drmlib Documentation

Release v1.1

jeydoux@accelize.com

Nov 08, 2018

Contents

CHAPTER 1

C++ API

1.1 Namespaces

namespace DRMLib

class Exception : **public** runtime_error
#include </home/docs/checkouts/readthedocs.org/user_builds/sd-drmlib/checkouts/latest/include/accelize/drm/error.h>
Exception class with error code for the DRMLib.

This class is an exception that may be thrown by the DRMLib in case of synchronous error

class MeteringSessionManager
#include </home/docs/checkouts/readthedocs.org/user_builds/sd-drmlib/checkouts/latest/include/accelize/drm/session_man
manages a session of metering on a DRM controller.

The session of metering life-cycle is:

- 1. starting a session : from a locked DRM, install the first license from Accelize WebService which unlocks the protected IPs.
- 2. keep protected IPs unlocked by regularly supplying licenses and uploading metering : this operation will be done in a dedicated thread managed by this class. It will involve communication with the DRM controller and the Accelize WebService.
- 3. May be paused : the DRM is no longer supplied with new licenses. Between pause(3) and resume(4) the end of license time maybe reached and therefore the protected IPs may be locked.
- 4. May be resumed after paused : the manager makes sure that the protected IPs is unlocked back again.
- 5. stop a session : uploads last metering data to the Accelize WebService and locks the protected IPs. After stop the session is correctly ended on the Accelize Webservice.

1.2 Classes

class MeteringSessionManager
manages a session of metering on a DRM controller.

The session of metering life-cycle is:

- 1. starting a session : from a locked DRM, install the first license from Accelize WebService which unlocks the protected IPs.
- 2. keep protected IPs unlocked by regularly supplying licenses and uploading metering : this operation will be done in a dedicated thread managed by this class. It will involve communication with the DRM controller and the Accelize WebService.
- 3. May be paused : the DRM is no longer supplied with new licenses. Between pause(3) and resume(4) the end of license time maybe reached and therefore the protected IPs may be locked.
- 4. May be resumed after paused : the manager makes sure that the protected IPs is unlocked back again.
- 5. stop a session : uploads last metering data to the Accelize WebService and locks the protected IPs. After stop the session is correctly ended on the Accelize Webservice.

class Exception : **public** runtime_error
Exception class with error code for the DRMLib.

This class is an exception that may be thrown by the DRMLib in case of synchronous error

1.3 Files

1.3.1 session_manager.h

namespace Accelize

namespace DRMLib

class MeteringSessionManager
*#include </home/docs/checkouts/readthedocs.org/user_builds/sd-drmlib/checkouts/latest/include/accelize/drm/session_*
manages a session of metering on a DRM controller.

The session of metering life-cycle is:
- 1. starting a session : from a locked DRM, install the first license from Accelize WebService which unlocks the protected IPs.
- 2. keep protected IPs unlocked by regularly supplying licenses and uploading metering : this operation will be done in a dedicated thread managed by this class. It will involve communication with the DRM controller and the Accelize WebService.
- 3. May be paused : the DRM is no longer supplied with new licenses. Between pause(3) and resume(4) the end of license time maybe reached and therefore the protected IPs may be locked.
- 4. May be resumed after paused : the manager makes sure that the protected IPs is unlocked back again.
- 5. stop a session : uploads last metering data to the Accelize WebService and locks the protected IPs. After stop the session is correctly ended on the Accelize Webservice.

1.3.1.0.1 Public Types

typedef std::function<int (uint32_t, uint32_t *) > **ReadReg32ByOffsetHandler**
Read callback function by offset.

Offset is relative to first register of DRM controller
Warning This function must be thread-safe in case of concurrency on the register bus

typedef std::function<int (uint32_t, uint32_t) > **WriteReg32ByOffsetHandler**
Write callback function by offset.

Offset is relative to first register of DRM controller
Warning This function must be thread-safe in case of concurrency on the register bus

typedef std::function<int (**const** std::string&, uint32_t *) > **ReadReg32ByNameHandler**
Read callback function by register name.

Note This function is intended to be used with QuickPlayLib that manages registers by names
Warning This function must be thread-safe in case of concurrency on the register bus

typedef std::function<int (**const** std::string&, uint32_t) > **WriteReg32ByNameHandler**
Write callback function by register name.

Note This function is intended to be used with QuickPlayLib that manages registers by names
Warning This function must be thread-safe in case of concurrency on the register bus

typedef std::function<void (**const** std::string&) > **ErrorCallBackHandler**
Asynchronous Error callback function.

This function is called in case of asynchronous error during operation

1.3.1.0.2 Public Functions

MeteringSessionManager ()
No default constructor.

MeteringSessionManager (**const** std::string *&conf_file_path*, **const** std::string *&cred_file_path*, *ReadReg32ByOffsetHandler f_drm_read32*, *WriteReg32ByOffsetHandler f_drm_write32*, *ErrorCallBackHandler f_error_cb*)
Constructor with "ByOffset" functions.

Constructs a MeteringSessionManager with ByOffset callback functions.
See *ReadReg32ByOffsetHandler WriteReg32ByOffsetHandler*
Parameters

- `conf_file_path`: : Path to json configuration file with configuration of the manager (Accelize WebService URL, Board type, Udid, ...)
- `cred_file_path`: : Path to json file containing your accelstore.accelize.com Account credentials (client_id, client_private)
- `f_drm_read32`: : Callback function for reading DRM 32bit registers by offset
- `f_drm_write32`: : Callback function for writing DRM 32bit registers by offset
- `f_error_cb`: : Callback function in case of asynchronous error

MeteringSessionManager (*ReadReg32ByOffsetHandler f_drm_read32*, *WriteReg32ByOffsetHandler f_drm_write32*, *ErrorCallBackHandler f_error_cb*)
Constructor with "ByOffset" functions, uninitialized webservice connection.

Constructs a MeteringSessionManager with ByOffset callback functions.
See *ReadReg32ByOffsetHandler WriteReg32ByOffsetHandler*
Warning Only used for Debug purposes
Parameters
- `f_drm_read32`: : Callback function for reading DRM 32bit registers by offset
- `f_drm_write32`: : Callback function for writing DRM 32bit registers by offset
- `f_error_cb`: : Callback function in case of asynchronous error

MeteringSessionManager(**const** std::string *&conf_file_path*, **const** std::string *&cred_file_path*, *ReadReg32ByNameHandler f_drm_read32*, *WriteReg32ByNameHandler f_drm_write32*, *ErrorCallBackHandler f_error_cb*)
Constructor with "ByName" functions.

Constructs a MeteringSessionManager with ByName callback functions.
See *ReadReg32ByNameHandler WriteReg32ByNameHandler*
Note This function is intended to be used with QuickPlayLib that manages IP registers by names
Parameters
- `conf_file_path`: : Path to json configuration file with configuration of the manager (Accelize WebService URL, Board type, Udid, ...)
- `cred_file_path`: : Path to json file containing your accelstore.accelize.com Account credentials (client_id, client_private)
- `f_drm_read32`: : Callback function for reading DRM 32bit registers by name
- `f_drm_write32`: : Callback function for writing DRM 32bit registers by name
- `f_error_cb`: : Callback function in case of asynchronous error

MeteringSessionManager(*ReadReg32ByNameHandler f_drm_read32*, *WriteReg32ByNameHandler f_drm_write32*, *ErrorCallBackHandler f_error_cb*)
Constructor with "ByName" functions, uninitialized webservice connection.

Constructs a MeteringSessionManager with ByName callback functions.
See *ReadReg32ByNameHandler WriteReg32ByNameHandler*
Note This function is intended to be used with QuickPlayLib that manages IP registers by names
Warning Only used for Debug purposes
Parameters
- `f_drm_read32`: : Callback function for reading DRM 32bit registers by name
- `f_drm_write32`: : Callback function for writing DRM 32bit registers by name
- `f_error_cb`: : Callback function in case of asynchronous error

MeteringSessionManager(**const** *MeteringSessionManager*&)
Non-copyable.

MeteringSessionManager(*MeteringSessionManager*&&)
Support move.

~MeteringSessionManager()
Destructor.

void **setUdid**(**const** std::string *&udid*)
Set Udid.

This function can be used in case the Udid is not already set in the configuration file. In case the Udid has already been set by the configuration file, the new provided Udid will be used. The Udid will be transmitted to the Accelize WebService as additional information about the metering session. Please contact us if you don't know what Udid to use.
Parameters

- `udid`: : udid as a std::string, format "00000000-0000-0000-0000-000000000000"

void **setBoardType**(**const** std::string &*boardType*)
Set BoardType.

This function can be used in case the BoardType is not already set in the configuration file. In case the BoardType has already been set by the configuration file, the new provided BoardType will be used. The BoardType will be transmitted to the Accelize WebService as additional information about the metering session. Please contact us if you don't know what BoardType to use.
Parameters
- `boardType`: : boardType as a std::string

void **start_session**()
Start a session (1)

This function will always start a new session of metering. The behavior is undefined in case a session was already started. This function will start a std::thread to keep session active with new licenses when necessary (2). Any error during this thread execution will be reported asynchronously via the provided ErrorCallBackHandler. Once this function returns successfully the protected IPs are guaranteed to be unlocked.

void **stop_session**()
Stop a session (5)

This function will stop a session of metering. This function will join the thread keeping the session active (2) that may have been started by *auto_start_session()*, *start_session()* or *resume_session()*. Once this function returns successfully the protected IPs are guaranteed to be locked and the session has been correctly ended on Accelize WebService.

void **pause_session**()
Pause a session (3)

This function will pause a session of metering. This function will join the thread keeping the session active (2) that may have been started by *auto_start_session()*, *start_session()* or *resume_session()*. Once this function returns successfully the protected IPs may or may not be locked depending on the time left for licenses currently installed on the DRM controller. Though it is guaranteed that protected IPs will be locked if those licenses reach the end of their validity time

void **resume_session**()
Resume a session (4)

This function will resume a session of metering. This function will start a std::thread to keep session active with new licenses when necessary (2). Any error during this thread execution will be reported asynchronously via the provided ErrorCallBackHandler. Once this function returns successfully the protected IPs are guaranteed to be unlocked.

void **auto_start_session**()
Automatically Start or Resume a session (4) This function will automatically start or resume a session of metering. Depending on detected state of the DRM controller this function will call *start_session()* or *resume_session()*. As a consequence : This function will start a std::thread to keep session active with new licenses when necessary (2). Once this function returns successfully the protected IPs are guaranteed to be unlocked.

void **dump_drm_hw_report**(std::ostream &*os*)
Dump DRM Hardware report.

Note Only for Debug purposes
Parameters

- `os` : : std::ostream on which hardware report will be dumped as text

1.3.1.0.3 Private Members

Impl ***pImpl**
Internal representation.

Internal representation

1.3.2 error.h

namespace Accelize

namespace DRMLib

class Exception : **public** runtime_error
#include </home/docs/checkouts/readthedocs.org/user_builds/sd-drmlib/checkouts/latest/include/accelize/drm/error.h>
Exception class with error code for the DRMLib.

This class is an exception that may be thrown by the DRMLib in case of synchronous error

1.3.2.0.1 Public Functions

template <**class** S>
Exception(*DRMLibErrorCode errCode*, S &&*errMsg*)

virtual ~Exception()

DRMLibErrorCode **getErrCode**() **const**

virtual const char ***what**() **const**

1.3.2.0.2 Protected Attributes

DRMLibErrorCode **errCode**
error code from the DRMLibErrorCode enum

std::string **errWhat**
internal error message to be accessed from *what()*

1.3.3 version.h

CHAPTER 2

C API

2.1 Files

2.1.1 common.h

2.1.1.0.1 Defines

DRMLIB_EXPORT

DRMLIB_LOCAL

2.1.2 errorcode.h

Header defining all error codes of DRMLib.

2.1.2.0.1 Enums

enum DRMLibErrorCode
Error code enum.

Values:

DRMLibOK = 0
Function returned successfully

DRMBadArg = 00001
Bad argument provided

DRMBadFormat = 00002
Bad format of provided input or config file

DRMExternFail = 00003
Fail happened in an external library

DRMBadUsage = 00004
Wrong usage of the DRMLib

DRMWSRespError = 10001
A malformed response has been received from Accelize WebService

DRMWSReqError = 10002
Failed during HTTP request to Accelize WebService

DRMWSError = 10003
Error returned from Accelize WebService

DRMWSMayRetry = 10004
Error with request to Accelize Webservice, retry advised

DRMCtlrError = 20001
An error happened on a command on the DRM controller

DRMLibFatal = 90001
Fatal error, unknown error (Please contact Accelize)

DRMLibAssert = 90002
Assertion failed internally (Please contact Accelize)

2.1.3 metering.h

Header for using MeteringSessionManager feature of the DRMLib.

Note This C API is wrapping C++ API, please refer to C++ API

2.1.3.0.1 Typedefs

typedef struct MeteringSessionManager_s **MeteringSessionManager**
Wrapper struct around C++ Accelize::DRMLib::MeteringSessionManager.

typedef int (***ReadReg32ByOffsetHandler**) (uint32_t, uint32_t *, void *user_p)
Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::ReadReg32ByOffsetHandler*.

Note user_p is the user pointer provided at construction to MeteringSessionManager_alloc

typedef int (***WriteReg32ByOffsetHandler**) (uint32_t, uint32_t, void *user_p)
Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::WriteReg32ByOffsetHandler*.

Note user_p is the user pointer provided at construction to MeteringSessionManager_alloc

typedef void (***ErrorCallBackHandler**) (**const** char *, void *user_p)
Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::ErrorCallBackHandler*.

Note user_p is the user pointer provided at construction to MeteringSessionManager_alloc

2.1.3.0.2 Functions

DRMLibErrorCode **MeteringSessionManager_alloc**(*MeteringSessionManager* ***p_m*, **const** char ***conf_file_path*, **const** char ***cred_file_path*, *ReadReg32ByOffsetHandler* *f_drm_read32*, *WriteReg32ByOffsetHandler* *f_drm_write32*, *ErrorCallBackHandler* *f_error_cb*, void ***user_p*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::MeteringSessionManager* constructor.

Parameters

- `**p_m`: : pointer to a MeteringSessionManager pointer that will be set to the new constructed object
- `conf_file_pathcred_file_pathf_drm_read32f_drm_write32f_error_cb`: : see C++ API documentation
- `user_p`: : user pointer that will be passed to the callback functions

DRMLibErrorCode **MeteringSessionManager_free**(*MeteringSessionManager* ***p_m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::~MeteringSessionManager* destructor.

Parameters

- `**p_m`: : pointer to a MeteringSessionManager pointer that will be freed. After *p_m == NULL.

DRMLibErrorCode **MeteringSessionManager_start_session**(*MeteringSessionManager* **m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::start_session*.

Parameters

- `*m`: : pointer to a MeteringSessionManager object

DRMLibErrorCode **MeteringSessionManager_stop_session**(*MeteringSessionManager* **m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::stop_session*.

Parameters

- `*m`: : pointer to a MeteringSessionManager object

DRMLibErrorCode **MeteringSessionManager_pause_session**(*MeteringSessionManager* **m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::pause_session*.

Parameters

- `*m`: : pointer to a MeteringSessionManager object

DRMLibErrorCode **MeteringSessionManager_resume_session**(*MeteringSessionManager* **m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::resume_session*.

Parameters

- `*m`: : pointer to a MeteringSessionManager object

DRMLibErrorCode **MeteringSessionManager_auto_start_session**(*MeteringSessionManager* **m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::auto_start_session*.

Parameters

- *m: : pointer to a MeteringSessionManager object

DRMLibErrorCode **MeteringSessionManager_dump_drm_hw_report**(*MeteringSessionManager* **m*)

Wrapper typedef around C++ *Accelize::DRMLib::MeteringSessionManager::dump_drm_hw_report*.

Parameters

- *m: : pointer to a MeteringSessionManager object

2.1.4 version.h

Header for getting version of the DRMLib.

Note This C API is wrapping C++ API, please refer to C++ API

2.1.4.0.1 Functions

const char ***DRMLib_get_version**()

Return version of the library as text.

Index

R

W