NAME

libmtp −

SYNOPSIS

Macros

#define **LIBMTP_FILES_AND_FOLDERS_ROOT** 0xffffffff

Typedefs

typedef void(* **LIBMTP_event_cb_fn**) (int, LIBMTP_event_t, uint32_t, void *)

Functions

LIBMTP_file_t * **LIBMTP_new_file_t** (void)
void **LIBMTP_destroy_file_t** (**LIBMTP_file_t** *)
char const * **LIBMTP_Get_Filetype_Description** (**LIBMTP_filetype_t**)
LIBMTP_file_t * **LIBMTP_Get_Filelisting** (**LIBMTP_mtpdevice_t** *)
LIBMTP_file_t * **LIBMTP_Get_Filelisting_With_Callback** (**LIBMTP_mtpdevice_t** *, **LIBMTP_progressfunc_t** const, void const *const)
LIBMTP_file_t * **LIBMTP_Get_Files_And_Folders** (**LIBMTP_mtpdevice_t** *, uint32_t const, uint32_t const)
LIBMTP_file_t * **LIBMTP_Get_Filemetadata** (**LIBMTP_mtpdevice_t** *, uint32_t const)
int **LIBMTP_Get_File_To_File** (**LIBMTP_mtpdevice_t** *, uint32_t, char const *const, **LIBMTP_progressfunc_t** const, void const *const)
int **LIBMTP_Get_File_To_File_Descriptor** (**LIBMTP_mtpdevice_t** *, uint32_t const, int const, **LIBMTP_progressfunc_t** const, void const *const)
int **LIBMTP_Get_File_To_Handler** (**LIBMTP_mtpdevice_t** *, uint32_t const, **MTPDataPutFunc**, void *, **LIBMTP_progressfunc_t** const, void const *const)
int **LIBMTP_Send_File_From_File** (**LIBMTP_mtpdevice_t** *, char const *const, **LIBMTP_file_t** *const, **LIBMTP_progressfunc_t** const, void const *const)
int **LIBMTP_Send_File_From_File_Descriptor** (**LIBMTP_mtpdevice_t** *, int const, **LIBMTP_file_t** *const, **LIBMTP_progressfunc_t** const, void const *const)
int **LIBMTP_Send_File_From_Handler** (**LIBMTP_mtpdevice_t** *, **MTPDataGetFunc**, void *, **LIBMTP_file_t** *const, **LIBMTP_progressfunc_t** const, void const *const)
int **LIBMTP_Set_File_Name** (**LIBMTP_mtpdevice_t** *, **LIBMTP_file_t** *, const char *)
LIBMTP_filesampledata_t * **LIBMTP_new_filesampledata_t** (void)
void **LIBMTP_destroy_filesampledata_t** (**LIBMTP_filesampledata_t** *)
int **LIBMTP_Get_Representative_Sample_Format** (**LIBMTP_mtpdevice_t** *, **LIBMTP_filetype_t** const, **LIBMTP_filesampledata_t** **)
int **LIBMTP_Send_Representative_Sample** (**LIBMTP_mtpdevice_t** *, uint32_t const, **LIBMTP_filesampledata_t** *)
int **LIBMTP_Get_Representative_Sample** (**LIBMTP_mtpdevice_t** *, uint32_t const, **LIBMTP_filesampledata_t** *)
int **LIBMTP_Get_Thumbnail** (**LIBMTP_mtpdevice_t** *, uint32_t const, unsigned char **data, unsigned int *size)
int **LIBMTP_Read_Event** (**LIBMTP_mtpdevice_t** *, LIBMTP_event_t *, uint32_t *)
int **LIBMTP_Read_Event_Async** (**LIBMTP_mtpdevice_t** *, LIBMTP_event_cb_fn, void *)
int **LIBMTP_Handle_Events_Timeout_Completed** (struct timeval *, int *)

Detailed Description

Function Documentation

void LIBMTP_destroy_file_t (LIBMTP_file_t * file)

This destroys a file metadata structure and deallocates the memory used by it, including any strings. Never use a file metadata structure again after calling this function on it.

Parameters:

file the file metadata to destroy.

See also:

LIBMTP_new_file_t()

References LIBMTP_file_struct::filename.

void LIBMTP_destroy_filesampledata_t (LIBMTP_filesampledata_t * sample)
This destroys a file sample metadata type.

Parameters:
sample the file sample metadata to be destroyed.

References LIBMTP_filesampledata_struct::data.

int LIBMTP_Get_File_To_File (LIBMTP_mtpdevice_t * device, uint32_t const id, char const *const path, LIBMTP_progressfunc_t const callback, void const *const data)
This gets a file off the device to a local file identified by a filename.

Parameters:
device a pointer to the device to get the track from.
id the file ID of the file to retrieve.
path a filename to use for the retrieved file.
callback a progress indicator function or NULL to ignore.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:
0 if the transfer was successful, any other value means failure.

See also:
LIBMTP_Get_File_To_File_Descriptor()

Referenced by LIBMTP_Get_Track_To_File().

int LIBMTP_Get_File_To_File_Descriptor (LIBMTP_mtpdevice_t * device, uint32_t const id, int const fd, LIBMTP_progressfunc_t const callback, void const *const data)
This gets a file off the device to a file identified by a file descriptor.

This function can potentially be used for streaming files off the device for playback or broadcast for example, by downloading the file into a stream sink e.g. a socket.

Parameters:
device a pointer to the device to get the file from.
id the file ID of the file to retrieve.
fd a local file descriptor to write the file to.
callback a progress indicator function or NULL to ignore.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:
0 if the transfer was successful, any other value means failure.

See also:
LIBMTP_Get_File_To_File()

References LIBMTP_mtpdevice_struct::params, and LIBMTP_mtpdevice_struct::usbinfo.

Referenced by LIBMTP_Get_Track_To_File_Descriptor().

int LIBMTP_Get_File_To_Handler (LIBMTP_mtpdevice_t * device, uint32_t const id, MTPDataPutFunc put_func, void * priv, LIBMTP_progressfunc_t const callback, void const *const data)
This gets a file off the device and calls put_func with chunks of data

Parameters:
device a pointer to the device to get the file from.
id the file ID of the file to retrieve.
put_func the function to call when we have data.
priv the user-defined pointer that is passed to `put_func`.
callback a progress indicator function or NULL to ignore.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:
0 if the transfer was successful, any other value means failure.

References LIBMTP_mtpdevice_struct::params, and LIBMTP_mtpdevice_struct::usbinfo.

Referenced by LIBMTP_Get_Track_To_Handler().

LIBMTP_file_t* LIBMTP_Get_Filelisting (LIBMTP_mtpdevice_t * device)

THIS FUNCTION IS DEPRECATED. PLEASE UPDATE YOUR CODE IN ORDER NOT TO USE IT.

See also:

LIBMTP_Get_Filelisting_With_Callback()

References LIBMTP_Get_Filelisting_With_Callback().

LIBMTP_file_t* LIBMTP_Get_Filelisting_With_Callback (LIBMTP_mtpdevice_t * device, LIBMTP_progressfunc_t const callback, void const *const data)

This returns a long list of all files available on the current MTP device. Folders will not be returned, but abstract entities like playlists and albums will show up as 'files'. Typical usage:

```
LIBMTP_file_t *filelist;

filelist = LIBMTP_Get_Filelisting_With_Callback(device, callback, data);
while (filelist != NULL) {
  LIBMTP_file_t *tmp;

  // Do something on each element in the list here...
  tmp = filelist;
  filelist = filelist->next;
  LIBMTP_destroy_file_t(tmp);
}
```

If you want to group your file listing by storage (per storage unit) or arrange files into folders, you must dereference the `storage_id` and/or `parent_id` field of the returned `LIBMTP_file_t` struct. To arrange by folders or files you typically have to create the proper trees by calls to **LIBMTP_Get_Storage() and/or LIBMTP_Get_Folder_List() first.**

Parameters:

device a pointer to the device to get the file listing for.
callback a function to be called during the tracklisting retrieveal for displaying progress bars etc, or NULL if you don't want any callbacks.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:

a list of files that can be followed using the `next` field of the `LIBMTP_file_t` data structure. Each of the metadata tags must be freed after use, and may contain only partial metadata information, i.e. one or several fields may be NULL or 0.

See also:

LIBMTP_Get_Filemetadata()

References LIBMTP_mtpdevice_struct::params.

Referenced by LIBMTP_Get_Filelisting().

LIBMTP_file_t* LIBMTP_Get_Filemetadata (LIBMTP_mtpdevice_t * device, uint32_t const fileid)

This function retrieves the metadata for a single file off the device.

Do not call this function repeatedly! The file handles are linearly searched O(n) and the call may involve (slow) USB traffic, so use **LIBMTP_Get_Filelisting() and cache the file, preferably as an efficient data structure such as a hash list.**

Incidentally this function will return metadata for a folder (association) as well, but this is not a proper use of it, it is intended for file manipulation, not folder manipulation.

Parameters:

device a pointer to the device to get the file metadata from.
fileid the object ID of the file that you want the metadata for.

Returns:

a metadata entry on success or NULL on failure.

See also:
LIBMTP_Get_Filelisting()

References LIBMTP_mtpdevice_struct::cached, and LIBMTP_mtpdevice_struct::params.

Referenced by LIBMTP_Set_Object_Filename().

LIBMTP_file_t* LIBMTP_Get_Files_And_Folders (LIBMTP_mtpdevice_t * device, uint32_t const storage, uint32_t const parent)
This function retrieves the contents of a certain folder with id parent on a certain storage on a certain device. The result contains both files and folders. The device used with this operations must have been opened with **LIBMTP_Open_Raw_Device_Uncached()** or it will fail.

NOTE: the request will always perform I/O with the device.

Parameters:
device a pointer to the MTP device to report info from.
storage a storage on the device to report info from. If 0 is passed in, the files for the given parent will be searched across all available storages.
parent the parent folder id.

References LIBMTP_mtpdevice_struct::cached, LIBMTP_mtpdevice_struct::params, and LIBMTP_mtpdevice_struct::usbinfo.

char const* LIBMTP_Get_Filetype_Description (LIBMTP_filetype_t intype)
This helper function returns a textual description for a libmtp file type to be used in dialog boxes etc.

Parameters:
intype the libmtp internal filetype to get a description for.

Returns:
a string representing the filetype, this must **NOT** be free():ed by the caller!

int LIBMTP_Get_Representative_Sample (LIBMTP_mtpdevice_t * device, uint32_t const id, LIBMTP_filesampledata_t * sampledata)
This routine gets representative sample data for an object. This uses the RepresentativeSampleData property of the album, if the device supports it.

Parameters:
device a pointer to the device which the object is on.
id unique id of the object to get data for.
pointer to LIBMTP_filesampledata_t struct to receive data

Returns:
0 on success, any other value means failure.

See also:
LIBMTP_Send_Representative_Sample()

LIBMTP_Get_Representative_Sample_Format()

LIBMTP_Create_New_Album()

References LIBMTP_mtpdevice_struct::params.

int LIBMTP_Get_Representative_Sample_Format (LIBMTP_mtpdevice_t * device, LIBMTP_filetype_t const filetype, LIBMTP_filesampledata_t ** sample)
This routine figures out whether a certain filetype supports representative samples (small thumbnail images) or not. This typically applies to JPEG files, MP3 files and Album abstract playlists, but in theory any filetype could support representative samples.

Parameters:
device a pointer to the device which is to be examined.
filetype the fileype to examine, and return the representative sample properties for.
sample this will contain a new sample type with the fields filled in with suitable default values. For example, the supported sample type will be set, the supported height and width will be set to max values if it is an image sample, and duration will also be given some suitable default value which should not be exceeded on audio samples. If the device does not support samples for this filetype, this pointer will be NULL. If it is not NULL, the user must destroy this struct with

LIBMTP_destroy_filesampledata_t() after use.

Returns:
0 on success, any other value means failure.

See also:
LIBMTP_Send_Representative_Sample()

LIBMTP_Create_New_Album()

References LIBMTP_mtpdevice_struct::params.

int LIBMTP_Get_Thumbnail (LIBMTP_mtpdevice_t * device, uint32_t const id, unsigned char ** data, unsigned int * size)
Retrieve the thumbnail for a file.

Parameters:
device a pointer to the device to get the thumbnail from.
id the object ID of the file to retrieve the thumbnail for.

Returns:
0 on success, any other value means failure.

References LIBMTP_mtpdevice_struct::params.

int LIBMTP_Handle_Events_Timeout_Completed (struct timeval * tv, int * completed)
Trivial wrapper around the most generic libusb method for polling for events. Can be used to drive asynchronous event detection.

LIBMTP_file_t* LIBMTP_new_file_t (void)
This creates a new file metadata structure and allocates memory for it. Notice that if you add strings to this structure they will be freed by the corresponding `LIBMTP_destroy_file_t` operation later, so be careful of using strdup() when assigning strings, e.g.:

LIBMTP_file_t *file = **LIBMTP_new_file_t()**;
file->filename = strdup(namestr);
....
LIBMTP_destroy_file_t(file);

Returns:
a pointer to the newly allocated metadata structure.

See also:
LIBMTP_destroy_file_t()

References LIBMTP_file_struct::filename.

LIBMTP_filesampledata_t* LIBMTP_new_filesampledata_t (void)
This creates a new sample data metadata structure and allocates memory for it. Notice that if you add strings to this structure they will be freed by the corresponding `LIBMTP_destroy_sampledata_t` operation later, so be careful of using strdup() when assigning strings.

Returns:
a pointer to the newly allocated metadata structure.

See also:
LIBMTP_destroy_sampledata_t()

References LIBMTP_filesampledata_struct::height.

int LIBMTP_Read_Event (LIBMTP_mtpdevice_t * device, LIBMTP_event_t * event, uint32_t * out1)
To read events sent by the device, repeatedly call this function from a secondary thread until the return value is < 0.

Parameters:
device a pointer to the MTP device to poll for events.
event contains a pointer to be filled in with the event retrieved if the call is successful.
out1 contains the param1 value from the raw event.

Returns:

0 on success, any other value means the polling loop shall be terminated immediately for this session.

References LIBMTP_mtpdevice_struct::params.

int LIBMTP_Read_Event_Async (LIBMTP_mtpdevice_t * device, LIBMTP_event_cb_fn cb, void * user_data)

This function reads events sent by the device, in a non-blocking manner. The callback function will be called when an event is received, but for the function to make progress, polling must take place, using LIBMTP_Handle_Events_Timeout_Completed.

After an event is received, this function should be called again to listen for the next event.

For now, this non-blocking mechanism only works with libusb-1.0, and not any of the other usb library backends. Attempting to call this method with another backend will always return an error.

Parameters:

device a pointer to the MTP device to poll for events.
cb a callback to be invoked when an event is received.
user_data arbitrary user data passed to the callback.

Returns:

0 on success, any other value means that the callback was not registered and no event notification will take place.

int LIBMTP_Send_File_From_File (LIBMTP_mtpdevice_t * device, char const *const path, LIBMTP_file_t *const filedata, LIBMTP_progressfunc_t const callback, void const *const data)

This function sends a local file to an MTP device. A filename and a set of metadata must be given as input.

Parameters:

device a pointer to the device to send the track to.
path the filename of a local file which will be sent.
filedata a file metadata set to be written along with the file. After this call the field `filedata->item_id` will contain the new file ID. Other fields such as the `filedata->filename`, `filedata->parent_id` or `filedata->storage_id` may also change during this operation due to device restrictions, so do not rely on the contents of this struct to be preserved in any way.

- `filedata->parent_id` should be set to the parent (e.g. folder) to store this file in. If this is 0, the file will be stored in the root folder.
- `filedata->storage_id` should be set to the desired storage (e.g. memory card or whatever your device presents) to store this file in. Setting this to 0 will store the file on the primary storage.

callback a progress indicator function or NULL to ignore.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:

0 if the transfer was successful, any other value means failure.

See also:

LIBMTP_Send_File_From_File_Descriptor()
LIBMTP_Delete_Object()

int LIBMTP_Send_File_From_File_Descriptor (LIBMTP_mtpdevice_t * device, int const fd, LIBMTP_file_t *const filedata, LIBMTP_progressfunc_t const callback, void const *const data)

This function sends a generic file from a file descriptor to an MTP device. A filename and a set of metadata must be given as input.
This can potentially be used for sending in a stream of unknown length. Send music files with **LIBMTP_Send_Track_From_File_Descriptor()**

Parameters:

device a pointer to the device to send the file to.
fd the filedescriptor for a local file which will be sent.
filedata a file metadata set to be written along with the file. After this call the field `filedata->item_id` will contain the new file ID. Other fields such as the

`filedata->filename`, `filedata->parent_id` or `filedata->storage_id` may also change during this operation due to device restrictions, so do not rely on the contents of this struct to be preserved in any way.

- `filedata->parent_id` should be set to the parent (e.g. folder) to store this file in. If this is 0, the file will be stored in the root folder.
- `filedata->storage_id` should be set to the desired storage (e.g. memory card or whatever your device presents) to store this file in. Setting this to 0 will store the file on the primary storage.

callback a progress indicator function or NULL to ignore.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:
0 if the transfer was successful, any other value means failure.

See also:
LIBMTP_Send_File_From_File()
LIBMTP_Send_Track_From_File_Descriptor()
LIBMTP_Delete_Object()

References LIBMTP_mtpdevice_struct::params, and LIBMTP_mtpdevice_struct::usbinfo.

int LIBMTP_Send_File_From_Handler (LIBMTP_mtpdevice_t * device, MTPDataGetFunc get_func, void * priv, LIBMTP_file_t *const filedata, LIBMTP_progressfunc_t const callback, void const *const data)

This function sends a generic file from a handler function to an MTP device. A filename and a set of metadata must be given as input.
This can potentially be used for sending in a stream of unknown length. Send music files with **LIBMTP_Send_Track_From_Handler()**

Parameters:
device a pointer to the device to send the file to.
get_func the function to call to get data to write
priv a user-defined pointer that is passed along to `get_func`. If not used, this is set to NULL.
filedata a file metadata set to be written along with the file. After this call the field `filedata->item_id` will contain the new file ID. Other fields such as the `filedata->filename`, `filedata->parent_id` or `filedata->storage_id` may also change during this operation due to device restrictions, so do not rely on the contents of this struct to be preserved in any way.

- `filedata->parent_id` should be set to the parent (e.g. folder) to store this file in. If this is 0, the file will be stored in the root folder.
- `filedata->storage_id` should be set to the desired storage (e.g. memory card or whatever your device presents) to store this file in. Setting this to 0 will store the file on the primary storage.

callback a progress indicator function or NULL to ignore.
data a user-defined pointer that is passed along to the `progress` function in order to pass along some user defined data to the progress updates. If not used, set this to NULL.

Returns:
0 if the transfer was successful, any other value means failure.

See also:
LIBMTP_Send_File_From_File()
LIBMTP_Send_Track_From_File_Descriptor()
LIBMTP_Delete_Object()

References LIBMTP_mtpdevice_struct::params, and LIBMTP_mtpdevice_struct::usbinfo.

int LIBMTP_Send_Representative_Sample (LIBMTP_mtpdevice_t * device, uint32_t const id, LIBMTP_filesampledata_t * sampledata)

This routine sends representative sample data for an object. This uses the RepresentativeSampleData property of the album, if the device supports it. The data should be of a format acceptable to the player (for iRiver and Creative, this seems to be JPEG) and must not be too large. (for a Creative, max seems to be about 20KB.) Check by calling **LIBMTP_Get_Representative_Sample_Format()** to get maximum size, dimensions, etc..

Parameters:

device a pointer to the device which the object is on.
id unique id of the object to set artwork for.
pointer to LIBMTP_filesampledata_t struct containing data

Returns:

0 on success, any other value means failure.

See also:

LIBMTP_Get_Representative_Sample()
LIBMTP_Get_Representative_Sample_Format()
LIBMTP_Create_New_Album()

References LIBMTP_mtpdevice_struct::params, and LIBMTP_mtpdevice_struct::usbinfo.

int LIBMTP_Set_File_Name (LIBMTP_mtpdevice_t * device, LIBMTP_file_t * file, const char * newname)

This function renames a single file. This simply means that the PTP_OPC_ObjectFileName property is updated, if this is supported by the device.

Parameters:

device a pointer to the device that contains the file.
file the file metadata of the file to rename. On success, the filename member is updated. Be aware, that this name can be different than newname depending of device restrictions.
newname the new filename for this object.

Returns:

0 on success, any other value means failure.

Author

Generated automatically by Doxygen for libmtp from the source code.