ADOM - Application-centric Data Object Management

Generated by Doxygen 1.9.1

Chapter 1

Todo List

Class DataObjectInstance

o Umbenennen, eher DataObjectIdentifier

Member Directory::other_directory_descriptor_

make private

Member Directory::sendDataRequest (EntityDescriptor home_directory, std::shared_ptr$<$ ObjectSample $>$, std::vector$<$ uint16_t $>$ block_ids)

make private

make private

Member translate_from_uchar (unsigned char $*$data, Structure structure)

o check for allignement issues, is the new data buffer big enough etc

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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 files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 adom Namespace Reference

Functions

- void init_file_log (std::string log_prefix, std::string log_suffix)
- void init_console_log ()

5.1.1 Function Documentation

5.1.1.1 init_console_log()

```
void adom::init_console_log ( )
```

5.1.1.2 init_file_log()

```
void adom::init_file_log (
             std::string log_prefix,
             std::string log_suffix )
```

Chapter 6

Class Documentation

6.1 AdomMessage Class Reference

Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports.

```
#include <protocol.hpp>
```

Public Member Functions

- AdomMessage ()
- AdomMessage (MessageType type, uint16_t payload_length, char ∗msg)
- void adomMessageToNet (char ∗msg)

  This function is called to serialize Application Data Object Management Messages for transport.
- void netToAdomMessage (char ∗msg)

  This function is called to deserialize received Application Data Object Management Messages.
- void clear ()

  This function is called to clear all attributes of the Application Data Object Management Messages for reuse.

Public Attributes

- enum MessageType type_
- uint16_t payload_length_
- std::vector< char > payload_
- uint16_t adom_message_length_

6.1.1 Detailed Description

Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AdomMessage() [1/2]

```
AdomMessage::AdomMessage ( )  [inline]
```

6.1.2.2 AdomMessage() [2/2]

```
AdomMessage::AdomMessage (
             MessageType type,
             uint16_t payload_length,
             char * msg )  [inline]
```

6.1.3 Member Function Documentation

6.1.3.1 adomMessageToNet()

```
void AdomMessage::adomMessageToNet (
             char * msg )
```

This function is called to serialize Application Data Object Management Messages for transport.

Parameters

msg	Pointer to buffer to write the serialized message into

6.1.3.2 clear()

```
void AdomMessage::clear ( )
```

This function is called to clear all attributes of the Application Data Object Management Messages for reuse.

This function is called to deserialize received Application Data Object Management Messages.

Parameters

msg	Pointer to buffer to the serialized message

6.1.3.3 netToAdomMessage()

```
void AdomMessage::netToAdomMessage (
            char * msg )
```

This function is called to deserialize received Application Data Object Management Messages.

Parameters

msg	Pointer to buffer to the serialized message

6.1.4 Member Data Documentation

6.1.4.1 adom_message_length_

```
uint16_t AdomMessage::adom_message_length_
```

6.1.4.2 payload_

```
std::vector<char> AdomMessage::payload_
```

6.1.4.3 payload_length_

```
uint16_t AdomMessage::payload_length_
```

6.1.4.4 type_

```
enum MessageType AdomMessage::type_
```

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

- include/adom/protocol.hpp
- src/cpp/adom/protocol.cpp

6.2 ApplicationInterface Class Reference

```
#include <application_interface.hpp>
```

Public Member Functions

- ApplicationInterface (EntityDescriptor home_directory_descriptor)

  Construct a new Application Interface object.
- void initialize (void)

  Read static definition of other directories and topic structures etc.
- void registerNewTopic (Topic type, Structure structure)

  Send a Request to the associated Directory to: Write a new (complete) sample to the managed shared memory (called from a "Publisher" or "Writer")
- void registerNewData (Topic type, unsigned char ∗data)

  Send a Request to the associated Directory to: Write a new (complete) sample (called from a "Publisher" or "Writer")
- std::future< int > readPartialData (Topic type, std::vector< uint16_t > associated_blocks, unsigned char ∗output_data)

  Send a Request to the associated Directory to: Partially read data from a sample (called from a "Subscriber" or "Reader")
- std::future< int > readPartialData (Topic type, int sequence_nr, std::vector< uint16_t > associated_blocks, unsigned char ∗output_data)

  Send a Request to the associated Directory to: Partially read data from a sample (called from a "Subscriber" or "Reader")
- void stop ()

  Stopping all processes within the interface.

6.2.1 Constructor & Destructor Documentation

6.2.1.1 ApplicationInterface()

```
ApplicationInterface::ApplicationInterface (
            EntityDescriptor home_directory_descriptor )  [inline]
```

Construct a new Application Interface object.

Parameters

associated_directory_descriptor	

6.2.2 Member Function Documentation

6.2.2.1 initialize()

```
void ApplicationInterface::initialize (
            void  )
```

Read static definition of other directories and topic structures etc.

Temporary static definition of other directory

6.2.2.2 readPartialData() [1/2]

```
std::future< int > ApplicationInterface::readPartialData (
            Topic type,
            int sequence_nr,
            std::vector< uint16_t > associated_blocks,
            unsigned char * output_data )
```

Send a Request to the associated Directory to: Partially read data from a sample (called from a "Subscriber" or "Reader")

Parameters

topic_name	name of the topic, whose object sample is read partially
sequence_nr	sequence number of the sample
associated_blocks	associated_blocks for this read access
output_data	data to write requested block into

6.2.2.3 readPartialData() [2/2]

```
std::future< int > ApplicationInterface::readPartialData (
            Topic type,
            std::vector< uint16_t > associated_blocks,
            unsigned char * output_data )
```

Send a Request to the associated Directory to: Partially read data from a sample (called from a "Subscriber" or "Reader")

Parameters

type	
associated_blocks	
output_data	

Returns

std::future<int>

Parameters

topic_name	name of the topic, whose object sample is read partially
associated_blocks	associated_blocks for this read access
output_data	data to write requested block into

6.2.2.4 registerNewData()

```
void ApplicationInterface::registerNewData (
            Topic type,
            unsigned char * data )
```

Send a Request to the associated Directory to: Write a new (complete) sample (called from a "Publisher" or "Writer")

Send a Request to the associated Directory to: Write a new (complete) sample to the managed shared memory (called from a "Publisher" or "Writer")

Parameters

type	name of the topic, whose object sample is written
data	data to be registered

6.2.2.5 registerNewTopic()

```
void ApplicationInterface::registerNewTopic (
            Topic type,
            Structure data_structure )
```

Send a Request to the associated Directory to: Write a new (complete) sample to the managed shared memory (called from a "Publisher" or "Writer")

Parameters

type	name of the topic, whose object sample is written
structure	structure of the data type required for parsing and specification of blocks
type	name of the topic, whose object sample is written
structure	structure of the data type required for parsing specification of blocks

6.2.2.6 stop()

```
void ApplicationInterface::stop ( )
```

Stopping all processes within the interface.

Stopping all processes within directory.

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

- include/adom/application_interface.hpp
- src/cpp/adom/application_interface.cpp

6.3 AppRequest Struct Reference

```
#include <protocol.hpp>
```

Public Attributes

- enum Topic topic
- uint16_t sequence_nr
- void ∗ object_address
- void ∗ access_start_address
- void ∗ access_end_address

6.3.1 Member Data Documentation

6.3.1.1 access_end_address

```
void* AppRequest::access_end_address
```

6.3.1.2 access_start_address

```
void* AppRequest::access_start_address
```

6.3.1.3 object_address

```
void* AppRequest::object_address
```

6.3.1.4 sequence_nr

```
uint16_t AppRequest::sequence_nr
```

6.3.1.5 topic

```
enum Topic AppRequest::topic
```

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

- include/adom/protocol.hpp

6.4 DataAnnouncement Class Reference

DataAnnouncement is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample.

```
#include <protocol.hpp>
```

Public Member Functions

- DataAnnouncement ()
- DataAnnouncement (EntityDescriptor home_directory, Topic object_type, uint16_t object_sequence_nr)
- DataAnnouncement (EntityDescriptor home_directory, Topic object_type, uint16_t object_sequence_nr, Structure structure)
- void dataAnnouncementToNet (char ∗msg)

  This function is called to serialize DataAnnouncement for transport.
- void netToDataAnnouncement (char ∗msg)

  This function is called to deserialize a received DataAnnouncement.
- void clear ()

  This function is called to clear all attributes of the DataAnnouncement for reuse.
- void print ()

  This function prints out the DataAnnouncement.

Public Attributes

- EntityDescriptor home_directory_
- DataObjectInstance associated_object_
- Structure structure_
- uint16_t message_length_

6.4.1 Detailed Description

DataAnnouncement is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 DataAnnouncement() [1/3]

```
DataAnnouncement::DataAnnouncement ( )  [inline]
```

6.4.2.2 DataAnnouncement() [2/3]

```
DataAnnouncement::DataAnnouncement (
            EntityDescriptor home_directory,
            Topic object_type,
            uint16_t object_sequence_nr )  [inline]
```

6.4.2.3 DataAnnouncement() [3/3]

```
DataAnnouncement::DataAnnouncement (
            EntityDescriptor home_directory,
            Topic object_type,
            uint16_t object_sequence_nr,
            Structure structure )  [inline]
```

6.4.3 Member Function Documentation

6.4.3.1 clear()

```
void DataAnnouncement::clear ( )
```

This function is called to clear all attributes of the DataAnnouncement for reuse.

6.4.3.2 dataAnnouncementToNet()

```
void DataAnnouncement::dataAnnouncementToNet (
            char * msg )
```

This function is called to serialize DataAnnouncement for transport.

Parameters

msg	Pointer to buffer to write the serialized DataAnnouncement into

6.4.3.3 netToDataAnnouncement()

```
void DataAnnouncement::netToDataAnnouncement (
            char * msg )
```

This function is called to deserialize a received DataAnnouncement.

Parameters

msg	Pointer to buffer to the serialized DataAnnouncement

6.4.3.4 print()

```
void DataAnnouncement::print ( )
```

This function prints out the DataAnnouncement.

6.4.4 Member Data Documentation

6.4.4.1 associated_object_

```
DataObjectInstance DataAnnouncement::associated_object_
```

6.4.4.2 home_directory_

```
EntityDescriptor DataAnnouncement::home_directory_
```

6.4.4.3 message_length_

```
uint16_t DataAnnouncement::message_length_
```

6.4.4.4 structure_

```
Structure DataAnnouncement::structure_
```

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

- include/adom/protocol.hpp
- src/cpp/adom/protocol.cpp

6.5 DataBlockHeader Struct Reference

Struct to describe a Data Block, that is part of an Object instance.

```
#include <protocol.hpp>
```

Public Member Functions

- DataBlockHeader ()
- DataBlockHeader (Topic topic, uint16_t updated_sequence_nr, uint16_t block_id, uint16_t block_size, long data_address_offset)

Public Attributes

- DataObjectInstance associated_object
- uint16_t block_id
- uint16_t block_size
- long data_address_offset

6.5.1 Detailed Description

Struct to describe a Data Block, that is part of an Object instance.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 DataBlockHeader() [1/2]

```
DataBlockHeader::DataBlockHeader ( )  [inline]
```

6.5.2.2 DataBlockHeader() [2/2]

```
DataBlockHeader::DataBlockHeader (
            Topic topic,
            uint16_t updated_sequence_nr,
            uint16_t block_id,
            uint16_t block_size,
            long data_address_offset )  [inline]
```

6.5.3 Member Data Documentation

6.5.3.1 associated_object

```
DataObjectInstance DataBlockHeader::associated_object
```

6.5.3.2 block_id

```
uint16_t DataBlockHeader::block_id
```

6.5.3.3 block_size

```
uint16_t DataBlockHeader::block_size
```

6.5.3.4 data_address_offset

```
long DataBlockHeader::data_address_offset
```

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

- include/adom/protocol.hpp

6.6 DataObjectInstance Struct Reference

Struct to describe an Object instance with its sequence number within a topic or object type.

```
#include <protocol.hpp>
```

Public Member Functions

- DataObjectInstance ()

Public Attributes

- enum Topic object_type
- uint16_t object_sequence_nr

6.6.1 Detailed Description

Struct to describe an Object instance with its sequence number within a topic or object type.

Todo o Umbenennen, eher DataObjectIdentifier

6.6.2 Constructor & Destructor Documentation

6.6.2.1 DataObjectInstance()

```
DataObjectInstance::DataObjectInstance ( )  [inline]
```

6.6.3 Member Data Documentation

6.6.3.1 object_sequence_nr

```
uint16_t DataObjectInstance::object_sequence_nr
```

6.6.3.2 object_type

```
enum Topic DataObjectInstance::object_type
```

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

- include/adom/protocol.hpp

6.7 DataRequest Class Reference

DataRequest is used by a remote (subscribed) directory (reader_application_) to request specific blocks of a sample object managed by the data's (publihser/) home directory (home_directory_) via a block_validity_matrix.

```
#include <protocol.hpp>
```

Public Member Functions

- DataRequest ()
- DataRequest (EntityDescriptor reader_application, EntityDescriptor home_directory, DataObjectInstance object_instance, uint16_t object_block_count, int ∗object_block_validity, int request_id)
- void dataRequestToNet (char ∗msg)

  This function is called to serialize DataRequest for transport.
- void netToDataRequest (char ∗msg)

  This function is called to deserialize received DataRequest.
- void clear ()

  This function is called to clear all attributes of the DataRequest for reuse.
- void print ()

  This function prints out the DataRequest.

Public Attributes

- uint16_t request_id_
- EntityDescriptor reader_application_
- EntityDescriptor home_directory_
- DataObjectInstance object_instance_
- uint16_t object_block_count_
- std::vector< uint8_t > object_block_validity_
- uint16_t message_length_

6.7.1 Detailed Description

DataRequest is used by a remote (subscribed) directory (reader_application_) to request specific blocks of a sample object managed by the data's (publihser/) home directory (home_directory_) via a block_validity_matrix.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 DataRequest() [1/2]

```
DataRequest::DataRequest ( )  [inline]
```

6.7.2.2 DataRequest() [2/2]

```
DataRequest::DataRequest (
             EntityDescriptor reader_application,
             EntityDescriptor home_directory,
             DataObjectInstance object_instance,
             uint16_t object_block_count,
             int * object_block_validity,
             int request_id )  [inline]
```

6.7.3 Member Function Documentation

6.7.3.1 clear()

```
void DataRequest::clear ( )
```

This function is called to clear all attributes of the DataRequest for reuse.

6.7.3.2 dataRequestToNet()

```
void DataRequest::dataRequestToNet (
             char * msg )
```

This function is called to serialize DataRequest for transport.

Parameters

msg	Pointer to buffer to write the serialized request into

6.7.3.3 netToDataRequest()

```
void DataRequest::netToDataRequest (
             char * msg )
```

This function is called to deserialize received DataRequest.

Parameters

msg	Pointer to buffer to the serialized request

6.7.3.4 print()

```
void DataRequest::print ( )
```

This function prints out the DataRequest.

6.7.4 Member Data Documentation

6.7.4.1 home_directory_

```
EntityDescriptor DataRequest::home_directory_
```

6.7.4.2 message_length_

```
uint16_t DataRequest::message_length_
```

6.7.4.3 object_block_count_

```
uint16_t DataRequest::object_block_count_
```

6.7.4.4 object_block_validity_

```
std::vector<uint8_t> DataRequest::object_block_validity_
```

6.7.4.5 object_instance_

```
DataObjectInstance DataRequest::object_instance_
```

6.7.4.6 reader_application_

```
EntityDescriptor DataRequest::reader_application_
```

6.7.4.7 request_id_

```
uint16_t DataRequest::request_id_
```

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

- include/adom/protocol.hpp
- src/cpp/adom/protocol.cpp

6.8 DataTransport Class Reference

DataTransport is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory.

```
#include <protocol.hpp>
```

Public Member Functions

- DataTransport ()
- DataTransport (DataBlockHeader block_header, unsigned char ∗data, uint16_t associated_request_id)
- void dataTransportToNet (char ∗msg)

  This function is called to serialize DataTransport for transport.
- void netToDataTransport (char ∗msg)

  This function is called to deserialize received DataTransport.
- void clear ()

  This function is called to clear all attributes of the DataTransport for reuse.
- void print ()

  This function prints out the DataTransport.

Public Attributes

- uint16_t associated_request_id_
- DataObjectInstance associated_object_
- uint16_t block_id_
- uint16_t block_size_
- std::vector< char > data_
- uint16_t message_length_

6.8.1 Detailed Description

DataTransport is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 DataTransport() [1/2]

```
DataTransport::DataTransport ( )  [inline]
```

6.8.2.2 DataTransport() [2/2]

```
DataTransport::DataTransport (
            DataBlockHeader block_header,
            unsigned char * data,
            uint16_t associated_request_id )  [inline]
```

6.8.3 Member Function Documentation

6.8.3.1 clear()

```
void DataTransport::clear ( )
```

This function is called to clear all attributes of the DataTransport for reuse.

6.8.3.2 dataTransportToNet()

```
void DataTransport::dataTransportToNet (
            char * msg )
```

This function is called to serialize DataTransport for transport.

Parameters

msg	Pointer to buffer to write the serialized DataTransport into

6.8.3.3 netToDataTransport()

```
void DataTransport::netToDataTransport (
             char * msg )
```

This function is called to deserialize received DataTransport.

Parameters

msg	Pointer to buffer to the serialized DataTransport

6.8.3.4 print()

```
void DataTransport::print ( )
```

This function prints out the DataTransport.

6.8.4 Member Data Documentation

6.8.4.1 associated_object_

```
DataObjectInstance DataTransport::associated_object_
```

6.8.4.2 associated_request_id_

```
uint16_t DataTransport::associated_request_id_
```

6.8.4.3 block_id_

```
uint16_t DataTransport::block_id_
```

6.8.4.4 block_size_

```
uint16_t DataTransport::block_size_
```

6.8.4.5 data_

```
std::vector<char> DataTransport::data_
```

6.8.4.6 message_length_

```
uint16_t DataTransport::message_length_
```

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

- include/adom/protocol.hpp
- src/cpp/adom/protocol.cpp

6.9 Directory Class Reference

```
#include <directory.hpp>
```

Public Member Functions

- Directory (EntityDescriptor entity_descriptor, SafeQueue< std::pair< DataRequest, udp::endpoint >> &directory_request_queue, SafeQueue< DataTransport > &directory_reply_queue, SafeQueue< std::pair< DataAnnouncement, udp::endpoint >> &directory_announcement_queue)

  Construct a new Directory object.
- void initiate ()

  Initialize the new directory. Handler threads are started and an ingress endpoint for communication is created.
- void addNewObjectSample (std::shared_ptr< ObjectSample > new_sample)

  This function is called at the application interface, as soon as new data is available. New data is either added by an user application directly, or through the reception of a data announcement.
- std::vector< uint16_t > checkAvailability (std::shared_ptr< ObjectSample > sample, std::vector< uint16_t > block_ids)

  This function is called to check whether a given set of blocks (identified by their IDs) for a given object sample is currently locally available.
- int sendDataRequest (EntityDescriptor home_directory, std::shared_ptr< ObjectSample >, std::vector< uint16_t > block_ids)

  This function is called from within a read process triggered by the application. After the data to be read and the associated blocks are determined, ths function is called to iniate the process of sending the required data requests.
- std::shared_ptr< ObjectSample > getLastObjectSample (Topic type)

  This function is called to obtain a shared pointer to the most up-to-date sample for a specified topic that is managed by the directory.
- std::shared_ptr< ObjectSample > getSample (Topic type, int sequence_nr)

This function is called to obtain a shared pointer to the sample for a specified topic and a specified sequence number that is managed by the directory.

- void printTrackedObjectSamples ()

  Print out all objects tracked and cached by the directory.
- int getNumberOfTrackedObjects (Topic topic)

  This function returns the number of tracked samples for a given topic.
- uint16_t getLatestSequencenumberOfTrackedTopic (Topic topic)

  This funtion returns the sequnce number of the most up-to-date sample of a specified topic.
- void setRequestedBlocks (int request_id, const std::set< uint16_t > &requested_blocks)

  Set the Expected Blocks object for a specific request.
- void join ()

  Joining all threads for smooth exit.
- void terminate ()

  Terminating all threads.
- void freeAllObjects ()

  Freeing all Objects.
- void stop ()

  Stopping all processes within directory.

Public Attributes

- EntityDescriptor other_directory_descriptor_

  information/ descriptor on other directories. Temporary solution as compensatoin for a subscriber list, that is currently missing as no discovery process is implemented
- std::mutex request_tracking_lock_
- std::unordered_map< int, RequestTracking > request_tracking_

  List to track received Blocks for pending requests.

6.9.1 Constructor & Destructor Documentation

6.9.1.1 Directory()

```
Directory::Directory (
            EntityDescriptor entity_descriptor,
            SafeQueue< std::pair< DataRequest, udp::endpoint >> & directory_request_queue,
            SafeQueue< DataTransport > & directory_reply_queue,
            SafeQueue< std::pair< DataAnnouncement, udp::endpoint >> & directory_announcement↩
_queue )  [inline]
```

Construct a new Directory object.

Parameters

entity_descriptor	

6.9.2 Member Function Documentation

6.9.2.1 addNewObjectSample()

```
void Directory::addNewObjectSample (
             std::shared_ptr< ObjectSample > new_sample )
```

This function is called at the application interface, as soon as new data is available. New data is either added by an user application directly, or through the reception of a data announcement.

This function specifically is called, when the sample sequence number is not directly given, e.g. by the user application. Therefore, the last tracked sequence number has to be determined.

Parameters

new_sample	shared Pointer to new ObjectSample to be tracked and distributed if requested

6.9.2.2 checkAvailability()

```
std::vector< uint16_t > Directory::checkAvailability (
             std::shared_ptr< ObjectSample > sample,
             std::vector< uint16_t > block_ids )
```

This function is called to check whether a given set of blocks (identified by their IDs) for a given object sample is currently locally available.

Parameters

sample	associated object sample
block_ids	given set of requested block ID

6.9.2.3 freeAllObjects()

```
void Directory::freeAllObjects ( )
```

Freeing all Objects.

6.9.2.4 getLastObjectSample()

```
std::shared_ptr< ObjectSample > Directory::getLastObjectSample (
            Topic type )
```

This function is called to obtain a shared pointer to the most up-to-date sample for a specified topic that is managed by the directory.

Parameters

type	specified object sample topic

Returns

std::shared_ptr<ObjectSample> shared pointer to most up-to-date sample

6.9.2.5 getLatestSequencenumberOfTrackedTopic()

```
uint16_t Directory::getLatestSequencenumberOfTrackedTopic (
            Topic type )
```

This funtion returns the sequnce number of the most up-to-date sample of a specified topic.

Parameters

type	specified object sample topic

Returns

uint16_t latest sequence number

6.9.2.6 getNumberOfTrackedObjects()

```
int Directory::getNumberOfTrackedObjects (
            Topic topic )
```

This function returns the number of tracked samples for a given topic.

Parameters

topic	Topic to check the tracked sample number on

Returns

int Number of tracked samples for the given topic

6.9.2.7 getSample()

```
std::shared_ptr< ObjectSample > Directory::getSample (
            Topic type,
            int sequence_nr )
```

This function is called to obtain a shared pointer to the sample for a specified topic and a specified sequence number that is managed by the directory.

Parameters

type	specified object sample topic
sequence↩_nr	specified sequence number

Returns

std::shared_ptr<ObjectSample> shared pointer to most up-to-date sample

6.9.2.8 initiate()

```
void Directory::initiate ( )
```

Initialize the new directory. Handler threads are started and an ingress endpoint for communication is created.

6.9.2.9 join()

```
void Directory::join ( )
```

Joining all threads for smooth exit.

6.9.2.10 printTrackedObjectSamples()

```
void Directory::printTrackedObjectSamples ( )
```

Print out all objects tracked and cached by the directory.

6.9.2.11 sendDataRequest()

```
int Directory::sendDataRequest (
             EntityDescriptor home_directory,
             std::shared_ptr< ObjectSample > sample,
             std::vector< uint16_t > block_ids )
```

This function is called from within a read process triggered by the application. After the data to be read and the associated blocks are determined, ths function is called to iniate the process of sending the required data requests.

Todo make private

Parameters

home_directory	Home directory for the data that is requested
sample	Affiliated sample the read process is directed to

Returns

Request ID

Todo make private

Parameters

home_directory	Home directory for the data that is requested
sample	Affiliated sample the read process is directed to

Returns

Request ID

6.9.2.12 setRequestedBlocks()

```
void Directory::setRequestedBlocks (
            int request_id,
            const std::set< uint16_t > & requested_blocks )
```

Set the Expected Blocks object for a specific request.

Parameters

request_id	associated request ID
expected_blocks	given requested and therefore expected blocks

6.9.2.13 stop()

```
void Directory::stop ( )
```

Stopping all processes within directory.

6.9.2.14 terminate()

```
void Directory::terminate ( )
```

Terminating all threads.

Joining all threads for smooth exit.

6.9.3 Member Data Documentation

6.9.3.1 other_directory_descriptor_

```
EntityDescriptor Directory::other_directory_descriptor_
```

information/ descriptor on other directories. Temporary solution as compensatoin for a subscriber list, that is currently missing as no discovery process is implemented

Todo make private

6.9.3.2 request_tracking_

```
std::unordered_map<int, RequestTracking> Directory::request_tracking_
```

List to track received Blocks for pending requests.

6.9.3.3 request_tracking_lock_

```
std::mutex Directory::request_tracking_lock_
```

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

- include/adom/directory.hpp
- src/cpp/adom/directory.cpp

6.10 EntityDescriptor Struct Reference

This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself.

```
#include <protocol.hpp>
```

Public Member Functions

- EntityDescriptor ()

Public Attributes

- uint16_t entity_id
- char ip_address [string_length]
- uint16_t ingress_port

6.10.1 Detailed Description

This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 EntityDescriptor()

```
EntityDescriptor::EntityDescriptor ( )  [inline]
```

6.10.3 Member Data Documentation

6.10.3.1 entity_id

```
uint16_t EntityDescriptor::entity_id
```

6.10.3.2 ingress_port

```
uint16_t EntityDescriptor::ingress_port
```

6.10.3.3 ip_address

```
char EntityDescriptor::ip_address[string_length]
```

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

- include/adom/protocol.hpp

6.11 ObjectSample Class Reference

Class to decribe an ObjectSample.

```
#include <directory.hpp>
```

Public Member Functions

- ObjectSample (EntityDescriptor home_directory, Topic topic, uint16_t updated_sequence_nr, unsigned char ∗data, Structure structure)
- EntityDescriptor getHomeDirectory ()
- Topic getTopic ()
- uint16_t getSequenceNumber ()
- uint16_t getBlockCount ()
- Structure getStructure ()
- DataBlockHeader getHeader (int block_id)
- std::vector< DataBlockHeader > getHeader ()
- int isAvailable (int block_id)
- void makeAvailable (int block_id)
- void printObjectSample ()

Public Attributes

- std::vector< unsigned char > object_sample_data_

6.11.1 Detailed Description

Class to decribe an ObjectSample.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 ObjectSample()

```
ObjectSample::ObjectSample (
            EntityDescriptor home_directory,
            Topic topic,
            uint16_t updated_sequence_nr,
            unsigned char * data,
            Structure structure )  [inline]
```

6.11.3 Member Function Documentation

6.11.3.1 getBlockCount()

```
uint16_t ObjectSample::getBlockCount ( )  [inline]
```

6.11.3.2 getHeader() [1/2]

```
std::vector<DataBlockHeader> ObjectSample::getHeader ( )  [inline]
```

6.11.3.3 getHeader() [2/2]

```
DataBlockHeader ObjectSample::getHeader (
             int block_id )  [inline]
```

6.11.3.4 getHomeDirectory()

```
EntityDescriptor ObjectSample::getHomeDirectory ( )  [inline]
```

6.11.3.5 getSequenceNumber()

```
uint16_t ObjectSample::getSequenceNumber ( )  [inline]
```

6.11.3.6 getStructure()

```
Structure ObjectSample::getStructure ( )  [inline]
```

6.11.3.7 getTopic()

```
Topic ObjectSample::getTopic ( )  [inline]
```

6.11.3.8 isAvailable()

```
int ObjectSample::isAvailable (
            int block_id )  [inline]
```

6.11.3.9 makeAvailable()

```
void ObjectSample::makeAvailable (
            int block_id )  [inline]
```

6.11.3.10 printObjectSample()

```
void ObjectSample::printObjectSample ( )  [inline]
```

6.11.4 Member Data Documentation

6.11.4.1 object_sample_data_

```
std::vector<unsigned char> ObjectSample::object_sample_data_
```

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

- include/adom/directory.hpp

6.12 RequestTracking Struct Reference

```
#include <directory.hpp>
```

Public Attributes

- std::set< uint16_t > received_blocks
- std::set< uint16_t > requested_blocks
- std::condition_variable cv

6.12.1 Member Data Documentation

6.12.1.1 cv

```
std::condition_variable RequestTracking::cv
```

6.12.1.2 received_blocks

```
std::set<uint16_t> RequestTracking::received_blocks
```

6.12.1.3 requested_blocks

```
std::set<uint16_t> RequestTracking::requested_blocks
```

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

- include/adom/directory.hpp

6.13 SafeQueue< T > Class Template Reference

A thread safe queue.

```
#include <safe_queue.hpp>
```

Public Member Functions

- SafeQueue (void)

  Construct a new (empty) Safe Queue object.
- ∼SafeQueue (void)

  Destroy the Safe Queue object (default)
- void enqueue (T value)

  Thread safe wrapper for std::queue push. A thread waiting on dequeue is notified after the enqueue operation.
- T dequeue (void)

  Get the "front"-element. If the queue is empty, wait till an element is available.
- T dequeue (bool blocking=true)

  Get the "front"-element. (!) Note: A non blocking call to an empty queue will lead to undefined behavior. Thus the empty() method needs to be called first. (!) Note: A clean implementation would use exceptions to avoid a call to an empty queue.
- T dequeue (bool blocking=true, std::chrono::milliseconds timeout=std::chrono::milliseconds::zero())
- bool empty ()

  Thread safe wrapper for std::queue empty()

Protected Attributes

- std::queue< T > safe_queue

  The underlying queue which is accessed via the queue_lock (mutex)
- std::mutex queue_lock

  The mutex controlling the concurrent queue access.
- std::condition_variable queue_event

  Condition variable used for an enqueue event. The blocking dequeue call waits for an enqueue event if the queue is empty.

6.13.1 Detailed Description

template<class T>
class SafeQueue< T >

A thread safe queue.

Template Parameters

T	the data type to be stored in the queue

6.13.2 Constructor & Destructor Documentation

6.13.2.1 SafeQueue()

```
template<class T >
SafeQueue< T >::SafeQueue (
            void ) [inline]
```

Construct a new (empty) Safe Queue object.

6.13.2.2 ~SafeQueue()

```
template<class T >
SafeQueue< T >::~SafeQueue (
            void ) [inline]
```

Destroy the Safe Queue object (default)

6.13.3 Member Function Documentation

6.13.3.1 dequeue() [1/3]

```
template<class T >
T SafeQueue< T >::dequeue (
            bool blocking = true )  [inline]
```

Get the "front"-element. (!) Note: A non blocking call to an empty queue will lead to undefined behavior. Thus the empty() method needs to be called first. (!) Note: A clean implementation would use exceptions to avoid a call to an empty queue.

Parameters

blocking	

Returns

T The dequeued value

6.13.3.2 dequeue() [2/3]

```
template<class T >
T SafeQueue< T >::dequeue (
            bool blocking = true,
            std::chrono::milliseconds timeout = std::chrono::milliseconds::zero() )  [inline]
```

6.13.3.3 dequeue() [3/3]

```
template<class T >
T SafeQueue< T >::dequeue (
            void )  [inline]
```

Get the "front"-element. If the queue is empty, wait till an element is available.

Returns

T The dequeued value

6.13.3.4 empty()

```
template<class T >
bool SafeQueue< T >::empty ( )  [inline]
```

Thread safe wrapper for std::queue empty()

Returns

true empty queue

false non empty queue

6.13.3.5 enqueue()

```
template<class T >
void SafeQueue< T >::enqueue (
            T value )  [inline]
```

Thread safe wrapper for std::queue push. A thread waiting on dequeue is notified after the enqueue operation.

Parameters

value	the value to be enqueued

6.13.4 Member Data Documentation

6.13.4.1 queue_event

```
template<class T >
std::condition_variable SafeQueue< T >::queue_event  [protected]
```

Condition variable used for an enqueue event. The blocking dequeue call waits for an enqueue event if the queue is empty.

6.13.4.2 queue_lock

```
template<class T >
std::mutex SafeQueue< T >::queue_lock  [mutable], [protected]
```

The mutex controlling the concurrent queue access.

6.13.4.3 safe_queue

```
template<class T >
std::queue<T> SafeQueue< T >::safe_queue  [protected]
```

The underlying queue which is accessed via the queue_lock (mutex)

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

- include/adom/safe_queue.hpp

6.14 socket_endpoint Struct Reference

Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions.

```
#include <socket_endpoint.hpp>
```

Public Member Functions

- socket_endpoint ()
- socket_endpoint (std::string ip, int p)

Public Attributes

- std::string ip_addr
- int port

6.14.1 Detailed Description

Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 socket_endpoint() [1/2]

```
socket_endpoint::socket_endpoint ( )  [inline]
```

6.14.2.2 socket_endpoint() [2/2]

```
socket_endpoint::socket_endpoint (
              std::string ip,
              int p )  [inline]
```

6.14.3 Member Data Documentation

6.14.3.1 ip_addr

```
std::string socket_endpoint::ip_addr
```

6.14.3.2 port

```
int socket_endpoint::port
```

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

- include/adom/socket_endpoint.hpp

6.15 Structure Struct Reference

To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample.

```
#include <translation.h>
```

Public Member Functions

- Structure ()
- Structure (StructureType type, uint16_t block_rows, uint16_t block_cols, uint16_t object_height, uint16_↩
t object_width, uint16_t object_channels)

Public Attributes

- StructureType type
- uint16_t block_rows
- uint16_t block_cols
- uint16_t object_height
- uint16_t object_width
- uint16_t object_channels

6.15.1 Detailed Description

To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample.

1D data structure: Data blocks are composed of one (or several successive) memory "lines" (equals standard hardware data caching) –For 1D data, block_line_width equals about 1kB to fill an etherent frame and block_line↩
_count is 1.– For 1D data, object_height and block_rows is 1.

2D data structure: Data blocks are composed of several NON successive memory "lines" –For 2D data, the following applies: block_line_width $*$ block_line_count $<=$ 1kB and block_line_count $>$ 1.–

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Structure() [1/2]

```
Structure::Structure ( )  [inline]
```

6.15.2.2 Structure() [2/2]

```
Structure::Structure (
            StructureType type,
            uint16_t block_rows,
            uint16_t block_cols,
            uint16_t object_height,
            uint16_t object_width,
            uint16_t object_channels )  [inline]
```

6.15.3 Member Data Documentation

6.15.3.1 block_cols

```
uint16_t Structure::block_cols
```

6.15.3.2 block_rows

```
uint16_t Structure::block_rows
```

6.15.3.3 object_channels

```
uint16_t Structure::object_channels
```

6.15.3.4 object_height

```
uint16_t Structure::object_height
```

6.15.3.5 object_width

```
uint16_t Structure::object_width
```

6.15.3.6 type

```
StructureType Structure::type
```

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

- include/adom/translation.h

Chapter 7

File Documentation

7.1 include/adom/application_interface.hpp File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <vector>
#include "protocol.hpp"
#include "directory.hpp"
#include "log.hpp"
```

Classes

- class ApplicationInterface

7.2 include/adom/directory.hpp File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <condition_variable>
#include <unordered_map>
#include <mutex>
#include <vector>
#include <set>
#include <list>
#include <thread>
#include <boost/asio.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <adom/safe_queue.hpp>
#include <adom/protocol.hpp>
#include <adom/parameters.h>
#include <adom/log.hpp>
```

Classes

- struct RequestTracking
- class ObjectSample

  Class to decribe an ObjectSample.
- class Directory

7.3 include/adom/log.hpp File Reference

```
#include <boost/log/core.hpp>
#include <boost/log/trivial.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/sinks/text_file_backend.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/console.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
#include <boost/log/sources/severity_logger.hpp>
#include <boost/log/sources/record_ostream.hpp>
#include <boost/log/utility/setup/settings.hpp>
#include <boost/log/attributes/timer.hpp>
#include <boost/log/attributes/named_scope.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/log/attributes/clock.hpp>
#include <boost/log/support/date_time.hpp>
#include <adom/parameters.h>
#include <cstdlib>
#include <iostream>
```

Namespaces

- adom

Macros

- #define logTrace(msg)

  Preprocessor wrapper for boost logging library.
- #define logDebug(msg)
- #define logInfo(msg)
- #define logWarning(msg)
- #define logError(msg)
- #define logFatal(msg)
- #define AppLog(msg)

Functions

- void adom::init_file_log (std::string log_prefix, std::string log_suffix)
- void adom::init_console_log ()

7.3.1 Macro Definition Documentation

7.3.1.1 AppLog

```
#define AppLog(
              msg )
```

7.3.1.2 logDebug

```
#define logDebug(
              msg )
```

7.3.1.3 logError

```
#define logError(
              msg )
```

7.3.1.4 logFatal

```
#define logFatal(
              msg )
```

7.3.1.5 logInfo

```
#define logInfo(
              msg )
```

7.3.1.6 logTrace

```
#define logTrace(
              msg )
```

Preprocessor wrapper for boost logging library.

7.3.1.7 logWarning

```
#define logWarning(
            msg )
```

7.4 include/adom/opencv_helper.h File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <vector>
#include <adom/translation.h>
```

Functions

- uint16_t findBlockFromAddress (const unsigned char ∗readDataAddress, const unsigned char ∗objectStart↩ Data, Structure data_structure)

  Finds associated data block for a given data address and returns the number of the associated block.
- uint16_t findBlock (int x, int y, Structure data_structure)

  Finds associated data block for a given data address and returns the number of the associated block.
- std::vector< uint16_t > findBlocksForReadAccess (int first_pixel_x, int first_pixel_y, int last_pixel_x, int last↩ _pixel_y, Structure data_structure)

  Finds associated data blocks for a read access to an image that is specified by the first pixel and the last pixel.

7.4.1 Function Documentation

7.4.1.1 findBlock()

```
uint16_t findBlock (
           int x,
           int y,
           Structure data_structure )  [inline]
```

Finds associated data block for a given data address and returns the number of the associated block.

Parameters

readDataAddress	given data address (through read process)
objectStartData	start address of data objects
data_structure	internal structure of the data object and how it is cached

Returns

int

7.4.1.2 findBlockFromAddress()

```
uint16_t findBlockFromAddress (
            const unsigned char * readDataAddress,
            const unsigned char * objectStartData,
            Structure data_structure ) [inline]
```

Finds associated data block for a given data address and returns the number of the associated block.

Parameters

readDataAddress	given data address (through read process)
objectStartData	start address of data objects
data_structure	internal structure of the data object and how it is cached

Returns

int

7.4.1.3 findBlocksForReadAccess()

```
std::vector<uint16_t> findBlocksForReadAccess (
            int first_pixel_x,
            int first_pixel_y,
            int last_pixel_x,
            int last_pixel_y,
            Structure data_structure ) [inline]
```

Finds associated data blocks for a read access to an image that is specified by the first pixel and the last pixel.

Parameters

first_pixel_x	x coordindate of start pixel
first_pixel_y	y coordindate of start pixel
last_pixel_x	x coordindate of end pixel
last_pixel_y	y coordindate of end pixel
data_structure	structure of the image, e.g. number of columns and rows...

Returns

std::vector<uint16_t> block IDs of the associated blocks

7.5 include/adom/parameters.h File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <boost/asio.hpp>
```

Macros

- #define PROTOCOL_TEST 1
- #define MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC 3
- #define PRINTF_BINARY_PATTERN_INT8 "%c%c%c%c%c%c%c%c"
- #define PRINTF_BYTE_TO_BINARY_INT8(i)

Enumerations

- enum Topic { NONE = 0 , IMAGE = 1 }

  Enum to describe different topics.

Variables

- const std::string log_directory = "/ApplicationDataObjectManagement/examples/adom_logs/"
- const char publisher_ip [20] = "192.168.3.100"
- const char subscriber_ip [20] = "192.168.3.101"
- const uint16_t home_dir_ingress_port = 5400
- const uint16_t req_dir_ingress_port = 5402
- const uint16_t string_length = 20
- const uint16_t adom_message_overhead = 8
- const uint16_t max_buffer_length = 1400
- const uint16_t TOTAL_BLOCK_SIZE = 1200
- const uint16_t HD_BLOCKS_IN_ROW = 64
- const uint16_t HD_BLOCK_ROWS = 36
- const uint16_t HD_V_PIXEL_PER_IMAGE = 720
- const uint16_t HD_H_PIXEL_PER_IMAGE = 1280
- const uint16_t FULL_HD_BLOCKS_IN_ROW = 96
- const uint16_t FULL_HD_BLOCK_ROWS = 54
- const uint16_t FULL_HD_V_PIXEL_PER_IMAGE = 1080
- const uint16_t FULL_HD_H_PIXEL_PER_IMAGE = 1920
- const uint16_t V_PIXEL_PER_BLOCK = 20
- const uint16_t H_PIXEL_PER_BLOCK = 20
- const uint16_t CHANNEL_NUMBER = 3
- const uint16_t MAX_NUM_TRACKED_SAMPLES = 5
- const uint16_t EMPTY_QUEUE_ERROR = 20

7.5.1 Macro Definition Documentation

7.5.1.1 MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC

```
#define MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC 3
```

7.5.1.2 PRINTF_BINARY_PATTERN_INT8

```
#define PRINTF_BINARY_PATTERN_INT8 "%c%c%c%c%c%c%c%c"
```

7.5.1.3 PRINTF_BYTE_TO_BINARY_INT8

```
#define PRINTF_BYTE_TO_BINARY_INT8(
              i )
```

Value:

```
    (((i) & 0x80ll) ? '1' : '0'), \
    (((i) & 0x40ll) ? '1' : '0'), \
    (((i) & 0x20ll) ? '1' : '0'), \
    (((i) & 0x10ll) ? '1' : '0'), \
    (((i) & 0x08ll) ? '1' : '0'), \
    (((i) & 0x04ll) ? '1' : '0'), \
    (((i) & 0x02ll) ? '1' : '0'), \
    (((i) & 0x01ll) ? '1' : '0')
```

7.5.1.4 PROTOCOL_TEST

```
#define PROTOCOL_TEST 1
```

7.5.2 Enumeration Type Documentation

7.5.2.1 Topic

```
enum Topic
```

Enum to describe different topics.

Enumerator

NONE	
IMAGE	

7.5.3 Variable Documentation

7.5.3.1 adom_message_overhead

```
const uint16_t adom_message_overhead = 8
```

7.5.3.2 CHANNEL_NUMBER

```
const uint16_t CHANNEL_NUMBER = 3
```

7.5.3.3 EMPTY_QUEUE_ERROR

```
const uint16_t EMPTY_QUEUE_ERROR = 20
```

7.5.3.4 FULL_HD_BLOCK_ROWS

```
const uint16_t FULL_HD_BLOCK_ROWS = 54
```

7.5.3.5 FULL_HD_BLOCKS_IN_ROW

```
const uint16_t FULL_HD_BLOCKS_IN_ROW = 96
```

7.5.3.6 FULL_HD_H_PIXEL_PER_IMAGE

```
const uint16_t FULL_HD_H_PIXEL_PER_IMAGE = 1920
```

7.5.3.7 FULL_HD_V_PIXEL_PER_IMAGE

```
const uint16_t FULL_HD_V_PIXEL_PER_IMAGE = 1080
```

7.5.3.8 H_PIXEL_PER_BLOCK

```
const uint16_t H_PIXEL_PER_BLOCK = 20
```

7.5.3.9 HD_BLOCK_ROWS

```
const uint16_t HD_BLOCK_ROWS = 36
```

7.5.3.10 HD_BLOCKS_IN_ROW

```
const uint16_t HD_BLOCKS_IN_ROW = 64
```

7.5.3.11 HD_H_PIXEL_PER_IMAGE

```
const uint16_t HD_H_PIXEL_PER_IMAGE = 1280
```

7.5.3.12 HD_V_PIXEL_PER_IMAGE

```
const uint16_t HD_V_PIXEL_PER_IMAGE = 720
```

7.5.3.13 home_dir_ingress_port

```
const uint16_t home_dir_ingress_port = 5400
```

7.5.3.14 log_directory

```
const std::string log_directory = "/ApplicationDataObjectManagement/examples/adom_logs/"
```

7.5.3.15 max_buffer_length

```
const uint16_t max_buffer_length = 1400
```

7.5.3.16 MAX_NUM_TRACKED_SAMPLES

```
const uint16_t MAX_NUM_TRACKED_SAMPLES = 5
```

7.5.3.17 publisher_ip

```
const char publisher_ip[20] = "192.168.3.100"
```

7.5.3.18 req_dir_ingress_port

```
const uint16_t req_dir_ingress_port = 5402
```

7.5.3.19 string_length

```
const uint16_t string_length = 20
```

7.5.3.20 subscriber_ip

```
const char subscriber_ip[20] = "192.168.3.101"
```

7.5.3.21 TOTAL_BLOCK_SIZE

```
const uint16_t TOTAL_BLOCK_SIZE = 1200
```

7.5.3.22 V_PIXEL_PER_BLOCK

```
const uint16_t V_PIXEL_PER_BLOCK = 20
```

7.6 include/adom/protocol.hpp File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <iostream>
#include <vector>
#include <adom/parameters.h>
#include <adom/translation.h>
#include <adom/debug.h>
```

Classes

- struct EntityDescriptor

  This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself.
- struct DataObjectInstance

  Struct to describe an Object instance with its sequence number within a topic or object type.
- struct DataBlockHeader

  Struct to describe a Data Block, that is part of an Object instance.
- class AdomMessage

  Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports.
- class DataRequest

  DataRequest is used by a remote (subscribed) directory (reader_application_) to request specific blocks of a sample object managed by the data's (publihser/) home directory (home_directory_) via a block_validity_matrix.
- class DataTransport

  DataTransport is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory.
- class DataAnnouncement

  DataAnnouncement is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample.
- struct AppRequest

Enumerations

- enum MessageType { UNKNOWN = 0 , REQUEST = 1 , REPLY = 2 , ANNOUNCEMENT }

  Enum to describe the different protocol message types.
- enum BlockValidity { INVALID = 0 , REQUESTED , VALID , SHARED }

  Enum to describe the status of a block within a validity matrix of an object.

Functions

- bool operator== (const EntityDescriptor &lhs, const EntityDescriptor &rhs)

7.6.1 Enumeration Type Documentation

7.6.1.1 BlockValidity

`enum BlockValidity`

Enum to describe the status of a block within a validity matrix of an object.

Enumerator

INVALID	
REQUESTED	
VALID	
SHARED	

7.6.1.2 MessageType

```
enum MessageType
```

Enum to describe the different protocol message types.

Enumerator

UNKNOWN	
REQUEST	
REPLY	
ANNOUNCEMENT	

7.6.2 Function Documentation

7.6.2.1 operator==()

```
bool operator== (
            const EntityDescriptor & lhs,
            const EntityDescriptor & rhs )  [inline]
```

7.7 include/adom/safe_queue.hpp File Reference

```
#include <queue>
#include <mutex>
#include <condition_variable>
#include <adom/parameters.h>
```

Classes

- class SafeQueue< T >

  A thread safe queue.

7.8 include/adom/socket_endpoint.hpp File Reference

```
#include <string>
#include <chrono>
```

Classes

- struct socket_endpoint

    Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions.

7.9 include/adom/translation.h File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <iostream>
#include <vector>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <adom/parameters.h>
```

Classes

- struct Structure

    To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample.

Enumerations

- enum StructureType { NOT_SPECIFIED = 0 , ONE_DIMENSIONAL = 1 , TWO_DIMENSIONAL = 2 }

    Enum to describe different types of decomposition strategies for objects samples.

Functions

- std::vector< unsigned char > translate_from_uchar (unsigned char ∗data, Structure structure)

    This function takes data in form of unsigned char ∗data, e.g. unsigned char array, and translates in into the data format required to construct a cachable Object Sample.
- int translate_uchar_access (const unsigned char ∗pixel_address, const unsigned char ∗original_data, Structure structure)
- unsigned char ∗ translate_to_uchar (std::vector< unsigned char > object_sample_data, Structure structure)
- void translate_to_uchar (std::vector< unsigned char > object_sample_data, unsigned char ∗output_data, Structure structure)

7.9.1 Enumeration Type Documentation

7.9.1.1 StructureType

```
enum StructureType
```

Enum to describe different types of decomposition strategies for objects samples.

Enumerator

NOT_SPECIFIED	
ONE_DIMENSIONAL	
TWO_DIMENSIONAL	

7.9.2 Function Documentation

7.9.2.1 translate_from_uchar()

```
std::vector<unsigned char> translate_from_uchar (
             unsigned char * data,
             Structure structure )  [inline]
```

This function takes data in form of unsigned char ∗data, e.g. unsigned char array, and translates in into the data format required to construct a cachable Object Sample.

Todo o check for allignement issues, is the new data buffer big enough etc

7.9.2.2 translate_to_uchar() [1/2]

```
unsigned char* translate_to_uchar (
             std::vector< unsigned char > object_sample_data,
             Structure structure )  [inline]
```

7.9.2.3 translate_to_uchar() [2/2]

```
void translate_to_uchar (
             std::vector< unsigned char > object_sample_data,
             unsigned char * output_data,
             Structure structure )  [inline]
```

7.9.2.4 translate_uchar_access()

```
int translate_uchar_access (
             const unsigned char * pixel_address,
             const unsigned char * original_data,
             Structure structure )  [inline]
```

7.10 src/cpp/adom/application_interface.cpp File Reference

```
#include <adom/application_interface.hpp>
#include <adom/opencv_helper.h>
#include <boost/log/trivial.hpp>
#include <boost/log/core.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/sources/severity_logger.hpp>
#include <boost/log/sources/record_ostream.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
#include <boost/log/utility/setup/console.hpp>
```

7.11 src/cpp/adom/directory.cpp File Reference

```
#include <adom/directory.hpp>
#include <boost/log/trivial.hpp>
#include <boost/log/core.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/sources/severity_logger.hpp>
#include <boost/log/sources/record_ostream.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
#include <boost/log/utility/setup/console.hpp>
#include <chrono>
```

7.12 src/cpp/adom/log.cpp File Reference

```
#include <adom/log.hpp>
#include <boost/date_time.hpp>
```

7.13 src/cpp/adom/protocol.cpp File Reference

```
#include <adom/protocol.hpp>
```

Index