

gconfmm

Generated by Doxygen 1.9.2



<b>1 gconfmm Reference Manual</b>	<b>1</b>
1.1 Description	1
1.2 Basic Usage	1
<b>2 Module Index</b>	<b>3</b>
2.1 Modules	3
<b>3 Namespace Index</b>	<b>5</b>
3.1 Namespace List	5
<b>4 Hierarchical Index</b>	<b>7</b>
4.1 Class Hierarchy	7
<b>5 Class Index</b>	<b>9</b>
5.1 Class List	9
<b>6 Module Documentation</b>	<b>11</b>
6.1 gconfmm Enums and Flags	11
6.1.1 Detailed Description	11
6.1.2 Enumeration Type Documentation	11
6.1.2.1 ClientErrorHandlingMode	11
6.1.2.2 ClientPreloadType	12
6.1.2.3 UnsetFlags	12
6.1.2.4 ValueType	12
<b>7 Namespace Documentation</b>	<b>13</b>
7.1 Glib Namespace Reference	13
7.2 Gnome Namespace Reference	13
7.3 Gnome::Conf Namespace Reference	13
7.3.1 Typedef Documentation	14
7.3.1.1 Callback	14
7.3.1.2 ValuePair	14
7.3.1.3 ValueTypePair	14
7.3.2 Function Documentation	14
7.3.2.1 init()	14
<b>8 Class Documentation</b>	<b>15</b>
8.1 Gnome::Conf::ChangeSet Class Reference	15
8.1.1 Detailed Description	16
8.1.2 Member Typedef Documentation	17
8.1.2.1 ForeachSlot	17
8.1.3 Constructor & Destructor Documentation	17
8.1.3.1 ChangeSet() [1/3]	17
8.1.3.2 ChangeSet() [2/3]	17
8.1.3.3 ChangeSet() [3/3]	17

8.1.3.4 ~ChangeSet()	17
8.1.4 Member Function Documentation	17
8.1.4.1 clear()	18
8.1.4.2 exists()	18
8.1.4.3 for_each()	18
8.1.4.4 gobj() [1/2]	18
8.1.4.5 gobj() [2/2]	18
8.1.4.6 gobj_copy()	18
8.1.4.7 operator=()	19
8.1.4.8 remove()	19
8.1.4.9 set() [1/6]	19
8.1.4.10 set() [2/6]	19
8.1.4.11 set() [3/6]	19
8.1.4.12 set() [4/6]	20
8.1.4.13 set() [5/6]	20
8.1.4.14 set() [6/6]	20
8.1.4.15 size()	20
8.1.4.16 unset()	20
8.1.5 Member Data Documentation	20
8.1.5.1 gobject_	21
8.2 Gnome::Conf::Client Class Reference	21
8.2.1 Detailed Description	24
8.2.2 Member Typedef Documentation	24
8.2.2.1 SListHandleBools	24
8.2.2.2 SListHandleFloats	24
8.2.2.3 SListHandleInts	25
8.2.3 Constructor & Destructor Documentation	25
8.2.3.1 ~Client()	25
8.2.4 Member Function Documentation	25
8.2.4.1 add_dir()	25
8.2.4.2 all_dirs()	25
8.2.4.3 all_entries()	26
8.2.4.4 change_set_commit()	26
8.2.4.5 change_set_from_current()	27
8.2.4.6 change_set_reverse()	27
8.2.4.7 clear_cache()	28
8.2.4.8 dir_exists()	28
8.2.4.9 error()	29
8.2.4.10 get()	29
8.2.4.11 get_bool()	29
8.2.4.12 get_bool_list()	29
8.2.4.13 get_client_for_engine()	29

8.2.4.14 <code>get_default_client()</code> . . . . .	30
8.2.4.15 <code>get_default_from_schema()</code> . . . . .	30
8.2.4.16 <code>get_entry()</code> [1/2] . . . . .	30
8.2.4.17 <code>get_entry()</code> [2/2] . . . . .	31
8.2.4.18 <code>get_float()</code> . . . . .	31
8.2.4.19 <code>get_float_list()</code> . . . . .	32
8.2.4.20 <code>get_int()</code> . . . . .	32
8.2.4.21 <code>get_int_list()</code> . . . . .	32
8.2.4.22 <code>get_pair()</code> . . . . .	33
8.2.4.23 <code>get_schema()</code> . . . . .	33
8.2.4.24 <code>get_schema_list()</code> . . . . .	33
8.2.4.25 <code>get_string()</code> . . . . .	34
8.2.4.26 <code>get_string_list()</code> . . . . .	34
8.2.4.27 <code>get_without_default()</code> . . . . .	34
8.2.4.28 <code>gobj()</code> [1/2] . . . . .	34
8.2.4.29 <code>gobj()</code> [2/2] . . . . .	35
8.2.4.30 <code>gobj_copy()</code> . . . . .	35
8.2.4.31 <code>key_is_writable()</code> . . . . .	35
8.2.4.32 <code>notify()</code> . . . . .	35
8.2.4.33 <code>notify_add()</code> . . . . .	36
8.2.4.34 <code>notify_remove()</code> . . . . .	36
8.2.4.35 <code>on_error()</code> . . . . .	36
8.2.4.36 <code>on_unreturned_error()</code> . . . . .	37
8.2.4.37 <code>on_value_changed()</code> . . . . .	37
8.2.4.38 <code>preload()</code> . . . . .	37
8.2.4.39 <code>recursive_unset()</code> . . . . .	37
8.2.4.40 <code>remove_dir()</code> . . . . .	38
8.2.4.41 <code>set()</code> [1/6] . . . . .	38
8.2.4.42 <code>set()</code> [2/6] . . . . .	38
8.2.4.43 <code>set()</code> [3/6] . . . . .	39
8.2.4.44 <code>set()</code> [4/6] . . . . .	39
8.2.4.45 <code>set()</code> [5/6] . . . . .	40
8.2.4.46 <code>set()</code> [6/6] . . . . .	40
8.2.4.47 <code>set_bool_list()</code> . . . . .	41
8.2.4.48 <code>set_error_handling()</code> . . . . .	41
8.2.4.49 <code>set_float_list()</code> . . . . .	41
8.2.4.50 <code>set_int_list()</code> . . . . .	41
8.2.4.51 <code>set_schema_list()</code> . . . . .	41
8.2.4.52 <code>set_string_list()</code> . . . . .	41
8.2.4.53 <code>signal_error()</code> . . . . .	42
8.2.4.54 <code>signal_value_changed()</code> . . . . .	42
8.2.4.55 <code>suggest_sync()</code> . . . . .	42

8.2.4.56 unset()	42
8.2.4.57 value_changed()	43
8.2.5 Friends And Related Function Documentation	43
8.2.5.1 wrap()	43
8.3 Gnome::Conf::Entry Class Reference	43
8.3.1 Detailed Description	45
8.3.2 Constructor & Destructor Documentation	45
8.3.2.1 Entry() [1/4]	45
8.3.2.2 Entry() [2/4]	45
8.3.2.3 Entry() [3/4]	45
8.3.2.4 ~Entry()	45
8.3.2.5 Entry() [4/4]	46
8.3.3 Member Function Documentation	46
8.3.3.1 get_is_default()	46
8.3.3.2 get_is_writable()	46
8.3.3.3 get_key()	46
8.3.3.4 get_schema_name()	46
8.3.3.5 get_value()	46
8.3.3.6 gobj() [1/2]	47
8.3.3.7 gobj() [2/2]	47
8.3.3.8 gobj_copy()	47
8.3.3.9 operator=()	47
8.3.3.10 set_is_default()	47
8.3.3.11 set_is_writable()	47
8.3.3.12 set_schema_name()	48
8.3.3.13 set_value()	48
8.3.4 Friends And Related Function Documentation	48
8.3.4.1 wrap()	48
8.3.5 Member Data Documentation	48
8.3.5.1 gobject_	48
8.4 Gnome::Conf::Error Class Reference	49
8.4.1 Detailed Description	50
8.4.2 Member Enumeration Documentation	50
8.4.2.1 Code	50
8.4.3 Constructor & Destructor Documentation	50
8.4.3.1 Error() [1/2]	50
8.4.3.2 Error() [2/2]	51
8.4.4 Member Function Documentation	51
8.4.4.1 code()	51
8.5 Gnome::Conf::Schema Class Reference	51
8.5.1 Constructor & Destructor Documentation	52
8.5.1.1 Schema() [1/3]	52

8.5.1.2 Schema() [2/3]	53
8.5.1.3 Schema() [3/3]	53
8.5.1.4 ~Schema()	53
8.5.2 Member Function Documentation	53
8.5.2.1 get_car_type()	53
8.5.2.2 get_cdr_type()	53
8.5.2.3 get_default_value()	53
8.5.2.4 get_list_type()	53
8.5.2.5 get_locale()	54
8.5.2.6 get_long_desc()	54
8.5.2.7 get_owner()	54
8.5.2.8 get_short_desc()	54
8.5.2.9 get_type()	54
8.5.2.10 gobj() [1/2]	54
8.5.2.11 gobj() [2/2]	54
8.5.2.12 gobj_copy()	55
8.5.2.13 operator=()	55
8.5.2.14 set_car_type()	55
8.5.2.15 set_cdr_type()	55
8.5.2.16 set_default_value()	55
8.5.2.17 set_list_type()	55
8.5.2.18 set_locale()	56
8.5.2.19 set_long_desc()	56
8.5.2.20 set_owner()	56
8.5.2.21 set_short_desc()	56
8.5.2.22 set_type()	56
8.5.3 Friends And Related Function Documentation	56
8.5.3.1 wrap()	56
8.5.4 Member Data Documentation	57
8.5.4.1 gobject_	57
8.6 Gnome::Conf::SetInterface Class Reference	57
8.6.1 Detailed Description	58
8.6.2 Member Function Documentation	58
8.6.2.1 set() [1/7]	58
8.6.2.2 set() [2/7]	58
8.6.2.3 set() [3/7]	59
8.6.2.4 set() [4/7]	59
8.6.2.5 set() [5/7]	59
8.6.2.6 set() [6/7]	59
8.6.2.7 set() [7/7]	59
8.6.2.8 set_bool_list()	60
8.6.2.9 set_float_list()	60

8.6.2.10 set_int_list()	60
8.6.2.11 set_schema_list()	60
8.6.2.12 set_string_list()	60
8.7 Gnome::Conf::Value Class Reference	61
8.7.1 Detailed Description	63
8.7.2 Constructor & Destructor Documentation	63
8.7.2.1 Value() [1/3]	63
8.7.2.2 Value() [2/3]	63
8.7.2.3 ~Value()	63
8.7.2.4 Value() [3/3]	63
8.7.3 Member Function Documentation	64
8.7.3.1 get_bool()	64
8.7.3.2 get_bool_list()	64
8.7.3.3 get_car()	64
8.7.3.4 get_cdr()	64
8.7.3.5 get_float()	65
8.7.3.6 get_float_list()	65
8.7.3.7 get_int()	65
8.7.3.8 get_int_list()	65
8.7.3.9 get_list_type()	66
8.7.3.10 get_schema()	66
8.7.3.11 get_schema_list()	66
8.7.3.12 get_string()	66
8.7.3.13 get_string_list()	66
8.7.3.14 get_type()	67
8.7.3.15 gobj() [1/2]	67
8.7.3.16 gobj() [2/2]	67
8.7.3.17 gobj_copy()	67
8.7.3.18 operator=()	67
8.7.3.19 set() [1/5]	67
8.7.3.20 set() [2/5]	68
8.7.3.21 set() [3/5]	68
8.7.3.22 set() [4/5]	68
8.7.3.23 set() [5/5]	68
8.7.3.24 set_bool_list()	68
8.7.3.25 set_car()	69
8.7.3.26 set_cdr()	69
8.7.3.27 set_float_list()	69
8.7.3.28 set_int_list()	69
8.7.3.29 set_list_type()	69
8.7.3.30 set_schema_list()	70
8.7.3.31 set_string_list()	70



---

8.7.3.32 to_string() . . . . .	70
8.7.4 Friends And Related Function Documentation . . . . .	70
8.7.4.1 wrap() . . . . .	70
8.7.5 Member Data Documentation . . . . .	71
8.7.5.1 gobject_ . . . . .	71
<b>Index</b>	<b>73</b>



# Chapter 1

## gconfmm Reference Manual

### 1.1 Description

gconfmm is the official C++ interface for the GConf client API for storing and retrieving configuration data. See [Gnome::Conf::Client](#).

### 1.2 Basic Usage

Include the gconfmm header:

```
#include <gconfmm.h>
```

(You may include individual headers, such as `gconfmm/client.h` instead.)

If your source file is `program.cc`, you can compile it with:

```
g++ program.cc -o program `pkg-config --cflags --libs gconfmm-2.6`
```

Alternatively, if using autoconf, use the following in `configure.ac`:

```
PKG_CHECK_MODULES([GCONFMM], [gconfmm-2.4])
```

Then use the generated `GCONFMM_CFLAGS` and `GCONFMM_LIBS` variables in the project `Makefile.am` files. For example:

```
program_CPPFLAGS = $(GCONFMM_CFLAGS)
program_LDADD = $(GCONFMM_LIBS)
```



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

gconfmm Enums and Flags . . . . .	11
-----------------------------------	----



## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">Glib</a>	.....	<a href="#">13</a>
<a href="#">Gnome</a>	.....	<a href="#">13</a>
<a href="#">Gnome::Conf</a>	.....	<a href="#">13</a>





## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Glib::Error	
Gnome::Conf::Error . . . . .	49
Glib::Object	
Gnome::Conf::Client . . . . .	21
Gnome::Conf::Entry . . . . .	43
Gnome::Conf::Schema . . . . .	51
Gnome::Conf::SetInterface . . . . .	57
Gnome::Conf::ChangeSet . . . . .	15
Gnome::Conf::Client . . . . .	21
Gnome::Conf::Value . . . . .	61



## Chapter 5

# Class Index

### 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Gnome::Conf::ChangeSet</a>	
A <a href="#">ChangeSet</a> is a set of changes to the GConf database that can be committed and reversed easily . . . . .	15
<a href="#">Gnome::Conf::Client</a>	
The main <a href="#">Gnome::Conf</a> object . . . . .	21
<a href="#">Gnome::Conf::Entry</a>	
An <a href="#">Entry</a> stores an entry from a GConf "directory", including a key-value pair, the name of the <a href="#">Schema</a> applicable to this entry, whether the value is a default value, and whether GConf can write a new value at this key . . . . .	43
<a href="#">Gnome::Conf::Error</a>	
Exception class for <a href="#">Gnome::Conf::Client</a> errors . . . . .	49
<a href="#">Gnome::Conf::Schema</a> . . . . .	51
<a href="#">Gnome::Conf::SetInterface</a>	
Common Interface for key-value settable objects . . . . .	57
<a href="#">Gnome::Conf::Value</a>	
Wrapper for primitive types . . . . .	61



## Chapter 6

# Module Documentation

### 6.1 gconfmm Enums and Flags

#### Enumerations

- enum `Gnome::Conf::ClientErrorHandlingMode` {  
    `Gnome::Conf::CLIENT_HANDLE_NONE` ,  
    `Gnome::Conf::CLIENT_HANDLE_UNRETURNED` ,  
    `Gnome::Conf::CLIENT_HANDLE_ALL` }
- enum `Gnome::Conf::ClientPreloadType` {  
    `Gnome::Conf::CLIENT_PRELOAD_NONE` ,  
    `Gnome::Conf::CLIENT_PRELOAD_ONELEVEL` ,  
    `Gnome::Conf::CLIENT_PRELOAD_RECURSIVE` }
- enum `Gnome::Conf::ValueType` {  
    `Gnome::Conf::VALUE_INVALID` ,  
    `Gnome::Conf::VALUE_STRING` ,  
    `Gnome::Conf::VALUE_INT` ,  
    `Gnome::Conf::VALUE_FLOAT` ,  
    `Gnome::Conf::VALUE_BOOL` ,  
    `Gnome::Conf::VALUE_SCHEMA` ,  
    `Gnome::Conf::VALUE_LIST` ,  
    `Gnome::Conf::VALUE_PAIR` }
- enum `Gnome::Conf::UnsetFlags` { `Gnome::Conf::UNSET_INCLUDING_SCHEMA_NAMES` }

#### 6.1.1 Detailed Description

#### 6.1.2 Enumeration Type Documentation

##### 6.1.2.1 ClientErrorHandlingMode

```
enum Gnome::Conf::ClientErrorHandlingMode
```

**Enumerator**

CLIENT_HANDLE_NONE	
CLIENT_HANDLE_UNRETURNED	
CLIENT_HANDLE_ALL	

**6.1.2.2 ClientPreloadType**

enum `Gnome::Conf::ClientPreloadType`

**Enumerator**

CLIENT_PRELOAD_NONE	
CLIENT_PRELOAD_ONELEVEL	
CLIENT_PRELOAD_RECURSIVE	

**6.1.2.3 UnsetFlags**

enum `Gnome::Conf::UnsetFlags`

**Enumerator**

UNSET_INCLUDING_SCHEMA_NAMES	
------------------------------	--

**6.1.2.4 ValueType**

enum `Gnome::Conf::ValueType`

**Enumerator**

VALUE_INVALID	
VALUE_STRING	
VALUE_INT	
VALUE_FLOAT	
VALUE_BOOL	
VALUE_SCHEMA	
VALUE_LIST	
VALUE_PAIR	

## Chapter 7

# Namespace Documentation

### 7.1 Glib Namespace Reference

### 7.2 Gnome Namespace Reference

#### Namespaces

- namespace [Conf](#)

### 7.3 Gnome::Conf Namespace Reference

#### Classes

- class [ChangeSet](#)  
*A [ChangeSet](#) is a set of changes to the GConf database that can be committed and reversed easily.*
- class [Client](#)  
*The main [Gnome::Conf](#) object.*
- class [Entry](#)  
*An [Entry](#) stores an entry from a GConf "directory", including a key-value pair, the name of the [Schema](#) applicable to this entry, whether the value is a default value, and whether GConf can write a new value at this key.*
- class [Error](#)  
*Exception class for [Gnome::Conf::Client](#) errors.*
- class [Schema](#)
- class [SetInterface](#)  
*Common Interface for key-value settable objects.*
- class [Value](#)  
*Wrapper for primitive types.*

#### Typedefs

- typedef **std::pair**< [Value](#), [Value](#) > [ValuePair](#)
- typedef **std::pair**< [ValueType](#), [ValueType](#) > [ValueTypePair](#)
- typedef sigc::slot< void, guint, [Entry](#) > [Callback](#)

## Enumerations

- enum [ClientErrorHandlingMode](#) {  
    [CLIENT\\_HANDLE\\_NONE](#) ,  
    [CLIENT\\_HANDLE\\_UNRETURNED](#) ,  
    [CLIENT\\_HANDLE\\_ALL](#) }
- enum [ClientPreloadType](#) {  
    [CLIENT\\_PRELOAD\\_NONE](#) ,  
    [CLIENT\\_PRELOAD\\_ONELEVEL](#) ,  
    [CLIENT\\_PRELOAD\\_RECURSIVE](#) }
- enum [ValueType](#) {  
    [VALUE\\_INVALID](#) ,  
    [VALUE\\_STRING](#) ,  
    [VALUE\\_INT](#) ,  
    [VALUE\\_FLOAT](#) ,  
    [VALUE\\_BOOL](#) ,  
    [VALUE\\_SCHEMA](#) ,  
    [VALUE\\_LIST](#) ,  
    [VALUE\\_PAIR](#) }
- enum [UnsetFlags](#) { [UNSET\\_INCLUDING\\_SCHEMA\\_NAMES](#) }

## Functions

- void [init](#) ()

### 7.3.1 Typedef Documentation

#### 7.3.1.1 Callback

```
typedef sigc::slot<void, guint, Entry> Gnome::Conf::Callback
```

#### 7.3.1.2 ValuePair

```
typedef std::pair<Value, Value> Gnome::Conf::ValuePair
```

#### 7.3.1.3 ValueTypePair

```
typedef std::pair<ValueType, ValueType> Gnome::Conf::ValueTypePair
```

### 7.3.2 Function Documentation

#### 7.3.2.1 init()

```
void Gnome::Conf::init ( )
```



## Chapter 8

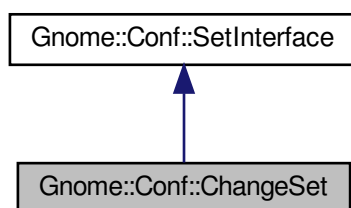
# Class Documentation

### 8.1 Gnome::Conf::ChangeSet Class Reference

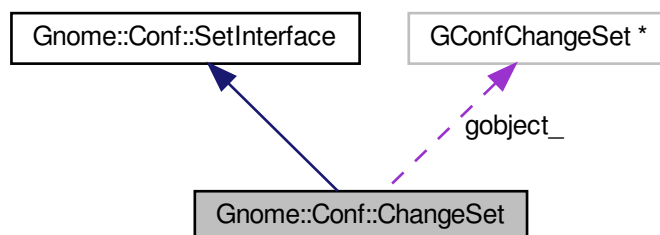
A [ChangeSet](#) is a set of changes to the GConf database that can be committed and reversed easily.

```
#include <gconfmm/changeset.h>
```

Inheritance diagram for Gnome::Conf::ChangeSet:



Collaboration diagram for Gnome::Conf::ChangeSet:



## Public Types

- typedef sigc::slot< void, const Glib::ustring &, const [Value](#) & > [ForeachSlot](#)

## Public Member Functions

- [ChangeSet](#) ()
- [ChangeSet](#) (GConfChangeSet \*castitem, bool make\_a\_copy=false)
- [ChangeSet](#) (const [ChangeSet](#) &src)
- [ChangeSet](#) & [operator=](#) (const [ChangeSet](#) &src)
- virtual [~ChangeSet](#) ()
- GConfChangeSet \* [gobj](#) ()
- const GConfChangeSet \* [gobj](#) () const
- GConfChangeSet \* [gobj\\_copy](#) () const
- void [clear](#) ()  
*Clear all entries.*
- unsigned int [size](#) () const  
*Returns the number of keys in the changeset.*
- void [remove](#) (const Glib::ustring &key)  
*Remove the specified key from the changeset.*
- [Value](#) \* [exists](#) (const Glib::ustring &key) const  
*Check whether the given key will be modified by a commit operation.*
- void [unset](#) (const Glib::ustring &key)  
*Unset the given key.*
- virtual void [set](#) (const Glib::ustring &key, const [Value](#) &value)
- virtual void [set](#) (const Glib::ustring &key, bool **what**)
- virtual void [set](#) (const Glib::ustring &key, int **what**)
- virtual void [set](#) (const Glib::ustring &key, double **what**)
- virtual void [set](#) (const Glib::ustring &key, const Glib::ustring &**what**)
- virtual void [set](#) (const Glib::ustring &key, const [Schema](#) &**what**)
- void [for\\_each](#) (const [ForeachSlot](#) &slot)  
*Iterate over the keys marked in this [ChangeSet](#).*

## Protected Attributes

- GConfChangeSet \* [gobject\\_](#)

### 8.1.1 Detailed Description

A [ChangeSet](#) is a set of changes to the GConf database that can be committed and reversed easily.

The changes can be both set and unset operations. Currently the [ChangeSet](#) operations are not atomic, and not specially optimized for. However, it is suitable for use, for instance, preferences dialogs.

The set\*() methods do not throw errors, they simply store the keys and the values.

See also

[Client::change\\_set\\_from\\_current\(\)](#), [Client::change\\_set\\_commit\(\)](#), [Client::change\\_set\\_reverse\(\)](#).

## 8.1.2 Member Typedef Documentation

### 8.1.2.1 ForeachSlot

```
typedef sigc::slot<void, const Glib::ustring&, const Value&> Gnome::Conf::ChangeSet::ForeachSlot
```

## 8.1.3 Constructor & Destructor Documentation

### 8.1.3.1 ChangeSet() [1/3]

```
Gnome::Conf::ChangeSet::ChangeSet ( )
```

### 8.1.3.2 ChangeSet() [2/3]

```
Gnome::Conf::ChangeSet::ChangeSet (
    GConfChangeSet * castitem,
    bool make_a_copy = false ) [explicit]
```

### 8.1.3.3 ChangeSet() [3/3]

```
Gnome::Conf::ChangeSet::ChangeSet (
    const ChangeSet & src )
```

### 8.1.3.4 ~ChangeSet()

```
virtual Gnome::Conf::ChangeSet::~~ChangeSet ( ) [virtual]
```

## 8.1.4 Member Function Documentation

#### 8.1.4.1 clear()

```
void Gnome::Conf::ChangeSet::clear ( )
```

Clear all entries.

After this method, committing the changeset is a no-op.

#### 8.1.4.2 exists()

```
Value * Gnome::Conf::ChangeSet::exists (
    const Glib::ustring & key ) const
```

Check whether the given key will be modified by a commit operation.

##### Returns

0 if the key will not be modified, else the modified value. Remember to delete the [Value](#).

#### 8.1.4.3 for\_each()

```
void Gnome::Conf::ChangeSet::for_each (
    const ForeachSlot & slot )
```

Iterate over the keys marked in this [ChangeSet](#).

Calls `slot` for each key-value pair that is marked in the [ChangeSet](#). Keys marked unset will have a [Value](#) with type `VALUE_INVALID`.

#### 8.1.4.4 gobj() [1/2]

```
GConfChangeSet * Gnome::Conf::ChangeSet::gobj ( ) [inline]
```

#### 8.1.4.5 gobj() [2/2]

```
const GConfChangeSet * Gnome::Conf::ChangeSet::gobj ( ) const [inline]
```

#### 8.1.4.6 gobj\_copy()

```
GConfChangeSet * Gnome::Conf::ChangeSet::gobj_copy ( ) const
```

#### 8.1.4.7 operator=()

```
ChangeSet & Gnome::Conf::ChangeSet::operator= (
    const ChangeSet & src )
```

#### 8.1.4.8 remove()

```
void Gnome::Conf::ChangeSet::remove (
    const Glib::ustring & key )
```

Remove the specified key from the changeset.

This means that the given key will not be modified by a commit.

#### 8.1.4.9 set() [1/6]

```
virtual void Gnome::Conf::ChangeSet::set (
    const Glib::ustring & key,
    bool what ) [virtual]
```

Implements [Gnome::Conf::SetInterface](#).

#### 8.1.4.10 set() [2/6]

```
virtual void Gnome::Conf::ChangeSet::set (
    const Glib::ustring & key,
    const Glib::ustring & what ) [virtual]
```

Implements [Gnome::Conf::SetInterface](#).

#### 8.1.4.11 set() [3/6]

```
virtual void Gnome::Conf::ChangeSet::set (
    const Glib::ustring & key,
    const Schema & what ) [virtual]
```

Implements [Gnome::Conf::SetInterface](#).

#### 8.1.4.12 set() [4/6]

```
virtual void Gnome::Conf::ChangeSet::set (
    const Glib::ustring & key,
    const Value & value ) [virtual]
```

Implements [Gnome::Conf::SetInterface](#).

#### 8.1.4.13 set() [5/6]

```
virtual void Gnome::Conf::ChangeSet::set (
    const Glib::ustring & key,
    double what ) [virtual]
```

Implements [Gnome::Conf::SetInterface](#).

#### 8.1.4.14 set() [6/6]

```
virtual void Gnome::Conf::ChangeSet::set (
    const Glib::ustring & key,
    int what ) [virtual]
```

Implements [Gnome::Conf::SetInterface](#).

#### 8.1.4.15 size()

```
unsigned int Gnome::Conf::ChangeSet::size ( ) const
```

Returns the number of keys in the changeset.

#### 8.1.4.16 unset()

```
void Gnome::Conf::ChangeSet::unset (
    const Glib::ustring & key )
```

Unset the given key.

Mark the key, so that it will be removed from the configuration database during a commit.

### 8.1.5 Member Data Documentation

### 8.1.5.1 gobject\_

```
GConfChangeSet* Gnome::Conf::ChangeSet::gobject_ [protected]
```

The documentation for this class was generated from the following file:

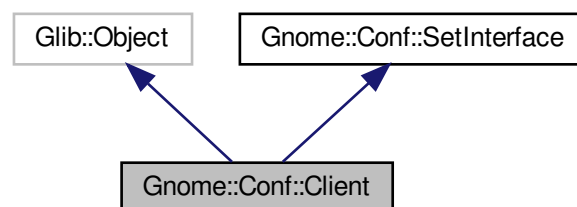
- gconfmm/changeset.h

## 8.2 Gnome::Conf::Client Class Reference

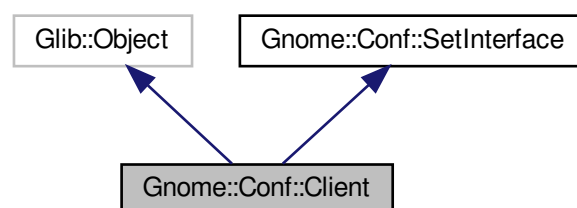
The main [Gnome::Conf](#) object.

```
#include <gconfmm/client.h>
```

Inheritance diagram for Gnome::Conf::Client:



Collaboration diagram for Gnome::Conf::Client:



### Public Types

- typedef Glib::SListHandle< int, BasicTypeTraits< int > > [SListHandleInts](#)
- typedef Glib::SListHandle< bool, BasicTypeTraits< bool > > [SListHandleBools](#)
- typedef Glib::SListHandle< double, BasicTypeTraits< double > > [SListHandleFloats](#)

## Public Member Functions

- virtual `~Client` ()
- `GConfClient * gobj` ()  
*Provides access to the underlying C GObject.*
- const `GConfClient * gobj` () const  
*Provides access to the underlying C GObject.*
- `GConfClient * gobj_copy` ()  
*Provides access to the underlying C instance. The caller is responsible for unrefing it. Use when directly setting fields in structs.*
- void `add_dir` (const Glib::ustring &dir, `ClientPreloadType` preload=`CLIENT_PRELOAD_NONE`)  
*Add a directory to the list of directories the `Client` will watch.*
- void `remove_dir` (const Glib::ustring &dir)  
*Remove a directory from the list of directories the `Client` will watch.*
- guint `notify_add` (const Glib::ustring &namespace\_section, `Callback` callback)  
*Request notification of changes to namespace\_section.*
- void `notify_remove` (guint cnxn)  
*Cancel a notification request.*
- void `notify` (const Glib::ustring &key)  
*Emits the value\_changed signal and notifies listeners as if key had been changed.*
- void `set_error_handling` (`ClientErrorHandlingMode` mode)
- void `clear_cache` ()  
*Clear the client-side cache.*
- void `preload` (const Glib::ustring &dirname, `ClientPreloadType` type)  
*Preloads a directory.*
- `Value` `get` (const Glib::ustring &key) const  
*Get the value of a configuration key.*
- `Value` `get_without_default` (const Glib::ustring &key) const  
*Get the value of a configuration key, without falling back to the default if the key has not been set.*
- `Value` `get_default_from_schema` (const Glib::ustring &key) const  
*Get the default value of this key by looking it up in the appropriate schema.*
- `Entry` `get_entry` (const Glib::ustring &key, bool use\_schema\_default=true) const  
*Get the complete `Entry` of the specified key.*
- `Entry` `get_entry` (const Glib::ustring &key, const char \*locale, bool use\_schema\_default=true) const  
*Get the complete `Entry` of the specified key.*
- void `unset` (const Glib::ustring &key)  
*Unset a configuration key.*
- void `recursive_unset` (const Glib::ustring &key, `UnsetFlags` flags=`UNSET_INCLUDING_SCHEMA_NAMES`)  
*Unsets all keys below key, including key itself.*
- Glib::SListHandle< `Entry` > `all_entries` (const Glib::ustring &dir) const  
*Retrieve all keys in the given configuration directory.*
- Glib::SListHandle< Glib::ustring > `all_dirs` (const Glib::ustring &dir) const  
*Retrieve all subdirectories of a given configuration directory.*
- void `suggest_sync` ()  
*Suggest to the GConf server that a sync of cached data to stable storage would be appropriate now.*
- bool `dir_exists` (const Glib::ustring &p1) const  
*Determine whether a given configuration directory exists.*
- bool `key_is_writable` (const Glib::ustring &p1) const  
*Determine whether a given configuration key is writeable by the application.*
- double `get_float` (const Glib::ustring &key) const  
*Get the float value at the given configuration key.*



- gint [get\\_int](#) (const Glib::ustring &key) const  
*Get the integer at the given configuration key.*
- bool [get\\_bool](#) (const Glib::ustring &key) const  
*Get the boolean at the given configuration key.*
- Glib::ustring [get\\_string](#) (const Glib::ustring &key) const  
*Get the string at the given configuration key.*
- [Schema](#) [get\\_schema](#) (const Glib::ustring &key) const  
*Get the [Schema](#) at the given configuration key.*
- SListHandle\_ValueInt [get\\_int\\_list](#) (const Glib::ustring &key) const  
*Get the list of integers at the given configuration key.*
- SListHandle\_ValueBool [get\\_bool\\_list](#) (const Glib::ustring &key) const  
*Get the list of booleans at the given configuration key.*
- SListHandle\_ValueFloat [get\\_float\\_list](#) (const Glib::ustring &key) const  
*Get the list of doubles at the given configuration key.*
- SListHandle\_ValueSchema [get\\_schema\\_list](#) (const Glib::ustring &key) const  
*Get the list of Schemas at the given configuration key.*
- SListHandle\_ValueString [get\\_string\\_list](#) (const Glib::ustring &key) const  
*Get the list of strings at the given configuration key.*
- [ValuePair](#) [get\\_pair](#) (const Glib::ustring &key, [ValueTypePair](#) types) const  
*Get the pair at the given configuration key.*
- void [set](#) (const Glib::ustring &key, int **what**)  
*Set the given configuration key to the specified integer value.*
- void [set](#) (const Glib::ustring &key, bool **what**)  
*Set the given configuration key to the specified boolean value.*
- void [set](#) (const Glib::ustring &key, double **what**)  
*Set the given configuration key to the specified double value.*
- void [set](#) (const Glib::ustring &key, const Glib::ustring & **what**)  
*Set the given configuration key to the specified string.*
- void [set](#) (const Glib::ustring &key, const [Schema](#) & **what**)  
*Set the given configuration key to the specified [Schema](#).*
- void [set](#) (const Glib::ustring &key, const [Value](#) & **what**)  
*Set the given configuration key to the specified [Value](#).*
- void [set\\_int\\_list](#) (const Glib::ustring &key, const SListHandleInts & **what**)
- void [set\\_bool\\_list](#) (const Glib::ustring &key, const SListHandleBools & **what**)
- void [set\\_float\\_list](#) (const Glib::ustring &key, const SListHandleFloats & **what**)
- void [set\\_schema\\_list](#) (const Glib::ustring &key, const Glib::SListHandle< [Schema](#) > & **what**)
- void [set\\_string\\_list](#) (const Glib::ustring &key, const Glib::SListHandle< Glib::ustring > & **what**)
- [ChangeSet](#) [change\\_set\\_from\\_current](#) (const Glib::SArray &[set](#))  
*Create a [ChangeSet](#) from the current values of the configuration database.*
- void [change\\_set\\_commit](#) ([ChangeSet](#) &[set](#), bool remove\_committed)  
*Commit the [ChangeSet](#) to the configuration database.*
- [ChangeSet](#) [change\\_set\\_reverse](#) (const [ChangeSet](#) &[set](#))  
*Creates a [ChangeSet](#) to reverse the effects of the given [ChangeSet](#).*
- Glib::SignalProxy2< void, const Glib::ustring &, const [Value](#) & > [signal\\_value\\_changed](#) ()  
*A signal emitted when a value changes.*
- void [value\\_changed](#) (const Glib::ustring &key, const [Value](#) &value)
- Glib::SignalProxy1< void, const Glib::Error & > [signal\\_error](#) ()  
*A signal emitted when an error occurs.*
- void [error](#) (const Glib::Error &error)

## Static Public Member Functions

- static Glib::RefPtr< [Client](#) > [get\\_default\\_client](#) ()  
*Get the default client object for this application.*
- static Glib::RefPtr< [Client](#) > [get\\_client\\_for\\_engine](#) (GConfEngine \*engine)

## Protected Member Functions

- virtual void [on\\_value\\_changed](#) (const Glib::ustring &key, const [Value](#) &value)
- virtual void [on\\_unreturned\\_error](#) (const Glib::Error &error)
- virtual void [on\\_error](#) (const Glib::Error &error)

## Related Functions

(Note that these are not member functions.)

- Glib::RefPtr< [Gnome::Conf::Client](#) > [wrap](#) (GConfClient \*object, bool take\_copy=false)  
*A Glib::wrap() method for this object.*

### 8.2.1 Detailed Description

The main [Gnome::Conf](#) object.

This class allows you to interface with the [Gnome](#) configuration system. Generally, it stores key-value pairs. The keys have an hierarchical namespace, with elements separated by slashes. The values are either typed primitives (int, bool, string, float or a [Schema](#)), or lists of primitives or pairs of primitives (for limits on the compound values, see [Value](#)). For conventions on the names of keys, see the GConf documentation.

### 8.2.2 Member Typedef Documentation

#### 8.2.2.1 SListHandleBools

```
typedef Glib::SListHandle< bool, BasicTypeTraits<bool> > Gnome::Conf::Client::SListHandleBools
```

#### 8.2.2.2 SListHandleFloats

```
typedef Glib::SListHandle< double, BasicTypeTraits<double> > Gnome::Conf::Client::SListHandleFloats
```

### 8.2.2.3 SListHandleInts

```
typedef Glib::SListHandle< int, BasicTypeTraits<int> > Gnome::Conf::Client::SListHandleInts
```

## 8.2.3 Constructor & Destructor Documentation

### 8.2.3.1 ~Client()

```
virtual Gnome::Conf::Client::~~Client ( ) [virtual]
```

## 8.2.4 Member Function Documentation

### 8.2.4.1 add\_dir()

```
void Gnome::Conf::Client::add_dir (
    const Glib::ustring & dir,
    ClientPreloadType preload = CLIENT_PRELOAD_NONE )
```

Add a directory to the list of directories the [Client](#) will watch.

Any changes to keys below this directory will cause the "value\_changed" signal to be emitted. When you add the directory, you can request that the [Client](#) preloads its contents - see ClientPreloadType for details.

Added directories may not overlap. That is, if you add "/foo", you may not add "/foo/bar". However you can add "/foo" and "/bar". You can also add "/foo" multiple times; if you add a directory multiple times, it will not be removed until you call [remove\\_dir\(\)](#) an equal number of times.

#### Parameters

<i>dir</i>	the directory to watch.
<i>preload</i>	the preload type (if any) to be performed.

### 8.2.4.2 all\_dirs()

```
Glib::SListHandle< Glib::ustring > Gnome::Conf::Client::all_dirs (
    const Glib::ustring & dir ) const
```

Retrieve all subdirectories of a given configuration directory.

**Parameters**

<i>dir</i>	the configuration directory to scan.
------------	--------------------------------------

**Returns**

a container with the names of the subdirectories.

**Exceptions**

<i>Gnome::Conf::Error.</i>	
----------------------------	--

**8.2.4.3 all\_entries()**

```
Glib::SListHandle< Entry > Gnome::Conf::Client::all_entries (
    const Glib::ustring & dir ) const
```

Retrieve all keys in the given configuration directory.

Get all the configuration keys in the given directory, without recursion.

**Parameters**

<i>dir</i>	the configuration directory to scan.
------------	--------------------------------------

**Returns**

a container with the names of the configuration keys.

**Exceptions**

<i>Gnome::Conf::Error.</i>	
----------------------------	--

**8.2.4.4 change\_set\_commit()**

```
void Gnome::Conf::Client::change_set_commit (
    ChangeSet & set,
    bool remove_committed )
```

Commit the [ChangeSet](#) to the configuration database.

Commits the configuration changes in the [ChangeSet](#) to the database. If `remove_committed` is `true`, all successfully committed keys will be removed from the [ChangeSet](#). If an error occurs, a [Gnome::Conf::Error](#) will be thrown. This operation is not atomic - an error will be thrown on the first error.

## Parameters

<i>set</i>	the <a href="#">ChangeSet</a> to commit.
<i>remove_committed</i>	whether to remove successfully-committed keys from the <a href="#">ChangeSet</a> .

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

## See also

[ChangeSet](#)**8.2.4.5 change\_set\_from\_current()**

```
ChangeSet Gnome::Conf::Client::change_set_from_current (
    const Glib::SArray & set )
```

Create a [ChangeSet](#) from the current values of the configuration database.

Creates a [ChangeSet](#) containing the current values of all the keys listed in the *set*. For instance, this could be used in a preferences dialog as an undo operation.

## Parameters

<i>set</i>	A container of the configuration keys to backup.
------------	--

## Returns

the [ChangeSet](#) with the current values.

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

## See also

[ChangeSet](#)**8.2.4.6 change\_set\_reverse()**

```
ChangeSet Gnome::Conf::Client::change_set_reverse (
    const ChangeSet & set )
```

Creates a [ChangeSet](#) to reverse the effects of the given [ChangeSet](#).

Creates a [ChangeSet](#) that contains the current values of the keys in `set`, effectively creating a back-up of the values in the database that will be modified when the `set` will be committed. For instance, this allows you to create a back-up changeset to use in case of errors, or an undo facility for preferences.

#### Parameters

<code>set</code>	the <a href="#">ChangeSet</a> to reverse.
------------------	---

#### Returns

the reverse [ChangeSet](#).

#### Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

#### See also

[ChangeSet](#)

#### 8.2.4.7 clear\_cache()

```
void Gnome::Conf::Client::clear_cache ( )
```

Clear the client-side cache.

#### 8.2.4.8 dir\_exists()

```
bool Gnome::Conf::Client::dir_exists (
    const Glib::ustring & p1 ) const
```

Determine whether a given configuration directory exists.

#### Returns

true if the directory exists.

#### Exceptions

<a href="#">Gnome::Conf::Error</a> .	
--------------------------------------	--

#### 8.2.4.9 error()

```
void Gnome::Conf::Client::error (
    const Glib::Error & error )
```

#### 8.2.4.10 get()

```
Value Gnome::Conf::Client::get (
    const Glib::ustring & key ) const
```

Get the value of a configuration key.

@parameter key: the configuration key to retrieve.

##### Returns

the [Value](#) of the key.

##### Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

#### 8.2.4.11 get\_bool()

```
bool Gnome::Conf::Client::get_bool (
    const Glib::ustring & key ) const
```

Get the boolean at the given configuration key.

#### 8.2.4.12 get\_bool\_list()

```
SListHandle_ValueBool Gnome::Conf::Client::get_bool_list (
    const Glib::ustring & key ) const
```

Get the list of booleans at the given configuration key.

#### 8.2.4.13 get\_client\_for\_engine()

```
static Glib::RefPtr< Client > Gnome::Conf::Client::get_client_for_engine (
    GConfEngine * engine ) [static]
```

#### 8.2.4.14 `get_default_client()`

```
static Glib::RefPtr< Client > Gnome::Conf::Client::get_default_client ( ) [static]
```

Get the default client object for this application.

The object is a Singleton, so you will always get the same instance. Most applications should use this.

#### 8.2.4.15 `get_default_from_schema()`

```
Value Gnome::Conf::Client::get_default_from_schema (
    const Glib::ustring & key ) const
```

Get the default value of this key by looking it up in the appropriate schema.

@parameter key: the configuration key to retrieve.

##### Returns

the default [Value](#) of the key.

##### Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

#### 8.2.4.16 `get_entry()` [1/2]

```
Entry Gnome::Conf::Client::get_entry (
    const Glib::ustring & key,
    bool use_schema_default = true ) const
```

Get the complete [Entry](#) of the specified key.

Uses the default locale

##### Parameters

<i>key</i>	the configuration key to retrieve.
<i>use_schema_default</i>	whether to fall back to the <a href="#">Schema</a> default value if the specified configuration key has not been set.

##### Returns

an [Entry](#) for the corresponding configuration key.



## Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

**8.2.4.17** `get_entry()` [2/2]

```
Entry Gnome::Conf::Client::get_entry (
    const Glib::ustring & key,
    const char * locale,
    bool use_schema_default = true ) const
```

Get the complete [Entry](#) of the specified key.

## Parameters

<i>key</i>	the configuration key to retrieve.
<i>locale</i>	the locale for the user-visible strings in the <a href="#">Entry</a> 's <a href="#">Schema</a> . Use 0 to use the default.
<i>use_schema_default</i>	whether to fall back to the <a href="#">Schema</a> default value if the specified configuration key has not been set.

## Returns

an [Entry](#) for the corresponding configuration key.

## Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

**8.2.4.18** `get_float()`

```
double Gnome::Conf::Client::get_float (
    const Glib::ustring & key ) const
```

Get the float value at the given configuration key.

Throws an error if the key does not contain the appropriate type.

## Parameters

<i>key</i>	the configuration key to fetch.
------------	---------------------------------

**Returns**

the value at the specified configuration key.

**Exceptions**

<a href="#"><i>Gnome::Conf::Error</i></a>	
---	--

**8.2.4.19 get\_float\_list()**

```
SListHandle_ValueFloat Gnome::Conf::Client::get_float_list (
    const Glib::ustring & key ) const
```

Get the list of doubles at the given configuration key.

**8.2.4.20 get\_int()**

```
gint Gnome::Conf::Client::get_int (
    const Glib::ustring & key ) const
```

Get the integer at the given configuration key.

**8.2.4.21 get\_int\_list()**

```
SListHandle_ValueInt Gnome::Conf::Client::get_int_list (
    const Glib::ustring & key ) const
```

Get the list of integers at the given configuration key.

If the given key is not a list, or the list elements are not of the appropriate type, an error will be thrown.

**Parameters**

<i>key</i>	the configuration key that contains the list.
------------	---

**Returns**

a Glib::SListHandle of the appropriate type.

**Exceptions**

<a href="#"><i>Gnome::Conf::Error</i></a>	
---	--

#### 8.2.4.22 get\_pair()

```
ValuePair Gnome::Conf::Client::get_pair (
    const Glib::ustring & key,
    ValueTypePair types ) const
```

Get the pair at the given configuration key.

The pair's elements must have the types given in `types` respectively. If the value is not a pair or the types do not match, an error will be thrown.

##### Parameters

<i>key</i>	the configuration key that contains the pair.
<i>types</i>	a pair of the expected types of the values.

##### Returns

a ValuePair.

##### Exceptions

<a href="#"><i>Gnome::Conf::Error</i></a>	
---	--

#### 8.2.4.23 get\_schema()

```
Schema Gnome::Conf::Client::get_schema (
    const Glib::ustring & key ) const
```

Get the [Schema](#) at the given configuration key.

#### 8.2.4.24 get\_schema\_list()

```
SListHandle_ValueSchema Gnome::Conf::Client::get_schema_list (
    const Glib::ustring & key ) const
```

Get the list of Schemas at the given configuration key.

**8.2.4.25 get\_string()**

```
Glib::ustring Gnome::Conf::Client::get_string (
    const Glib::ustring & key ) const
```

Get the string at the given configuration key.

**8.2.4.26 get\_string\_list()**

```
SListHandle_ValueString Gnome::Conf::Client::get_string_list (
    const Glib::ustring & key ) const
```

Get the list of strings at the given configuration key.

**8.2.4.27 get\_without\_default()**

```
Value Gnome::Conf::Client::get_without_default (
    const Glib::ustring & key ) const
```

Get the value of a configuration key, without falling back to the default if the key has not been set.

In that case, the type of the value will be VALUE\_INVALID.

**Parameters**

<i>key</i>	the configuration key to retrieve.
------------	------------------------------------

**Returns**

the [Value](#) of the key.

**Exceptions**

<i>Gnome::Conf::Error.</i>	
----------------------------	--

**8.2.4.28 gobj() [1/2]**

```
GConfClient * Gnome::Conf::Client::gobj ( ) [inline]
```

Provides access to the underlying C GObject.

#### 8.2.4.29 gobj() [2/2]

```
const GConfClient * Gnome::Conf::Client::gobj ( ) const [inline]
```

Provides access to the underlying C GObject.

#### 8.2.4.30 gobj\_copy()

```
GConfClient * Gnome::Conf::Client::gobj_copy ( )
```

Provides access to the underlying C instance. The caller is responsible for unrefing it. Use when directly setting fields in structs.

#### 8.2.4.31 key\_is\_writable()

```
bool Gnome::Conf::Client::key_is_writable (
    const Glib::ustring & pl ) const
```

Determine whether a given configuration key is writeable by the application.

##### Returns

true if the key is writeable.

##### Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

#### 8.2.4.32 notify()

```
void Gnome::Conf::Client::notify (
    const Glib::ustring & key )
```

Emits the value\_changed signal and notifies listeners as if key had been changed.

##### Parameters

key	The key that has changed.
-----	---------------------------

@newin2p24

### 8.2.4.33 notify\_add()

```
guint Gnome::Conf::Client::notify_add (
    const Glib::ustring & namespace_section,
    Callback callback )
```

Request notification of changes to namespace\_section.

This includes the key namespace\_section itself, and any keys below it. For the notification to happen, namespace\_section must be equal to or below one of the directories added with [add\\_dir\(\)](#). You can still call [notify\\_add\(\)](#) for other directories, but no notification will be received until you add a directory above or equal to namespace\_section. One implication of this is that [remove\\_dir\(\)](#) temporarily disables notifications that were below the removed directory.

The callback will be called with the key that changed and the [Entry](#) that holds the new [Value](#). If the [Value](#) has a type of VALUE\_INVALID, then the key has been unset.

The function returns a connection ID you can use when calling [notify\\_remove\(\)](#).

#### Parameters

<i>namespace_section</i>	the namespace section for which notification is required.
<i>callback</i>	the sigc::slot to call when the a key under namespace_section changes.

#### Returns

a connection id that can be passed to [notify\\_remove\(\)](#) to cancel the notification request.

### 8.2.4.34 notify\_remove()

```
void Gnome::Conf::Client::notify_remove (
    guint cnxn )
```

Cancel a notification request.

#### Parameters

<i>cnxn</i>	a connection id, previously returned by <a href="#">notify_add()</a>
-------------	--

#### See also

[notify\\_add\(\)](#)

### 8.2.4.35 on\_error()

```
virtual void Gnome::Conf::Client::on_error (
    const Glib::Error & error ) [protected], [virtual]
```

#### 8.2.4.36 on\_unreturned\_error()

```
virtual void Gnome::Conf::Client::on_unreturned_error (
    const Glib::Error & error ) [protected], [virtual]
```

#### 8.2.4.37 on\_value\_changed()

```
virtual void Gnome::Conf::Client::on_value_changed (
    const Glib::ustring & key,
    const Value & value ) [protected], [virtual]
```

#### 8.2.4.38 preload()

```
void Gnome::Conf::Client::preload (
    const Glib::ustring & dirname,
    ClientPreloadType type )
```

Preloads a directory.

Normally this happens automatically with [add\\_dir\(\)](#), but if you've called [clear\\_cache\(\)](#) you may need to do it again.

See also

[add\\_dir\(\)](#)

#### 8.2.4.39 recursive\_unset()

```
void Gnome::Conf::Client::recursive_unset (
    const Glib::ustring & key,
    UnsetFlags flags = UNSET_INCLUDING_SCHEMA_NAMES )
```

Unsets all keys below *key*, including *key* itself.

If any unset fails, it continues on to unset as much as it can. The first failure is then thrown as an exception.

##### Parameters

<i>key</i>	The configuration key to unset.
<i>flags</i>	Change how the unset is done.

##### Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

@newin2p24

#### 8.2.4.40 remove\_dir()

```
void Gnome::Conf::Client::remove_dir (
    const Glib::ustring & dir )
```

Remove a directory from the list of directories the [Client](#) will watch.

See also

[add\\_dir\(\)](#)

#### 8.2.4.41 set() [1/6]

```
void Gnome::Conf::Client::set (
    const Glib::ustring & key,
    bool what ) [virtual]
```

Set the given configuration key to the specified boolean value.

Set the given configuration key to the specified integer value.

Parameters

<i>key</i>	the configuration key to set.
<i>what</i>	the value to set it to.

Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

Implements [Gnome::Conf::SetInterface](#).

#### 8.2.4.42 set() [2/6]

```
void Gnome::Conf::Client::set (
    const Glib::ustring & key,
    const Glib::ustring & what ) [virtual]
```

Set the given configuration key to the specified string.

Set the given configuration key to the specified integer value.



## Parameters

<i>key</i>	the configuration key to set.
<i>what</i>	the value to set it to.

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

Implements [Gnome::Conf::SetInterface](#).

**8.2.4.43 set()** [3/6]

```
void Gnome::Conf::Client::set (
    const Glib::ustring & key,
    const Schema & what ) [virtual]
```

Set the given configuration key to the specified [Schema](#).

Set the given configuration key to the specified integer value.

## Parameters

<i>key</i>	the configuration key to set.
<i>what</i>	the value to set it to.

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

Implements [Gnome::Conf::SetInterface](#).

**8.2.4.44 set()** [4/6]

```
void Gnome::Conf::Client::set (
    const Glib::ustring & key,
    const Value & what ) [virtual]
```

Set the given configuration key to the specified [Value](#).

Set the given configuration key to the specified integer value.

## Parameters

<i>key</i>	the configuration key to set.
<i>what</i>	the value to set it to.

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

Implements [Gnome::Conf::SetInterface](#).

**8.2.4.45 set()** [5/6]

```
void Gnome::Conf::Client::set (  
    const Glib::ustring & key,  
    double what ) [virtual]
```

Set the given configuration key to the specified double value.

Set the given configuration key to the specified integer value.

## Parameters

<i>key</i>	the configuration key to set.
<i>what</i>	the value to set it to.

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

Implements [Gnome::Conf::SetInterface](#).

**8.2.4.46 set()** [6/6]

```
void Gnome::Conf::Client::set (  
    const Glib::ustring & key,  
    int what ) [virtual]
```

Set the given configuration key to the specified integer value.

## Parameters

<i>key</i>	the configuration key to set.
<i>what</i>	the value to set it to.

## Exceptions

<a href="#">Gnome::Conf::Error</a>	
------------------------------------	--

Implements [Gnome::Conf::SetInterface](#).

#### 8.2.4.47 set\_bool\_list()

```
void Gnome::Conf::Client::set_bool_list (
    const Glib::ustring & key,
    const SListHandleBools & what )
```

#### 8.2.4.48 set\_error\_handling()

```
void Gnome::Conf::Client::set_error_handling (
    ClientErrorHandlingMode mode )
```

#### 8.2.4.49 set\_float\_list()

```
void Gnome::Conf::Client::set_float_list (
    const Glib::ustring & key,
    const SListHandleFloats & what )
```

#### 8.2.4.50 set\_int\_list()

```
void Gnome::Conf::Client::set_int_list (
    const Glib::ustring & key,
    const SListHandleInts & what )
```

#### 8.2.4.51 set\_schema\_list()

```
void Gnome::Conf::Client::set_schema_list (
    const Glib::ustring & key,
    const Glib::SListHandle< Schema > & what )
```

#### 8.2.4.52 set\_string\_list()

```
void Gnome::Conf::Client::set_string_list (
    const Glib::ustring & key,
    const Glib::SListHandle< Glib::ustring > & what )
```

**8.2.4.53 signal\_error()**

```
Glib::SignalProxy1< void, const Glib::Error & > Gnome::Conf::Client::signal_error ( )
```

A signal emitted when an error occurs.

This signal will be emitted when an error occurs, right before the throw() of the error.

**Prototype:**

```
void on_my_error(const Glib::Error& error)
```

**8.2.4.54 signal\_value\_changed()**

```
Glib::SignalProxy2< void, const Glib::ustring &, const Value & > Gnome::Conf::Client::signal_value_changed ( )
```

A signal emitted when a value changes.

This signal will only be called for directories added with [add\\_dir\(\)](#).

**Prototype:**

```
void on_my_value_changed(const Glib::ustring& key, const Value& value)
```

**8.2.4.55 suggest\_sync()**

```
void Gnome::Conf::Client::suggest_sync ( )
```

Suggest to the GConf server that a sync of cached data to stable storage would be appropriate now.

**Exceptions**

<i>Gnome::Conf::Error.</i>	
----------------------------	--

**8.2.4.56 unset()**

```
void Gnome::Conf::Client::unset (
    const Glib::ustring & key )
```

Unset a configuration key.

## Parameters

<i>key</i>	the configuration key to unset.
------------	---------------------------------

## Exceptions

<i>Gnome::Conf::Error.</i>	
----------------------------	--

## 8.2.4.57 value\_changed()

```
void Gnome::Conf::Client::value_changed (
    const Glib::ustring & key,
    const Value & value )
```

## 8.2.5 Friends And Related Function Documentation

## 8.2.5.1 wrap()

```
Glib::RefPtr< Gnome::Conf::Client > wrap (
    GConfClient * object,
    bool take_copy = false ) [related]
```

A Glib::wrap() method for this object.

## Parameters

<i>object</i>	The C instance.
<i>take_copy</i>	False if the result should take ownership of the C instance. True if it should take a new copy or ref.

## Returns

A C++ instance that wraps this C instance.

The documentation for this class was generated from the following file:

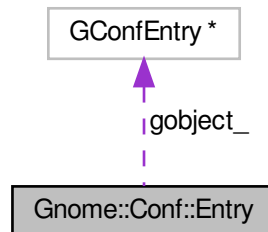
- gconfmm/client.h

## 8.3 Gnome::Conf::Entry Class Reference

An [Entry](#) stores an entry from a GConf "directory", including a key-value pair, the name of the [Schema](#) applicable to this entry, whether the value is a default value, and whether GConf can write a new value at this key.

```
#include <gconfmm/entry.h>
```

Collaboration diagram for Gnome::Conf::Entry:



## Public Member Functions

- [Entry](#) ()
  - [Entry](#) (GConfEntry \*castitem, bool make\_a\_copy=false)
  - [Entry](#) (const [Entry](#) &src)
  - [Entry](#) & [operator=](#) (const [Entry](#) &src)
  - [~Entry](#) ()
  - GConfEntry \* [gobj](#) ()
  - const GConfEntry \* [gobj](#) () const
  - GConfEntry \* [gobj\\_copy](#) () const
- Provides access to the underlying C instance. The caller is responsible for freeing it. Use when directly setting fields in structs.*
- [Entry](#) (const Glib::ustring &key, const [Value](#) &value)
- Construct an [Entry](#) with the given *key* and *value*.*
- void [set\\_value](#) (const [Value](#) &val)
- Set the [Value](#) of the entry.*
- void [set\\_schema\\_name](#) (const Glib::ustring &val)
- Set the [Schema](#) name of the entry.*
- void [set\\_is\\_default](#) (bool is\_default=true)
- Set whether the value has originated from the default given in the [Schema](#).*
- void [set\\_is\\_writable](#) (bool is\_writable=true)
- Set whether the given configuration key is writeable.*
- [Value](#) [get\\_value](#) () const
- Retrieve the value of the entry.*
- Glib::ustring [get\\_schema\\_name](#) () const
- Retrieve the [Schema](#) name associated with the given entry.*
- Glib::ustring [get\\_key](#) () const
  - bool [get\\_is\\_default](#) () const
  - bool [get\\_is\\_writable](#) () const

## Protected Attributes

- GConfEntry \* [gobject\\_](#)

## Related Functions

(Note that these are not member functions.)

- [Gnome::Conf::Entry wrap](#) (GConfEntry \*object, bool take\_copy=false)  
*A Glib::wrap() method for this object.*

### 8.3.1 Detailed Description

An [Entry](#) stores an entry from a GConf "directory", including a key-value pair, the name of the [Schema](#) applicable to this entry, whether the value is a default value, and whether GConf can write a new value at this key.

The key should be an absolute key, not a relative key.

### 8.3.2 Constructor & Destructor Documentation

#### 8.3.2.1 Entry() [1/4]

```
Gnome::Conf::Entry::Entry ( )
```

#### 8.3.2.2 Entry() [2/4]

```
Gnome::Conf::Entry::Entry (
    GConfEntry * castitem,
    bool make_a_copy = false ) [explicit]
```

#### 8.3.2.3 Entry() [3/4]

```
Gnome::Conf::Entry::Entry (
    const Entry & src )
```

#### 8.3.2.4 ~Entry()

```
Gnome::Conf::Entry::~Entry ( )
```

### 8.3.2.5 Entry() [4/4]

```
Gnome::Conf::Entry::Entry (
    const Glib::ustring & key,
    const Value & value )
```

Construct an [Entry](#) with the given key and value.

## 8.3.3 Member Function Documentation

### 8.3.3.1 get\_is\_default()

```
bool Gnome::Conf::Entry::get_is_default ( ) const
```

### 8.3.3.2 get\_is\_writable()

```
bool Gnome::Conf::Entry::get_is_writable ( ) const
```

### 8.3.3.3 get\_key()

```
Glib::ustring Gnome::Conf::Entry::get_key ( ) const
```

### 8.3.3.4 get\_schema\_name()

```
Glib::ustring Gnome::Conf::Entry::get_schema_name ( ) const
```

Retrieve the [Schema](#) name associated with the given entry.

### 8.3.3.5 get\_value()

```
Value Gnome::Conf::Entry::get_value ( ) const
```

Retrieve the value of the entry.

#### Returns

a copy the entry's value.



### 8.3.3.6 gobj() [1/2]

```
GConfEntry * Gnome::Conf::Entry::gobj ( ) [inline]
```

### 8.3.3.7 gobj() [2/2]

```
const GConfEntry * Gnome::Conf::Entry::gobj ( ) const [inline]
```

### 8.3.3.8 gobj\_copy()

```
GConfEntry * Gnome::Conf::Entry::gobj_copy ( ) const
```

Provides access to the underlying C instance. The caller is responsible for freeing it. Use when directly setting fields in structs.

### 8.3.3.9 operator=()

```
Entry & Gnome::Conf::Entry::operator= (
    const Entry & src )
```

### 8.3.3.10 set\_is\_default()

```
void Gnome::Conf::Entry::set_is_default (
    bool is_default = true )
```

Set whether the value has originated from the default given in the [Schema](#).

### 8.3.3.11 set\_is\_writable()

```
void Gnome::Conf::Entry::set_is_writable (
    bool is_writable = true )
```

Set whether the given configuration key is writeable.

#### 8.3.3.12 set\_schema\_name()

```
void Gnome::Conf::Entry::set_schema_name (
    const Glib::ustring & val )
```

Set the [Schema](#) name of the entry.

#### 8.3.3.13 set\_value()

```
void Gnome::Conf::Entry::set_value (
    const Value & val )
```

Set the [Value](#) of the entry.

### 8.3.4 Friends And Related Function Documentation

#### 8.3.4.1 wrap()

```
Gnome::Conf::Entry wrap (
    GConfEntry * object,
    bool take_copy = false ) [related]
```

A Glib::wrap() method for this object.

##### Parameters

<i>object</i>	The C instance.
<i>take_copy</i>	False if the result should take ownership of the C instance. True if it should take a new copy or ref.

##### Returns

A C++ instance that wraps this C instance.

### 8.3.5 Member Data Documentation

#### 8.3.5.1 gobject\_

```
GConfEntry* Gnome::Conf::Entry::gobject_ [protected]
```

The documentation for this class was generated from the following file:

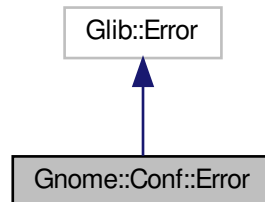
- gconfmm/entry.h

## 8.4 Gnome::Conf::Error Class Reference

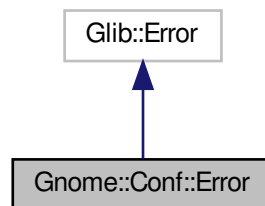
Exception class for [Gnome::Conf::Client](#) errors.

```
#include <gconfmm/client.h>
```

Inheritance diagram for Gnome::Conf::Error:



Collaboration diagram for Gnome::Conf::Error:



### Public Types

- enum [Code](#) {  
    [SUCCESS](#) = 0 ,  
    [NO\\_SERVER](#) = 2 ,  
    [NO\\_PERMISSION](#) = 3 ,  
    [BAD\\_ADDRESS](#) = 4 ,  
    [PARSE\\_ERROR](#) = 6 ,  
    [CORRUPT](#) = 7 ,  
    [TYPE\\_MISMATCH](#) = 8 ,  
    [IS\\_DIR](#) = 9 ,  
    [IS\\_KEY](#) = 10 ,  
    [OVERRIDDEN](#) = 11 ,  
    [OAF\\_ERROR](#) = 12 ,  
    [LOCAL\\_ENGINE](#) = 13 ,  
    [LOCK\\_FAILED](#) = 14 ,  
    [NO\\_WRITABLE\\_DATABASE](#) = 15 ,  
    [IN\\_SHUTDOWN](#) = 16 }

## Public Member Functions

- [Error](#) ([Code](#) error\_code, const Glib::ustring &error\_message)
- [Error](#) (GError \*gobject)
- [Code](#) code () const

### 8.4.1 Detailed Description

Exception class for [Gnome::Conf::Client](#) errors.

### 8.4.2 Member Enumeration Documentation

#### 8.4.2.1 Code

```
enum Gnome::Conf::Error::Code
```

Enumerator

SUCCESS	
NO_SERVER	
NO_PERMISSION	
BAD_ADDRESS	
PARSE_ERROR	
CORRUPT	
TYPE_MISMATCH	
IS_DIR	
IS_KEY	
OVERRIDDEN	
OAF_ERROR	
LOCAL_ENGINE	
LOCK_FAILED	
NO_WRITABLE_DATABASE	
IN_SHUTDOWN	

### 8.4.3 Constructor & Destructor Documentation

#### 8.4.3.1 Error() [1/2]

```
Gnome::Conf::Error::Error (  
    Code error_code,  
    const Glib::ustring & error_message )
```

### 8.4.3.2 Error() [2/2]

```
Gnome::Conf::Error::Error (
    GError * gobject ) [explicit]
```

## 8.4.4 Member Function Documentation

### 8.4.4.1 code()

```
Code Gnome::Conf::Error::code ( ) const
```

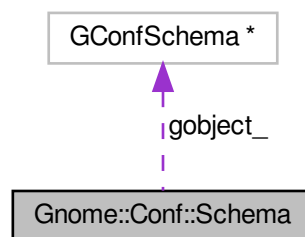
The documentation for this class was generated from the following file:

- gconfmm/client.h

## 8.5 Gnome::Conf::Schema Class Reference

```
#include <gconfmm/schema.h>
```

Collaboration diagram for Gnome::Conf::Schema:



## Public Member Functions

- [Schema](#) ()
- [Schema](#) (GConfSchema \*castitem, bool make\_a\_copy=false)
- [Schema](#) (const [Schema](#) &src)
- [Schema](#) & [operator=](#) (const [Schema](#) &src)
- [~Schema](#) ()
- GConfSchema \* [gobj](#) ()
- const GConfSchema \* [gobj](#) () const
- GConfSchema \* [gobj\\_copy](#) () const

*Provides access to the underlying C instance. The caller is responsible for freeing it. Use when directly setting fields in structs.*

- void [set\\_type](#) (ValueType type)
- void [set\\_list\\_type](#) (ValueType type)
- void [set\\_car\\_type](#) (ValueType type)
- void [set\\_cdr\\_type](#) (ValueType type)
- void [set\\_locale](#) (const **std::string** &locale)
- void [set\\_short\\_desc](#) (const Glib::ustring &desc)
- void [set\\_long\\_desc](#) (const Glib::ustring &desc)
- void [set\\_owner](#) (const Glib::ustring &owner)
- void [set\\_default\\_value](#) (const [Value](#) &value)
- [ValueType](#) [get\\_type](#) () const
- [ValueType](#) [get\\_list\\_type](#) () const
- [ValueType](#) [get\\_car\\_type](#) () const
- [ValueType](#) [get\\_cdr\\_type](#) () const
- **std::string** [get\\_locale](#) () const
- Glib::ustring [get\\_short\\_desc](#) () const
- Glib::ustring [get\\_long\\_desc](#) () const
- Glib::ustring [get\\_owner](#) () const
- [Value](#) [get\\_default\\_value](#) () const

## Protected Attributes

- GConfSchema \* [gobject\\_](#)

## Related Functions

(Note that these are not member functions.)

- [Gnome::Conf::Schema wrap](#) (GConfSchema \*object, bool take\_copy=false)

*A Glib::wrap() method for this object.*

## 8.5.1 Constructor & Destructor Documentation

### 8.5.1.1 Schema() [1/3]

```
Gnome::Conf::Schema::Schema ( )
```

### 8.5.1.2 Schema() [2/3]

```
Gnome::Conf::Schema::Schema (
    GConfSchema * castitem,
    bool make_a_copy = false ) [explicit]
```

### 8.5.1.3 Schema() [3/3]

```
Gnome::Conf::Schema::Schema (
    const Schema & src )
```

### 8.5.1.4 ~Schema()

```
Gnome::Conf::Schema::~Schema ( )
```

## 8.5.2 Member Function Documentation

### 8.5.2.1 get\_car\_type()

```
ValueType Gnome::Conf::Schema::get_car_type ( ) const
```

### 8.5.2.2 get\_cdr\_type()

```
ValueType Gnome::Conf::Schema::get_cdr_type ( ) const
```

### 8.5.2.3 get\_default\_value()

```
Value Gnome::Conf::Schema::get_default_value ( ) const
```

### 8.5.2.4 get\_list\_type()

```
ValueType Gnome::Conf::Schema::get_list_type ( ) const
```

#### 8.5.2.5 get\_locale()

```
std::string Gnome::Conf::Schema::get_locale ( ) const
```

#### 8.5.2.6 get\_long\_desc()

```
Glib::ustring Gnome::Conf::Schema::get_long_desc ( ) const
```

#### 8.5.2.7 get\_owner()

```
Glib::ustring Gnome::Conf::Schema::get_owner ( ) const
```

#### 8.5.2.8 get\_short\_desc()

```
Glib::ustring Gnome::Conf::Schema::get_short_desc ( ) const
```

#### 8.5.2.9 get\_type()

```
ValueType Gnome::Conf::Schema::get_type ( ) const
```

#### 8.5.2.10 gobj() [1/2]

```
GConfSchema * Gnome::Conf::Schema::gobj ( ) [inline]
```

#### 8.5.2.11 gobj() [2/2]

```
const GConfSchema * Gnome::Conf::Schema::gobj ( ) const [inline]
```



#### 8.5.2.12 gobj\_copy()

```
GConfSchema * Gnome::Conf::Schema::gobj_copy ( ) const
```

Provides access to the underlying C instance. The caller is responsible for freeing it. Use when directly setting fields in structs.

#### 8.5.2.13 operator=()

```
Schema & Gnome::Conf::Schema::operator= (
    const Schema & src )
```

#### 8.5.2.14 set\_car\_type()

```
void Gnome::Conf::Schema::set_car_type (
    ValueType type )
```

#### 8.5.2.15 set\_cdr\_type()

```
void Gnome::Conf::Schema::set_cdr_type (
    ValueType type )
```

#### 8.5.2.16 set\_default\_value()

```
void Gnome::Conf::Schema::set_default_value (
    const Value & value )
```

#### 8.5.2.17 set\_list\_type()

```
void Gnome::Conf::Schema::set_list_type (
    ValueType type )
```

#### 8.5.2.18 set\_locale()

```
void Gnome::Conf::Schema::set_locale (
    const std::string & locale )
```

#### 8.5.2.19 set\_long\_desc()

```
void Gnome::Conf::Schema::set_long_desc (
    const Glib::ustring & desc )
```

#### 8.5.2.20 set\_owner()

```
void Gnome::Conf::Schema::set_owner (
    const Glib::ustring & owner )
```

#### 8.5.2.21 set\_short\_desc()

```
void Gnome::Conf::Schema::set_short_desc (
    const Glib::ustring & desc )
```

#### 8.5.2.22 set\_type()

```
void Gnome::Conf::Schema::set_type (
    ValueType type )
```

### 8.5.3 Friends And Related Function Documentation

#### 8.5.3.1 wrap()

```
Gnome::Conf::Schema wrap (
    GConfSchema * object,
    bool take_copy = false ) [related]
```

A Glib::wrap() method for this object.

## Parameters

<i>object</i>	The C instance.
<i>take_copy</i>	False if the result should take ownership of the C instance. True if it should take a new copy or ref.

## Returns

A C++ instance that wraps this C instance.

## 8.5.4 Member Data Documentation

### 8.5.4.1 gobject\_

```
GConfSchema* Gnome::Conf::Schema::gobject_ [protected]
```

The documentation for this class was generated from the following file:

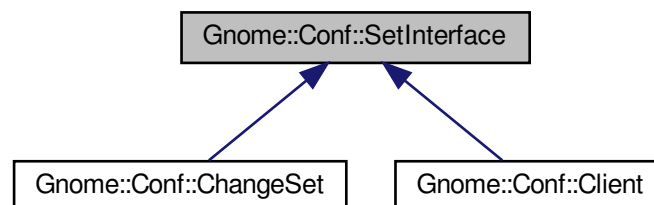
- gconfmm/schema.h

## 8.6 Gnome::Conf::SetInterface Class Reference

Common Interface for key-value settable objects.

```
#include <gconfmm/setinterface.h>
```

Inheritance diagram for Gnome::Conf::SetInterface:



## Public Member Functions

- virtual void [set](#) (const Glib::ustring &key, const [Value](#) &value)=0
- virtual void [set](#) (const Glib::ustring &key, bool **what**)=0
- virtual void [set](#) (const Glib::ustring &key, int **what**)=0
- virtual void [set](#) (const Glib::ustring &key, double **what**)=0
- virtual void [set](#) (const Glib::ustring &key, const Glib::ustring &**what**)=0
- virtual void [set](#) (const Glib::ustring &key, const [Schema](#) &**what**)=0
- void [set](#) (const Glib::ustring &key, const [ValuePair](#) &pair)
- void [set\\_int\\_list](#) (const Glib::ustring &key, const SListHandle\_ValueInt &list)
- void [set\\_bool\\_list](#) (const Glib::ustring &key, const SListHandle\_ValueBool &list)
- void [set\\_float\\_list](#) (const Glib::ustring &key, const SListHandle\_ValueFloat &list)
- void [set\\_string\\_list](#) (const Glib::ustring &key, const SListHandle\_ValueString &list)
- void [set\\_schema\\_list](#) (const Glib::ustring &key, const SListHandle\_ValueSchema &list)

### 8.6.1 Detailed Description

Common Interface for key-value settable objects.

This class defines a common interface for GConfmm objects that implement the [set\(\)](#) methods for configuration keys. It also provides the implementations for the [set\\_\\*\\_list\(\)](#) family of methods.

The only classes that support this interface are [Client](#) and [ChangeSet](#).

The [set\\_\\*\\_list\(\)](#) methods take as a parameter any STL-compatible container that has the appropriate `value_type`.

### 8.6.2 Member Function Documentation

#### 8.6.2.1 [set\(\)](#) [1/7]

```
virtual void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    bool what ) [pure virtual]
```

Implemented in [Gnome::Conf::ChangeSet](#), and [Gnome::Conf::Client](#).

#### 8.6.2.2 [set\(\)](#) [2/7]

```
virtual void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    const Glib::ustring & what ) [pure virtual]
```

Implemented in [Gnome::Conf::ChangeSet](#), and [Gnome::Conf::Client](#).

### 8.6.2.3 set() [3/7]

```
virtual void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    const Schema & what ) [pure virtual]
```

Implemented in [Gnome::Conf::ChangeSet](#), and [Gnome::Conf::Client](#).

### 8.6.2.4 set() [4/7]

```
virtual void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    const Value & value ) [pure virtual]
```

Implemented in [Gnome::Conf::ChangeSet](#), and [Gnome::Conf::Client](#).

### 8.6.2.5 set() [5/7]

```
void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    const ValuePair & pair )
```

### 8.6.2.6 set() [6/7]

```
virtual void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    double what ) [pure virtual]
```

Implemented in [Gnome::Conf::ChangeSet](#), and [Gnome::Conf::Client](#).

### 8.6.2.7 set() [7/7]

```
virtual void Gnome::Conf::SetInterface::set (
    const Glib::ustring & key,
    int what ) [pure virtual]
```

Implemented in [Gnome::Conf::ChangeSet](#), and [Gnome::Conf::Client](#).

#### 8.6.2.8 set\_bool\_list()

```
void Gnome::Conf::SetInterface::set_bool_list (
    const Glib::ustring & key,
    const SListHandle_ValueBool & list )
```

#### 8.6.2.9 set\_float\_list()

```
void Gnome::Conf::SetInterface::set_float_list (
    const Glib::ustring & key,
    const SListHandle_ValueFloat & list )
```

#### 8.6.2.10 set\_int\_list()

```
void Gnome::Conf::SetInterface::set_int_list (
    const Glib::ustring & key,
    const SListHandle_ValueInt & list )
```

#### 8.6.2.11 set\_schema\_list()

```
void Gnome::Conf::SetInterface::set_schema_list (
    const Glib::ustring & key,
    const SListHandle_ValueSchema & list )
```

#### 8.6.2.12 set\_string\_list()

```
void Gnome::Conf::SetInterface::set_string_list (
    const Glib::ustring & key,
    const SListHandle_ValueString & list )
```

The documentation for this class was generated from the following file:

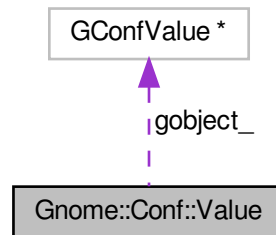
- gconfmm/setinterface.h

## 8.7 Gnome::Conf::Value Class Reference

Wrapper for primitive types.

```
#include <gconfmm/value.h>
```

Collaboration diagram for Gnome::Conf::Value:



### Public Member Functions

- [Value](#) (GConfValue \*castitem, bool make\_a\_copy=false)
- [Value](#) (const [Value](#) &src)
- [Value](#) & [operator=](#) (const [Value](#) &src)
- [~Value](#) ()
- GConfValue \* [gobj](#) ()
- const GConfValue \* [gobj](#) () const
- GConfValue \* [gobj\\_copy](#) () const
- *Provides access to the underlying C instance. The caller is responsible for freeing it. Use when directly setting fields in structs.*
- [Value](#) (ValueType type=VALUE\_INVALID)
- *Create a [Value](#).*
- void [set](#) (gint val)
- *Set the integer value of a [Value](#) whose type is VALUE\_INT.*
- void [set](#) (gdouble val)
- *Set the float value of a [Value](#) whose type is VALUE\_FLOAT.*
- void [set](#) (bool val)
- *Set the boolean value of a [Value](#) whose type is VALUE\_BOOL.*
- void [set](#) (const [Schema](#) &sc)
- *Set the [Schema](#) of a [Value](#) whose type is VALUE\_SCHEMA.*
- void [set\\_car](#) (const [Value](#) &car)
- *Set the car (in a pair, the first element) of a [Value](#) whose type is VALUE\_PAIR.*
- void [set\\_cdr](#) (const [Value](#) &cdr)
- *Set the cdr (in a pair, the second element) of a [Value](#) whose type is VALUE\_PAIR.*
- void [set](#) (const Glib::ustring &val)
- *Set the string of a [Value](#) whose type is VALUE\_STRING.*
- void [set\\_list\\_type](#) (ValueType type)
- *Sets the type of the elements of a [Value](#) with type VALUE\_LIST.*

- void [set\\_int\\_list](#) (const SListHandle\_ValueInt &list)  
*Sets the [Value](#) to contain a list of integers.*
- void [set\\_bool\\_list](#) (const SListHandle\_ValueBool &list)  
*Sets the [Value](#) to contain a list of bools.*
- void [set\\_float\\_list](#) (const SListHandle\_ValueFloat &list)  
*Sets the [Value](#) to contain a list of doubles.*
- void [set\\_string\\_list](#) (const SListHandle\_ValueString &list)  
*Sets the [Value](#) to contain a list of strings.*
- void [set\\_schema\\_list](#) (const SListHandle\_ValueSchema &list)  
*Sets the [Value](#) to contain a list of [Schema](#).*
- [ValueType](#) [get\\_type](#) () const  
*Get the type of the [Value](#).*
- [ValueType](#) [get\\_list\\_type](#) () const  
*Get the type of the list elements of the [Value](#).*
- int [get\\_int](#) () const  
*Get the integer that the [Value](#) contains.*
- bool [get\\_bool](#) () const  
*Get the boolean that the [Value](#) contains.*
- double [get\\_float](#) () const  
*Get the double that the [Value](#) contains.*
- Glib::ustring [get\\_string](#) () const  
*Get the string that the [Value](#) contains.*
- [Schema](#) [get\\_schema](#) () const  
*Get a copy of the [Schema](#) of the value.*
- [Value](#) [get\\_car](#) () const  
*Get a copy of the car of a [VALUE\\_PAIR](#) [Value](#).*
- [Value](#) [get\\_cdr](#) () const  
*Get a copy of the cdr of a [VALUE\\_PAIR](#) [Value](#).*
- SListHandle\_ValueFloat [get\\_float\\_list](#) () const  
*Gets a list of doubles from the [Value](#).*
- SListHandle\_ValueInt [get\\_int\\_list](#) () const  
*Retrieves the list of integers from the [Value](#).*
- SListHandle\_ValueBool [get\\_bool\\_list](#) () const  
*Retrieves the list of booleans from the [Value](#).*
- SListHandle\_ValueString [get\\_string\\_list](#) () const  
*Retrieves the list of strings from the [Value](#).*
- SListHandle\_ValueSchema [get\\_schema\\_list](#) () const  
*Retrieves the list of Schemas from the [Value](#).*
- Glib::ustring [to\\_string](#) () const  
*Convert the [Value](#) to a string.*

## Protected Attributes

- GConfValue \* [gobject\\_](#)

## Related Functions

(Note that these are not member functions.)

- [Gnome::Conf::Value](#) [wrap](#) (GConfValue \*object, bool take\_copy=false)  
*A [Glib::wrap\(\)](#) method for this object.*



### 8.7.1 Detailed Description

Wrapper for primitive types.

This class wraps the primitive types that are passed to and from instances of [Gnome::Conf::Client](#). It has an associated `ValueType`, which is specified at creation time, but can be changed with assignment. If the type is `VALUE_INVALID` then the effect of the set and get methods is undefined. Using a default-constructed [Value](#) without using any of the set methods produces undefined behaviour.

Compound Values of type `VALUE_PAIR` and `VALUE_LIST` can only have elements whose types are neither `VALUE_PAIR` or `VALUE_LIST` - they can only have primitive types.

The [Value](#) class has copy-by-value semantics - all arguments to the set methods are copied.

Note that while the type is named `VALUE_FLOAT`, the accessors for floating-point values use `double`, not `float`, to preserve accuracy.

### 8.7.2 Constructor & Destructor Documentation

#### 8.7.2.1 Value() [1/3]

```
Gnome::Conf::Value::Value (
    GConfValue * castitem,
    bool make_a_copy = false ) [explicit]
```

#### 8.7.2.2 Value() [2/3]

```
Gnome::Conf::Value::Value (
    const Value & src )
```

#### 8.7.2.3 ~Value()

```
Gnome::Conf::Value::~~Value ( )
```

#### 8.7.2.4 Value() [3/3]

```
Gnome::Conf::Value::Value (
    ValueType type = VALUE_INVALID )
```

Create a [Value](#).

You should call a [set\(\)](#) method before using the [Value](#).

**Parameters**

<i>type</i>	The type of the produced value.
-------------	---------------------------------

## 8.7.3 Member Function Documentation

### 8.7.3.1 `get_bool()`

```
bool Gnome::Conf::Value::get_bool ( ) const
```

Get the boolean that the [Value](#) contains.

### 8.7.3.2 `get_bool_list()`

```
SListHandle_ValueBool Gnome::Conf::Value::get_bool_list ( ) const
```

Retrieves the list of booleans from the [Value](#).

See also

[get\\_float\\_list](#)

### 8.7.3.3 `get_car()`

```
Value Gnome::Conf::Value::get_car ( ) const
```

Get a copy of the car of a VALUE\_PAIR [Value](#).

### 8.7.3.4 `get_cdr()`

```
Value Gnome::Conf::Value::get_cdr ( ) const
```

Get a copy of the cdr of a VALUE\_PAIR [Value](#).

### 8.7.3.5 get\_float()

```
double Gnome::Conf::Value::get_float ( ) const
```

Get the double that the [Value](#) contains.

### 8.7.3.6 get\_float\_list()

```
SListHandle_ValueFloat Gnome::Conf::Value::get_float_list ( ) const
```

Gets a list of doubles from the [Value](#).

Typical usage is

```
std::vector<double> foo = value.get_float_list();
```

.

#### Returns

: an STL-compatible container with doubles as its value type. Assign to an **std::vector**, list or deque for proper use.

### 8.7.3.7 get\_int()

```
int Gnome::Conf::Value::get_int ( ) const
```

Get the integer that the [Value](#) contains.

### 8.7.3.8 get\_int\_list()

```
SListHandle_ValueInt Gnome::Conf::Value::get_int_list ( ) const
```

Retrieves the list of integers from the [Value](#).

#### See also

[get\\_float\\_list](#)

#### 8.7.3.9 `get_list_type()`

```
ValueType Gnome::Conf::Value::get_list_type ( ) const
```

Get the type of the list elements of the [Value](#).

Do not call this method on non-list Values.

##### Returns

the type of the list elements.

#### 8.7.3.10 `get_schema()`

```
Schema Gnome::Conf::Value::get_schema ( ) const
```

Get a copy of the [Schema](#) of the value.

#### 8.7.3.11 `get_schema_list()`

```
SListHandle_ValueSchema Gnome::Conf::Value::get_schema_list ( ) const
```

Retrieves the list of Schemas from the [Value](#).

@See `get_float_list`

#### 8.7.3.12 `get_string()`

```
Glib::ustring Gnome::Conf::Value::get_string ( ) const
```

Get the string that the [Value](#) contains.

#### 8.7.3.13 `get_string_list()`

```
SListHandle_ValueString Gnome::Conf::Value::get_string_list ( ) const
```

Retrieves the list of strings from the [Value](#).

##### See also

[get\\_float\\_list](#)

#### 8.7.3.14 get\_type()

```
ValueType Gnome::Conf::Value::get_type ( ) const
```

Get the type of the [Value](#).

##### Returns

the type of the [Value](#)

#### 8.7.3.15 gobj() [1/2]

```
GConfValue * Gnome::Conf::Value::gobj ( ) [inline]
```

#### 8.7.3.16 gobj() [2/2]

```
const GConfValue * Gnome::Conf::Value::gobj ( ) const [inline]
```

#### 8.7.3.17 gobj\_copy()

```
GConfValue * Gnome::Conf::Value::gobj_copy ( ) const
```

Provides access to the underlying C instance. The caller is responsible for freeing it. Use when directly setting fields in structs.

#### 8.7.3.18 operator=()

```
Value & Gnome::Conf::Value::operator= (
    const Value & src )
```

#### 8.7.3.19 set() [1/5]

```
void Gnome::Conf::Value::set (
    bool val )
```

Set the boolean value of a [Value](#) whose type is VALUE\_BOOL.

**8.7.3.20 set()** [2/5]

```
void Gnome::Conf::Value::set (
    const Glib::ustring & val )
```

Set the string of a [Value](#) whose type is VALUE\_STRING.

**8.7.3.21 set()** [3/5]

```
void Gnome::Conf::Value::set (
    const Schema & sc )
```

Set the [Schema](#) of a [Value](#) whose type is VALUE\_SCHEMA.

**8.7.3.22 set()** [4/5]

```
void Gnome::Conf::Value::set (
    gdouble val )
```

Set the float value of a [Value](#) whose type is VALUE\_FLOAT.

**Parameters**

<i>val</i>	the double this <a href="#">Value</a> will be se to.
------------	--

**8.7.3.23 set()** [5/5]

```
void Gnome::Conf::Value::set (
    gint val )
```

Set the integer value of a [Value](#) whose type is VALUE\_INT.

**8.7.3.24 set\_bool\_list()**

```
void Gnome::Conf::Value::set_bool_list (
    const SListHandle_ValueBool & list )
```

Sets the [Value](#) to contain a list of bools.

See also

[set\\_int\\_list](#)

#### 8.7.3.25 set\_car()

```
void Gnome::Conf::Value::set_car (
    const Value & car )
```

Set the car (in a pair, the first element) of a [Value](#) whose type is VALUE\_PAIR.

#### 8.7.3.26 set\_cdr()

```
void Gnome::Conf::Value::set_cdr (
    const Value & cdr )
```

Set the cdr (in a pair, the second element) of a [Value](#) whose type is VALUE\_PAIR.

#### 8.7.3.27 set\_float\_list()

```
void Gnome::Conf::Value::set_float_list (
    const SListHandle_ValueFloat & list )
```

Sets the [Value](#) to contain a list of doubles.

See also

[set\\_int\\_list](#)

#### 8.7.3.28 set\_int\_list()

```
void Gnome::Conf::Value::set_int_list (
    const SListHandle_ValueInt & list )
```

Sets the [Value](#) to contain a list of integers.

set\_list\_type(VALUE\_INT) must have been called prior this call.

##### Parameters

<i>list</i>	an STL-compatible container whose value_type is <code>int</code>
-------------	--

#### 8.7.3.29 set\_list\_type()

```
void Gnome::Conf::Value::set_list_type (
```

```
ValueType type )
```

Sets the type of the elements of a [Value](#) with type VALUE\_LIST.

#### 8.7.3.30 set\_schema\_list()

```
void Gnome::Conf::Value::set_schema_list (
    const SListHandle_ValueSchema & list )
```

Sets the [Value](#) to contain a list of [Schema](#).

See also

[set\\_int\\_list](#)

#### 8.7.3.31 set\_string\_list()

```
void Gnome::Conf::Value::set_string_list (
    const SListHandle_ValueString & list )
```

Sets the [Value](#) to contain a list of strings.

See also

[set\\_int\\_list](#)

#### 8.7.3.32 to\_string()

```
Glib::ustring Gnome::Conf::Value::to_string ( ) const
```

Convert the [Value](#) to a string.

The string is not machine-parseable. Do not depend on the format of the string.

### 8.7.4 Friends And Related Function Documentation

#### 8.7.4.1 wrap()

```
Gnome::Conf::Value wrap (
    GConfValue * object,
    bool take_copy = false ) [related]
```

A Glib::wrap() method for this object.



## Parameters

<i>object</i>	The C instance.
<i>take_copy</i>	False if the result should take ownership of the C instance. True if it should take a new copy or ref.

## Returns

A C++ instance that wraps this C instance.

## 8.7.5 Member Data Documentation

### 8.7.5.1 gobject\_

GConfValue\* Gnome::Conf::Value::gobject\_ [protected]

The documentation for this class was generated from the following file:

- gconfmm/value.h



# Index

- ~ChangeSet
  - Gnome::Conf::ChangeSet, [17](#)
- ~Client
  - Gnome::Conf::Client, [25](#)
- ~Entry
  - Gnome::Conf::Entry, [45](#)
- ~Schema
  - Gnome::Conf::Schema, [53](#)
- ~Value
  - Gnome::Conf::Value, [63](#)
- add\_dir
  - Gnome::Conf::Client, [25](#)
- all\_dirs
  - Gnome::Conf::Client, [25](#)
- all\_entries
  - Gnome::Conf::Client, [26](#)
- BAD\_ADDRESS
  - Gnome::Conf::Error, [50](#)
- Callback
  - Gnome::Conf, [14](#)
- change\_set\_commit
  - Gnome::Conf::Client, [26](#)
- change\_set\_from\_current
  - Gnome::Conf::Client, [27](#)
- change\_set\_reverse
  - Gnome::Conf::Client, [27](#)
- ChangeSet
  - Gnome::Conf::ChangeSet, [17](#)
- clear
  - Gnome::Conf::ChangeSet, [17](#)
- clear\_cache
  - Gnome::Conf::Client, [28](#)
- CLIENT\_HANDLE\_ALL
  - gconfmm Enums and Flags, [12](#)
- CLIENT\_HANDLE\_NONE
  - gconfmm Enums and Flags, [12](#)
- CLIENT\_HANDLE\_UNRETURNED
  - gconfmm Enums and Flags, [12](#)
- CLIENT\_PRELOAD\_NONE
  - gconfmm Enums and Flags, [12](#)
- CLIENT\_PRELOAD\_ONELEVEL
  - gconfmm Enums and Flags, [12](#)
- CLIENT\_PRELOAD\_RECURSIVE
  - gconfmm Enums and Flags, [12](#)
- ClientErrorHandlingMode
  - gconfmm Enums and Flags, [11](#)
- ClientPreloadType
  - gconfmm Enums and Flags, [12](#)
- gconfmm Enums and Flags, [12](#)
  - CLIENT\_HANDLE\_ALL, [12](#)
  - CLIENT\_HANDLE\_NONE, [12](#)
  - CLIENT\_HANDLE\_UNRETURNED, [12](#)
  - CLIENT\_PRELOAD\_NONE, [12](#)
  - CLIENT\_PRELOAD\_ONELEVEL, [12](#)
  - CLIENT\_PRELOAD\_RECURSIVE, [12](#)
  - ClientErrorHandlingMode, [11](#)
  - ClientPreloadType, [12](#)
  - UNSET\_INCLUDING\_SCHEMA\_NAMES, [12](#)
  - UnsetFlags, [12](#)
  - VALUE\_BOOL, [12](#)
  - VALUE\_FLOAT, [12](#)
  - VALUE\_INT, [12](#)
  - VALUE\_INVALID, [12](#)
  - VALUE\_LIST, [12](#)
  - VALUE\_PAIR, [12](#)
  - VALUE\_SCHEMA, [12](#)
  - VALUE\_STRING, [12](#)
  - ValueType, [12](#)
- get
  - Gnome::Conf::Client, [29](#)
- get\_bool
  - Gnome::Conf::Client, [29](#)
  - Gnome::Conf::Value, [64](#)
- Code
  - Gnome::Conf::Error, [50](#)
- code
  - Gnome::Conf::Error, [51](#)
- CORRUPT
  - Gnome::Conf::Error, [50](#)
- dir\_exists
  - Gnome::Conf::Client, [28](#)
- Entry
  - Gnome::Conf::Entry, [45](#)
- Error
  - Gnome::Conf::Error, [50](#)
- error
  - Gnome::Conf::Client, [28](#)
- exists
  - Gnome::Conf::ChangeSet, [18](#)
- for\_each
  - Gnome::Conf::ChangeSet, [18](#)
- ForeachSlot
  - Gnome::Conf::ChangeSet, [17](#)
- gconfmm Enums and Flags, [11](#)
  - CLIENT\_HANDLE\_ALL, [12](#)
  - CLIENT\_HANDLE\_NONE, [12](#)
  - CLIENT\_HANDLE\_UNRETURNED, [12](#)
  - CLIENT\_PRELOAD\_NONE, [12](#)
  - CLIENT\_PRELOAD\_ONELEVEL, [12](#)
  - CLIENT\_PRELOAD\_RECURSIVE, [12](#)
  - ClientErrorHandlingMode, [11](#)
  - ClientPreloadType, [12](#)
  - UNSET\_INCLUDING\_SCHEMA\_NAMES, [12](#)
  - UnsetFlags, [12](#)
  - VALUE\_BOOL, [12](#)
  - VALUE\_FLOAT, [12](#)
  - VALUE\_INT, [12](#)
  - VALUE\_INVALID, [12](#)
  - VALUE\_LIST, [12](#)
  - VALUE\_PAIR, [12](#)
  - VALUE\_SCHEMA, [12](#)
  - VALUE\_STRING, [12](#)
  - ValueType, [12](#)

- get\_bool\_list
  - Gnome::Conf::Client, 29
  - Gnome::Conf::Value, 64
- get\_car
  - Gnome::Conf::Value, 64
- get\_car\_type
  - Gnome::Conf::Schema, 53
- get\_cdr
  - Gnome::Conf::Value, 64
- get\_cdr\_type
  - Gnome::Conf::Schema, 53
- get\_client\_for\_engine
  - Gnome::Conf::Client, 29
- get\_default\_client
  - Gnome::Conf::Client, 29
- get\_default\_from\_schema
  - Gnome::Conf::Client, 30
- get\_default\_value
  - Gnome::Conf::Schema, 53
- get\_entry
  - Gnome::Conf::Client, 30, 31
- get\_float
  - Gnome::Conf::Client, 31
  - Gnome::Conf::Value, 64
- get\_float\_list
  - Gnome::Conf::Client, 32
  - Gnome::Conf::Value, 65
- get\_int
  - Gnome::Conf::Client, 32
  - Gnome::Conf::Value, 65
- get\_int\_list
  - Gnome::Conf::Client, 32
  - Gnome::Conf::Value, 65
- get\_is\_default
  - Gnome::Conf::Entry, 46
- get\_is\_writable
  - Gnome::Conf::Entry, 46
- get\_key
  - Gnome::Conf::Entry, 46
- get\_list\_type
  - Gnome::Conf::Schema, 53
  - Gnome::Conf::Value, 65
- get\_locale
  - Gnome::Conf::Schema, 53
- get\_long\_desc
  - Gnome::Conf::Schema, 54
- get\_owner
  - Gnome::Conf::Schema, 54
- get\_pair
  - Gnome::Conf::Client, 33
- get\_schema
  - Gnome::Conf::Client, 33
  - Gnome::Conf::Value, 66
- get\_schema\_list
  - Gnome::Conf::Client, 33
  - Gnome::Conf::Value, 66
- get\_schema\_name
  - Gnome::Conf::Entry, 46
- get\_short\_desc
  - Gnome::Conf::Schema, 54
- get\_string
  - Gnome::Conf::Client, 33
  - Gnome::Conf::Value, 66
- get\_string\_list
  - Gnome::Conf::Client, 34
  - Gnome::Conf::Value, 66
- get\_type
  - Gnome::Conf::Schema, 54
  - Gnome::Conf::Value, 66
- get\_value
  - Gnome::Conf::Entry, 46
- get\_without\_default
  - Gnome::Conf::Client, 34
- Glib, 13
- Gnome, 13
- Gnome::Conf, 13
  - Callback, 14
  - init, 14
  - ValuePair, 14
  - ValueTypePair, 14
- Gnome::Conf::ChangeSet, 15
  - ~ChangeSet, 17
  - ChangeSet, 17
  - clear, 17
  - exists, 18
  - for\_each, 18
  - ForeachSlot, 17
  - gobj, 18
  - gobj\_copy, 18
  - gobject\_, 20
  - operator=, 18
  - remove, 19
  - set, 19, 20
  - size, 20
  - unset, 20
- Gnome::Conf::Client, 21
  - ~Client, 25
  - add\_dir, 25
  - all\_dirs, 25
  - all\_entries, 26
  - change\_set\_commit, 26
  - change\_set\_from\_current, 27
  - change\_set\_reverse, 27
  - clear\_cache, 28
  - dir\_exists, 28
  - error, 28
  - get, 29
  - get\_bool, 29
  - get\_bool\_list, 29
  - get\_client\_for\_engine, 29
  - get\_default\_client, 29
  - get\_default\_from\_schema, 30
  - get\_entry, 30, 31
  - get\_float, 31
  - get\_float\_list, 32
  - get\_int, 32

- get\_int\_list, 32
- get\_pair, 33
- get\_schema, 33
- get\_schema\_list, 33
- get\_string, 33
- get\_string\_list, 34
- get\_without\_default, 34
- gobj, 34
- gobj\_copy, 35
- key\_is\_writable, 35
- notify, 35
- notify\_add, 35
- notify\_remove, 36
- on\_error, 36
- on\_unreturned\_error, 36
- on\_value\_changed, 37
- preload, 37
- recursive\_unset, 37
- remove\_dir, 38
- set, 38–40
- set\_bool\_list, 41
- set\_error\_handling, 41
- set\_float\_list, 41
- set\_int\_list, 41
- set\_schema\_list, 41
- set\_string\_list, 41
- signal\_error, 41
- signal\_value\_changed, 42
- SListHandleBools, 24
- SListHandleFloats, 24
- SListHandleInts, 24
- suggest\_sync, 42
- unset, 42
- value\_changed, 43
- wrap, 43
- Gnome::Conf::Entry, 43
  - ~Entry, 45
  - Entry, 45
  - get\_is\_default, 46
  - get\_is\_writable, 46
  - get\_key, 46
  - get\_schema\_name, 46
  - get\_value, 46
  - gobj, 46, 47
  - gobj\_copy, 47
  - gobject\_, 48
  - operator=, 47
  - set\_is\_default, 47
  - set\_is\_writable, 47
  - set\_schema\_name, 47
  - set\_value, 48
  - wrap, 48
- Gnome::Conf::Error, 49
  - BAD\_ADDRESS, 50
  - Code, 50
  - code, 51
  - CORRUPT, 50
  - Error, 50
  - IN\_SHUTDOWN, 50
  - IS\_DIR, 50
  - IS\_KEY, 50
  - LOCAL\_ENGINE, 50
  - LOCK\_FAILED, 50
  - NO\_PERMISSION, 50
  - NO\_SERVER, 50
  - NO\_WRITABLE\_DATABASE, 50
  - OAF\_ERROR, 50
  - OVERRIDDEN, 50
  - PARSE\_ERROR, 50
  - SUCCESS, 50
  - TYPE\_MISMATCH, 50
- Gnome::Conf::Schema, 51
  - ~Schema, 53
  - get\_car\_type, 53
  - get\_cdr\_type, 53
  - get\_default\_value, 53
  - get\_list\_type, 53
  - get\_locale, 53
  - get\_long\_desc, 54
  - get\_owner, 54
  - get\_short\_desc, 54
  - get\_type, 54
  - gobj, 54
  - gobj\_copy, 54
  - gobject\_, 57
  - operator=, 55
  - Schema, 52, 53
  - set\_car\_type, 55
  - set\_cdr\_type, 55
  - set\_default\_value, 55
  - set\_list\_type, 55
  - set\_locale, 55
  - set\_long\_desc, 56
  - set\_owner, 56
  - set\_short\_desc, 56
  - set\_type, 56
  - wrap, 56
- Gnome::Conf::SetInterface, 57
  - set, 58, 59
  - set\_bool\_list, 59
  - set\_float\_list, 60
  - set\_int\_list, 60
  - set\_schema\_list, 60
  - set\_string\_list, 60
- Gnome::Conf::Value, 61
  - ~Value, 63
  - get\_bool, 64
  - get\_bool\_list, 64
  - get\_car, 64
  - get\_cdr, 64
  - get\_float, 64
  - get\_float\_list, 65
  - get\_int, 65
  - get\_int\_list, 65
  - get\_list\_type, 65
  - get\_schema, 66

- get\_schema\_list, 66
- get\_string, 66
- get\_string\_list, 66
- get\_type, 66
- gobj, 67
- gobj\_copy, 67
- gobject\_, 71
- operator=, 67
- set, 67, 68
- set\_bool\_list, 68
- set\_car, 68
- set\_cdr, 69
- set\_float\_list, 69
- set\_int\_list, 69
- set\_list\_type, 69
- set\_schema\_list, 70
- set\_string\_list, 70
- to\_string, 70
- Value, 63
- wrap, 70
- gobj
  - Gnome::Conf::ChangeSet, 18
  - Gnome::Conf::Client, 34
  - Gnome::Conf::Entry, 46, 47
  - Gnome::Conf::Schema, 54
  - Gnome::Conf::Value, 67
- gobj\_copy
  - Gnome::Conf::ChangeSet, 18
  - Gnome::Conf::Client, 35
  - Gnome::Conf::Entry, 47
  - Gnome::Conf::Schema, 54
  - Gnome::Conf::Value, 67
- gobject\_
  - Gnome::Conf::ChangeSet, 20
  - Gnome::Conf::Entry, 48
  - Gnome::Conf::Schema, 57
  - Gnome::Conf::Value, 71
- IN\_SHUTDOWN
  - Gnome::Conf::Error, 50
- init
  - Gnome::Conf, 14
- IS\_DIR
  - Gnome::Conf::Error, 50
- IS\_KEY
  - Gnome::Conf::Error, 50
- key\_is\_writable
  - Gnome::Conf::Client, 35
- LOCAL\_ENGINE
  - Gnome::Conf::Error, 50
- LOCK\_FAILED
  - Gnome::Conf::Error, 50
- NO\_PERMISSION
  - Gnome::Conf::Error, 50
- NO\_SERVER
  - Gnome::Conf::Error, 50
- NO\_WRITABLE\_DATABASE
  - Gnome::Conf::Error, 50
- notify
  - Gnome::Conf::Client, 35
- notify\_add
  - Gnome::Conf::Client, 35
- notify\_remove
  - Gnome::Conf::Client, 36
- OAF\_ERROR
  - Gnome::Conf::Error, 50
- on\_error
  - Gnome::Conf::Client, 36
- on\_unreturned\_error
  - Gnome::Conf::Client, 36
- on\_value\_changed
  - Gnome::Conf::Client, 37
- operator=
  - Gnome::Conf::ChangeSet, 18
  - Gnome::Conf::Entry, 47
  - Gnome::Conf::Schema, 55
  - Gnome::Conf::Value, 67
- OVERRIDDEN
  - Gnome::Conf::Error, 50
- PARSE\_ERROR
  - Gnome::Conf::Error, 50
- preload
  - Gnome::Conf::Client, 37
- recursive\_unset
  - Gnome::Conf::Client, 37
- remove
  - Gnome::Conf::ChangeSet, 19
- remove\_dir
  - Gnome::Conf::Client, 38
- Schema
  - Gnome::Conf::Schema, 52, 53
- set
  - Gnome::Conf::ChangeSet, 19, 20
  - Gnome::Conf::Client, 38–40
  - Gnome::Conf::SetInterface, 58, 59
  - Gnome::Conf::Value, 67, 68
- set\_bool\_list
  - Gnome::Conf::Client, 41
  - Gnome::Conf::SetInterface, 59
  - Gnome::Conf::Value, 68
- set\_car
  - Gnome::Conf::Value, 68
- set\_car\_type
  - Gnome::Conf::Schema, 55
- set\_cdr
  - Gnome::Conf::Value, 69
- set\_cdr\_type
  - Gnome::Conf::Schema, 55
- set\_default\_value
  - Gnome::Conf::Schema, 55
- set\_error\_handling

- Gnome::Conf::Client, 41
- set\_float\_list
  - Gnome::Conf::Client, 41
  - Gnome::Conf::SetInterface, 60
  - Gnome::Conf::Value, 69
- set\_int\_list
  - Gnome::Conf::Client, 41
  - Gnome::Conf::SetInterface, 60
  - Gnome::Conf::Value, 69
- set\_is\_default
  - Gnome::Conf::Entry, 47
- set\_is\_writable
  - Gnome::Conf::Entry, 47
- set\_list\_type
  - Gnome::Conf::Schema, 55
  - Gnome::Conf::Value, 69
- set\_locale
  - Gnome::Conf::Schema, 55
- set\_long\_desc
  - Gnome::Conf::Schema, 56
- set\_owner
  - Gnome::Conf::Schema, 56
- set\_schema\_list
  - Gnome::Conf::Client, 41
  - Gnome::Conf::SetInterface, 60
  - Gnome::Conf::Value, 70
- set\_schema\_name
  - Gnome::Conf::Entry, 47
- set\_short\_desc
  - Gnome::Conf::Schema, 56
- set\_string\_list
  - Gnome::Conf::Client, 41
  - Gnome::Conf::SetInterface, 60
  - Gnome::Conf::Value, 70
- set\_type
  - Gnome::Conf::Schema, 56
- set\_value
  - Gnome::Conf::Entry, 48
- signal\_error
  - Gnome::Conf::Client, 41
- signal\_value\_changed
  - Gnome::Conf::Client, 42
- size
  - Gnome::Conf::ChangeSet, 20
- SListHandleBools
  - Gnome::Conf::Client, 24
- SListHandleFloats
  - Gnome::Conf::Client, 24
- SListHandleInts
  - Gnome::Conf::Client, 24
- SUCCESS
  - Gnome::Conf::Error, 50
- suggest\_sync
  - Gnome::Conf::Client, 42
- to\_string
  - Gnome::Conf::Value, 70
- TYPE\_MISMATCH
  - Gnome::Conf::Error, 50
- unset
  - Gnome::Conf::ChangeSet, 20
  - Gnome::Conf::Client, 42
- UNSET\_INCLUDING\_SCHEMA\_NAMES
  - gconfmm Enums and Flags, 12
- UnsetFlags
  - gconfmm Enums and Flags, 12
- Value
  - Gnome::Conf::Value, 63
- VALUE\_BOOL
  - gconfmm Enums and Flags, 12
- value\_changed
  - Gnome::Conf::Client, 43
- VALUE\_FLOAT
  - gconfmm Enums and Flags, 12
- VALUE\_INT
  - gconfmm Enums and Flags, 12
- VALUE\_INVALID
  - gconfmm Enums and Flags, 12
- VALUE\_LIST
  - gconfmm Enums and Flags, 12
- VALUE\_PAIR
  - gconfmm Enums and Flags, 12
- VALUE\_SCHEMA
  - gconfmm Enums and Flags, 12
- VALUE\_STRING
  - gconfmm Enums and Flags, 12
- ValuePair
  - Gnome::Conf, 14
- ValueType
  - gconfmm Enums and Flags, 12
- ValueTypePair
  - Gnome::Conf, 14
- wrap
  - Gnome::Conf::Client, 43
  - Gnome::Conf::Entry, 48
  - Gnome::Conf::Schema, 56
  - Gnome::Conf::Value, 70