libvalhalla

1.0.1

Generated by Doxygen 1.6.1

Mon Jan 25 13:04:53 2010

Contents

1 External Metadata

1.1 External Metadata

See also:

valhalla_db_metadata_insert().
valhalla_db_metadata_update().
valhalla_db_metadata_delete().

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)

  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.
- int enable

  Different of 0 if the grabber is enabled.
- unsigned int stat_success

  Total of successes for the session.
- unsigned int stat_failure

  Total of failures for the session.
- struct timespec stat_difftime

  Total time used by the grabber.

4.1.1 Detailed Description

Structure for a grabber.

Definition at line 79 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 85 of file grabber_common.h.

4.1.2.2 int grabber_list_t::enable

Different of 0 if the grabber is enabled.

Definition at line 155 of file grabber_common.h.

4.1.2.3 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:

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

Returns:

0 for success, != 0 on error.

4.1.2.4 int(∗ grabber_list_t::init)(void ∗priv)

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:

← ***priv*** Private structure registered with the grabber.

Returns:

0 for success, != 0 on error.

4.1.2.5 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:

← ***priv*** Private structure registered with the grabber.

4.1.2.6 const char∗ grabber_list_t::name

Textual identification of the grabber.

Definition at line 83 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 152 of file grabber_common.h.

4.1.2.8 struct timespec grabber_list_t::stat_difftime `[read]`

Total time used by the grabber.

Definition at line 161 of file grabber_common.h.

4.1.2.9 unsigned int grabber_list_t::stat_failure

Total of failures for the session.

Definition at line 159 of file grabber_common.h.

4.1.2.10 unsigned int grabber_list_t::stat_success

Total of successes for the session.

Definition at line 157 of file grabber_common.h.

4.1.2.11 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:

← ***priv*** Private structure registered with the grabber.

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

- grabber_common.h

4.2 valhalla_db_filemeta_t Struct Reference

Results for valhalla_db_file_get().

`#include <valhalla.h>`

4.2.1 Detailed Description

Results for valhalla_db_file_get().

Definition at line 576 of file valhalla.h.

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

- valhalla.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 561 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 545 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 553 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 568 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 <time.h>
#include "utils.h"
```

Data Structures

- struct grabber_list_t

  Structure for a grabber.

Defines

- #define GRABBER_REGISTER(p_name, p_caps,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 69 of file grabber_common.h.

5.1.2.2 #define GRABBER_CAP_IMAGE (1 << 2)

grab for image files

Definition at line 71 of file grabber_common.h.

5.1.2.3 #define GRABBER_CAP_VIDEO (1 << 1)

grab for video files

Definition at line 70 of file grabber_common.h.

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

Value:

```
grabber_list_t *                                                        \
  vh_grabber_##p_name##_register (void)                                  \
  {                                                                      \
    grabber_list_t *grabber;                                             \
                                                                         \
    valhalla_log (VALHALLA_MSG_VERBOSE, __FUNCTION__);                   \
                                                                         \
    grabber = calloc (1, sizeof (grabber_list_t));                       \
    if (!grabber)                                                        \
      return NULL;                                                       \
                                                                         \
    grabber->name       = #p_name;                                       \
    grabber->caps_flag = p_caps;                                         \
    grabber->enable     = 1;                                             \
    grabber->priv       = fct_priv ();                                   \
                                                                         \
    grabber->init       = fct_init;                                      \
    grabber->uninit     = fct_uninit;                                    \
    grabber->grab       = fct_grab;                                      \
    grabber->loop       = fct_loop;                                           \
                                                                              \
    return grabber;                                                           \
  }
```

Macro to register and populate a grabber structure. See struct grabber_list_s for more informations on the structure and the functions.

Parameters:

← ***p_name*** Grabber's name.

← ***p_caps*** Capabilities flags.

← ***fct_priv*** Function to retrieve the private data pointer.

← ***fct_init*** grabber_list_t::init().

← ***fct_uninit*** grabber_list_t::uninit().

← ***fct_grab*** grabber_list_t::grab().

← ***fct_loop*** grabber_list_t::loop().

Definition at line 180 of file grabber_common.h.

5.2 valhalla.h File Reference

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

Data Structures

- 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.

- struct valhalla_db_filemeta_t

  Results for valhalla_db_file_get().

Defines

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_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_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"

Macros for selection functions handling.

- #define VALHALLA_DB_SEARCH_ID(meta_id, group) VALHALLA_DB_SEARCH (meta_-id, NULL, group, ID)

  Set valhalla_db_item_t local variable for an id.

- #define VALHALLA_DB_SEARCH_TEXT(meta_name, group) VALHALLA_DB_SEARCH (0, meta_name, group, TEXT)

  Set valhalla_db_item_t local variable for a text.

- #define VALHALLA_DB_SEARCH_GRP(group) VALHALLA_DB_SEARCH (0, NULL, group, GROUP)

  Set valhalla_db_item_t local variable for a group.

- #define VALHALLA_DB_RESTRICT_INT(op, meta, data) VALHALLA_DB_RESTRICT (op, meta, data, NULL, NULL, ID, ID)

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

- #define VALHALLA_DB_RESTRICT_STR(op, meta, data) VALHALLA_DB_RESTRICT (op, 0, 0, meta, data, TEXT, TEXT)

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

- #define VALHALLA_DB_RESTRICT_INTSTR(op, meta, data) VALHALLA_DB_RESTRICT (op, meta, 0, NULL, data, ID, TEXT)

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

- #define VALHALLA_DB_RESTRICT_STRINT(op, meta, data) VALHALLA_DB_RESTRICT (op, 0, data, meta, NULL, TEXT, ID)

  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.

- #define VALHALLA_DB_FILEMETA_FREE(meta)

  Free a valhalla_db_filemeta_t pointer.

Typedefs

- typedef struct valhalla_s valhalla_t

  Scanner handle.

Enumerations

- enum valhalla_errno

  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

  Destinations for downloading.

- enum valhalla_event_t { VALHALLA_EVENT_PARSED = 0, VALHALLA_EVENT_GRABBED, VALHALLA_EVENT_ENDED }

  Events for valhalla_ondemand() callback.

- 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.

Functions

Valhalla Handling.

- valhalla_t ∗ valhalla_init (const char ∗db, unsigned int parser_nb, int decrapifier, unsigned int commit_int, void(∗od_cb)(const char ∗file, valhalla_event_t e, const char ∗id, void ∗data), void ∗od_data)

  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.

- void valhalla_path_add (valhalla_t ∗handle, const char ∗location, int recursive)

  Add a path to the scanner.

- void valhalla_suffix_add (valhalla_t ∗handle, const char ∗suffix)

  Add a file suffix for the scanner.

- void valhalla_bl_keyword_add (valhalla_t ∗handle, const char ∗keyword)

  Add a keyword in the blacklist for the decrapifier.

- void valhalla_downloader_dest_set (valhalla_t ∗handle, valhalla_dl_t dl, const char ∗dst)

  Set a destination for the downloader.

- void valhalla_grabber_state_set (valhalla_t ∗handle, const char ∗id, int enable)

  Set the state of a grabber.

- const char ∗ valhalla_grabber_list_get (valhalla_t ∗handle, const char ∗id)

  Retrieve the ID of all grabbers compiled in Valhalla.

- int valhalla_run (valhalla_t ∗handle, int loop, uint16_t timeout, 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.

Database selections.

- int valhalla_db_metalist_get (valhalla_t ∗handle, valhalla_db_item_t ∗search, valhalla_db_-restrict_t ∗restriction, int(∗result_cb)(void ∗data, valhalla_db_metares_t ∗res), void ∗data)
  Retrieve a list of metadata according to a condition.
- int valhalla_db_filelist_get (valhalla_t ∗handle, valhalla_file_type_t filetype, valhalla_db_-restrict_t ∗restriction, int(∗result_cb)(void ∗data, valhalla_db_fileres_t ∗res), void ∗data)
  Retrieve a list of files.
- int valhalla_db_file_get (valhalla_t ∗handle, int64_t id, const char ∗path, valhalla_db_restrict_t ∗restriction, valhalla_db_filemeta_t ∗∗res)
  Retrieve all metadata of a file.

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_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)
  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.

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_DB_FILEMETA_FREE(meta)

Value:

```
do {                                                    \
    typeof (meta) tmp;                                    \
    while (meta) {                                        \
      if ((meta)->meta_name)  free ((meta)->meta_name);   \
      if ((meta)->data_value) free ((meta)->data_value);  \
      tmp = (meta)->next; free (meta); meta = tmp;}       \
  } while (0)
```

Free a valhalla_db_filemeta_t pointer.

Definition at line 629 of file valhalla.h.

5.2.2.2 #define VALHALLA_DB_RESTRICT_INT(op, meta, data) VALHALLA_DB_-RESTRICT (op, meta, data, NULL, NULL, ID, ID)

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

Definition at line 613 of file valhalla.h.

5.2.2.3 #define VALHALLA_DB_RESTRICT_INTSTR(op, meta, data) VALHALLA_DB_-RESTRICT (op, meta, 0, NULL, data, ID, TEXT)

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

Definition at line 619 of file valhalla.h.

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

Link two valhalla_db_restrict_t variables together.

Definition at line 625 of file valhalla.h.

5.2.2.5 #define VALHALLA_DB_RESTRICT_STR(op, meta, data) VALHALLA_DB_-RESTRICT (op, 0, 0, meta, data, TEXT, TEXT)

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

Definition at line 616 of file valhalla.h.

5.2.2.6 #define VALHALLA_DB_RESTRICT_STRINT(op, meta, data) VALHALLA_DB_-RESTRICT (op, 0, data, meta, NULL, TEXT, ID)

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

Definition at line 622 of file valhalla.h.

5.2.2.7 #define VALHALLA_DB_SEARCH_GRP(group) VALHALLA_DB_SEARCH (0, NULL, group, GROUP)

Set valhalla_db_item_t local variable for a group.

Definition at line 609 of file valhalla.h.

5.2.2.8 #define VALHALLA_DB_SEARCH_ID(meta_id, group) VALHALLA_DB_SEARCH (meta_id, NULL, group, ID)

Set valhalla_db_item_t local variable for an id.

Definition at line 603 of file valhalla.h.

5.2.2.9 #define VALHALLA_DB_SEARCH_TEXT(meta_name, group) VALHALLA_DB_-SEARCH (0, meta_name, group, TEXT)

Set valhalla_db_item_t local variable for a text.

Definition at line 606 of file valhalla.h.

5.2.3 Typedef Documentation

5.2.3.1 typedef struct valhalla_s valhalla_t

Scanner handle.

Definition at line 67 of file valhalla.h.

5.2.4 Enumeration Type Documentation

5.2.4.1 enum valhalla_dl_t

Destinations for downloading.

Definition at line 89 of file valhalla.h.

5.2.4.2 enum valhalla_errno

Error code returned by valhalla_run().

Definition at line 70 of file valhalla.h.

5.2.4.3 enum valhalla_event_t

Events for valhalla_ondemand() callback.

Enumerator:

VALHALLA_EVENT_PARSED Parsed data available in DB.

VALHALLA_EVENT_GRABBED Grabbed data available in DB.

VALHALLA_EVENT_ENDED Nothing more (downloading included).

Definition at line 98 of file valhalla.h.

5.2.4.4 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 365 of file valhalla.h.

5.2.4.5 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 79 of file valhalla.h.

5.2.5 Function Documentation

5.2.5.1 void valhalla_bl_keyword_add (valhalla_t ∗ *handle*, const char ∗ *keyword*)

Add a keyword in the blacklist for the decrapifier. This function 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.

Warning:

This function must be called before valhalla_run()!

Parameters:

← ***handle*** Handle on the scanner.

← ***keyword*** Keyword to blacklist.

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

Retrieve all metadata of a 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.

`*res` must be freed by VALHALLA_DB_FILEMETA_FREE().

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

```
restriction_1 = VALHALLA_DB_RESTRICT_STR (EQUAL, "track", NULL);
restriction_2 = VALHALLA_DB_RESTRICT_STR (EQUAL, "title", NULL);
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:

← ***handle*** Handle on the scanner.

← ***id*** File ID or 0.

← ***path*** Path or NULL.

← ***restriction*** Restrictions on the list of meta.

→ ***res*** Pointer on the linked list to populate.

Returns:

!=0 on error.

5.2.5.3 int valhalla_db_filelist_get (valhalla_t ∗ *handle*, valhalla_file_type_t *filetype*, valhalla_db_restrict_t ∗ *restriction*, int(∗)(void ∗data, valhalla_db_fileres_t ∗res) *result_cb*, void ∗ *data*)

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):

```
restriction_1 = VALHALLA_DB_RESTRICT_STR (IN, "author", "John Doe");
restriction_2 = VALHALLA_DB_RESTRICT_STR (NOTIN, "album", NULL);
VALHALLA_DB_RESTRICT_LINK (restriction_2, restriction_1);
```

Parameters:

← ***handle*** Handle on the scanner.

← ***filetype*** File type.

← ***restriction*** Restrictions on the list.

→ ***result_cb*** Result callback.

↔ ***data*** Data for the first callback argument.

Returns:

!=0 on error.

5.2.5.4 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:

← ***handle*** Handle on the scanner.

← ***path*** Path on the file.

← ***meta*** Meta name.

← ***data*** Data value.

Returns:

!=0 on error.

5.2.5.5 int valhalla_db_metadata_insert (valhalla_t ∗ *handle*, const char ∗ *path*, const char ∗ *meta*, const char ∗ *data*, 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` 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
valhalla_db_filemeta_t

Please, refer to External Metadata.

Parameters:

← ***handle*** Handle on the scanner.

← ***path*** Path on the file.

← ***meta*** Meta name.

← ***data*** Data value.

← ***group*** Group.

Returns:

!=0 on error.

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

Update an external metadata in the database. The previous `data` is necessary for Valhalla to identify the association for the update.

Please, refer to External Metadata.

Parameters:

← ***handle*** Handle on the scanner.

← ***path*** Path on the file.

← ***meta*** Meta name.

← ***data*** Current data value.

← ***ndata*** New data value.

Returns:

!=0 on error.

5.2.5.7 int valhalla_db_metalist_get (valhalla_t ∗ *handle*, valhalla_db_item_t ∗ *search*, valhalla_db_restrict_t ∗ *restriction*, int(∗)(void ∗data, valhalla_db_metares_t ∗res) *result_cb*, void ∗ *data*)

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

Example (to list all albums of an author):

```
search      = VALHALLA_DB_SEARCH_TEXT ("album", TITLES);
restriction = VALHALLA_DB_RESTRICT_STR (IN, "author", "John Doe");
```

Parameters:

← ***handle*** Handle on the scanner.

← ***search*** Condition for the search.

← ***restriction*** Restrictions on the list.

→ ***result_cb*** Result callback.

↔ ***data*** Data for the first callback argument.

Returns:

!=0 on error.

5.2.5.8 void valhalla_downloader_dest_set (valhalla_t ∗ *handle*, valhalla_dl_t *dl*, const char ∗ *dst*)

Set a destination for the downloader. The default destination is used when a specific destination is NULL. VALHALLA_DL_LAST is only used for internal purposes.

Warning:

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

Parameters:

← ***handle*** Handle on the scanner.

← ***dl*** Type of destination to set.

← ***dst*** Path for the destination.

5.2.5.9 const char∗ valhalla_grabber_list_get (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:

← ***handle*** Handle on the scanner.

← ***id*** Grabber ID or NULL to retrieve the first.

Returns:

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

5.2.5.10 void valhalla_grabber_state_set (valhalla_t ∗ *handle*, const char ∗ *id*, int *enable*)

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

Warning:

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

Parameters:

← ***handle*** Handle on the scanner.

← ***id*** Grabber ID.

← ***enable*** 0 to disable, !=0 to enable.

5.2.5.11 valhalla_t∗ valhalla_init (const char ∗ *db*, unsigned int *parser_nb*, int *decrapifier*, unsigned int *commit_int*, void(∗)(const char ∗file, valhalla_event_t e, const char ∗id, void ∗data) *od_cb*, void ∗ *od_data*)

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.

Several parsers (`parser_nb`) for metadata can be created in parallel.

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 grabber support is compiled because the title is used as keywords in a lot of grabbers.

The interval for `commit_int` is the number of data to be inserted or updated in one pass. A value between 100 and 200 is a good choice. If the value is <=0, then the default interval is used (128).

Events

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_EVENT_-GRABBED can be sent before VALHALLA_EVENT_PARSED. VALHALLA_EVENT_GRABBED is sent for each grabber and `id` is its textual identifier (for example: "amazon", "exif", etc, ...). Only VALHALLA_EVENT_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_EVENT_ENDED is sent to the callback.

Parameters:

← ***db*** Path on the database.

← ***parser_nb*** Number of parsers to create.

← ***decrapifier*** Use the decrapifier, !=0 to enable.

← ***commit_int*** File Interval between database commits.

← ***od_cb*** Callback for ondemand, NULL to ignore.

← ***od_data*** User data for ondemand callback.

Returns:

The handle.

5.2.5.12 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:

← ***handle*** Handle on the scanner.

← ***file*** Target.

5.2.5.13 void valhalla_path_add (valhalla_t ∗ *handle*, const char ∗ *location*, int *recursive*)

Add a path to the scanner. At least one path must be added to the scanner, otherwise an error is returned by valhalla_run(). If the same path is added several times, only one is saved in the scanner.

Warning:

This function must be called before valhalla_run()!

Parameters:

← ***handle*** Handle on the scanner.

← ***location*** The path to be scanned.

← ***recursive*** 1 to scan all folders recursively, 0 otherwise.

5.2.5.14 int valhalla_run (valhalla_t ∗ *handle*, int *loop*, uint16_t *timeout*, 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 (default)
- 1 to 19 : lower priorities
- -1 to -20 : higher priorities

Parameters:

← ***handle*** Handle on the scanner.

← ***loop*** Number of loops (<=0 for infinite).

← ***timeout*** Timeout between loops, 0 to disable [seconds].

← ***priority*** Priority set to all threads.

Returns:

0 for success and <0 on error (

See also:

enum valhalla_errno).

5.2.5.15 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:

← ***handle*** Handle on the scanner.

5.2.5.16 void valhalla_suffix_add (valhalla_t ∗ *handle*, const char ∗ *suffix*)

Add a file suffix for the scanner. 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.

Warning:

This function must be called before valhalla_run()!

Parameters:

← ***handle*** Handle on the scanner.

← ***suffix*** File suffix to add.

5.2.5.17 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:

← ***handle*** Handle on the scanner.

5.2.5.18 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:

← ***level*** Level provided by valhalla_verb_t.

5.2.5.19 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().

Warning:

This function can be used only after valhalla_run()!

Parameters:

← ***handle*** Handle on the scanner.

Index