libvalhalla

2.1.0

Generated by Doxygen 1.7.6.1

Sun Aug 12 2012 12:56:58

Contents

1 External Metadata

1.1 External Metadata

See also

valhalla_db_metadata_insert().
valhalla_db_metadata_update().
valhalla_db_metadata_delete().
valhalla_db_metadata_priority() (only 6.).

1. A data inserted/updated by these functions can not be updated by Valhalla.
2. The metadata are only inserted/updated and deleted in the database, the tags in the files are not modified.
3. If a metadata is changed in a file, a new metadata will be inserted by Valhalla but your entries (inserted or updated by these functions) will not be altered (consequence, you can have duplicated informations if the value is not exactly the same).
4. If a metadata was already inserted by Valhalla and you use these functions to insert or to update the same entry, this metadata will be changed to be considered like an external metadata (see point 1).
5. If a file is no longer available, when Valhalla removes all metadata, the metadata inserted and updated with these functions are removed too.
6. If valhalla_uninit() is called shortly after one of these functions, there is no guarenteed that the metadata is handled.

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

4 Data Structure Documentation

4.1 grabber_list_t Struct Reference

Structure for a grabber.

```
#include <grabber_common.h>
```

Data Fields

- const char ∗ name

  Textual identification of the grabber.
- int caps_flag

  Flags to define the capabilities of the grabber.
- int(∗ init)(void ∗priv, const grabber_param_t ∗param)

  Init function for the grabber.
- void(∗ uninit)(void ∗priv)

  Uninit function for the grabber.
- int(∗ grab)(void ∗priv, file_data_t ∗data)

  Grabbing function for the grabber.
- void(∗ loop)(void ∗priv)

  Function called for each end of scan loop.
- void ∗ priv

  Private data for the grabber.
- grabber_param_t param

  Parameters for the grabber.

4.1.1 Detailed Description

Structure for a grabber.

Definition at line 91 of file grabber_common.h.

4.1.2 Field Documentation

4.1.2.1 int grabber_list_t::caps_flag

Flags to define the capabilities of the grabber.

Definition at line 97 of file grabber_common.h.

4.1.2.2 int(∗ grabber_list_t::grab)(void ∗priv, file_data_t ∗data)

Grabbing function for the grabber.

This function is called in order to populate the attributes `meta_grabber` and `list_downloader` in the `data` structure. All others attributes must be considered as read-only! Only them are thread-safe for writing.

To add new metadata in the database, the function vh_metadata_add() must be used on `meta_grabber`.

It is proibited to download files (images for example) with this function. Only textual metadata are proceeded here. But the reference on an image can be saved in the meta_grabber attribute. To download a file, the URL and the destination must be prepared for the downloader with the function vh_file_dl_add() in order to populate the `list_downloader` attribute. The files will be downloaded after the grabbing step.

To read `meta_parser` (attribute populated by the parser), you must use the function vh_metadata_get().

Parameters

in	*priv*	Private structure registered with the grabber.
in	*data*	File structure where some data must be populated.

Returns

0 for success, != 0 on error.

Definition at line 147 of file grabber_common.h.

4.1.2.3 int(∗ grabber_list_t::init)(void ∗priv, const grabber_param_t ∗param)

Init function for the grabber.

This initialization is called only at the init of an instance of valhalla. The private structure (`priv`) must be created before this initialization.

Parameters

in	*priv*	Private structure registered with the grabber.
in	*param*	Parameters, see grabber_param_t.

Returns

0 for success, != 0 on error.

Definition at line 109 of file grabber_common.h.

4.1.2.4 void(∗ grabber_list_t::loop)(void ∗priv)

Function called for each end of scan loop.

This function is optional, it is called after each scanner loop if there are more than one loop. This function is never called after the last loop. It is useful to make some cleanup in the grabber before the next scan.

Parameters

in	*priv*	Private structure registered with the grabber.

Definition at line 158 of file grabber_common.h.

4.1.2.5 const char∗ grabber_list_t::name

Textual identification of the grabber.

Definition at line 95 of file grabber_common.h.

4.1.2.6 grabber_param_t grabber_list_t::param

Parameters for the grabber.

Definition at line 168 of file grabber_common.h.

4.1.2.7 void∗ grabber_list_t::priv

Private data for the grabber.

The data is registered at the same time that the grabber.

Definition at line 165 of file grabber_common.h.

4.1.2.8 void(∗ grabber_list_t::uninit)(void ∗priv)

Uninit function for the grabber.

This unititialization is called only at the uninit of an instance of valhalla. The private structure (`priv`) must be released in this function.

Parameters

`in`	*priv*	Private structure registered with the grabber.

Definition at line 120 of file grabber_common.h.

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

- grabber_common.h

4.2 grabber_param_t Struct Reference

Structure for the init of a grabber.

```
#include <grabber_common.h>
```

Data Fields

- metadata_plist_t ∗ pl

  List of priorities for metadata.
- struct url_ctl_s ∗ url_ctl

  This pointer is intended to be used with all vh_url_new().

4.2.1 Detailed Description

Structure for the init of a grabber.

Definition at line 81 of file grabber_common.h.

4.2.2 Field Documentation

4.2.2.1 metadata_plist_t* grabber_param_t::pl

List of priorities for metadata.

Definition at line 83 of file grabber_common.h.

4.2.2.2 struct url_ctl_s* grabber_param_t::url_ctl

This pointer is intended to be used with all vh_url_new().

Definition at line 85 of file grabber_common.h.

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

- grabber_common.h

4.3 valhalla_db_fileres_t Struct Reference

Results for valhalla_db_filelist_get().

```
#include <valhalla.h>
```

4.3.1 Detailed Description

Results for valhalla_db_filelist_get().

Definition at line 854 of file valhalla.h.

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

- valhalla.h

4.4 valhalla_db_item_t Struct Reference

Main structure to search in the DB.

```
#include <valhalla.h>
```

4.4.1 Detailed Description

Main structure to search in the DB.

Definition at line 835 of file valhalla.h.

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

- valhalla.h

4.5 valhalla_db_metares_t Struct Reference

Results for valhalla_db_metalist_get().

```
#include <valhalla.h>
```

4.5.1 Detailed Description

Results for valhalla_db_metalist_get().

Definition at line 845 of file valhalla.h.

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

- valhalla.h

4.6 valhalla_db_restrict_t Struct Reference

Restriction.

```
#include <valhalla.h>
```

4.6.1 Detailed Description

Restriction.

Definition at line 861 of file valhalla.h.

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

- valhalla.h

4.7 valhalla_file_t Struct Reference

File structure for general purpose.

```
#include <valhalla.h>
```

4.7.1 Detailed Description

File structure for general purpose.

Definition at line 346 of file valhalla.h.

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

- valhalla.h

4.8 valhalla_init_param_t Struct Reference

Parameters for valhalla_init().

```
#include <valhalla.h>
```

Data Fields

- unsigned int parser_nb
- unsigned int grabber_nb
- unsigned int commit_int
- unsigned int decrapifier: 1
- unsigned int od_meta: 1
- void(∗ od_cb)(const char ∗file, valhalla_event_od_t e, const char ∗id, void ∗data)
- void ∗ od_data
- void(∗ gl_cb)(valhalla_event_gl_t e, void ∗data)
- void ∗ gl_data
- void(∗ md_cb)(valhalla_event_md_t e, const char ∗id, const valhalla_file_t ∗file, const valhalla_metadata_t ∗md, void ∗data)
- void ∗ md_data

4.8.1 Detailed Description

Parameters for valhalla_init().

Definition at line 495 of file valhalla.h.

4.8.2 Field Documentation

4.8.2.1 unsigned int **valhalla_init_param_t::commit_int**

Number of data (set of metadata) to be inserted or updated in one pass in the database (BEGIN and COMMIT sql mechanisms). A value between 100 and 200 is a good choice. The default interval is 128.

Definition at line 516 of file valhalla.h.

4.8.2.2 unsigned int **valhalla_init_param_t::decrapifier**

If the "title" metadata is not available with a file, the decrapifier can be used to create this metadata by using the filename. This feature is very useful when the grabbing support is enabled, because the title is used as keywords in a lot of grabbers. By default the decrapifier is disabled.

Definition at line 524 of file valhalla.h.

4.8.2.3 void(∗ valhalla_init_param_t::gl_cb)(valhalla_event_gl_t e, void ∗data)

When `gl_cb` is defined, events can be sent by Valhalla according to some global actions. See valhalla_event_gl_t for details on the events.

Definition at line 551 of file valhalla.h.

4.8.2.4 void∗ valhalla_init_param_t::gl_data

User data for global event callback.

Definition at line 553 of file valhalla.h.

4.8.2.5 unsigned int valhalla_init_param_t::grabber_nb

Number of threads for grabbing (max 16); the grabbers are concurrent as long as their ID are different. The default number of threads is 2. To use many threads will not increase a lot the use of memory, but it can increase significantly the use of the bandwidth for Internet and the CPU load. Set this parameter to 1, in order to serialize the calls on the grabbers. A value of 3 or 4 is a good choice for most of the uses.

Definition at line 510 of file valhalla.h.

4.8.2.6 void(∗ valhalla_init_param_t::md_cb)(valhalla_event_md_t e, const char ∗id, const valhalla_file_t ∗file, const valhalla_metadata_t ∗md, void ∗data)

When `md_cb` is defined, events can be sent by Valhalla each time that a file metadata set is completed. Where `id` is the textual identifier (for example: "amazon", "exif", etc, ...) of the grabber when the event `e` is VALHALLA_EVENTMD_GRABBER. This callback is called for each metadata. If there are 10 metadata in one set, then this callback is called 10 times. The use of this callback is not recommanded. It may increase significantly the use of memory because all metadata are kept (and duplicated when it comes from the parser) until a set is fully read.

Definition at line 565 of file valhalla.h.

4.8.2.7 void∗ valhalla_init_param_t::md_data

User data for metadata event callback.

Definition at line 569 of file valhalla.h.

4.8.2.8 void(∗ valhalla_init_param_t::od_cb)(const char ∗file, valhalla_event_od_t e, const char ∗id, void ∗data)

When `od_cb` is defined, an event is sent for each step with an on demand query. If an event arrives, the data are really inserted in the DB. The order for the events is not determinative, VALHALLA_EVENTOD_GRABBED can be sent before VALHALLA_E-VENTOD_PARSED. VALHALLA_EVENTOD_GRABBED is sent for each grabber and `id` is its textual identifier (for example: "amazon", "exif", etc, ...). Only VALHALLA_E-VENTOD_ENDED is always sent at the end, but this one has not a high priority unlike other events. If the file is already (fully) inserted in the DB, only VALHALLA_EVENTO-D_ENDED is sent to the callback.

Definition at line 542 of file valhalla.h.

4.8.2.9 void∗ valhalla_init_param_t::od_data

User data for ondemand callback.

Definition at line 545 of file valhalla.h.

4.8.2.10 unsigned int valhalla_init_param_t::od_meta

If the attribute is set, then the meta keys can be retrieved from the ondemand callback by using the function valhalla_ondemand_cb_meta().

Definition at line 529 of file valhalla.h.

4.8.2.11 unsigned int valhalla_init_param_t::parser_nb

Number of threads for parsing (max 8); the parsers are concurrent. The default number of threads is 2.

Definition at line 500 of file valhalla.h.

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

- valhalla.h

4.9 valhalla_metadata_t Struct Reference

Metadata structure for general purpose.

`#include <valhalla.h>`

4.9.1 Detailed Description

Metadata structure for general purpose.

Definition at line 338 of file valhalla.h.

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

- valhalla.h

5 File Documentation

5.1 grabber_common.h File Reference

`#include <pthread.h>` `#include <string.h>` `#include "stats.h"` `#include "utils.h"`

Data Structures

- struct grabber_param_t

  Structure for the init of a grabber.
- struct grabber_list_t

  Structure for a grabber.

Defines

- #define GRABBER_REGISTER(p_name, p_caps, p_pl, p_tw,fct_priv, fct_init, fct-_uninit, fct_grab, fct_loop)

  Macro to register and populate a grabber structure.

Flags for the capabilities of the grabbers.

- #define GRABBER_CAP_AUDIO (1 << 0)

  grab for audio files
- #define GRABBER_CAP_VIDEO (1 << 1)

  grab for video files
- #define GRABBER_CAP_IMAGE (1 << 2)

  grab for image files

5.1.1 Detailed Description

GeeXboX Valhalla Grabber private API header.

To add a new grabber, a good approach is to copy an existing grabber like grabber_-dummy.[ch] in order to have at least the structure. A grabber must not use global/static variables. A grabber must be thread-safe in the case where more than one instance of Valhalla are running concurrently. But, the functions in one grabber are not called in concurrency in one instance of Valhalla.

Some others points to consider:

- grabber_list_t::init() and grabber_list_t::uninit() functions are called only one time by a Valhalla instance.
- grabber_list_t::grab() function is called only between the grabber_list_t::init() and grabber_list_t::uninit() functions.
- grabber_list_t::loop() function is called only between two scanner loops. The function is not called if only one loop is configured for the instance of Valhalla.

Some utils are available for the grabbers:

- grabber_utils.h utils specific to grabbers
- xml_utils.h for XML parsing (based on libxml2)
- url_utils.h for downloading (based on libcurl)

- logs.h for logging
- md5.h to compute the MD5 sum
- list.h to handle very simple linked-lists

Main header for all grabbers:

- grabber_common.h
  - metadata.h to save metadata retrieved by grabber_list_t::grab()
  - utils.h to prepare files (images, ...) for downloading

See also

grabber_list_t for details on the functions.

Definition in file grabber_common.h.

5.1.2 Define Documentation

5.1.2.1 #define GRABBER_CAP_AUDIO (1 $<<$ 0)

grab for audio files

Definition at line 71 of file grabber_common.h.

5.1.2.2 #define GRABBER_CAP_IMAGE (1 $<<$ 2)

grab for image files

Definition at line 73 of file grabber_common.h.

5.1.2.3 #define GRABBER_CAP_VIDEO (1 $<<$ 1)

grab for video files

Definition at line 72 of file grabber_common.h.

5.1.2.4 #define GRABBER_REGISTER(*p_name, p_caps, p_pl, p_tw, fct_priv, fct_init, fct_uninit, fct_grab, fct_loop*)

Macro to register and populate a grabber structure.

See struct grabber_list_s for more informations on the structure and the functions.

Parameters

`in`	*p_name*	Grabber's name.
`in`	*p_caps*	Capabilities flags.
`in`	*p_pl*	List of metadata priorities.
`in`	*p_tw*	Min time to wait [ms] between grabber_list_t::grab().

in	*fct_priv*	Function to retrieve the private data pointer.
in	*fct_init*	grabber_list_t::init().
in	*fct_uninit*	grabber_list_t::uninit().
in	*fct_grab*	grabber_list_t::grab().
in	*fct_loop*	grabber_list_t::loop().

Definition at line 209 of file grabber_common.h.

5.2 valhalla.h File Reference

`#include <inttypes.h> #include <stdarg.h>`

Data Structures

- struct valhalla_metadata_t

  Metadata structure for general purpose.
- struct valhalla_file_t

  File structure for general purpose.
- struct valhalla_init_param_t

  Parameters for valhalla_init().
- struct valhalla_db_item_t

  Main structure to search in the DB.
- struct valhalla_db_metares_t

  Results for valhalla_db_metalist_get().
- struct valhalla_db_fileres_t

  Results for valhalla_db_filelist_get().
- struct valhalla_db_restrict_t

  Restriction.

Defines

- #define VH_CFG_RANGE 8
- #define VH_VOID_T (0 << VH_CFG_RANGE)
- #define VH_VOIDP_T (1 << VH_CFG_RANGE)
- #define VH_INT_T (2 << VH_CFG_RANGE)
- #define VH_VOIDP_2_T (4 << VH_CFG_RANGE)
- #define VH_CFG_INIT(name, type, num) VALHALLA_CFG_##name = ((type) + (num))

  Macro to init items in valhalla_cfg_t.

List of common metadata.

- #define **VALHALLA_METADATA_CATEGORY** "category"

- #define **VALHALLA_METADATA_EPISODE** "episode"
- #define **VALHALLA_METADATA_GENRE** "genre"
- #define **VALHALLA_METADATA_MPAA** "mpaa"
- #define **VALHALLA_METADATA_RUNTIME** "runtime"
- #define **VALHALLA_METADATA_SEASON** "season"
- #define **VALHALLA_METADATA_SYNOPSIS** "synopsis"
- #define **VALHALLA_METADATA_SYNOPSIS_SHOW** "synopsis_show"
- #define **VALHALLA_METADATA_BUDGET** "budget"
- #define **VALHALLA_METADATA_COUNTRY** "country"
- #define **VALHALLA_METADATA_REVENUE** "revenue"
- #define **VALHALLA_METADATA_STUDIO** "studio"
- #define **VALHALLA_METADATA_ACTOR** "actor"
- #define **VALHALLA_METADATA_ARTIST** "artist"
- #define **VALHALLA_METADATA_AUTHOR** "author"
- #define **VALHALLA_METADATA_CASTING** "casting"
- #define **VALHALLA_METADATA_COMPOSER** "composer"
- #define **VALHALLA_METADATA_CREDITS** "credits"
- #define **VALHALLA_METADATA_DIRECTOR** "director"
- #define **VALHALLA_METADATA_DIRECTOR_PHOTO** "director_photo"
- #define **VALHALLA_METADATA_EDITOR** "editor"
- #define **VALHALLA_METADATA_PRODUCER** "producer"
- #define **VALHALLA_METADATA_WRITER** "writer"
- #define **VALHALLA_METADATA_COVER** "cover"
- #define **VALHALLA_METADATA_COVER_SEASON** "cover_season"
- #define **VALHALLA_METADATA_COVER_SHOW** "cover_show"
- #define **VALHALLA_METADATA_COVER_SHOW_HEADER** "cover_show-_header"
- #define **VALHALLA_METADATA_FAN_ART** "fanart"
- #define **VALHALLA_METADATA_LYRICS** "lyrics"
- #define **VALHALLA_METADATA_THUMBNAIL** "thumbnail"
- #define **VALHALLA_METADATA_TRACK** "track"
- #define **VALHALLA_METADATA_PLAY_COUNT** "playcount"
- #define **VALHALLA_METADATA_RATING** "rating"
- #define **VALHALLA_METADATA_WATCHED** "watched"
- #define **VALHALLA_METADATA_AUDIO_BITRATE** "audio_bitrate"
- #define **VALHALLA_METADATA_AUDIO_CHANNELS** "audio_channels"
- #define **VALHALLA_METADATA_AUDIO_CODEC** "audio_codec"
- #define **VALHALLA_METADATA_AUDIO_LANG** "audio_lang"
- #define **VALHALLA_METADATA_AUDIO_STREAMS** "audio_streams"
- #define **VALHALLA_METADATA_DURATION** "duration"
- #define **VALHALLA_METADATA_FILESIZE** "filesize"
- #define **VALHALLA_METADATA_HEIGHT** "height"
- #define **VALHALLA_METADATA_PICTURE_ORIENTATION** "picture_-orientation"
- #define **VALHALLA_METADATA_SUB_LANG** "sub_lang"
- #define **VALHALLA_METADATA_SUB_STREAMS** "sub_streams"
- #define **VALHALLA_METADATA_VIDEO_ASPECT** "video_aspect"
- #define **VALHALLA_METADATA_VIDEO_BITRATE** "video_bitrate"
- #define **VALHALLA_METADATA_VIDEO_CODEC** "video_codec"
- #define **VALHALLA_METADATA_VIDEO_STREAMS** "video_streams"
- #define **VALHALLA_METADATA_WIDTH** "width"
- #define **VALHALLA_METADATA_DATE** "date"
- #define **VALHALLA_METADATA_PREMIERED** "premiered"
- #define **VALHALLA_METADATA_YEAR** "year"
- #define **VALHALLA_METADATA_ALBUM** "album"

- #define **VALHALLA_METADATA_TITLE** "title"
- #define **VALHALLA_METADATA_TITLE_ALTERNATIVE** "title_alternative"
- #define **VALHALLA_METADATA_TITLE_SHOW** "title_show"
- #define **VALHALLA_METADATA_TITLE_STREAM** "title_stream"

Macros for selection functions handling.

- #define VALHALLA_DB_SEARCH(id, txt, g, t, l, p)

  Set valhalla_db_item_t local variable.
- #define VALHALLA_DB_RESTRICT(op, m_id, d_id, m_txt, d_txt, m_t, d_t, l, p)

  Set valhalla_db_restrict_t local variable.
- #define VALHALLA_DB_SEARCH_ID(meta_id, group, l, p) VALHALLA_DB_-SEARCH (meta_id, NULL, group, ID, l, p)

  Set valhalla_db_item_t local variable for an id.
- #define VALHALLA_DB_SEARCH_TEXT(meta_name, group, l, p) VALHALL-A_DB_SEARCH (0, meta_name, group, TEXT, l, p)

  Set valhalla_db_item_t local variable for a text.
- #define VALHALLA_DB_SEARCH_GRP(group, l, p) VALHALLA_DB_SEAR-CH (0, NULL, group, GROUP, l, p)

  Set valhalla_db_item_t local variable for a group.
- #define VALHALLA_DB_RESTRICT_INT(op, meta, data, l, p) VALHALLA_D-B_RESTRICT (op, meta, data, NULL, NULL, ID, ID, l, p)

  Set valhalla_db_restrict_t local variable for meta.id, data.id.
- #define VALHALLA_DB_RESTRICT_STR(op, meta, data, l, p) VALHALLA_-DB_RESTRICT (op, 0, 0, meta, data, TEXT, TEXT, l, p)

  Set valhalla_db_restrict_t local variable for meta.text, data.text.
- #define VALHALLA_DB_RESTRICT_INTSTR(op, meta, data, l, p) VALHALL-A_DB_RESTRICT (op, meta, 0, NULL, data, ID, TEXT, l, p)

  Set valhalla_db_restrict_t local variable for meta.id, data.text.
- #define VALHALLA_DB_RESTRICT_STRINT(op, meta, data, l, p) VALHALL-A_DB_RESTRICT (op, 0, data, meta, NULL, TEXT, ID, l, p)

  Set valhalla_db_restrict_t local variable for meta.text, data.id.
- #define VALHALLA_DB_RESTRICT_LINK(from, to) do {(to).next = &(from);} while (0)

  Link two valhalla_db_restrict_t variables together.

Typedefs

- typedef struct valhalla_s valhalla_t

  Scanner handle.
- typedef struct valhalla_db_stmt_s valhalla_db_stmt_t

  Prepared statement.

Enumerations

- enum valhalla_lang_t { VALHALLA_LANG_ALL = -1, VALHALLA_LANG_UNDE-F = 0, VALHALLA_LANG_DE, VALHALLA_LANG_EN, VALHALLA_LANG_ES, VALHALLA_LANG_FR, VALHALLA_LANG_IT }

Languages for metadata.

- enum valhalla_meta_grp_t { VALHALLA_META_GRP_NIL = 0, VALHALLA_META_GRP_CLASSIFICATION, VALHALLA_META_GRP_COMMERCIAL, VALHALLA_META_GRP_CONTACT, VALHALLA_META_GRP_ENTITIES, VALHALLA_META_GRP_IDENTIFIER, VALHALLA_META_GRP_LEGAL, VALHALLA_META_GRP_MISCELLANEOUS, VALHALLA_META_GRP_MUSICAL, VALHALLA_META_GRP_ORGANIZATIONAL, VALHALLA_META_GRP_PERSONAL, VALHALLA_META_GRP_SPACIAL, VALHALLA_META_GRP_TECHNICAL, VALHALLA_META_GRP_TEMPORAL, VALHALLA_META_GRP_TITLES }

  Groups for metadata.
- enum valhalla_errno { VALHALLA_ERROR_DEAD = -4, VALHALLA_ERROR_PATH = -3, VALHALLA_ERROR_HANDLER = -2, VALHALLA_ERROR_THREAD = -1, VALHALLA_SUCCESS = 0 }

  Error code returned by valhalla_run().
- enum valhalla_verb_t { VALHALLA_MSG_NONE, VALHALLA_MSG_VERBOSE, VALHALLA_MSG_INFO, VALHALLA_MSG_WARNING, VALHALLA_MSG_ERROR, VALHALLA_MSG_CRITICAL }

  Verbosity level.
- enum valhalla_dl_t { VALHALLA_DL_DEFAULT = 0, VALHALLA_DL_COVER, VALHALLA_DL_THUMBNAIL, VALHALLA_DL_FAN_ART }

  Destinations for downloading.
- enum valhalla_event_od_t { VALHALLA_EVENTOD_PARSED = 0, VALHALLA_EVENTOD_GRABBED, VALHALLA_EVENTOD_ENDED }

  Events for valhalla_ondemand() callback.
- enum valhalla_event_gl_t { VALHALLA_EVENTGL_SCANNER_BEGIN = 0, VALHALLA_EVENTGL_SCANNER_END, VALHALLA_EVENTGL_SCANNER_SLEEP, VALHALLA_EVENTGL_SCANNER_ACKS, VALHALLA_EVENTGL_SCANNER_EXIT }

  Events for general actions in Valhalla.
- enum valhalla_event_md_t { VALHALLA_EVENTMD_PARSER = 0, VALHALLA_EVENTMD_GRABBER }

  Events for metadata callback.
- enum valhalla_stats_type_t { VALHALLA_STATS_TIMER = 0, VALHALLA_STATS_COUNTER }

  Type of statistic.
- enum valhalla_metadata_pl_t { VALHALLA_METADATA_PL_HIGHEST = -128, VALHALLA_METADATA_PL_HIGHER = -96, VALHALLA_METADATA_PL_HIGH = -64, VALHALLA_METADATA_PL_ABOVE = -32, VALHALLA_METADATA_PL_NORMAL = 0, VALHALLA_METADATA_PL_BELOW = 32, VALHALLA_METADATA_PL_LOW = 64, VALHALLA_METADATA_PL_LOWER = 96, VALHALLA_METADATA_PL_LOWEST = 128 }

  Priorities for the metadata.
- enum valhalla_cfg_t { VALHALLA_CFG_DOWNLOADER_DEST = (((1 << 8) | (2 << 8)) + (2)), VALHALLA_CFG_GRABBER_PRIORITY = (((1 << 8) | (2 << 8) | (4 << 8)) + (0)), VALHALLA_CFG_GRABBER_STATE = (((1 << 8) | (2 << 8)) + (0)), VALHALLA_CFG_PARSER_KEYWORD = (((1 << 8)) + (0)), VALHALLA_CFG_SCANNER_PATH = (((1 << 8) | (2 << 8)) + (1)), VALHALLA_CFG_SCANNER_SUFFIX = (((1 << 8)) + (1)) }

List of parameters available for the configuration.
- enum valhalla_db_type_t

  Type of field.
- enum valhalla_db_operator_t

  Operator for a restriction.

Functions

- unsigned int libvalhalla_version (void)

  Return LIBVALHALLA_VERSION_INT constant.

Database selections.

- valhalla_db_stmt_t ∗ valhalla_db_metalist_get (valhalla_t ∗handle, valhalla-_db_item_t ∗search, valhalla_file_type_t filetype, valhalla_db_restrict_t ∗restriction)

  Init a statement to retrieve a list of metadata.
- const valhalla_db_metares_t ∗ valhalla_db_metalist_read (valhalla_t ∗handle, valhalla_db_stmt_t ∗vhstmt)

  Read the next row of a 'metalist' statement.
- valhalla_db_stmt_t ∗ valhalla_db_filelist_get (valhalla_t ∗handle, valhalla_file-_type_t filetype, valhalla_db_restrict_t ∗restriction)

  Init a statement to retrieve a list of files.
- const valhalla_db_fileres_t ∗ valhalla_db_filelist_read (valhalla_t ∗handle, valhalla_db_stmt_t ∗vhstmt)

  Read the next row of a 'filelist' statement.
- valhalla_db_stmt_t ∗ valhalla_db_file_get (valhalla_t ∗handle, int64_t id, const char ∗path, valhalla_db_restrict_t ∗restriction)

  Init a statement to retrieve the metadata of file.
- const valhalla_db_metares_t ∗ valhalla_db_file_read (valhalla_t ∗handle, valhalla_db_stmt_t ∗vhstmt)

  Read the next row of a 'file' statement.

Database insertions/updates/deletions.

With these functions, you can insert/update and delete metadata for a particular file (`path`). They should not be used to provide grabbing functionalities with the front-end (implement a grabber in Valhalla is the better way); but in some exceptional cases it can be necessary.

For example, you can use this functionality to write data like "playcount" or "last_-position" (to replay a file from the last position).

- int valhalla_db_metadata_insert (valhalla_t ∗handle, const char ∗path, const char ∗meta, const char ∗data, valhalla_lang_t lang, valhalla_meta_grp_-t group)

  Insert an external metadata in the database.
- int valhalla_db_metadata_update (valhalla_t ∗handle, const char ∗path, const char ∗meta, const char ∗data, const char ∗ndata, valhalla_lang_t lang)

  Update an external metadata in the database.

- int valhalla_db_metadata_delete (valhalla_t ∗handle, const char ∗path, const char ∗meta, const char ∗data)

  Delete an external metadata in the database.
- int valhalla_db_metadata_priority (valhalla_t ∗handle, const char ∗path, const char ∗meta, const char ∗data, valhalla_metadata_pl_t p)

  Change the priority for one or more metadata in the database.

Valhalla Handling.

- #define valhalla_config_set(handle, conf, arg...) valhalla_config_set_orig (handle, VALHALLA_CFG_##conf, ##arg, ∼0)

  Configure an handle.
- valhalla_t ∗ valhalla_init (const char ∗db, valhalla_init_param_t ∗param)

  Init a scanner and the database.
- void valhalla_uninit (valhalla_t ∗handle)

  Uninit an handle.
- void valhalla_verbosity (valhalla_verb_t level)

  Change verbosity level.
- const char ∗ valhalla_metadata_group_str (valhalla_meta_grp_t group)

  Retrieve an human readable string according to a group number.
- const char ∗ valhalla_grabber_next (valhalla_t ∗handle, const char ∗id)

  Retrieve the ID of all grabbers compiled in Valhalla.
- valhalla_metadata_pl_t valhalla_grabber_priority_read (valhalla_t ∗handle, const char ∗id, const char ∗∗meta)

  Retrieve the priority for a metadata according to a grabber.
- const char ∗ valhalla_stats_group_next (valhalla_t ∗handle, const char ∗id)

  Retrieve the ID of all groups in the statistics.
- uint64_t valhalla_stats_read_next (valhalla_t ∗handle, const char ∗id, valhalla_stats_type_t type, const char ∗∗item)

  Retrieve the value of a timer or a counter in the statistics.
- int valhalla_run (valhalla_t ∗handle, int loop, uint16_t timeout, uint16_t delay, int priority)

  Run the scanner, the database manager and all parsers.
- void valhalla_wait (valhalla_t ∗handle)

  Wait until the scanning is finished.
- void valhalla_scanner_wakeup (valhalla_t ∗handle)

  Force to wake up the scanner.
- void valhalla_ondemand (valhalla_t ∗handle, const char ∗file)

  Force Valhalla to retrieve metadata on-demand for a file.
- const char ∗ valhalla_ondemand_cb_meta (valhalla_t ∗handle, const char ∗meta)

  Retrieve the meta key when running in the ondemand callback.

5.2.1 Detailed Description

GeeXboX Valhalla public API header.

Definition in file valhalla.h.

5.2.2 Define Documentation

5.2.2.1 #define valhalla_config_set(*handle, conf, arg...*) valhalla_config_set_orig (handle, VALHALLA_CFG_##conf, ##arg, ~0)

Configure an handle.

The list of available parameters is defined by enum valhalla_cfg_t. VALHALLA_CFG_ is automatically prepended to `conf`.

The function must be used as follow (for example):

```
ret = valhalla_config_set (handle, GRABBER_STATE, "ffmpeg", 0);
```

Because it uses variadic arguments, there is a check on the number of arguments passed to the function and it returns a critical error if it fails. But it can't detect all bad uses. It is the job of the programmer to use correctly this function in all cases.

Warning

This function must be called before valhalla_run()!

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*conf*	Parameter to configure.
`in`	*arg*	List of arguments.

Returns

!=0 on error.

Definition at line 605 of file valhalla.h.

5.2.2.2 #define VALHALLA_DB_RESTRICT(*op, m_id, d_id, m_txt, d_txt, m_t, d_t, l, p*)

Value:

```
{                                                                      \
    /* .next = */ NULL,                                                 \
    /* .op   = */ VALHALLA_DB_OPERATOR_##op,                            \
    /* .meta = */ VALHALLA_DB_SEARCH (m_id, m_txt, NIL, m_t, l, p),     \
    /* .data = */ VALHALLA_DB_SEARCH (d_id, d_txt, NIL, d_t, l, p)      \
  }
```

Set valhalla_db_restrict_t local variable.

If possible, prefer the macros VALHALLA_DB_RESTRICT_∗() instead of this one.

Parameters

in	*op*	Operator applied on the restriction.
in	*m_id*	Meta ID.
in	*d_id*	Data ID.
in	*m_txt*	Meta text.
in	*d_txt*	Data text.
in	*m_t*	Type of field for meta.
in	*d_t*	Type of field for data.
in	*l*	Language.
in	*p*	Minimum priority.

Definition at line 911 of file valhalla.h.

5.2.2.3 #define **VALHALLA_DB_RESTRICT_INT**(*op, meta, data, l, p*) **VALHALLA_DB_RESTRICT** (op, meta, data, NULL, NULL, ID, ID, l, p)

Set valhalla_db_restrict_t local variable for meta.id, data.id.

Definition at line 930 of file valhalla.h.

5.2.2.4 #define **VALHALLA_DB_RESTRICT_INTSTR**(*op, meta, data, l, p*) **VALHALLA_DB_RESTRICT** (op, meta, 0, NULL, data, ID, TEXT, l, p)

Set valhalla_db_restrict_t local variable for meta.id, data.text.

Definition at line 936 of file valhalla.h.

5.2.2.5 #define **VALHALLA_DB_RESTRICT_LINK**(*from, to*) do {(to).next = &(from);} while (0)

Link two valhalla_db_restrict_t variables together.

Definition at line 942 of file valhalla.h.

5.2.2.6 #define **VALHALLA_DB_RESTRICT_STR**(*op, meta, data, l, p*) **VALHALLA_DB_RESTRICT** (op, 0, 0, meta, data, TEXT, TEXT, l, p)

Set valhalla_db_restrict_t local variable for meta.text, data.text.

Definition at line 933 of file valhalla.h.

5.2.2.7 #define **VALHALLA_DB_RESTRICT_STRINT**(*op, meta, data, l, p*) **VALHALLA_DB_RESTRICT** (op, 0, data, meta, NULL, TEXT, ID, l, p)

Set valhalla_db_restrict_t local variable for meta.text, data.id.

Definition at line 939 of file valhalla.h.

5.2.2.8 #define VALHALLA_DB_SEARCH(*id,* *txt,* *g,* *t,* *l,* *p*)

Value:

```
{                                                  \
    /* .type     = */ VALHALLA_DB_TYPE_##t,          \
    /* .id       = */ id,                            \
    /* .text     = */ txt,                           \
    /* .group    = */ VALHALLA_META_GRP_##g,         \
    /* .lang     = */ l,                             \
    /* .priority = */ p                              \
  }
```

Set valhalla_db_item_t local variable.

If possible, prefer the macros VALHALLA_DB_SEARCH_*() instead of this one.

Parameters

in	*id*	Meta or data ID.
in	*txt*	Meta or data text.
in	*g*	Meta group.
in	*t*	Type of field.
in	*l*	Language.
in	*p*	Minimum priority.

Definition at line 886 of file valhalla.h.

5.2.2.9 #define VALHALLA_DB_SEARCH_GRP(*group,* *l,* *p*) VALHALLA_DB_SEARCH (0, NULL, group, GROUP, l, p)

Set valhalla_db_item_t local variable for a group.

Definition at line 926 of file valhalla.h.

5.2.2.10 #define VALHALLA_DB_SEARCH_ID(*meta_id,* *group,* *l,* *p*) VALHALLA_DB_SEARCH (meta_id, NULL, group, ID, l, p)

Set valhalla_db_item_t local variable for an id.

Definition at line 920 of file valhalla.h.

5.2.2.11 #define VALHALLA_DB_SEARCH_TEXT(*meta_name,* *group,* *l,* *p*) VALHALLA_DB_SEARCH (0, meta_name, group, TEXT, l, p)

Set valhalla_db_item_t local variable for a text.

Definition at line 923 of file valhalla.h.

5.2.2.12 #define VH_CFG_INIT(*name,* *type,* *num*) VALHALLA_CFG_##name = ((type) + (num))

Macro to init items in valhalla_cfg_t.

Definition at line 361 of file valhalla.h.

5.2.2.13 #define VH_CFG_RANGE 8

256 possibilities for every combinations of type

Definition at line 353 of file valhalla.h.

5.2.2.14 #define VH_INT_T (2 << VH_CFG_RANGE)

int

Definition at line 357 of file valhalla.h.

5.2.2.15 #define VH_VOID_T (0 << VH_CFG_RANGE)

void

Definition at line 355 of file valhalla.h.

5.2.2.16 #define VH_VOIDP_2_T (4 << VH_CFG_RANGE)

void *

Definition at line 358 of file valhalla.h.

5.2.2.17 #define VH_VOIDP_T (1 << VH_CFG_RANGE)

void *

Definition at line 356 of file valhalla.h.

5.2.3 Typedef Documentation

5.2.3.1 typedef struct valhalla_db_stmt_s valhalla_db_stmt_t

Prepared statement.

Definition at line 818 of file valhalla.h.

5.2.3.2 typedef struct valhalla_s valhalla_t

Scanner handle.

Definition at line 262 of file valhalla.h.

5.2.4 Enumeration Type Documentation

5.2.4.1 enum valhalla_cfg_t

List of parameters available for the configuration.

These parameters must be used with valhalla_config_set().

When adding a new entry in the enum:

When an entry must be added in this enum, keep this one by alphabetical order. ABI is safely preserved as long as the types and the number provided with VH_CFG_INIT() are not changed.

Next `num` for the current combinations :

```
VH_VOIDP_T                              : 2
VH_VOIDP_T | VH_INT_T                   : 3
VH_VOIDP_T | VH_INT_T | VH_VOIDP_2_T : 1
```

See also

VH_CFG_INIT().

Enumerator:

VALHALLA_CFG_DOWNLOADER_DEST Set a destination for the downloader. The default destination is used when a specific destination is NULL.

`arg1` must be a null-terminated string.

Warning

There is no effect if the grabber support is not compiled.

Parameters

`in`	*arg1*	VH_VOIDP_T Path for the destination.
`in`	*arg2*	VH_INT_T Type of destination to set, valhalla_dl_t.

VALHALLA_CFG_GRABBER_PRIORITY Change the metadata priorities in the grabbers.

The argument `arg3` should be a name provided in the list of common metadata (above). If `arg1` is NULL, it affects all grabbers. If `arg3` is NULL, then it changes the default priority, but specific priorities are not modified.

The string `arg3` is not copied. The address must be valid until the call on valhalla_uninit().

`arg1` and `arg3` must be null-terminated strings.

Warning

There is no effect if the grabber support is not compiled.

Parameters

`in`	*arg1*	VH_VOIDP_T Grabber ID.
`in`	*arg2*	VH_INT_T The new priority, valhalla_metadata_pl_t.
`in`	*arg3*	VH_VOIDP_2_T Metadata.

VALHALLA_CFG_GRABBER_STATE Set the state of a grabber. By default, all grabbers are enabled.

`arg1` must be a null-terminated string.

Warning

There is no effect if the grabber support is not compiled.

Parameters

in	*arg1*	VH_VOIDP_T Grabber ID.
in	*arg2*	VH_INT_T 0 to disable, !=0 to enable.

VALHALLA_CFG_PARSER_KEYWORD This parameter is useful only if the decrapifier is enabled with valhalla_init().

The keywords are case insensitive except when a pattern (NUM, SE or EP) is used.

Available patterns (unsigned int):

- NUM to trim a number
- SE to trim and retrieve a "season" number (at least >= 1)
- EP to trim and retrieve an "episode" number (at least >= 1)

NUM can be used several time in the same keyword, like "NUMxNUM". But SE and EP must be used only one time by keyword. When a season or an episode is found, a new metadata is added for each one.

Examples:

- Blacklist : "xvid", "foobar", "fileNUM", "sSEeEP", "divx", "SExEP", "NumEP"
- Filename : "{XvID-Foobar}.file01.My_Movie.s02e10.avi"
- Result : "My Movie", season=2 and episode=10
- Filename : "My_Movie_2.s02e10_(5x3)_.mkv"
- Result : "My Movie 2", season=2, episode=10, season=5, episode=3
- Filename : "The-Episode.-.Pilot_DivX.(01x01)_FooBar.mkv"
- Result : "The Episode Pilot", season=1 and episode=1
- Filename : "_Name_of_the_episode_Num05.ogg"
- Result : "Name of the episode", episode=5

If the same keyword is added several times, only one is saved in the decrapifier.

`arg1` must be a null-terminated string.

Parameters

in	*arg1*	VH_VOIDP_T Keyword to blacklist.

VALHALLA_CFG_SCANNER_PATH Add a path to the scanner. If the same path is added several times, only one is saved in the scanner.

`arg1` must be a null-terminated string.

Parameters

in	*arg1*	VH_VOIDP_T The path to be scanned.
in	*arg2*	VH_INT_T 1 to scan all dirs recursively, 0 otherwise.

VALHALLA_CFG_SCANNER_SUFFIX If no suffix is added to the scanner, then all files will be parsed by FFmpeg without exception and it can be very slow.

It is highly recommanded to always set at least one suffix (file extension)! If the same suffix is added several times, only one is saved in the scanner. The suffixes are case insensitive.

`arg1` must be a null-terminated string.

Parameters

`in`	*arg1*	VH_VOIDP_T File suffix to add.

Definition at line 383 of file valhalla.h.

5.2.4.2 enum valhalla_db_operator_t

Operator for a restriction.

Definition at line 828 of file valhalla.h.

5.2.4.3 enum valhalla_db_type_t

Type of field.

Definition at line 821 of file valhalla.h.

5.2.4.4 enum valhalla_dl_t

Destinations for downloading.

Enumerator:

VALHALLA_DL_DEFAULT	Destination by default.
VALHALLA_DL_COVER	Destination for covers.
VALHALLA_DL_THUMBNAIL	Destination for thumbnails.
VALHALLA_DL_FAN_ART	Destination for fan-arts.

Definition at line 284 of file valhalla.h.

5.2.4.5 enum valhalla_errno

Error code returned by valhalla_run().

Enumerator:

VALHALLA_ERROR_DEAD	Valhalla is already running.
VALHALLA_ERROR_PATH	Problem with the paths for the scan.
VALHALLA_ERROR_HANDLER	Allocation memory error.
VALHALLA_ERROR_THREAD	Problem with at least one thread.
VALHALLA_SUCCESS	The Valkyries are running.

Definition at line 265 of file valhalla.h.

5.2.4.6 enum **valhalla_event_gl_t**

Events for general actions in Valhalla.

Enumerator:

VALHALLA_EVENTGL_SCANNER_BEGIN Begin the scanning of paths.
VALHALLA_EVENTGL_SCANNER_END All paths scanned.
VALHALLA_EVENTGL_SCANNER_SLEEP Scanner is sleeping.
VALHALLA_EVENTGL_SCANNER_ACKS All files fully handled.
VALHALLA_EVENTGL_SCANNER_EXIT Exit, end of all loops.

Definition at line 300 of file valhalla.h.

5.2.4.7 enum **valhalla_event_md_t**

Events for metadata callback.

Enumerator:

VALHALLA_EVENTMD_PARSER New parsed data.
VALHALLA_EVENTMD_GRABBER New grabbed data.

Definition at line 309 of file valhalla.h.

5.2.4.8 enum **valhalla_event_od_t**

Events for valhalla_ondemand() callback.

Enumerator:

VALHALLA_EVENTOD_PARSED Parsed data available in DB.
VALHALLA_EVENTOD_GRABBED Grabbed data available in DB.
VALHALLA_EVENTOD_ENDED Nothing more (downloading included).

Definition at line 293 of file valhalla.h.

5.2.4.9 enum **valhalla_lang_t**

Languages for metadata.

Enumerator:

VALHALLA_LANG_ALL All languages.
VALHALLA_LANG_UNDEF Undefined.
VALHALLA_LANG_DE German.
VALHALLA_LANG_EN English.
VALHALLA_LANG_ES Spanish.
VALHALLA_LANG_FR French.
VALHALLA_LANG_IT Italian.

Definition at line 66 of file valhalla.h.

5.2.4.10 enum valhalla_meta_grp_t

Groups for metadata.

Enumerator:

VALHALLA_META_GRP_NIL NULL value for a group attribution.

VALHALLA_META_GRP_CLASSIFICATION genre, mood, subject, synopsis, summary, description, keywords, mediatype, period, ...

VALHALLA_META_GRP_COMMERCIAL commercial, payment, purchase info, purchase price, purchase item, purchase owner, purchase currency, file owner, ...

VALHALLA_META_GRP_CONTACT url, email, address, phone, fax, ...

VALHALLA_META_GRP_ENTITIES artist, url, performer, accompaniment, band, ensemble, composer, arranger, lyricist, conductor, actor, character, author, director, producer, coproducer, executive producer, costume designer, label, choregrapher, sound engineer, production studio, publisher, ...

VALHALLA_META_GRP_IDENTIFIER isrc, mcdi, isbn, barcode, lccn, cdid, ufid, ...

VALHALLA_META_GRP_LEGAL copyright, terms of use, url, ownership, license, rights, ...

VALHALLA_META_GRP_MISCELLANEOUS user text, orig filename, picture, lyrics, ...

VALHALLA_META_GRP_MUSICAL bmp, measure, tunning, initial key, ...

VALHALLA_META_GRP_ORGANIZATIONAL track, disk, part number, track number, disc number, total tracks, total parts, ...

VALHALLA_META_GRP_PERSONAL comment, rating, play count, ...

VALHALLA_META_GRP_SPACIAL composition location, recording location, composer nationality, ...

VALHALLA_META_GRP_TECHNICAL encoder, playlist delay, buffer size, ...

VALHALLA_META_GRP_TEMPORAL date written, date recorded, date released, date digitized, date encoded, date tagged, date purchased, year, ...

VALHALLA_META_GRP_TITLES title, album, subtitle, title sort order, album sort order, part ...

Definition at line 85 of file valhalla.h.

5.2.4.11 enum valhalla_metadata_pl_t

Priorities for the metadata.

The values which are not mod 32, are only for internal use.

Enumerator:

VALHALLA_METADATA_PL_HIGHEST The highest priority.

VALHALLA_METADATA_PL_HIGHER The higher priority.

VALHALLA_METADATA_PL_HIGH High priority.

VALHALLA_METADATA_PL_ABOVE Priority above normal.

VALHALLA_METADATA_PL_NORMAL Normal (usual) priority.

VALHALLA_METADATA_PL_BELOW Priority below normal.

VALHALLA_METADATA_PL_LOW Low priority.

VALHALLA_METADATA_PL_LOWER The lower priority.

VALHALLA_METADATA_PL_LOWEST The lowest priority.

Definition at line 325 of file valhalla.h.

5.2.4.12 enum valhalla_stats_type_t

Type of statistic.

Enumerator:

VALHALLA_STATS_TIMER Read value for a timer.

VALHALLA_STATS_COUNTER Read value for a counter.

Definition at line 315 of file valhalla.h.

5.2.4.13 enum valhalla_verb_t

Verbosity level.

Enumerator:

VALHALLA_MSG_NONE No error messages.

VALHALLA_MSG_VERBOSE Super-verbose mode: mostly for debugging.

VALHALLA_MSG_INFO Working operations.

VALHALLA_MSG_WARNING Harmless failures.

VALHALLA_MSG_ERROR May result in hazardous behavior.

VALHALLA_MSG_CRITICAL Prevents lib from working.

Definition at line 274 of file valhalla.h.

5.2.5 Function Documentation

5.2.5.1 unsigned int libvalhalla_version (void)

Return LIBVALHALLA_VERSION_INT constant.

5.2.5.2 valhalla_db_stmt_t∗ valhalla_db_file_get (valhalla_t ∗ *handle,* int64_t *id,* const char ∗ *path,* valhalla_db_restrict_t ∗ *restriction*)

Init a statement to retrieve the metadata of file.

Only one parameter (`id` or `path`) must be set in order to retrieve a file. If both parameters are not null, then the `path` is ignored.

Example (to retrieve only the track and the title):

```
pmin          = VALHALLA_METADATA_PL_LOWEST;
restriction_1 = VALHALLA_DB_RESTRICT_STR (EQUAL, "track", NULL, pmin);
restriction_2 = VALHALLA_DB_RESTRICT_STR (EQUAL, "title", NULL, pmin);
VALHALLA_DB_RESTRICT_LINK (restriction_2, restriction_1);
```

If several tracks and(or) titles are returned, you must use the group id in the result, in order to know what metadata is the right.

Parameters

in	*handle*	Handle on the scanner.
in	*id*	File ID or 0.
in	*path*	Path or NULL.
in	*restriction*	Restrictions on the list.

Returns

the statement, NULL on error.

5.2.5.3 const valhalla_db_metares_t∗ valhalla_db_file_read (valhalla_t ∗ *handle,* valhalla_db_stmt_t ∗ *vhstmt*)

Read the next row of a 'file' statement.

The argument `vhstmt` must be initialized with valhalla_db_file_get(). It is freed when the returned value is NULL. The pointer returned by the function is valid as long as no new call is done for the `vhstmt`.

Parameters

in	*handle*	Handle on the scanner.
in	*vhstmt*	Statement.

Returns

the result, NULL if no more row or on error.

5.2.5.4 valhalla_db_stmt_t∗ valhalla_db_filelist_get (valhalla_t ∗ *handle,* valhalla_file_type_t *filetype,* valhalla_db_restrict_t ∗ *restriction*)

Init a statement to retrieve a list of files.

It is possible to retrieve a list of files according to restrictions on metadata and values.

Example (to list all files of an author, without album):

```
lang    = VALHALLA_LANG_ALL;
pmin    = VALHALLA_METADATA_PL_NORMAL;
restr_1 = VALHALLA_DB_RESTRICT_STR (IN, "author", "John Doe", lang, pmin);
restr_2 = VALHALLA_DB_RESTRICT_STR (NOTIN, "album", NULL, lang, pmin);
VALHALLA_DB_RESTRICT_LINK (restr_2, restr_1);
```

Parameters

in	*handle*	Handle on the scanner.
in	*filetype*	File type.
in	*restriction*	Restrictions on the list.

Returns

the statement, NULL on error.

5.2.5.5 const valhalla_db_fileres_t∗ valhalla_db_filelist_read (valhalla_t ∗ *handle*, valhalla_db_stmt_t ∗ *vhstmt*)

Read the next row of a 'filelist' statement.

The argument `vhstmt` must be initialized with valhalla_db_filelist_get(). It is freed when the returned value is NULL. The pointer returned by the function is valid as long as no new call is done for the `vhstmt`.

Parameters

in	*handle*	Handle on the scanner.
in	*vhstmt*	Statement.

Returns

the result, NULL if no more row or on error.

5.2.5.6 int valhalla_db_metadata_delete (valhalla_t ∗ *handle*, const char ∗ *path*, const char ∗ *meta*, const char ∗ *data*)

Delete an external metadata in the database.

Only a metadata inserted or updated with valhalla_db_metadata_insert(), and valhalla_db_metadata_update() can be deleted with this function.

Please, refer to External Metadata.

Parameters

in	*handle*	Handle on the scanner.
in	*path*	Path on the file.
in	*meta*	Meta name.
in	*data*	Data value.

Returns

!=0 on error.

5.2.5.7 int valhalla_db_metadata_insert (valhalla_t * *handle*, const char * *path*, const char * *meta*, const char * *data*, valhalla_lang_t *lang*, valhalla_meta_grp_t *group*)

Insert an external metadata in the database.

When a metadata is inserted with this function, you must use valhalla_db_metadata_update() to change the value, else two metadata will be available (for both values).

If the metadata is already available in the database and the `group` (or the `lang`) passed with this function is not the same, then the insertion is canceled and no error is returned, else the 'external' flag is set to 1.

See also

valhalla_db_metares_t

Please, refer to External Metadata.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*path*	Path on the file.
`in`	*meta*	Meta name.
`in`	*data*	Data value.
`in`	*lang*	Language.
`in`	*group*	Group.

Returns

!=0 on error.

5.2.5.8 int valhalla_db_metadata_priority (valhalla_t * *handle*, const char * *path*, const char * *meta*, const char * *data*, valhalla_metadata_pl_t *p*)

Change the priority for one or more metadata in the database.

If `meta` is NULL, all metadata are changed. If `data` is NULL, all metadata for a specific `meta` are changed. If `meta` is NULL, but `data` is set, then the function returns an error.

The 'external' flag is not altered by this function.

Please, refer to External Metadata.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*path*	Path on the file.
`in`	*meta*	Meta name.

in	*data*	Data value.
in	*p*	New priority.

Returns

!=0 on error.

5.2.5.9 int **valhalla_db_metadata_update** (**valhalla_t** * *handle*, const char * *path*, const char * *meta*, const char * *data*, const char * *ndata*, **valhalla_lang_t** *lang*)

Update an external metadata in the database.

The previous `data` is necessary for Valhalla to identify the association for the update.

If `ndata` already exists in the database, the language is not updated with the value passed by this function.

Please, refer to External Metadata.

Parameters

in	*handle*	Handle on the scanner.
in	*path*	Path on the file.
in	*meta*	Meta name.
in	*data*	Current data value.
in	*ndata*	New data value.
in	*lang*	Language.

Returns

!=0 on error.

5.2.5.10 **valhalla_db_stmt_t*** **valhalla_db_metalist_get** (**valhalla_t** * *handle*, **valhalla_db_item_t** * *search*, valhalla_file_type_t *filetype*, **valhalla_db_restrict_t** * *restriction*)

Init a statement to retrieve a list of metadata.

It is possible to retrieve a list of metadata according to restrictions on metadata and values.

Example (to list all albums of an author):

```
lang   = VALHALLA_LANG_ALL;
pmin   = VALHALLA_METADATA_PL_LOWEST;
search = VALHALLA_DB_SEARCH_TEXT ("album", TITLES, lang, pmin);
restr  = VALHALLA_DB_RESTRICT_STR (IN, "author", "John Doe", lang, pmin);
```

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*search*	Condition for the search.
`in`	*filetype*	File type.
`in`	*restriction*	Restrictions on the list.

Returns

the statement, NULL on error.

5.2.5.11 const **valhalla_db_metares_t**∗ **valhalla_db_metalist_read** (**valhalla_t** ∗ *handle,* **valhalla_db_stmt_t** ∗ *vhstmt*)

Read the next row of a 'metalist' statement.

The argument `vhstmt` must be initialized with valhalla_db_metalist_get(). It is freed when the returned value is NULL. The pointer returned by the function is valid as long as no new call is done for the `vhstmt`.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*vhstmt*	Statement.

Returns

the result, NULL if no more row or on error.

5.2.5.12 const char∗ **valhalla_grabber_next** (**valhalla_t** ∗ *handle,* const char ∗ *id*)

Retrieve the ID of all grabbers compiled in Valhalla.

The function returns the ID after `id`, or the first grabber ID if `id` is NULL.

Warning

This function must be called before valhalla_run()! There is no effect if the grabber support is not compiled.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*id*	Grabber ID or NULL to retrieve the first.

Returns

the next ID or NULL if `id` is the last (or on error).

5.2.5.13 valhalla_metadata_pl_t valhalla_grabber_priority_read (valhalla_t * *handle,* const char * *id,* const char ** *meta*)

Retrieve the priority for a metadata according to a grabber.

If `id` is NULL, the result is 0. To retrieve the default priority, the argument `*meta` must be set to NULL. On the return, `*meta` is the next metadata in the list, or NULL if there is nothing more. If on call, `*meta` is not found, then the result is 0 and `*meta` is not changed. If `meta` is NULL, the result is 0.

Please, note that 0 is a valid value for a priority and must not be used to detect errors. If this function is used correctly, no error is possible.

Use valhalla_grabber_next() in order to retrieve the IDs.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*id*	A valid grabber ID.
`in,out`	*meta*	A valid address; the next meta is returned.

Returns

the priority.

5.2.5.14 valhalla_t* valhalla_init (const char * *db,* valhalla_init_param_t * *param*)

Init a scanner and the database.

If a database already exists, then it is used. Otherwise, a new database is created to `db`. If more than one handles are created, you can't use the same database. You must specify a different `db` for each handle.

For a description of each parameters supported by this function:

See also

valhalla_init_param_t

When a parameter in `param` is 0 (or NULL), its default value is used. If `param` is NULL, then all default values are forced for all parameters.

Parameters

`in`	*db*	Path on the database.
`in`	*param*	Parameters, NULL for default values.

Returns

The handle.

5.2.5.15 const char∗ valhalla_metadata_group_str (valhalla_meta_grp_t *group*)

Retrieve an human readable string according to a group number.

The strings returned are the same that the strings saved in the database.

Warning

This function can be called in anytime.

Parameters

`in`	*group*	Group number.

Returns

the string.

5.2.5.16 void valhalla_ondemand (valhalla_t ∗ *handle,* const char ∗ *file*)

Force Valhalla to retrieve metadata on-demand for a file.

This functionality can be used on files in/out of paths defined for the scanner. This function is non-blocked and it has the top priority over the files retrieved by the scanner.

Warning

This function can be used only after valhalla_run()!

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*file*	Target.

5.2.5.17 const char∗ valhalla_ondemand_cb_meta (valhalla_t ∗ *handle,* const char ∗ *meta*)

Retrieve the meta key when running in the ondemand callback.

This function is a no-op when it is used elsewhere that an ondemand callback or if the od_meta attribute of valhalla_init_param_t is 0.

The function returns the key after `meta`, or the first key if `meta` is NULL. The returned pointer is valid as long as your are in the callback.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*meta*	Meta or NULL to retrieve the first.

Returns

the meta or NULL if `meta` is the last (or on error).

5.2.5.18 int **valhalla_run** (**valhalla_t** ∗ *handle,* int *loop,* uint16_t *timeout,* uint16_t *delay,* int *priority*)

Run the scanner, the database manager and all parsers.

The `priority` can be set to all thread especially to run the system in background with less priority. In the case of a user, you can change only for a lower priority.

0 (normal priority used by default) Linux : -20 (highest) to 19 (lowest) FreeBSD : -20 (highest) to 20 (lowest) Windows : -3 (highest) to 3 (lowest)

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*loop*	Number of loops (<=0 for infinite).
`in`	*timeout*	Timeout between loops, 0 to disable [seconds].
`in`	*delay*	Delay before the scanning begins [seconds].
`in`	*priority*	Priority set to all threads.

Returns

0 for success and <0 on error (see enum valhalla_errno).

5.2.5.19 void **valhalla_scanner_wakeup** (**valhalla_t** ∗ *handle*)

Force to wake up the scanner.

If the scanner is sleeping, this function will wake up this one independently of the time (`timeout`) set with valhalla_run(). If the number of loops is already reached or if the scanner is already working, this function has no effect.

Warning

This function can be used only after valhalla_run()!

Parameters

`in`	*handle*	Handle on the scanner.

5.2.5.20 const char∗ valhalla_stats_group_next (valhalla_t ∗ *handle,* const char ∗ *id*)

Retrieve the ID of all groups in the statistics.

The function returns the ID after `id`, or the first group ID if `id` is NULL.

Warning

This function can be called in anytime.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*id*	Group ID or NULL to retrieve the first.

Returns

the next ID or NULL if `id` is the last (or on error).

5.2.5.21 uint64_t valhalla_stats_read_next (valhalla_t ∗ *handle,* const char ∗ *id,* valhalla_stats_type_t *type,* const char ∗∗ *item*)

Retrieve the value of a timer or a counter in the statistics.

`item` ID is set according to the next timer or the next counter. If the `item` ID is not changed on the return, then an error was encountered.

Warning

This function can be called in anytime.

Parameters

`in`	*handle*	Handle on the scanner.
`in`	*id*	Group ID.
`in`	*type*	Timer or counter.
`in,out`	*item*	Item ID or NULL for the first.

Returns

the value (nanoseconds for the timers).

5.2.5.22 void valhalla_uninit (valhalla_t ∗ *handle*)

Uninit an handle.

If a scanner is running, this function stops immediatly all tasks before releasing all elements.

Parameters

`in`	*handle*	Handle on the scanner.

5.2.5.23 void valhalla_verbosity (valhalla_verb_t *level*)

Change verbosity level.

Default value is VALHALLA_MSG_INFO.

Warning

This function can be called in anytime.

Parameters

`in`	*level*	Level provided by valhalla_verb_t.

5.2.5.24 void valhalla_wait (valhalla_t * *handle*)

Wait until the scanning is finished.

This function wait until the scanning is finished for all loops. If the number of loops is infinite, then this function will wait forever. You must not break this function with valhalla-_uninit(), that is not safe! If you prefer stop the scanner even if it is not finished. In this case you must use _only_ valhalla_uninit().

If no path is defined (then the scanner is not running), this function returns immediately.

Warning

This function can be used only after valhalla_run()!

Parameters

`in`	*handle*	Handle on the scanner.