs to

Returns

the new instance or NULL in case of an error

bt_setup_remember_missing_machine ()

```
void
bt_setup_remember_missing_machine (const BtSetup * const self,
                                   const gchar * const str);
```

Loaders can use this function to collect information about machines that failed to load. The front-end can access this later by reading BtSetup::missing-machines property.

Parameters

self	the setup
str	human readable description of the missing machine

bt_setup_remove_machine ()

```
void
bt_setup_remove_machine (const BtSetup * const self,
                         const BtMachine * const machine);
```

Let the setup know that the supplied machine is removed from the song.

Parameters

self	the setup to remove the machine from
machine	the machine instance to remove

bt_setup_remove_wire ()

```
void
bt_setup_remove_wire (const BtSetup * const self,
                      const BtWire * const wire);
```

Let the setup know that the supplied wire is removed from the song.

Parameters

self	the setup to remove the wire from
wire	the wire instance to remove

Types and Values

struct BtSetup

```
struct BtSetup;
```

virtual hardware setup (contains BtMachine and BtWire objects)

Property Details

The "machines" property

```
  "machines"                gpointer
```

A copy of the list of machines.

Flags: Read

The "missing-machines" property

```
  "missing-machines"        gpointer
```

The list of missing machines, don't change.

Flags: Read

The "properties" property

```
  "properties"              gpointer
```

hashtable of setup properties.

Flags: Read

The “song” property

```
“song”                    BtSong~*
```

Set song object, the setup belongs to.

Flags: Read / Write / Construct Only

The “wires” property

```
“wires”                   gpointer
```

A copy of the list of wires.

Flags: Read

Signal Details

The “machine-added” signal

```
void
user_function (BtSetup   *self,
               BtMachine *machine,
               gpointer   user_data)
```

A new machine item has been added to the setup.

Parameters

self	the setup object that emitted the signal	
machine	the new machine	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The “machine-removed” signal

```
void
user_function (BtSetup   *self,
               BtMachine *machine,
               gpointer   user_data)
```

A machine item has been removed from the setup.

Parameters

self	the setup object that emitted the signal	
machine	the old machine	

user_data	user data set when the signal handler was connected.

Flags: No Hooks

The "wire-added" signal

```
void
user_function (BtSetup *self,
               BtWire  *wire,
               gpointer user_data)
```

A new wire item has been added to the setup.

Parameters

self	the setup object that emitted the signal
wire	the new wire
user_data	user data set when the signal handler was connected.

Flags: No Hooks

The "wire-removed" signal

```
void
user_function (BtSetup *self,
               BtWire  *wire,
               gpointer user_data)
```

A wire item has been removed from the setup.

Parameters

self	the setup object that emitted the signal
wire	the old wire
user_data	user data set when the signal handler was connected.

Flags: No Hooks

3.10 BtSinkBin

BtSinkBin — bin to be used by BtSinkMachine

Functions

gboolean	bt_sink_bin_is_record_format_supported ()

Properties

gpointer	analyzers	Read / Write
GstElement *	input-gain	Read / Write
gdouble	master-volume	Read / Write
BtSinkBinMode	mode	Read / Write
gchar *	record-file-name	Read / Write
BtSinkBinRecordFormat	record-format	Read / Write

Types and Values

struct	BtSinkBin
enum	BtSinkBinMode
enum	BtSinkBinRecordFormat

Object Hierarchy

```
    GEnum
    &#x251c;&#x2500;&#x2500; BtSinkBinMode
    &#x2570;&#x2500;&#x2500; BtSinkBinRecordFormat
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstElement
                &#x2570;&#x2500;&#x2500; GstBin
                    &#x2570;&#x2500;&#x2500; BtSinkBin
```

Implemented Interfaces

BtSinkBin implements GstChildProxy and GstBtTempo.

Includes

```
#include <libbtcore/core.h>
```

Description

The sink-bin provides switchable play and record facilities. It also provides controllable master-volume.

In play and record modes it plugs a chain of elements. In combined play and record mode it uses a tee and plugs both pipelines.

Functions

bt_sink_bin_is_record_format_supported ()

```
gboolean
bt_sink_bin_is_record_format_supported
                                        (BtSinkBinRecordFormat format);
```

Each record format might need a couple of GStreamer element to work. This function verifies that all needed element are available.

Parameters

format	the format to check

Returns

TRUE if a fomat is useable

Types and Values

struct BtSinkBin

```
struct BtSinkBin;
```

Sub-class of a GstBin that implements a signal output (a machine with inputs only).

enum BtSinkBinMode

BtSinkBin supports several modes of operation. Playing is the default mode. Passthru is only needed if the song is plugged in another pipeline.

Members

BT_SINK_BIN_MODE_PLAY	play the song
BT_SINK_BIN_MODE_RECORD	record to file
BT_SINK_BIN_MODE_PLAY_AND_RECORD	play and record together
BT_SINK_BIN_MODE_PASS_THRU	output audio on sometimes src pad

enum BtSinkBinRecordFormat

BtSinkMachine can record audio in several formats.

Members

BT_SINK_BIN_RECORD_FORMAT_OGG_VORBIS	ogg vor-bis
BT_SINK_BIN_RECORD_FORMAT_MP3	mp3
BT_SINK_BIN_RECORD_FORMAT_WAV	wav
BT_SINK_BIN_RECORD_FORMAT_OGG_FLAC	ogg flac
BT_SINK_BIN_RECORD_FORMAT_RAW	raw
BT_SINK_BIN_RECORD_FORMAT_MP4_AAC	mp4 aac
BT_SINK_BIN_RECORD_FORMAT_FLAC	flac
BT_SINK_BIN_RECORD_FORMAT_OGG_OPUS	ogg opus
BT_SINK_BIN_RECORD_FORMAT_COUNT	number of for-mats

Property Details

The “analyzers” property

```
“analyzers”                gpointer
```

list of master analyzers.

Flags: Read / Write

The “input-gain” property

```
“input-gain”               GstElement~*
```

the input-gain element, if any.

Flags: Read / Write

The “master-volume” property

```
“master-volume”            gdouble
```

master volume for the song.

Flags: Read / Write

Allowed values: [0,1]

Default value: 1

The “mode” property

```
“mode”                    BtSinkBinMode
```

mode of operation.

Flags: Read / Write

Default value: BT_SINK_BIN_MODE_PLAY

The “record-file-name” property

```
“record-file-name”        gchar~*
```

the file-name to use for recording.

Flags: Read / Write

Default value: NULL

The “record-format” property

```
“record-format”           BtSinkBinRecordFormat
```

format to use when in record mode.

Flags: Read / Write

Default value: .vorbis.ogg

3.11 BtSinkMachine

BtSinkMachine — class for signal processing machines with inputs only

Functions

BtSinkMachine *	bt_sink_machine_new ()

Types and Values

struct	BtSinkMachine
enum	BtSinkMachinePatternIndex

Object Hierarchy

```
GObject
&#x2570;&#x2500;&#x2500; GInitiallyUnowned
    &#x2570;&#x2500;&#x2500; GstObject
        &#x2570;&#x2500;&#x2500; GstElement
            &#x2570;&#x2500;&#x2500; GstBin
                &#x2570;&#x2500;&#x2500; BtMachine
                    &#x2570;&#x2500;&#x2500; BtSinkMachine
```

Implemented Interfaces

BtSinkMachine implements GstChildProxy and BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Sinks are machines that do playback or recording of the song. The sink-machine utilizes the BtSinkBin to transparently switch elements between record (encoding) and playback.

Functions

bt_sink_machine_new ()

```
BtSinkMachine~*
bt_sink_machine_new (const BtSong * const song,
                     const gchar * const id,
                     GError **err);
```

Create a new instance. The machine is automatically added to the setup from the given song object. You don't need to add the machine with `bt_setup_add_machine(setup,BT_MACHINE(machine));`.

The element used for this machine is BtSinkBin which is configured according to the use-case (playback, recording). The playback device is taken from the BtSettings.

Parameters

song	the song the new instance belongs to	
id	the id, we can use to lookup the machine	
err	inform about failed instance creation	

Returns

the new instance or NULL in case of an error

Types and Values

struct BtSinkMachine

```
struct BtSinkMachine;
```

Sub-class of a BtMachine that implements a signal output (a machine with inputs only).

enum BtSinkMachinePatternIndex

Use this with bt_machine_get_pattern_by_index() to get the command patterns.

Members

BT_SINK_MACHINE_PATTERN_INDEX_BREAK	stop the pat- tern
BT_SINK_MACHINE_PATTERN_INDEX_MUTE	mute the ma- chine
BT_SINK_MACHINE_PATTERN_INDEX_OFFSET	offset for real pat- tern ids

3.12 BtSong

BtSong — class of a song project object (contains BtSongInfo, BtSetup, BtSequence and BtWavetable)

Functions

gboolean	bt_song_continue ()
BtSong *	bt_song_new ()
gboolean	bt_song_pause ()
gboolean	bt_song_play ()
gboolean	bt_song_stop ()
gboolean	bt_song_update_playback_position ()

Properties

BtApplication *	app	Read / Write /
GstBin *	bin	Read
gboolean	is-idle	Read / Write
gboolean	is-playing	Read
BtSinkMachine *	master	Read / Write
gulong	play-pos	Read / Write
gdouble	play-rate	Read / Write
BtSequence *	sequence	Read
BtSetup *	setup	Read
BtSongInfo *	song-info	Read
BtSongIO *	song-io	Read / Write
BtWavetable *	wavetable	Read

Types and Values

struct	BtSong
struct	BtSongClass

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSong
```

Implemented Interfaces

BtSong implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

A song is the top-level container object to manage all song-related objects. The BtSetup contains the machines and their connections, the BtSequence contains the overall time-line, the BtWavetable holds a list of audio snippets and the BtSongInfo has a couple of meta-data items for the song.

To load or save a song, use a BtSongIO object. These implement loading and saving for different file-formats.

One can seek in a song by setting the “play-pos” property. Likewise one can watch the property to display the playback position.

The “play-rate” property can be used to change the playback speed and direction.

Functions

bt_song_continue ()

```
gboolean
bt_song_continue (const BtSong * const self);
```

Continues the playback of the specified song instance.

Parameters

self	the song that should be paused

Returns

TRUE for success

bt_song_new ()

```
BtSong~*
bt_song_new (const BtApplication * const app);
```

Create a new instance. The new song instance automatically has one instance of BtSetup, BtSequence and BtSongInfo. These instances can be retrieved via the respecting properties.

For example use following code to retrive a BtSequence from the song class:

```
BtSequence *sequence;
...
g_object_get(BT_SONG(song), "sequence", &sequence, NULL);
```

Parameters

app	the application object the songs belongs to.

Returns

the new instance or NULL in case of an error

bt_song_pause ()

```
gboolean
bt_song_pause (const BtSong * const self);
```

Pauses the playback of the specified song instance.

Parameters

self	the song that should be paused

Returns

TRUE for success

bt_song_play ()

```
gboolean
bt_song_play (const BtSong * const self);
```

Starts to play the specified song instance from beginning. This methods toggles the “is-playing” property.

Parameters

self	the song that should be played

Returns

TRUE for success

bt_song_stop ()

```
gboolean
bt_song_stop (const BtSong * const self);
```

Stops the playback of the specified song instance.

Parameters

self	the song that should be stopped

Returns

TRUE for success

bt_song_update_playback_position ()

```
gboolean
bt_song_update_playback_position (const BtSong * const self);
```

Updates the playback-position counter to fire all “play-pos” notify handlers.

Parameters

self	the song that should update its playback-pos counter

Returns

FALSE if the song is not playing

Types and Values

struct BtSong

```
struct BtSong;
```

Song project object (contains BtSongInfo, BtSetup and BtSequence)

struct BtSongClass

```
struct BtSongClass {
  const GObjectClass parent;
};
```

Base class for songs

Members

const GObjectClass *parent*;	parent class type

Property Details

The “app” property

```
“app”                          BtApplication~*
```

set application object, the song belongs to.

Flags: Read / Write / Construct Only

The “bin” property

```
“bin”                          GstBin~*
```

songs top-level GstElement container.

Flags: Read

The “is-idle” property

```
“is-idle”                      gboolean
```

request that the song should idle-loop if not playing.

Flags: Read / Write

Default value: FALSE

The “is-playing” property

```
“is-playing”                   gboolean
```

tell whether the song is playing right now or not.

Flags: Read

Default value: FALSE

The “master” property

```
“master”                       BtSinkMachine~*
```

songs audio_sink.

Flags: Read / Write

The “play-pos” property

```
“play-pos”                     gulong
```

position of the play cursor of the sequence in timeline bars.

Flags: Read / Write

Allowed values: <= G_MAXINT64

The “play-rate” property

```
“play-rate”                gdouble
```

playback rate of the sequence.

Flags: Read / Write

Allowed values: [-5,5]

Default value: 1

The “sequence” property

```
“sequence”                 BtSequence~*
```

songs sequence sub object.

Flags: Read

The “setup” property

```
“setup”                    BtSetup~*
```

songs setup sub object.

Flags: Read

The “song-info” property

```
“song-info”                BtSongInfo~*
```

songs metadata sub object.

Flags: Read

The “song-io” property

```
“song-io”                  BtSongIO~*
```

the song-io plugin during i/o operations.

Flags: Read / Write

The “wavetable” property

```
“wavetable”                BtWavetable~*
```

songs wavetable sub object.

Flags: Read

3.13 BtSongInfo

BtSongInfo — class that keeps the meta-data for a BtSong instance

Functions

BtSongInfo *	bt_song_info_new ()
const gchar *	bt_song_info_get_change_dts_in_local_tz ()
gint	bt_song_info_get_seconds_since_last_saved ()
GstClockTime	bt_song_info_tick_to_time ()
gulong	bt_song_info_time_to_tick ()
void	bt_song_info_time_to_m_s_ms ()
void	bt_song_info_tick_to_m_s_ms ()

Properties

gchar *	author	Read / Write
gulong	bars	Read / Write
gulong	bpm	Read / Write
gchar *	change-dts	Read / Write
gchar *	create-dts	Read / Write
gchar *	file-name	Read / Write
gchar *	genre	Read / Write
gchar *	info	Read / Write
gchar *	name	Read / Write
BtSong *	song	Read / Write /
gpointer	taglist	Read
guint64	tick-duration	Read
gulong	tpb	Read / Write

Types and Values

struct	BtSongInfo

Object Hierarchy

```
GObject
&#x2570;&#x2500;&#x2500; BtSongInfo
```

Implemented Interfaces

BtSongInfo implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

This class exposes the meta-data of a song as GObject properties. These are for one pure data fields such as author and song name. These fields get used when recording a song to a file (rendering) in the form of meta-tags.

Further there are fields that determine rythm and song-speed. The speed is determined by “bpm”. The rythm is determined by “bars” and “tpb”. If ’bars’ is 16, than on can have 1/16 notes. And if ’ticks per beat’ is 4 one will have 4 beats - a classic 4/4 meassure. For a 3/4 meassure, ’bars’ would be 12. Thus bars = beats * tpb.

Finally, the class offers a couple of timing related conversion functions.

Functions

bt_song_info_new ()

```
BtSongInfo~*
bt_song_info_new (const BtSong * const song);
```

Create a new instance

Parameters

song	the song the new instance belongs to

Returns

the new instance or NULL in case of an error

bt_song_info_get_change_dts_in_local_tz ()

```
const gchar~*
bt_song_info_get_change_dts_in_local_tz
                               (const BtSongInfo * const self);
```

Convert the BtSongInfo::change-dts to local time zone.

Return: the time stamp in iso 8601 format

Parameters

self	the song_info

bt_song_info_get_seconds_since_last_saved ()

```
gint
bt_song_info_get_seconds_since_last_saved
                               (const BtSongInfo * const self);
```

Calculate the seconds since last save time or the creation time if the song is new.

Return: the seconds

Parameters

self	the song_info

bt_song_info_tick_to_time ()

```
GstClockTime
bt_song_info_tick_to_time (const BtSongInfo * const self,
                           const gulong tick);
```

Convert a given tick position to the time in μs.

Parameters

self	the song_info	
tick	the tick position	

Returns

the time in μs

bt_song_info_time_to_tick ()

```
gulong
bt_song_info_time_to_tick (const BtSongInfo * const self,
                           const GstClockTime ts);
```

Convert a given time in μs to the tick position.

Parameters

self	the song_info	
ts	the time in μs	

Returns

the integer tick position

bt_song_info_time_to_m_s_ms ()

```
void
bt_song_info_time_to_m_s_ms (const BtSongInfo * const self,
                             gulong ts,
                             gulong *m,
                             gulong *s,
                             gulong *ms);
```

Convert a given time in μs to minutes, seconds and milliseconds.

Parameters

self	the song_info	
ts	the time in μs	
m	location for the minutes	
s	location for the seconds	
ms	location for the milliseconds	

bt_song_info_tick_to_m_s_ms ()

```
void
bt_song_info_tick_to_m_s_ms (const BtSongInfo * const self,
                             const gulong tick,
                             gulong *m,
                             gulong *s,
                             gulong *ms);
```

Convert a given tick position to minutes, seconds and milliseconds.

Parameters

self	the song_info	
tick	the tick position	
m	location for the minutes	
s	location for the seconds	
ms	location for the milliseconds	

Types and Values

struct BtSongInfo

```
struct BtSongInfo;
```

holds song metadata

Property Details

The "author" property

```
  "author"                      gchar~*
```

songs author.

Flags: Read / Write

Default value: NULL

The "bars" property

```
  "bars"                        gulong
```

how many bars per meassure.

Flags: Read / Write

Allowed values: [1,64]

The "bpm" property

```
  "bpm"                         gulong
```

how many beats should be played in a minute.

Flags: Read / Write

Allowed values: [1,1000]

The “change-dts” property

```
  “change-dts”                gchar~*
```

song changed date time stamp (iso 8601 format).

Flags: Read / Write

Default value: NULL

The “create-dts” property

```
  “create-dts”                gchar~*
```

song creation date time stamp (iso 8601 format).

Flags: Read / Write

Default value: NULL

The “file-name” property

```
  “file-name”                 gchar~*
```

songs file name.

Flags: Read / Write

Default value: NULL

The “genre” property

```
  “genre”                     gchar~*
```

songs genre.

Flags: Read / Write

Default value: NULL

The “info” property

```
  “info”                      gchar~*
```

songs freeform info.

Flags: Read / Write

Default value: "comment me!"

The “name” property

```
  “name”                      gchar~*
```

songs name.

Flags: Read / Write

Default value: "untitled song"

The “song” property

```
“song”                     BtSong~*
```

song object, the song-info belongs to.

Flags: Read / Write / Construct Only

The “taglist” property

```
“taglist”                  gpointer
```

songs meta data as a taglist.

Flags: Read

The “tick-duration” property

```
“tick-duration”            guint64
```

the duration for a tick in μs calculated form the song tempo.

Flags: Read

Allowed values: >= 1

Default value: 1

The “tpb” property

```
“tpb”                      gulong
```

event granularity in one beat.

Flags: Read / Write

Allowed values: [1,128]

3.14 BtSourceMachine

BtSourceMachine — class for signal processing machines with outputs only

Functions

BtSourceMachine *	bt_source_machine_new ()

Types and Values

struct	BtSourceMachine
enum	BtSourceMachinePatternIndex

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstElement
                &#x2570;&#x2500;&#x2500; GstBin
                    &#x2570;&#x2500;&#x2500; BtMachine
                        &#x2570;&#x2500;&#x2500; BtSourceMachine
```

Implemented Interfaces

BtSourceMachine implements GstChildProxy and BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Sources are machines that generate audio.

Functions

bt_source_machine_new ()

```
BtSourceMachine~*
bt_source_machine_new (const BtSong * const song,
                       const gchar * const id,
                       const gchar * const plugin_name,
                       const glong voices,
                       GError **err);
```

Create a new instance The machine is automaticly added to the setup from the given song object. You don't need to add the machine with `bt_setup_add_machine(setup,BT_MACHINE(machine));`.

Parameters

song	the song the new instance belongs to	
id	the id, we can use to lookup the machine	
plugin_name	the name of the gst-plugin the machine is using	
voices	the number of voices the machine should initially have	
err	inform about failed instance creation	

Returns

the new instance or NULL in case of an error

Types and Values

struct BtSourceMachine

```
struct BtSourceMachine;
```

Sub-class of a BtMachine that implements a signal generator (a machine with outputs only).

enum BtSourceMachinePatternIndex

Use this with bt_machine_get_pattern_by_index() to get the command patterns.

Members

BT_SOURCE_MACHINE_PATTERN_INDEX_BREAK	stop the pattern
BT_SOURCE_MACHINE_PATTERN_INDEX_MUTE	mute the machine
BT_SOURCE_MACHINE_PATTERN_INDEX_SOLO	play only this machine
BT_SOURCE_MACHINE_PATTERN_INDEX_OFFSET	offset for real pattern ids

3.15 BtValueGroup

BtValueGroup — a GValue array of parameter values

Functions

void	bt_value_group_blend_column ()
void	bt_value_group_blend_columns ()
BtValueGroup *	bt_value_group_copy ()
void	bt_value_group_clear_column ()
void	bt_value_group_clear_columns ()
void	bt_value_group_delete_full_row ()
void	bt_value_group_delete_row ()
gboolean	bt_value_group_deserialize_column ()
void	bt_value_group_flip_column ()
void	bt_value_group_flip_columns ()
gchar *	bt_value_group_get_event ()
GValue *	bt_value_group_get_event_data ()

void	bt_value_group_insert_full_row ()
void	bt_value_group_insert_row ()
BtValueGroup *	bt_value_group_new ()
void	bt_value_group_randomize_column ()
void	bt_value_group_randomize_columns ()
void	bt_value_group_range_randomize_column ()
void	bt_value_group_range_randomize_columns ()
void	bt_value_group_transpose_coarse_down_column ()
void	bt_value_group_transpose_coarse_down_columns ()
void	bt_value_group_transpose_coarse_up_column ()
void	bt_value_group_transpose_coarse_up_columns ()
void	bt_value_group_transpose_fine_down_column ()
void	bt_value_group_transpose_fine_down_columns ()
void	bt_value_group_transpose_fine_up_column ()
void	bt_value_group_transpose_fine_up_columns ()
void	bt_value_group_serialize_column ()
void	bt_value_group_serialize_columns ()
gboolean	bt_value_group_set_event ()
gboolean	bt_value_group_test_event ()
gboolean	bt_value_group_test_tick ()

Properties

gulong	length	Read / Write /
BtParameterGroup *	parameter-group	Read / Write /

Signals

void	group-changed	No Hooks
void	param-changed	No Hooks

Types and Values

struct	BtValueGroup

Object Hierarchy

```
GObject
&#x2570;&#x2500;&#x2500; BtValueGroup
```

Includes

```
#include <libbtcore/core.h>
```

Description

A group of GValues, such as used in patterns. The class provides a variety of methods to manipulate the data fields.

The value group maintains two blocks of data values. One for validated fields and one for plain fields. This allows step wise entry of data (multi column entry of sparse enums). The validated cells are only set as the plain value becomes valid. Invalid values are not copied nor are they stored in the song.

Functions

bt_value_group_blend_column ()

```
void
bt_value_group_blend_column (const BtValueGroup * const self,
                             const gulong start_tick,
                             const gulong end_tick,
                             const gulong param);
```

Fade values from `start_tick` to `end_tick` for `param` .

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.7

bt_value_group_blend_columns ()

```
void
bt_value_group_blend_columns (const BtValueGroup * const self,
                              const gulong start_tick,
                              const gulong end_tick);
```

Fade values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.7

bt_value_group_copy ()

```
BtValueGroup~*
bt_value_group_copy (const BtValueGroup * const self);
```

Create a new instance as a copy of the given instance.

Parameters

self	the value-group to create a copy from

Returns

the new instance or NULL in case of an error.

[transfer full]

Since: 0.7

bt_value_group_clear_column ()

```
void
bt_value_group_clear_column (const BtValueGroup * const self,
                             const gulong start_tick,
                             const gulong end_tick,
                             const gulong param);
```

Clears values from `start_tick` to `end_tick` for `param` .

Parameters

self	the pattern
start_tick	the start position for the range
end_tick	the end position for the range
param	the parameter

Since: 0.7

bt_value_group_clear_columns ()

```
void
bt_value_group_clear_columns (const BtValueGroup * const self,
                              const gulong start_tick,
                              const gulong end_tick);
```

Clear values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern
start_tick	the start position for the range
end_tick	the end position for the range

Since: 0.7

bt_value_group_delete_full_row ()

```
void
bt_value_group_delete_full_row (const BtValueGroup * const self,
                                const gulong tick);
```

Delete row for all parameters.

Parameters

self	the pattern	
tick	the position to delete	

Since: 0.7

bt_value_group_delete_row ()

```
void
bt_value_group_delete_row (const BtValueGroup * const self,
                           const gulong tick,
                           const gulong param);
```

Delete row for given `param` .

Parameters

self	the pattern	
tick	the position to delete	
param	the parameter	

Since: 0.7

bt_value_group_deserialize_column ()

```
gboolean
bt_value_group_deserialize_column (const BtValueGroup * const self,
                                   const gulong start_tick,
                                   const gulong end_tick,
                                   const gulong param,
                                   const gchar *data);
```

Deserializes values to `start_tick` to `end_tick` for `param` from `data` .

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	
data	the source data	

Returns

TRUE for success, FALSE e.g. to indicate incompatible GType values for the column specified by `param` and the `data` .

Since: 0.7

bt_value_group_flip_column ()

```
void
bt_value_group_flip_column (const BtValueGroup * const self,
                            const gulong start_tick,
                            const gulong end_tick,
                            const gulong param);
```

Flips values from `start_tick` to `end_tick` for `param` up-side down.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.7

bt_value_group_flip_columns ()

```
void
bt_value_group_flip_columns (const BtValueGroup * const self,
                             const gulong start_tick,
                             const gulong end_tick);
```

Flips values from `start_tick` to `end_tick` for all params up-side down.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.7

bt_value_group_get_event ()

```
gchar~*
bt_value_group_get_event (const BtValueGroup * const self,
                          const gulong tick,
                          const gulong param);
```

Returns the string representation of the specified cell. Free it when done.

Parameters

self	the pattern the cell belongs to
tick	the tick (time) position starting with 0
param	the number of the parameter starting with 0

Returns

a newly allocated string with the data or NULL on error

Since: 0.7

bt_value_group_get_event_data ()

```
GValue~*
bt_value_group_get_event_data (const BtValueGroup * const self,
                               const gulong tick,
                               const gulong param);
```

Fetches a cell from the given location in the pattern. If there is no event there, then the GValue is uninitialized. Test with BT_IS_GVALUE(event).

Parameters

self	the pattern to search for the param
tick	the tick (time) position starting with 0
param	the number of the parameter starting with 0

Returns

the GValue or NULL if out of the pattern range

Since: 0.7

bt_value_group_insert_full_row ()

```
void
bt_value_group_insert_full_row (const BtValueGroup * const self,
                                const gulong tick);
```

Insert one empty row for all parameters.

Parameters

self	the pattern
tick	the position to insert at

Since: 0.7

bt_value_group_insert_row ()

```
void
bt_value_group_insert_row (const BtValueGroup * const self,
                           const gulong tick,
                           const gulong param);
```

Insert one empty row for given `param` .

Parameters

self	the pattern
tick	the position to insert at
param	the parameter

Since: 0.7

bt_value_group_new ()

```
BtValueGroup~*
bt_value_group_new (const BtParameterGroup * const param_group,
                    const gulong length);
```

Create a new instance.

Parameters

param_group	the parameter-group
length	the number of ticks

Returns

the new instance or NULL in case of an error.

[transfer full]

Since: 0.7

bt_value_group_randomize_column ()

```
void
bt_value_group_randomize_column (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick,
                                 const gulong param);
```

Randomize values from `start_tick` to `end_tick` for `param` .

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.7

bt_value_group_randomize_columns ()

```
void
bt_value_group_randomize_columns (const BtValueGroup * const self,
                                  const gulong start_tick,
                                  const gulong end_tick);
```

Randomize values from `start_tick` to `end_tick` for all params.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.7

bt_value_group_range_randomize_column ()

```
void
bt_value_group_range_randomize_column (const BtValueGroup * const self,
                                       const gulong start_tick,
                                       const gulong end_tick,
                                       const gulong param);
```

Randomize values from `start_tick` to `end_tick` for `param` using the first and last value as bounds for the random values.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.7

bt_value_group_range_randomize_columns ()

```
void
bt_value_group_range_randomize_columns
                               (const BtValueGroup * const self,
                                const gulong start_tick,
                                const gulong end_tick);
```

Randomize values from *start_tick* to *end_tick* for all params using the first and last value as bounds for the random values.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.7

bt_value_group_transpose_coarse_down_column ()

```
void
bt_value_group_transpose_coarse_down_column
                               (const BtValueGroup * const self,
                                const gulong start_tick,
                                const gulong end_tick,
                                const gulong param);
```

Transposes values from *start_tick* to *end_tick* for *param* in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.11

bt_value_group_transpose_coarse_down_columns ()

```
void
bt_value_group_transpose_coarse_down_columns
                               (const BtValueGroup * const self,
                                const gulong start_tick,
                                const gulong end_tick);
```

Transposes values from *start_tick* to *end_tick* for all params in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_value_group_transpose_coarse_up_column ()

```
void
bt_value_group_transpose_coarse_up_column
                                (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick,
                                 const gulong param);
```

Transposes values from `start_tick` to `end_tick` for `param` in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.11

bt_value_group_transpose_coarse_up_columns ()

```
void
bt_value_group_transpose_coarse_up_columns
                                (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick);
```

Transposes values from `start_tick` to `end_tick` for all params in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_value_group_transpose_fine_down_column ()

```
void
bt_value_group_transpose_fine_down_column
                                (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick,
                                 const gulong param);
```

Transposes values from *start_tick* to *end_tick* for *param* in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.11

bt_value_group_transpose_fine_down_columns ()

```
void
bt_value_group_transpose_fine_down_columns
                                (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick);
```

Transposes values from *start_tick* to *end_tick* for all params in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_value_group_transpose_fine_up_column ()

```
void
bt_value_group_transpose_fine_up_column
                                (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick,
                                 const gulong param);
```

Transposes values from *start_tick* to *end_tick* for *param* in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	

Since: 0.11

bt_value_group_transpose_fine_up_columns ()

```
void
bt_value_group_transpose_fine_up_columns
                               (const BtValueGroup * const self,
                                const gulong start_tick,
                                const gulong end_tick);
```

Transposes values from *start_tick* to *end_tick* for all params in single steps.

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	

Since: 0.11

bt_value_group_serialize_column ()

```
void
bt_value_group_serialize_column (const BtValueGroup * const self,
                                 const gulong start_tick,
                                 const gulong end_tick,
                                 const gulong param,
                                 GString *data);
```

Serializes values from *start_tick* to *end_tick* for *param* into *data* .

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
param	the parameter	
data	the target	

Since: 0.7

bt_value_group_serialize_columns ()

```
void
bt_value_group_serialize_columns (const BtValueGroup * const self,
                                  const gulong start_tick,
                                  const gulong end_tick,
                                  GString *data);
```

Serializes values from `start_tick` to `end_tick` for all params into `data` .

Parameters

self	the pattern	
start_tick	the start position for the range	
end_tick	the end position for the range	
data	the target	

Since: 0.7

bt_value_group_set_event ()

```
gboolean
bt_value_group_set_event (const BtValueGroup * const self,
                          const gulong tick,
                          const gulong param,
                          const gchar * const value);
```

Stores the supplied value into the specified pattern cell.

Parameters

self	the pattern the cell belongs to	
tick	the tick (time) position starting with 0	
param	the number of the parameter starting with 0	
value	the string representation of the value to store	

Returns

TRUE for success

Since: 0.7

bt_value_group_test_event ()

```
gboolean
bt_value_group_test_event (const BtValueGroup * const self,
                           const gulong tick,
                           const gulong param);
```

Tests if there is an event in the specified cell.

Parameters

self	the pattern the cell belongs to
tick	the tick (time) position starting with 0
param	the number of the parameter starting with 0

Returns

TRUE if there is an event

Since: 0.7

bt_value_group_test_tick ()

```
gboolean
bt_value_group_test_tick (const BtValueGroup * const self,
                          const gulong tick);
```

Check if there are any event in the given pattern-row.

Parameters

self	the pattern to check
tick	the tick index in the pattern

Returns

TRUE is there are events, FALSE otherwise

Since: 0.7

Types and Values

struct BtValueGroup

```
struct BtValueGroup;
```

A group of parameters, such as used in machines or wires.

Property Details

The "length" property

```
  "length"                  gulong
```

length of the pattern in ticks.

Flags: Read / Write / Construct

The “parameter-group” property

```
“parameter-group”          BtParameterGroup~*
```

Parameter group for the values.

Flags: Read / Write / Construct Only

Signal Details

The “group-changed” signal

```
void
user_function (BtValueGroup     *self,
               BtParameterGroup *param_group,
               gboolean          intermediate,
               gpointer          user_data)
```

Signals that this value-group has been changed (more than in one place). When doing e.g. line inserts, one will receive two updates, one before and one after. The first will have *intermediate* =TRUE. Applications can use that to defer change-consolidation.

Parameters

self	the value-group object that emitted the signal	
param_group	the related BtParameterGroup	
intermediate	flag that is TRUE to signal that more change are coming	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The “param-changed” signal

```
void
user_function (BtValueGroup     *self,
               BtParameterGroup *param_group,
               gulong            tick,
               gulong            param,
               gpointer          user_data)
```

Signals that a param of this value-group has been changed.

Parameters

self	the value-group object that emitted the signal	
param_group	the parameter group	

tick	the tick position inside the pattern	
param	the parameter index	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

3.16 BtWave

BtWave — one BtWavetable entry that keeps a list of BtWavelevels

Functions

gboolean	bt_wave_add_wavelevel ()
BtWavelevel *	bt_wave_get_level_by_index ()
BtWave *	bt_wave_new ()

Properties

guint	channels	Read / Write /
gulong	index	Read / Write /
BtWaveLoopMode	loop-mode	Read / Write /
gchar *	name	Read / Write /
BtSong *	song	Read / Write /
gchar *	uri	Read / Write /
gdouble	volume	Read / Write /
gpointer	wavelevels	Read

Types and Values

struct	BtWave
enum	BtWaveLoopMode

Object Hierarchy

```
    GEnum
    &#x2570;&#x2500;&#x2500; BtWaveLoopMode
    GObject
    &#x2570;&#x2500;&#x2500; BtWave
```

Implemented Interfaces

BtWave implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Represents one instrument. Contains one or more BtWavelevels.

Functions

bt_wave_add_wavelevel ()

```
gboolean
bt_wave_add_wavelevel (const BtWave * const self,
                       const BtWavelevel * const wavelevel);
```

Add the supplied wavelevel to the wave. This is automatically done by bt_wavelevel_new().

Parameters

self	the wavetable to add the new wavelevel to
wavelevel	the new wavelevel instance

Returns

TRUE for success, FALSE otheriwse

bt_wave_get_level_by_index ()

```
BtWavelevel~*
bt_wave_get_level_by_index (const BtWave * const self,
                            const gulong index);
```

Search the wave for a wavelevel by the supplied index. The wavelevel must have been added previously to this wave with bt_wave_add_wavelevel().

Parameters

self	the wave to search for the wavelevel
index	the index of the wavelevel

Returns

BtWavelevel instance or NULL if not found. Unref the wavelevel, when done with it.

[transfer full]

bt_wave_new ()

```
BtWave~*
bt_wave_new (const BtSong * const song,
             const gchar * const name,
             const gchar * const uri,
             const gulong index,
             const gdouble volume,
             const BtWaveLoopMode loop_mode,
             const guint channels);
```

Create a new instance

Parameters

song	the song the new instance belongs to	
name	the display name for the new wave	
uri	the location of the sample data	
index	the list slot for the new wave	
volume	the volume of the wave	
loop_mode	loop playback mode	
channels	number of audio channels	

Returns

the new instance or NULL in case of an error

Types and Values

struct BtWave

```
struct BtWave;
```

A single waveform.

enum BtWaveLoopMode

BtWave clips can be played using several loop modes.

Members

BT_WAVE_LOOP_MODE_OFF	no loop
BT_WAVE_LOOP_MODE_FORWARD	forward loop-ing
BT_WAVE_LOOP_MODE_PINGPONG	forward/backward loop-ing

Property Details

The “channels” property

```
  “channels”                 guint
```

number of channels in the sample.

Flags: Read / Write / Construct

Allowed values: <= 2

Default value: 0

The “**`index`**” property

```
“index”                    gulong
```

The index of the wave in the wavtable.

Flags: Read / Write / Construct

Allowed values: >= 1

The “**`loop-mode`**” property

```
“loop-mode”                BtWaveLoopMode
```

mode of loop playback.

Flags: Read / Write / Construct

Default value: off

The “**`name`**” property

```
“name”                     gchar~*
```

The name of the wave.

Flags: Read / Write / Construct

Default value: "unamed wave"

The “**`song`**” property

```
“song”                     BtSong~*
```

Set song object, the wave belongs to.

Flags: Read / Write / Construct Only

The “**`uri`**” property

```
“uri”                      gchar~*
```

The uri of the wave.

Flags: Read / Write / Construct

Default value: NULL

The “volume” property

```
“volume”                    gdouble
```

The volume of the wave in the wavtable.

Flags: Read / Write / Construct

Allowed values: [0,1]

Default value: 1

The “wavelevels” property

```
“wavelevels”                gpointer
```

A copy of the list of wavelevels.

Flags: Read

3.17 BtWavelevel

BtWavelevel — a single part of a BtWave item

Functions

#define	BT_WAVELEVEL_DEFAULT_ROOT_NOTE
BtWavelevel *	bt_wavelevel_new ()

Properties

gpointer	data	Read / Write
gulong	length	Read / Write
gulong	loop-end	Read / Write
gulong	loop-start	Read / Write
gulong	rate	Read / Write
GstBtNote	root-note	Read / Write /
BtSong *	song	Read / Write /
BtWave *	wave	Read / Write /

Types and Values

struct	BtWavelevel

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtWavelevel
```

Implemented Interfaces

BtWavelevel implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

BtWavelevel contain the digital audio data of a BtWave to be used for a certain key-range.

Functions

BT_WAVELEVEL_DEFAULT_ROOT_NOTE

```
#define BT_WAVELEVEL_DEFAULT_ROOT_NOTE (1+(4*16))
```

Default base-note for a wavelevel (c-3).

bt_wavelevel_new ()

```
BtWavelevel~*
bt_wavelevel_new (const BtSong * const song,
                  const BtWave * const wave,
                  const GstBtNote root_note,
                  const gulong length,
                  const gulong loop_start,
                  const gulong loop_end,
                  const gulong rate,
                  gconstpointer sample);
```

Create a new instance

Parameters

song	the song the new instance belongs to	
wave	the wave the new wavelevel belongs to	
root_note	the keyboard note this sample is related	
length	the number of samples	
loop_start	the start of the loop	
loop_end	the end of the loop	
rate	the sampling rate	
sample	the sample data	

Returns

the new instance or NULL in case of an error.

[transfer full]

Types and Values

struct BtWavelevel

```
struct BtWavelevel;
```

A tone level for a BtWave. In most cases a BtWave has only one BtWavelevel.

Property Details

The "data" property

```
  "data"                     gpointer
```

the sample data.

Flags: Read / Write

The "length" property

```
  "length"                   gulong
```

length of the sample.

Flags: Read / Write

Allowed values: <= G_MAXINT64

The "loop-end" property

```
  "loop-end"                 gulong
```

end of the sample loop.

Flags: Read / Write

The "loop-start" property

```
  "loop-start"               gulong
```

start of the sample loop.

Flags: Read / Write

The "rate" property

```
  "rate"                     gulong
```

sampling rate of the sample.

Flags: Read / Write

The “root-note” property

```
“root-note”                   GstBtNote
```

the base note associated with the sample.

Flags: Read / Write / Construct

Default value: NONE

The “song” property

```
“song”                        BtSong~*
```

Set song object, the wavelevel belongs to.

Flags: Read / Write / Construct Only

The “wave” property

```
“wave”                        BtWave~*
```

Set wave object, the wavelevel belongs to.

Flags: Read / Write / Construct Only

3.18 BtWavetable

BtWavetable — the list of BtWave items in a BtSong

Functions

gboolean	bt_wavetable_add_wave ()
BtWave *	bt_wavetable_get_wave_by_index ()
BtWavetable *	bt_wavetable_new ()
void	bt_wavetable_remember_missing_wave ()
gboolean	bt_wavetable_remove_wave ()

Properties

gpointer	missing-waves	Read
BtSong *	song	Read / Write /
gpointer	waves	Read

Signals

void	wave-added	No Hooks
void	wave-removed	No Hooks

Types and Values

struct	BtWavetable

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtWavetable
```

Implemented Interfaces

BtWavetable implements BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Each wave table entry can consist of multiple BtWaves, were each of the waves has a BtWavelevel with the data for a note range.

The first entry starts at index pos 1. Index 0 is used in a BtPattern to indicate that no (new) wave is referenced.

Functions

bt_wavetable_add_wave ()

```
gboolean
bt_wavetable_add_wave (const BtWavetable * const self,
                       const BtWave * const wave);
```

Add the supplied wave to the wavetable. This is automatically done by bt_wave_new().

Parameters

self	the wavetable to add the wave to
wave	the new wave instance

Returns

TRUE for success, FALSE otheriwse

bt_wavetable_get_wave_by_index ()

```
BtWave~*
bt_wavetable_get_wave_by_index (const BtWavetable * const self,
                                const gulong index);
```

Search the wavetable for a wave by the supplied index. The wave must have been added previously to this wavetable with bt_wavetable_add_wave().

Parameters

self	the wavetable to search for the wave
index	the index of the wave

Returns

BtWave instance or NULL if not found. Unref the wave, when done with it.

[transfer full]

bt_wavetable_new ()

```
BtWavetable~*
bt_wavetable_new (const BtSong * const song);
```

Create a new instance

Parameters

song	the song the new instance belongs to

Returns

the new instance or NULL in case of an error

bt_wavetable_remember_missing_wave ()

```
void
bt_wavetable_remember_missing_wave (const BtWavetable * const self,
                                    const gchar * const str);
```

Loaders can use this function to collect information about wavetable entries that failed to load. The front-end can access this later by reading BtWavetable::missing-waves property.

Parameters

self	the wavetable
str	human readable description of the missing wave

bt_wavetable_remove_wave ()

```
gboolean
bt_wavetable_remove_wave (const BtWavetable * const self,
                          const BtWave * const wave);
```

Remove the supplied wave from the wavetable.

Parameters

self	the wavetable to remove the wave from
wave	the wave instance

Returns

TRUE for success, FALSE otheriwse

Types and Values

struct BtWavetable

```
struct BtWavetable;
```

A table of BtWave objects.

Property Details

The “missing-waves” property

```
  “missing-waves”           gpointer
```

The list of missing waves, don’t change.

Flags: Read

The “song” property

```
  “song”                    BtSong~*
```

Set song object, the wavetable belongs to.

Flags: Read / Write / Construct Only

The “waves” property

```
  “waves”                   gpointer
```

A copy of the list of waves.

Flags: Read

Signal Details

The “wave-added” signal

```
void
user_function (BtWavetable *self,
               BtWave      *wave,
               gpointer     user_data)
```

A new wave item has been added to the wavetable

Parameters

self	the wavetable object that emitted the signal	
wave	the new wave	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

The "wave-removed" signal

```
void
user_function (BtWavetable *self,
               BtWave       *wave,
               gpointer      user_data)
```

A wave item has been removed from the wavetable

Parameters

self	the setup object that emitted the signal	
wave	the old wave	
user_data	user data set when the signal handler was connected.	

Flags: No Hooks

3.19 BtWire

BtWire — class for a connection of two BtMachines

Functions

BtParameterGroup *	bt_wire_get_param_group ()
BtWire *	bt_wire_new ()
gboolean	bt_wire_reconnect ()
gboolean	bt_wire_can_link ()

Properties

gpointer	analyzers	Read / Write
gpointer	construction-error	Read / Write /
BtMachine *	dst	Read / Write /
GstElement *	gain	Read
gulong	num-params	Read / Write
GstElement *	pan	Read
gchar *	pretty-name	Read
gpointer	properties	Read

BtSong *	song	Read / Write /
BtMachine *	src	Read / Write /

Types and Values

#define	BT_WIRE_MAX_NUM_PARAMS
struct	BtWire

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; GInitiallyUnowned
        &#x2570;&#x2500;&#x2500; GstObject
            &#x2570;&#x2500;&#x2500; GstElement
                &#x2570;&#x2500;&#x2500; GstBin
                    &#x2570;&#x2500;&#x2500; BtWire
```

Implemented Interfaces

BtWire implements GstChildProxy and BtPersistence.

Includes

```
#include <libbtcore/core.h>
```

Description

Abstracts connection between two BtMachines. After creation, the elements are connected. In contrast to directly wiring GstElements this insert needed conversion elements automatically.

Furthermore each wire has a volume and if possible panorama/balance element. Volume and panorama/balance can be sequenced like machine parameters in wire groups of the BtPattern objects on the target machine (that means that source-machines don't have the controls).

Functions

bt_wire_get_param_group ()

```
BtParameterGroup~*
bt_wire_get_param_group (const BtWire * const self);
```

Get the parameter group.

Parameters

self	the machine

Returns

the BtParameterGroup or NULL.

[transfer none]

bt_wire_new ()

```
BtWire~*
bt_wire_new (const BtSong *song,
             const BtMachine *src_machine,
             const BtMachine *dst_machine,
             GError **err);
```

Create a new instance. The new wire is automatically added to a songs setup. You don't need to call `bt_setup_add_wire(setup,wire);`.

Parameters

song	the song the new instance belongs to	
src_machine	the data source (BtSourceMachine or BtProcessorMachine)	
dst_machine	the data sink (BtSinkMachine or BtProcessorMachine)	
err	inform about failed instance creation	

Returns

the new instance or NULL in case of an error

bt_wire_reconnect ()

```
gboolean
bt_wire_reconnect (BtWire *self);
```

Call this method after internal elements in a BtMachine have changed, but failed to link.

Parameters

self	the wire to re-link	

Returns

TRUE for success and FALSE otherwise

bt_wire_can_link ()

```
gboolean
bt_wire_can_link (const BtMachine * const src,
                  const BtMachine * const dst);
```

Check if we don't have such a wire yet and if we can connect the machines. We can connect if the data flow direction is correct (sources to effects to sinks) and if the new connect is not creating cycles in the graph.

Parameters

src	the src machine	
dst	the dst machine	

Returns

TRUE if we can link the machines

Types and Values

BT_WIRE_MAX_NUM_PARAMS

```
#define BT_WIRE_MAX_NUM_PARAMS 2
```

Maximum number of parameters per wire.

struct BtWire

```
struct BtWire;
```

A link between two BtMachine instances.

Property Details

The “analyzers” property

```
  “analyzers”                gpointer
```

list of wire analyzers.

Flags: Read / Write

The “construction-error” property

```
  “construction-error”       gpointer
```

signal failed instance creation.

Flags: Read / Write / Construct Only

The “dst” property

```
  “dst”                      BtMachine~*
```

dst machine object, the wire links to.

Flags: Read / Write / Construct Only

The “gain” property

```
“gain”                    GstElement~*
```

the gain element for the connection.

Flags: Read

The “num-params” property

```
“num-params”              gulong
```

number of params for the wire.

Flags: Read / Write

Allowed values: <= 2

The “pan” property

```
“pan”                     GstElement~*
```

the panorama element for the connection.

Flags: Read

The “pretty-name” property

```
“pretty-name”             gchar~*
```

pretty-printed name for display purposes.

Flags: Read

Default value: NULL

The “properties” property

```
“properties”              gpointer
```

list of wire properties.

Flags: Read

The “song” property

```
“song”                    BtSong~*
```

the song object, the wire belongs to.

Flags: Read / Write / Construct Only

The “src” property

```
“src”                     BtMachine~*
```

src machine object, the wire links to.

Flags: Read / Write / Construct Only

Chapter 4

Song IO Reference

4.1 BtSongIO

BtSongIO — base class for song input and output

Functions

gboolean	(*BtSongIOInit) ()
BtSongIO *	bt_song_io_from_data ()
BtSongIO *	bt_song_io_from_file ()
const GList *	bt_song_io_get_module_info_list ()
gboolean	bt_song_io_load ()
gboolean	bt_song_io_save ()
gboolean	(*bt_song_io_virtual_load) ()
gboolean	(*bt_song_io_virtual_save) ()

Properties

gpointer	data	Read / Write
guint	data-len	Read / Write
gchar *	file-name	Read
gchar *	status	Read / Write

Types and Values

#define	BT_SONG_IO_ERROR
#define	BT_SONG_IO_MODULE_INFO_MAX_FORMATS
struct	BtSongIO
struct	BtSongIOClass
enum	BtSongIOError
	BtSongIOFormatInfo
	BtSongIOModuleInfo

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSongIO
        &#x251c;&#x2500;&#x2500; BtSongIOBuzz
        &#x2570;&#x2500;&#x2500; BtSongIONative
```

Includes

```
#include <libbtcore/core.h>
```

Description

A base class for BtSong loader and saver implementations. A BtSongIO module needs to be installed as a shared library into LIBDIR/songio. It is recognized, if it exports a BtSongIOModuleInfo structure. At runtime the detect method of each module is called with the chosen file-name. The module should return its GType if it can handle the format or NULL else.

Such a module should overwrite the bt_song_io_load() and/or bt_song_io_save() default implementations.

There is an internal subclass of this called BtSongIONative.

Note
This API is not yet fully stable. Please discuss with the deverloper team if you intend to write a io plugin.

Functions

BtSongIOInit ()

```
gboolean
(*BtSongIOInit) (void);
```

Function to init the plugin.

Returns

TRUE if the plugin was initialized fine

bt_song_io_from_data ()

```
BtSongIO~*
bt_song_io_from_data (gpointer data,
                      guint len,
                      const gchar *media_type,
                      GError **err);
```

Create a new instance from the given parameters. Each installed plugin will test if it can handle the file type.

Parameters

data	in memory data of the song	
len	the siye of the `data` block	
media_type	the media-type of the song, if available	
err	where to store the error message in case of an error, or NULL	

Returns

the new instance or NULL in case of an error.

[transfer full]

bt_song_io_from_file ()

```
BtSongIO~*
bt_song_io_from_file (const gchar * const file_name,
                      GError **err);
```

Create a new instance from the given `file_name` . Each installed plugin will test if it can handle the file type.

Parameters

file_name	the file name of the song	
err	where to store the error message in case of an error, or NULL	

Returns

the new instance or NULL in case of an error.

[transfer full]

bt_song_io_get_module_info_list ()

```
const GList~*
bt_song_io_get_module_info_list (void);
```

Get read only access to list of BtSongIOModuleInfo entries.

Returns

the GList.

[element-type BuzztraxCore.SongIOModuleInfo][transfer none]

bt_song_io_load ()

```
gboolean
bt_song_io_load (BtSongIO const *self,
                 const BtSong * const song,
                 GError **err);
```

load the song from a file. The file is set in the constructor

Parameters

self	the BtSongIO instance to use	

song	the BtSong instance that should initialized	
err	where to store the error message in case of an error, or NULL	

Returns

TRUE for success

bt_song_io_save ()

```
gboolean
bt_song_io_save (BtSongIO const *self,
                 const BtSong * const song,
                 GError **err);
```

save the song to a file. The file is set in the constructor

Parameters

self	the BtSongIO instance to use	
song	the BtSong instance that should stored	
err	where to store the error message in case of an error, or NULL	

Returns

TRUE for success

bt_song_io_virtual_load ()

```
gboolean
(*bt_song_io_virtual_load) (gconstpointer self,
                            const BtSong * const song,
                            GError **err);
```

Subclasses will override this methods with the loader function.

Parameters

self	song-io instance	
song	song object to load	
err	where to store the error message in case of an error, or NULL	

Returns

TRUE for success

bt_song_io_virtual_save ()

```
gboolean
(*bt_song_io_virtual_save) (gconstpointer const self,
                            const BtSong * const song,
                            GError **err);
```

Subclasses will override this methods with the saver function.

Parameters

self	song-io instance	
song	song object to save	
err	where to store the error message in case of an error, or NULL	

Returns

TRUE for success

Types and Values

BT_SONG_IO_ERROR

```
#define BT_SONG_IO_ERROR      bt_song_io_error_quark ()
```

Error domain for the song-io subsystem. Errors in this domain will be from the BtSongIOError enumeration. See GError for information on error domains.

BT_SONG_IO_MODULE_INFO_MAX_FORMATS

```
#define BT_SONG_IO_MODULE_INFO_MAX_FORMATS 10
```

Maximum number of BtSongIOFormatInfo per plugin (10).

struct BtSongIO

```
struct BtSongIO;
```

base object for song input and output plugins

struct BtSongIOClass

```
struct BtSongIOClass {
  const GObjectClass parent;

  /* class methods */
  bt_song_io_virtual_load load;
  bt_song_io_virtual_save save;
};
```

Base class for song input and output plugins

Members

const GObjectClass *parent*;	parent class type
bt_song_io_virtual_load *load*;	virtual method for loading a song
bt_song_io_virtual_save *save*;	virtual method for saving a song

enum BtSongIOError

Error codes returned by the song-io subsystem in additions to GIOErrorEnum.

Members

BT_SONG_IO_ERROR_UNKNOWN_FORMAT	file is not in one of the supported formats
BT_SONG_IO_ERROR_UNSUPPORTED_METHOD	operation is not supported for this file type
BT_SONG_IO_ERROR_INVALID_FORMAT	file has structural errors

BtSongIOFormatInfo

```
typedef struct {
  GType type;
  const gchar *name;
  const gchar *mime_type;
  const gchar *extension;
} BtSongIOFormatInfo;
```

Metadata structure for BtSongIO plugins describing one format.

Members

GType *type*;	the io module GType
const gchar **name*;	format name
const gchar **mime_type*;	mime type
const gchar **extension*;	file extension

BtSongIOModuleInfo

```
typedef struct {
  BtSongIOInit init;
  BtSongIOFormatInfo formats[BT_SONG_IO_MODULE_INFO_MAX_FORMATS];
} BtSongIOModuleInfo;
```

Metadata structure for BtSongIO plugins.

Members

BtSongIOInit *init*;	pointer to init function, can be NULL.

BtSongIOFormatInfo *formats*[BT_SONG_IO_MODULE_INFO_MAX_FORMATS];	NULL terminated array of formats supported by this plugin

Property Details

The "data" property

```
"data"                    gpointer
```

in memory block pointer for load/save operations.

Flags: Read / Write

The "data-len" property

```
"data-len"                guint
```

in memory block length for load/save operations.

Flags: Read / Write

Default value: 0

The "file-name" property

```
"file-name"               gchar~*
```

full filename for load/save operations.

Flags: Read

Default value: NULL

The "status" property

```
"status"                  gchar~*
```

status of load/save operations.

Flags: Read / Write

Default value: NULL

4.2 BtSongIONative

BtSongIONative — class for song input and output in builtin native format

Types and Values

struct	BtSongIONative
struct	BtSongIONativeClass
extern BtSongIOModuleInfo	bt_song_io_native_module_info

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSongIO
        &#x2570;&#x2500;&#x2500; BtSongIONative
            &#x251c;&#x2500;&#x2500; BtSongIONativeBZT
            &#x2570;&#x2500;&#x2500; BtSongIONativeXML
```

Includes

```
#include <libbtcore/core.h>
```

Description

Buzztrax stores songs in an own file-format. This internal io-module implements loading and saving of this format. The format is an archive, that contains an XML file and optionally binary data, such as audio samples.

Functions

Types and Values

struct BtSongIONative

```
struct BtSongIONative;
```

object for song input and output in native zip/xml format

struct BtSongIONativeClass

```
struct BtSongIONativeClass {
  const BtSongIOClass parent;
};
```

Class for song input and output in native zip/xml format

Members

const BtSongIOClass *parent*;	parent class type

bt_song_io_native_module_info

```
extern BtSongIOModuleInfo bt_song_io_native_module_info;
```

Buzztrax native format song loader/saver metadata.

4.3 BtSongIONativeBZT

BtSongIONativeBZT — class for song input and output in builtin native format

Functions

gboolean	bt_song_io_native_bzt_copy_from_uri ()
gboolean	bt_song_io_native_bzt_copy_to_fd ()

Types and Values

struct	BtSongIONativeBZT
struct	BtSongIONativeBZTClass

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSongIO
        &#x2570;&#x2500;&#x2500; BtSongIONative
            &#x2570;&#x2500;&#x2500; BtSongIONativeBZT
```

Includes

```
#include <libbtcore/core.h>
```

Description

This internal BtSongIONative module implements loading and saving of an own xml format with externals. The format is an archive, that contains an XML file and optionally binary data, such as audio samples.

Functions

bt_song_io_native_bzt_copy_from_uri ()

```
gboolean
bt_song_io_native_bzt_copy_from_uri (const BtSongIONativeBZT * const self,
                                     const gchar *file_name,
                                     const gchar *uri);
```

Copies the file specified by *uri* to *file_name* into the song file.

This is a helper for BtSong persistence.

Parameters

self	the song-plugin	
file_name	the path to the file inside the song	
uri	location of the source file	

Returns

TRUE on success

bt_song_io_native_bzt_copy_to_fd ()

```
gboolean
bt_song_io_native_bzt_copy_to_fd (const BtSongIONativeBZT * const self,
                                  const gchar *file_name,
                                  gint fd);
```

Copies the file specified by *file_name* from the song file to the *fd* .

This is a helper for BtSong persistence.

Parameters

self	the song-plugin	
file_name	the path to the file inside the song	
fd	a file-descriptor of an opened file to copy *file_name* to	

Returns

TRUE on success

Types and Values

struct BtSongIONativeBZT

```
struct BtSongIONativeBZT;
```

object for song input and output in native zip/xml format

struct BtSongIONativeBZTClass

```
struct BtSongIONativeBZTClass {
  const BtSongIONativeClass parent;
};
```

Class for song input and output in native zip/xml format

Members

const BtSongIONativeClass *parent*;	parent class type

4.4 BtSongIONativeXML

BtSongIONativeXML — class for song input and output in builtin native format

Types and Values

struct	BtSongIONativeXML
struct	BtSongIONativeXMLClass

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSongIO
        &#x2570;&#x2500;&#x2500; BtSongIONative
            &#x2570;&#x2500;&#x2500; BtSongIONativeXML
```

Includes

```
#include <libbtcore/core.h>
```

Description

This internal BtSongIONative module implements loading and saving of an own xml format without externals.

Functions

Types and Values

struct BtSongIONativeXML

```
struct BtSongIONativeXML;
```

object for song input and output in native zip/xml format

struct BtSongIONativeXMLClass

```
struct BtSongIONativeXMLClass {
  const BtSongIONativeClass parent;
};
```

Class for song input and output in native zip/xml format

Members

const BtSongIONativeClass *parent*;	parent class type

4.5 BtSongIOBuzz

BtSongIOBuzz — class for song input in buzz bmx and bmw format

Types and Values

struct	BtSongIOBuzz

Object Hierarchy

```
    GObject
    &#x2570;&#x2500;&#x2500; BtSongIO
        &#x2570;&#x2500;&#x2500; BtSongIOBuzz
```

Includes

```
#include <libbtcore/core.h>
```

Description

This BtSongIO plugin implements loading and of songs made using Buzz. Both songs with and without embedded waveforms are supported. Most aspects of the file-format are implemented.

Functions

Types and Values

struct BtSongIOBuzz

```
struct BtSongIOBuzz;
```

object for song input and output in buzz zip/xml format

Part III

Appendix

Chapter 5

Object Hierarchy

```
    GObject
    &#x251c;&#x2500;&#x2500; GInitiallyUnowned
    &#x2502;   &#x2570;&#x2500;&#x2500; GstObject
    &#x2502;       &#x251c;&#x2500;&#x2500; GstElement
    &#x2502;       &#x2502;   &#x2570;&#x2500;&#x2500; GstBin
    &#x2502;       &#x2502;       &#x251c;&#x2500;&#x2500; BtMachine
    &#x2502;       &#x2502;       &#x2502;   &#x251c;&#x2500;&#x2500; BtProcessorMachine
    &#x2502;       &#x2502;       &#x2502;   &#x251c;&#x2500;&#x2500; BtSinkMachine
    &#x2502;       &#x2502;       &#x2502;   &#x2570;&#x2500;&#x2500; BtSourceMachine
    &#x2502;       &#x2502;       &#x251c;&#x2500;&#x2500; BtSinkBin
    &#x2502;       &#x2502;       &#x2570;&#x2500;&#x2500; BtWire
    &#x2502;       &#x2570;&#x2500;&#x2500; GstControlBinding
    &#x2502;           &#x251c;&#x2500;&#x2500; BtCmdPatternControlSource
    &#x2502;           &#x2570;&#x2500;&#x2500; BtPatternControlSource
    &#x251c;&#x2500;&#x2500; BtApplication
    &#x251c;&#x2500;&#x2500; BtAudioSession
    &#x251c;&#x2500;&#x2500; BtCmdPattern
    &#x2502;   &#x2570;&#x2500;&#x2500; BtPattern
    &#x251c;&#x2500;&#x2500; BtParameterGroup
    &#x251c;&#x2500;&#x2500; BtSequence
    &#x251c;&#x2500;&#x2500; BtSettings
    &#x251c;&#x2500;&#x2500; BtSetup
    &#x251c;&#x2500;&#x2500; BtSong
    &#x251c;&#x2500;&#x2500; BtSongInfo
    &#x251c;&#x2500;&#x2500; BtSongIO
    &#x2502;   &#x251c;&#x2500;&#x2500; BtSongIOBuzz
    &#x2502;   &#x2570;&#x2500;&#x2500; BtSongIONative
    &#x2502;       &#x251c;&#x2500;&#x2500; BtSongIONativeBZT
    &#x2502;       &#x2570;&#x2500;&#x2500; BtSongIONativeXML
    &#x251c;&#x2500;&#x2500; BtValueGroup
    &#x251c;&#x2500;&#x2500; BtWave
    &#x251c;&#x2500;&#x2500; BtWavelevel
    &#x2570;&#x2500;&#x2500; BtWavetable
    GInterface
    &#x251c;&#x2500;&#x2500; BtChildProxy
    &#x2570;&#x2500;&#x2500; BtPersistence
    GEnum
    &#x251c;&#x2500;&#x2500; BtSinkBinMode
    &#x251c;&#x2500;&#x2500; BtSinkBinRecordFormat
    &#x251c;&#x2500;&#x2500; BtPatternCmd
    &#x251c;&#x2500;&#x2500; BtMachineState
    &#x2570;&#x2500;&#x2500; BtWaveLoopMode
```

Chapter 6

Annotation Glossary

A

allow-none
NULL is OK, both for passing and for returning.

array
Parameter points to an array of items.

E

element-type
Generics and defining elements of containers and arrays.

I

inout
Parameter for input and for returning results. Default is transfer full.

O

out
Parameter for returning results. Default is transfer full.

S

scope async
The callback is valid until first called.

skip
Exposed in C code, not necessarily available in other languages.

T

transfer container

Free data container after the code is done.

transfer full

Free data after the code is done.

transfer none

Don’t free data after the code is done.

type

Override the parsed C type with given type.

Chapter 7

Index