SPDMAppLib

Generated by Doxygen 1.9.4

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 spdmapplib Namespace Reference

Namespaces

- namespace errorcodes

  spdmapplib error codes list.

Classes

- struct SPDMConfiguration

  SPDM configurations from EntityManager.
- struct spdmItem

  SPDM device context structure.
- class SPDMRequester

  The requester base class.
- class SPDMRequesterImpl

  SPDM requester implementation class.
- class SPDMResponder

  The responder base class.
- class SPDMResponderImpl

  SPDM responder implementation class.

Enumerations

- enum class SPDMVersions : uint32_t { **spdmv1p1** = 0x01 }

  SPDM version enum.
- enum class **SPDMDeviceEvent** : uint8_t { **deviceAdded** = 0x01 , **deviceRemoved** }

Functions

- std::shared_ptr< SPDMRequester > createRequester ()

  Requester object create Factory function.
- std::shared_ptr< SPDMResponder > createResponder ()

  Responder object create Factory function.
- void setCertificatePath (std::string &certPath)

  set cert file Path

5.1.1 Detailed Description

5.1.2 Function Documentation

5.1.2.1 createRequester()

```
std::shared_ptr< SPDMRequester > spdmapplib::createRequester ( )
```

Requester object create Factory function.

Returns

Pointer to Requester implementation object.

5.1.2.2 createResponder()

```
std::shared_ptr< SPDMResponder > spdmapplib::createResponder ( )
```

Responder object create Factory function.

Returns

Pointer to Responder implementation object.

5.1.2.3 setCertificatePath()

```
void spdmapplib::setCertificatePath (
            std::string & certPath )
```

set cert file Path

Parameters

certPath	: cert file location

5.2 spdmapplib::errorcodes Namespace Reference

spdmapplib error codes list.

Variables

- constexpr int **generalReturnError** = -1
- constexpr int **returnSuccess** = 0
- constexpr int **spdmConfigurationNotFoundInEntityManager** = 1
- constexpr int libspdmReturnError

5.2.1 Detailed Description

spdmapplib error codes list.

5.2.2 Variable Documentation

5.2.2.1 libspdmReturnError

```
constexpr int spdmapplib::errorcodes::libspdmReturnError  [inline], [constexpr]
```

Initial value:

```
=
    2
```

5.3 spdmtransport Namespace Reference

Classes

- class SPDMTransport

  SPDM transport layer class.
- class SPDMTransportMCTP

  SPDM transport layer implemented using MCTP.
- struct TransportEndPoint

  Endpoint information, could be extended.

Typedefs

- using **MsgReceiveCallback** = std::function< void(TransportEndPoint &transEP, const std::vector< uint8_t > &data)>
- using **AddRemoveDeviceCallback** = std::function< int(TransportEndPoint &transEP)>

Enumerations

- enum class TransportIdentifier : uint8_t { **mctpOverSMBus** = 0x01 , **mctpOverPCIe** = 0x02 , **pmtWatcher** = 0x03 }

  SPDM Transport type, could be extended.

5.3.1 Detailed Description

Chapter 6

Class Documentation

6.1 spdmapplib::SPDMConfiguration Struct Reference

SPDM configurations from EntityManager.

`#include <spdmapplib.hpp>`

Public Attributes

- uint32_t **version**
- uint32_t **capability**
- uint32_t **hash**
- uint32_t **measHash**
- uint32_t **asym**
- uint32_t **reqasym**
- uint32_t **dhe**
- uint32_t **aead**
- uint32_t **slotcount**
- std::string **certPath**

6.1.1 Detailed Description

SPDM configurations from EntityManager.

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

- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmapplib.hpp

6.2 spdmapplib::spdmItem Struct Reference

SPDM device context structure.

`#include <spdmapplib_impl.hpp>`

Public Attributes

- void * **pspdmContext**
- spdmtransport::TransportEndPoint **transEP**
- uint8_t **useSlotId**
- uint32_t **sessionId**
- uint32_t **useVersion**
- uint16_t **useReqAsymAlgo**
- uint32_t **useMeasurementHashAlgo**
- uint32_t **useAsymAlgo**
- uint32_t **useHashAlgo**
- libspdm_connection_state_t **connectStatus**
- std::vector< uint8_t > **data**
- std::vector< uint8_t > **dataCert**
- std::vector< uint8_t > **dataMeas**

6.2.1 Detailed Description

SPDM device context structure.

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

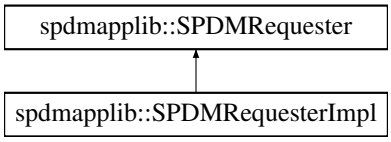
- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmapplib_impl.hpp

6.3 spdmapplib::SPDMRequester Class Reference

The requester base class.

`#include <spdmapplib.hpp>`

Inheritance diagram for spdmapplib::SPDMRequester:



Public Member Functions

- virtual int initRequester (std::shared_ptr< boost::asio::io_context > ioc, std::shared_ptr< sdbusplus::asio↩ ::connection > conn, std::shared_ptr< spdmtransport::SPDMTransport > trans, spdmtransport::TransportEndPoint &transResponder, SPDMConfiguration &pSpdmConfig)=0

  Initial function of SPDM requester.
- virtual int doAuthentication (void)=0

  The authentication function.
- virtual int doMeasurement (const uint32_t *sessionid)=0

  The measurement function.
- virtual std::optional< std::vector< uint8_t > > getMeasurements ()=0

  Get all measurement function.
- virtual std::optional< std::vector< uint8_t > > getCertificate ()=0

  Get certification function.

6.3.1 Detailed Description

The requester base class.

6.3.2 Member Function Documentation

6.3.2.1 doAuthentication()

```
virtual int spdmapplib::SPDMRequester::doAuthentication (
            void  )  [pure virtual]
```

The authentication function.

Returns

0: success, other: failed.

Implemented in spdmapplib::SPDMRequesterImpl.

6.3.2.2 doMeasurement()

```
virtual int spdmapplib::SPDMRequester::doMeasurement (
            const uint32_t * sessionid )  [pure virtual]
```

The measurement function.

Parameters

sessionid	The session id pointer(reserved for further use).

Returns

0: success, other: failed.

Implemented in spdmapplib::SPDMRequesterImpl.

6.3.2.3 getCertificate()

```
virtual std::optional< std::vector< uint8_t > > spdmapplib::SPDMRequester::getCertificate ( )
[pure virtual]
```

Get certification function.

Returns

vector of certification.

Implemented in spdmapplib::SPDMRequesterImpl.

6.3.2.4 getMeasurements()

```
virtual std::optional< std::vector< uint8_t > > spdmapplib::SPDMRequester::getMeasurements (
) [pure virtual]
```

Get all measurement function.

Returns

vector of all measurements.

Implemented in spdmapplib::SPDMRequesterImpl.

6.3.2.5 initRequester()

```
virtual int spdmapplib::SPDMRequester::initRequester (
            std::shared_ptr< boost::asio::io_context > ioc,
            std::shared_ptr< sdbusplus::asio::connection > conn,
            std::shared_ptr< spdmtransport::SPDMTransport > trans,
            spdmtransport::TransportEndPoint & transResponder,
            SPDMConfiguration & pSpdmConfig ) [pure virtual]
```

Initial function of SPDM requester.

Parameters

ioc	The shared_ptr to boost io_context object..
trans	The pointer of transport instance.
ptransResponder	The pointer to assigned responder EndPoint.

Returns

0: success, other: listed in spdmapplib::errorCodes.

Implemented in spdmapplib::SPDMRequesterImpl.

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

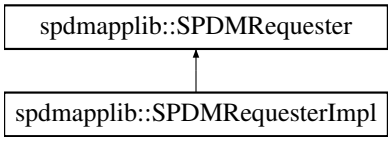
- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmapplib.hpp

6.4 spdmapplib::SPDMRequesterImpl Class Reference

SPDM requester implementation class.

`#include <spdmapplib_impl.hpp>`

Inheritance diagram for spdmapplib::SPDMRequesterImpl:



Public Member Functions

- int initRequester (std::shared_ptr< boost::asio::io_context > ioc, std::shared_ptr< sdbusplus::asio↩ ::connection > conn, std::shared_ptr< spdmtransport::SPDMTransport > trans, spdmtransport::TransportEndPoint &transResponder, SPDMConfiguration &spdmConfig) override

  Initial function of SPDM requester.
- int doAuthentication (void) override

  The authentication function.
- int doMeasurement (const uint32_t *sessionid) override

  The measurement function.
- std::optional< std::vector< uint8_t > > getMeasurements () override

  Get all measurement function.
- std::optional< std::vector< uint8_t > > getCertificate () override

  Get certification function.
- int addData (spdmtransport::TransportEndPoint &transEP, const std::vector< uint8_t > &data)

  Set received data to assigned endpoint.
- int checkResponderDevice (spdmtransport::TransportEndPoint &transEP)

  Function to check if found endpoint is the responder assigned by user.
- int msgRecvCallback (spdmtransport::TransportEndPoint &transEP, const std::vector< uint8_t > &data)

  Function to pass as parameter of syncSendRecvData of transport layer.
- return_status deviceSendMessage (void *spdmContext, const std::vector< uint8_t > &request, uint64_↩ t timeout)

  Register to libspdm for sending SPDM payload.
- return_status deviceReceiveMessage (void *spdmContext, std::vector< uint8_t > &response, uint64_t time-out)

  Register to libspdm for receiving SPDM response payload.

Protected Member Functions

- int setupResponder (const spdmtransport::TransportEndPoint &transEP)

  Setup the configuration of user assigned endpoint as target responder.
- int settingFromConfig (void)

  Function to setup user assigned endpoint initial configuration.

6.4.1 Detailed Description

SPDM requester implementation class.

6.4.2 Member Function Documentation

6.4.2.1 addData()

```
int spdmapplib::SPDMRequesterImpl::addData (
            spdmtransport::TransportEndPoint & transEP,
            const std::vector< uint8_t > & data )
```

Set received data to assigned endpoint.

Parameters

transEP	The Endpoint object to receive data.
trans	The pointer of transport instance.

Returns

0: success, other: failed.

6.4.2.2 checkResponderDevice()

```
int spdmapplib::SPDMRequesterImpl::checkResponderDevice (
            spdmtransport::TransportEndPoint & transEP )
```

Function to check if found endpoint is the responder assigned by user.

Parameters

transEP	The endpoint object to be checked.

Returns

0: success, other: failed.

6.4.2.3 deviceReceiveMessage()

```
return_status spdmapplib::SPDMRequesterImpl::deviceReceiveMessage (
            void * spdmContext,
```

```
            std::vector< uint8_t > & response,
            uint64_t timeout )
```

Register to libspdm for receiving SPDM response payload.

Parameters

spdmContext	The pointer of the spdmcontext.
response	The response data buffer vector.
timeout	The timeout time.

Returns

return_status defined in libspdm.

6.4.2.4 deviceSendMessage()

```
return_status spdmapplib::SPDMRequesterImpl::deviceSendMessage (
            void * spdmContext,
            const std::vector< uint8_t > & request,
            uint64_t timeout )
```

Register to libspdm for sending SPDM payload.

Parameters

spdmContext	The pointer of the spdmcontext.
request	The request payload data vector.
timeout	The timeout time.

Returns

return_status defined in libspdm.

6.4.2.5 doAuthentication()

```
int spdmapplib::SPDMRequesterImpl::doAuthentication (
            void  )  [override], [virtual]
```

The authentication function.

Returns

0: success, other: failed.

Implements spdmapplib::SPDMRequester.

6.4.2.6 doMeasurement()

```
int spdmapplib::SPDMRequesterImpl::doMeasurement (
            const uint32_t * sessionid )  [override], [virtual]
```

The measurement function.

Parameters

sessionid	The session id pointer(reserved for further use).

Returns

0: success, other: failed.

Implements spdmapplib::SPDMRequester.

6.4.2.7 getCertificate()

```
std::optional< std::vector< uint8_t > > spdmapplib::SPDMRequesterImpl::getCertificate ( )
[override], [virtual]
```

Get certification function.

Returns

vector of certification.

Implements spdmapplib::SPDMRequester.

6.4.2.8 getMeasurements()

```
std::optional< std::vector< uint8_t > > spdmapplib::SPDMRequesterImpl::getMeasurements ( )
[override], [virtual]
```

Get all measurement function.

Returns

vector of all measurements.

Implements spdmapplib::SPDMRequester.

6.4.2.9 initRequester()

```
int spdmapplib::SPDMRequesterImpl::initRequester (
            std::shared_ptr< boost::asio::io_context > ioc,
            std::shared_ptr< sdbusplus::asio::connection > conn,
            std::shared_ptr< spdmtransport::SPDMTransport > trans,
            spdmtransport::TransportEndPoint & transResponder,
            SPDMConfiguration & spdmConfig )  [override], [virtual]
```

Initial function of SPDM requester.

Parameters

ioc	boost io_context object..
trans	The pointer of transport instance.
ptransResponder	The pointer to assigned responder EndPoint.

Returns

0: success, other: listed in spdmapplib::errorCodes.

Implements spdmapplib::SPDMRequester.

6.4.2.10 msgRecvCallback()

```
int spdmapplib::SPDMRequesterImpl::msgRecvCallback (
            spdmtransport::TransportEndPoint & transEP,
            const std::vector< uint8_t > & data )
```

Function to pass as parameter of syncSendRecvData of transport layer.

The function will be called when send/receive is completed in transport layer.

Parameters

transEP	The endpoint to receive data after send. to.
data	The received data buffer.

Returns

0: success, other: failed.

6.4.2.11 settingFromConfig()

```
int spdmapplib::SPDMRequesterImpl::settingFromConfig (
            void ) [protected]
```

Function to setup user assigned endpoint initial configuration.

Returns

0: success, other: failed.

6.4.2.12 setupResponder()

```
int spdmapplib::SPDMRequesterImpl::setupResponder (
            const spdmtransport::TransportEndPoint & transEP ) [protected]
```

Setup the configuration of user assigned endpoint as target responder.

Parameters

transEP	The endpoint object to be configured.

Returns

return_status defined in libspdm.

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

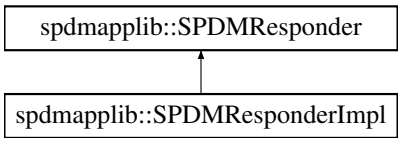
- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmapplib_impl.hpp

6.5 spdmapplib::SPDMResponder Class Reference

The responder base class.

`#include <spdmapplib.hpp>`

Inheritance diagram for spdmapplib::SPDMResponder:



Public Member Functions

- virtual int initResponder (std::shared_ptr< boost::asio::io_context > ioc, std::shared_ptr< sdbusplus↩ ::asio::connection > conn, std::shared_ptr< spdmtransport::SPDMTransport > trans, SPDMConfiguration &spdmConfig)=0

  Initial function of SPDM responder When the function is called, it will enter daemon mode and never return.

6.5.1 Detailed Description

The responder base class.

6.5.2 Member Function Documentation

6.5.2.1 initResponder()

```
virtual int spdmapplib::SPDMResponder::initResponder (
            std::shared_ptr< boost::asio::io_context > ioc,
            std::shared_ptr< sdbusplus::asio::connection > conn,
            std::shared_ptr< spdmtransport::SPDMTransport > trans,
            SPDMConfiguration & spdmConfig )  [pure virtual]
```

Initial function of SPDM responder When the function is called, it will enter daemon mode and never return.

Parameters

ioc	boost io_context object..
trans	The pointer of transport instance.
spdmConfig	Application assigned SPDMConfiguration.

Returns

0: success, other: listed in spdmapplib::errorCodes.

Implemented in spdmapplib::SPDMResponderImpl.

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

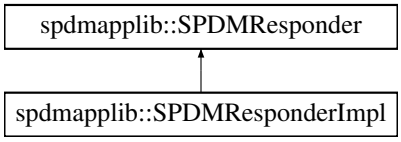
- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmapplib.hpp

6.6 spdmapplib::SPDMResponderImpl Class Reference

SPDM responder implementation class.

`#include <spdmapplib_impl.hpp>`

Inheritance diagram for spdmapplib::SPDMResponderImpl:



Public Member Functions

- int initResponder (std::shared_ptr< boost::asio::io_context > ioc, std::shared_ptr< sdbusplus::asio↩ ::connection > conn, std::shared_ptr< spdmtransport::SPDMTransport > trans, SPDMConfiguration &spdmConfig) override

  Initial function of SPDM responder.
- spdmItem & createSPDMItem (spdmtransport::TransportEndPoint &transEP)

  Called when need to create a new spdmItem.
- spdmItem & findSPDMItem (spdmtransport::TransportEndPoint &transEP)

  find assigned EndPoint's spdmItem reference in spdmPool If item not exist, create a new one add to spdmPool and return the item.
- int removeDevice (spdmtransport::TransportEndPoint &transEP)

  Called when endpoint remove is detected.
- int processSPDMMessage (spdmItem &transEP)

  Called when message received.
- int msgRecvCallback (spdmtransport::TransportEndPoint &transEP, const std::vector< uint8_t > &data)

  Register to transport layer for handling received data.
- return_status deviceSendMessage (void *spdmContext, const std::vector< uint8_t > &request, uint64_↩ t timeout)

Register to libspdm for sending SPDM payload.

- return_status deviceReceiveMessage (void *spdmContext, std::vector< uint8_t > &response, uint64_t timeout)

  Register to libspdm for receiving SPDM response payload.

- void processConnectionState (void *spdmContext, libspdm_connection_state_t connectionState)

  Register to libspdm for handling connection state change.

- void processSessionState (void *spdmContext, uint32_t sessionID, libspdm_session_state_t sessionState)

  Register to libspdm for handling session state change.

Protected Member Functions

- int settingFromConfig (uint8_t ItemIndex)

  Function to setup specific endpoint initial configuration.

6.6.1 Detailed Description

SPDM responder implementation class.

6.6.2 Member Function Documentation

6.6.2.1 createSPDMItem()

```
spdmItem & spdmapplib::SPDMResponderImpl::createSPDMItem (
            spdmtransport::TransportEndPoint & transEP )
```

Called when need to create a new spdmItem.

Parameters

transEP	The new endpoint object.

Returns

reference of spdmItem: success, other: throw exception.

6.6.2.2 deviceReceiveMessage()

```
return_status spdmapplib::SPDMResponderImpl::deviceReceiveMessage (
            void * spdmContext,
            std::vector< uint8_t > & response,
            uint64_t timeout )
```

Register to libspdm for receiving SPDM response payload.

Parameters

spdmContext	The pointer of the spdmcontext.
response	The response data buffer vector.
timeout	The timeout time.

Returns

return_status defined in libspdm.

6.6.2.3 deviceSendMessage()

```
return_status spdmapplib::SPDMResponderImpl::deviceSendMessage (
            void * spdmContext,
            const std::vector< uint8_t > & request,
            uint64_t timeout )
```

Register to libspdm for sending SPDM payload.

Parameters

spdmContext	The pointer of the spdmcontext.
request	The request payload data vector.
timeout	The timeout time.

Returns

return_status defined in libspdm.

6.6.2.4 findSPDMItem()

```
spdmItem & spdmapplib::SPDMResponderImpl::findSPDMItem (
            spdmtransport::TransportEndPoint & transEP )
```

find assigned EndPoint's spdmItem reference in spdmPool If item not exist, create a new one add to spdmPool and return the item.

Parameters

transEP	The new endpoint object.

Returns

reference of spdmItem: success, other: throw exception.

6.6.2.5 initResponder()

```
int spdmapplib::SPDMResponderImpl::initResponder (
            std::shared_ptr< boost::asio::io_context > ioc,
            std::shared_ptr< sdbusplus::asio::connection > conn,
            std::shared_ptr< spdmtransport::SPDMTransport > trans,
            SPDMConfiguration & spdmConfig )  [override], [virtual]
```

Initial function of SPDM responder.

The function will enter daemon mode. Accept request from assigned transport layer.

Parameters

ioc	boost io_context object..
trans	The pointer of transport instance.
spdmConfig	Application assigned SPDMConfiguration.

Returns

0: success, other: listed in spdmapplib::errorCodes.

Implements spdmapplib::SPDMResponder.

6.6.2.6 msgRecvCallback()

```
int spdmapplib::SPDMResponderImpl::msgRecvCallback (
            spdmtransport::TransportEndPoint & transEP,
            const std::vector< uint8_t > & data )
```

Register to transport layer for handling received data.

Parameters

transEP	The endpoint object to receive data.
data	The vector of received data.

Returns

0: success, other: failed.

6.6.2.7 processConnectionState()

```
void spdmapplib::SPDMResponderImpl::processConnectionState (
            void * spdmContext,
            libspdm_connection_state_t connectionState )
```

Register to libspdm for handling connection state change.

Parameters

spdmContext	The pointer of the spdmcontext.
connectionState	The connection state.

6.6.2.8 processSessionState()

```
void spdmapplib::SPDMResponderImpl::processSessionState (
            void * spdmContext,
            uint32_t sessionID,
            libspdm_session_state_t sessionState )
```

Register to libspdm for handling session state change.

Parameters

spdmContext	The pointer of the spdmcontext.
sessionID	The session ID.
sessionState	The session state.

6.6.2.9 processSPDMMessage()

```
int spdmapplib::SPDMResponderImpl::processSPDMMessage (
            spdmItem & transEP )
```

Called when message received.

The function is called in msgRecvCallback to process incoming received data.

Parameters

transEP	The endpoint object sending data.

Returns

0: success, other: failed.

6.6.2.10 removeDevice()

```
int spdmapplib::SPDMResponderImpl::removeDevice (
            spdmtransport::TransportEndPoint & transEP )
```

Called when endpoint remove is detected.

Parameters

transEP	The endpoint to be removed.

Returns

0: success, other: failed.

6.6.2.11 settingFromConfig()

```
int spdmapplib::SPDMResponderImpl::settingFromConfig (
            uint8_t ItemIndex )  [protected]
```

Function to setup specific endpoint initial configuration.

Parameters

ItemIndex	The endpoint index.

Returns

0: success, other: failed.

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

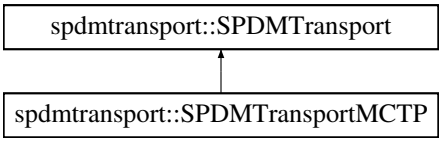
- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmapplib_impl.hpp

6.7 spdmtransport::SPDMTransport Class Reference

SPDM transport layer class.

`#include <spdmtransport.hpp>`

Inheritance diagram for spdmtransport::SPDMTransport:



Public Member Functions

- virtual int initTransport (std::shared_ptr< boost::asio::io_context > ioc, std::shared_ptr< sdbusplus::asio↩ ::connection > conn, AddRemoveDeviceCallback addCB, AddRemoveDeviceCallback delCB, MsgReceive↩ Callback msgRcvCB=nullptr)=0

  Initial function of transport instance.
- virtual TransportIdentifier getTransType (void)=0

  Get the interface type of transport layer.
- virtual int asyncSendData (TransportEndPoint &transEP, const std::vector< uint8_t > &request, uint64_↩ t timeout)=0

  The async send data function for responder nonblocking function to send message to remote endpoint.
- virtual int sendRecvData (TransportEndPoint &transEP, const std::vector< uint8_t > &request, uint64_↩ t timeout, MsgReceiveCallback rspRcvCB)=0

  The sync send and receive data function for requester blocking function to send SPDM payload and get response data.

6.7.1 Detailed Description

SPDM transport layer class.

6.7.2 Member Function Documentation

6.7.2.1 asyncSendData()

```
virtual int spdmtransport::SPDMTransport::asyncSendData (
            TransportEndPoint & transEP,
            const std::vector< uint8_t > & request,
            uint64_t timeout )  [pure virtual]
```

The async send data function for responder nonblocking function to send message to remote endpoint.

Parameters

transEP	The destination endpoint.
request	The vector of payload.
timeout	The timeout time.

Returns

0: success, other: failed.

Implemented in spdmtransport::SPDMTransportMCTP.

6.7.2.2 getTransType()

```
virtual TransportIdentifier spdmtransport::SPDMTransport::getTransType (
            void  )  [pure virtual]
```

Get the interface type of transport layer.

Returns

TransportIdentifier

Implemented in spdmtransport::SPDMTransportMCTP.

6.7.2.3 initTransport()

```
virtual int spdmtransport::SPDMTransport::initTransport (
            std::shared_ptr< boost::asio::io_context > ioc,
            std::shared_ptr< sdbusplus::asio::connection > conn,
            AddRemoveDeviceCallback addCB,
            AddRemoveDeviceCallback delCB,
            MsgReceiveCallback msgRcvCB = nullptr )  [pure virtual]
```

Initial function of transport instance.

Parameters

ioc	shared_ptr to boost io_context object.
conn	shared_ptr to already existing boost asio::connection.
addCB	The callback function for new endpoint detected.
delCB	The callback function for EndPoint removed.
msgRcvCB	The callback function for messages received(for responder used).

Returns

0: success, other: failed.

Implemented in spdmtransport::SPDMTransportMCTP.

6.7.2.4 sendRecvData()

```
virtual int spdmtransport::SPDMTransport::sendRecvData (
            TransportEndPoint & transEP,
            const std::vector< uint8_t > & request,
            uint64_t timeout,
            MsgReceiveCallback rspRcvCB )  [pure virtual]
```

The sync send and receive data function for requester blocking function to send SPDM payload and get response data.

Parameters

transEP	The destination endpoint.
request	The vector of data payload.
timeout	The timeout time.
rspRcvCB	The resRcvCB to be called when response data received.

Returns

0: success, other: failed.

Implemented in spdmtransport::SPDMTransportMCTP.

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

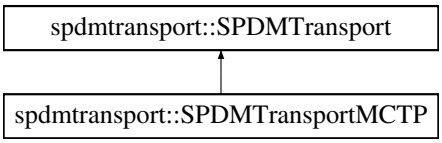
- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmtransport.hpp

6.8 spdmtransport::SPDMTransportMCTP Class Reference

SPDM transport layer implemented using MCTP.

`#include <spdmtransport_mctp.hpp>`

Inheritance diagram for spdmtransport::SPDMTransportMCTP:



Public Member Functions

- SPDMTransportMCTP (TransportIdentifier id)

  SPDMTransportMCTP constructor.
- int initTransport (std::shared_ptr< boost::asio::io_context > ioc, std::shared_ptr< sdbusplus::asio::connection > conn, AddRemoveDeviceCallback addCB, AddRemoveDeviceCallback delCB, MsgReceiveCallback msgRcvCB=nullptr) override

  Initial function of transport instance.
- TransportIdentifier getTransType (void) override

  Get the interface type of transport layer.
- int asyncSendData (TransportEndPoint &transEP, const std::vector< uint8_t > &request, uint64_t timeout) override

  The async send data function for responder nonblocking function to send message to remote endpoint.
- int sendRecvData (TransportEndPoint &transEP, const std::vector< uint8_t > &request, uint64_t timeout, MsgReceiveCallback rspRcvCB) override

  The sync send and receive data function for requester blocking function to send SPDM payload and get response data.

Protected Attributes

- TransportIdentifier **transType**
- std::shared_ptr< boost::asio::io_context > **pioc**
- std::shared_ptr< sdbusplus::asio::connection > **pconn**
- std::shared_ptr< mctpw::MCTPWrapper > **mctpWrapper**

6.8.1 Detailed Description

SPDM transport layer implemented using MCTP.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 SPDMTransportMCTP()

```
spdmtransport::SPDMTransportMCTP::SPDMTransportMCTP (
            TransportIdentifier id )  [inline]
```

SPDMTransportMCTP constructor.

Parameters

id	Transport layer interface id(here only for MCTPoverPCIe or MCTPoverSMBus).

6.8.3 Member Function Documentation

6.8.3.1 asyncSendData()

```
int spdmtransport::SPDMTransportMCTP::asyncSendData (
            TransportEndPoint & transEP,
            const std::vector< uint8_t > & request,
            uint64_t timeout )  [override], [virtual]
```

The async send data function for responder nonblocking function to send message to remote endpoint.

Parameters

transEP	The destination endpoint.
request	The buffer vector of data.
timeout	The timeout time.

Returns

0: success, other: failed.

Implements spdmtransport::SPDMTransport.

6.8.3.2 getTransType()

```
TransportIdentifier spdmtransport::SPDMTransportMCTP::getTransType (
            void  )  [inline], [override], [virtual]
```

Get the interface type of transport layer.

Returns

TransportIdentifier

Implements spdmtransport::SPDMTransport.

6.8.3.3 initTransport()

```
int spdmtransport::SPDMTransportMCTP::initTransport (
            std::shared_ptr< boost::asio::io_context > ioc,
            std::shared_ptr< sdbusplus::asio::connection > conn,
            AddRemoveDeviceCallback addCB,
            AddRemoveDeviceCallback delCB,
            MsgReceiveCallback msgRcvCB = nullptr )  [override], [virtual]
```

Initial function of transport instance.

Parameters

ioc	shared_ptr to boost io_context object.
conn	shared_ptr to already existing boost asio::connection.
addCB	The callback function for new endpoint detected.
delCB	The callback function for EndPoint removed.
msgRcvCB	The callback function for messages received(for responder used).

Returns

0: success, other: failed.

Implements spdmtransport::SPDMTransport.

6.8.3.4 sendRecvData()

```
int spdmtransport::SPDMTransportMCTP::sendRecvData (
            TransportEndPoint & transEP,
            const std::vector< uint8_t > & request,
            uint64_t timeout,
            MsgReceiveCallback rspRcvCB )  [override], [virtual]
```

The sync send and receive data function for requester blocking function to send SPDM payload and get response data.

Parameters

transEP	The destination endpoint.
request	The vector of data payload.
timeout	The timeout time.
rspRcvCB	The resRcvCB to be called when response data received.

Returns

0: success, other: failed.

Implements spdmtransport::SPDMTransport.

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

- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmtransport_mctp.hpp

6.9 spdmtransport::TransportEndPoint Struct Reference

Endpoint information, could be extended.

`#include <spdmtransport.hpp>`

Public Member Functions

- bool **operator==** (const TransportEndPoint &p2) const

Public Attributes

- TransportIdentifier **transType**
- uint8_t **devIdentifier**

6.9.1 Detailed Description

Endpoint information, could be extended.

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

- H:/bhs/openbmc-spdm/build/workspace/sources/spdmapplib/include/spdmtransport.hpp

Chapter 7

File Documentation

7.1 spdmapplib.hpp

```
1
17 #pragma once
18 #include "spdmapplib_errorcodes.hpp"
19 #include "spdmtransport.hpp"
20
21 namespace spdmapplib
22 {
27 struct SPDMConfiguration
28 {
29     uint32_t version;
30     /* library can support requester and responder roles */
31     uint32_t capability;
32     uint32_t hash;
33     uint32_t measHash;
34     uint32_t asym;
35     uint32_t reqasym;
36     uint32_t dhe;
37     uint32_t aead;
38     uint32_t slotcount;
39     std::string certPath;
40 };
41
46 class SPDMResponder
47 {
48   public:
49     virtual ~SPDMResponder() = default;
50     /*APIs called by SPDM responder daemon*/
60     virtual int
61         initResponder(std::shared_ptr<boost::asio::io_context> ioc,
62                       std::shared_ptr<sdbusplus::asio::connection> conn,
63                       std::shared_ptr<spdmtransport::SPDMTransport> trans,
64                       SPDMConfiguration& spdmConfig) = 0;
65 };
66
71 class SPDMRequester
72 {
73   public:
74     virtual ~SPDMRequester() = default;
75     /*Requester APIs*/
85     virtual int
86         initRequester(std::shared_ptr<boost::asio::io_context> ioc,
87                       std::shared_ptr<sdbusplus::asio::connection> conn,
88                       std::shared_ptr<spdmtransport::SPDMTransport> trans,
89                       spdmtransport::TransportEndPoint& transResponder,
90                       SPDMConfiguration& pSpdmConfig) = 0;
96     virtual int doAuthentication(void) = 0;
105     virtual int doMeasurement(const uint32_t* sessionid) = 0;
111     virtual std::optional<std::vector<uint8_t» getMeasurements() = 0;
117     virtual std::optional<std::vector<uint8_t» getCertificate() = 0;
118 };
119
126 std::shared_ptr<SPDMRequester> createRequester();
127
134 std::shared_ptr<SPDMResponder> createResponder();
135
136 } // namespace spdmapplib
```

7.2 spdmapplib_errorcodes.hpp

```
1
16 #pragma once
17 namespace spdmapplib
18 {
24 namespace errorcodes
25 {
26 inline constexpr int generalReturnError = -1;
27 inline constexpr int returnSuccess = 0;
28 inline constexpr int spdmConfigurationNotFoundInEntityManager = 1;
29 inline constexpr int libspdmReturnError =
30     2; // libspdm function calls return error.
31 } // namespace errorcodes
32 } // namespace spdmapplib
```

7.3 spdmapplib_impl.hpp

```
1
16 #pragma once
17 #include "spdmapplib.hpp"
18 // clang-format off
19 extern "C"
20 {
21 #include "library/spdm_common_lib.h"
22 #include "library/spdm_requester_lib.h"
23 #include "library/spdm_responder_lib.h"
24 #include "spdm_device_secret_lib_internal.h"
25 #include "library/malloclib.h"
26 }
27 // clang-format on
28
29 inline constexpr uint32_t exeConnectionVersionOnly = 0x1;
30 inline constexpr uint32_t exeConnectionDigest = 0x2;
31 inline constexpr uint32_t exeConnectionCert = 0x4;
32 inline constexpr uint32_t exeConnectionChal = 0x8;
33 inline constexpr uint32_t exeConnectionMeas = 0x10;
34
35 namespace spdmapplib
36 {
42 enum class SPDMVersions : uint32_t
43 {
44     spdmv1p1 = 0x01
45 };
46
47 enum class SPDMDeviceEvent : uint8_t
48 {
49     deviceAdded = 0x01,
50     deviceRemoved
51 };
52
57 typedef struct
58 {
59     void* pspdmContext;
60     spdmtransport::TransportEndPoint transEP;
61     uint8_t useSlotId;
62     uint32_t sessionId;
63     uint32_t useVersion;
64     uint16_t useReqAsymAlgo;
65     uint32_t useMeasurementHashAlgo;
66     uint32_t useAsymAlgo;
67     uint32_t useHashAlgo;
68     libspdm_connection_state_t connectStatus;
69     std::vector<uint8_t> data;
70     std::vector<uint8_t> dataCert;
71     std::vector<uint8_t> dataMeas;
72 } spdmItem;
73
78 class SPDMResponderImpl : public SPDMResponder
79 {
80   public:
81     /*APIs called by SPDM daemon*/
82     SPDMResponderImpl() = default;
83     virtual ~SPDMResponderImpl();
95     int initResponder(std::shared_ptr<boost::asio::io_context> ioc,
96                       std::shared_ptr<sdbusplus::asio::connection> conn,
97                       std::shared_ptr<spdmtransport::SPDMTransport> trans,
98                       SPDMConfiguration& spdmConfig) override;
99
100     /*APIs called by transport layer*/
108     spdmItem& createSPDMItem(spdmtransport::TransportEndPoint& transEP);
109
118     spdmItem& findSPDMItem(spdmtransport::TransportEndPoint& transEP);
```

```
119
127     int removeDevice(spdmtransport::TransportEndPoint& transEP);
128
138     int processSPDMMessage(spdmItem& transEP);
139
148     int msgRecvCallback(spdmtransport::TransportEndPoint& transEP,
149                         const std::vector<uint8_t>& data);
150
151     /*Cabllback functions implementation for libspdm */
161     return_status deviceSendMessage(void* spdmContext,
162                                     const std::vector<uint8_t>& request,
163                                     uint64_t timeout);
164
174     return_status deviceReceiveMessage(void* spdmContext,
175                                        std::vector<uint8_t>& response,
176                                        uint64_t timeout);
177
185     void processConnectionState(void* spdmContext,
186                                 libspdm_connection_state_t connectionState);
187
196     void processSessionState(void* spdmContext, uint32_t sessionID,
197                              libspdm_session_state_t sessionState);
198     /*Internal implementation*/
199   protected:
207     int settingFromConfig(uint8_t ItemIndex);
208
209   private:
210     std::shared_ptr<boost::asio::io_context> pioc;
211
212     uint8_t useSlotCount;
213     uint8_t curIndex;
214     uint32_t useResponderCapabilityFlags;
215     uint8_t useMutAuth;
216     uint8_t useBasicMutAuth;
217     SPDMConfiguration spdmResponderCfg;
218     std::vector<spdmItem> spdmPool;
219     std::shared_ptr<spdmtransport::SPDMTransport> spdmTrans;
220 };
221
226 class SPDMRequesterImpl : public SPDMRequester
227 {
228   public:
229     SPDMRequesterImpl() = default;
230     virtual ~SPDMRequesterImpl();
231     /* APIs for requester*/
240     int initRequester(std::shared_ptr<boost::asio::io_context> ioc,
241                       std::shared_ptr<sdbusplus::asio::connection> conn,
242                       std::shared_ptr<spdmtransport::SPDMTransport> trans,
243                       spdmtransport::TransportEndPoint& transResponder,
244                       SPDMConfiguration& spdmConfig) override;
250     int doAuthentication(void) override;
259     int doMeasurement(const uint32_t* sessionid) override;
265     std::optional<std::vector<uint8_t» getMeasurements() override;
271     std::optional<std::vector<uint8_t» getCertificate() override;
272
273     /*APIs called by transport layer*/
282     int addData(spdmtransport::TransportEndPoint& transEP,
283                 const std::vector<uint8_t>& data);
284
293     int checkResponderDevice(spdmtransport::TransportEndPoint& transEP);
294
307     int msgRecvCallback(spdmtransport::TransportEndPoint& transEP,
308                         const std::vector<uint8_t>& data);
309
310     /*Callback functions implementation for libspdm*/
320     return_status deviceSendMessage(void* spdmContext,
321                                     const std::vector<uint8_t>& request,
322                                     uint64_t timeout);
323
333     return_status deviceReceiveMessage(void* spdmContext,
334                                        std::vector<uint8_t>& response,
335                                        uint64_t timeout);
336
337     /*Internal implementation*/
338   protected:
347     int setupResponder(const spdmtransport::TransportEndPoint& transEP);
354     int settingFromConfig(void);
355
356   private:
357     bool bResponderFound;
358     std::shared_ptr<boost::asio::io_context> pioc;
359
360     uint8_t useSlotCount;
361     uint8_t useSlotId;
362     uint32_t useRequesterCapabilityFlags;
363     uint8_t useMutAuth;
364     uint8_t useBasicMutAuth;
```

```
365     uint16_t mUseReqAsymAlgo;
366     uint32_t mUseAsymAlgo;
367     uint32_t mUseHashAlgo;
368     uint32_t mExeConnection;
369     uint8_t mUseMeasurementSummaryHashType;
370     uint8_t mUseMeasurementOperation;
371     uint8_t mUseMeasurementAttribute;
372     SPDMConfiguration spdmRequesterCfg;
373     spdmItem spdmResponder; // only one instance for requester.
374     spdmtransport::TransportEndPoint transResponder;
375     std::shared_ptr<spdmtransport::SPDMTransport> spdmTrans;
376 };
377
378 /*Utility function*/
384 void setCertificatePath(std::string& certPath);
385
386 } // namespace spdmapplib
```

7.4 spdmtransport.hpp

```
1
17 #pragma once
18 #include <boost/asio.hpp>
19 #include <sdbusplus/asio/connection.hpp>
20 namespace spdmtransport
21 {
28 struct TransportEndPoint;
29 using MsgReceiveCallback = std::function<void(
30     TransportEndPoint& transEP, const std::vector<uint8_t>& data)>;
31 using AddRemoveDeviceCallback = std::function<int(TransportEndPoint& transEP)>;
32
37 enum class TransportIdentifier : uint8_t
38 {
39     mctpOverSMBus = 0x01,
40     mctpOverPCIe = 0x02,
41     pmtWatcher = 0x03, /*Intel specific transport*/
42 };
43
48 struct TransportEndPoint
49 {
50     TransportIdentifier transType; /*interface type.*/
51     uint8_t devIdentifier;
52     bool operator==(const TransportEndPoint& p2) const
53     {
54         const TransportEndPoint& p1 = (*this);
55         return p1.transType == p2.transType &&
56                p1.devIdentifier == p2.devIdentifier;
57     }
58 };
59
65 class SPDMTransport
66 {
67   public:
68     virtual ~SPDMTransport() = default;
69
70     /* APIs for requester and responder */
84     virtual int initTransport(
85         std::shared_ptr<boost::asio::io_context> ioc,
86         std::shared_ptr<sdbusplus::asio::connection> conn,
87         AddRemoveDeviceCallback addCB, AddRemoveDeviceCallback delCB,
88         MsgReceiveCallback msgRcvCB =
89             nullptr) = 0; // override this function in implementation
90
96     virtual TransportIdentifier getTransType(void) = 0;
97
98     /*************************************************
99         APIs to responder and interface that implementation should override
100     these pure virtual functions
101     *************************************************/
112     virtual int asyncSendData(TransportEndPoint& transEP,
113                               const std::vector<uint8_t>& request,
114                               uint64_t timeout) = 0;
115
116     /*************************************************
117         APIs for requester
118     *************************************************/
131     virtual int sendRecvData(TransportEndPoint& transEP,
132                              const std::vector<uint8_t>& request,
133                              uint64_t timeout, MsgReceiveCallback rspRcvCB) = 0;
134 };
135
136 } // namespace spdmtransport
```

7.5 spdmtransport_mctp.hpp

```
1
17 #pragma once
18 #include "mctp_wrapper.hpp"
19 #include "spdmapplib_errorcodes.hpp"
20 #include "spdmtransport.hpp"
21
22 namespace spdmtransport
23 {
29 class SPDMTransportMCTP : public SPDMTransport
30 {
31   public:
32     /*APIs called by spdmAppLib layer*/
40     SPDMTransportMCTP(TransportIdentifier id)
41     {
42         transType = id;
43     };
44
58     int initTransport(std::shared_ptr<boost::asio::io_context> ioc,
59                       std::shared_ptr<sdbusplus::asio::connection> conn,
60                       AddRemoveDeviceCallback addCB,
61                       AddRemoveDeviceCallback delCB,
62                       MsgReceiveCallback msgRcvCB = nullptr) override;
63
69     TransportIdentifier getTransType(void) override
70     {
71         return transType;
72     };
73
84     int asyncSendData(TransportEndPoint& transEP,
85                       const std::vector<uint8_t>& request,
86                       uint64_t timeout) override;
87
99     int sendRecvData(TransportEndPoint& transEP,
100                      const std::vector<uint8_t>& request, uint64_t timeout,
101                      MsgReceiveCallback rspRcvCB) override;
102
103    /*APIs called by mctpwrapper callback function*/
104  private:
110    int transAddNewDevice(const mctpw::eid_t eid);
116    int transRemoveDevice(const mctpw::eid_t eid);
117
122    void transMsgRecvCallback(void*, mctpw::eid_t srcEid, bool tagOwner,
123                              uint8_t msgTag, const std::vector<uint8_t>& data,
124                              int);
125
130    void transOnDeviceUpdate(void*, const mctpw::Event& evt,
131                             boost::asio::yield_context yield);
132
133    /* Callback function pointers */
134    AddRemoveDeviceCallback addNewDeviceCB = nullptr;
135    AddRemoveDeviceCallback removeDeviceCB = nullptr;
136    MsgReceiveCallback msgReceiveCB = nullptr;
137
138  protected:
139    TransportIdentifier transType; /*MCTP over PCIe, MCTP over SMBus, SDSi*/
140    std::shared_ptr<boost::asio::io_context> pioc;
141    std::shared_ptr<sdbusplus::asio::connection> pconn;
142    std::shared_ptr<mctpw::MCTPWrapper> mctpWrapper;
143 };
144 } // namespace spdmtransport
```

Index