Measurement Data Solution definitions and interfaces

0.0.1

Generated by Doxygen 1.9.1

Thu Oct 13 2022 12:03:00

1 MDS definitions readme

Back

Here are presents defintions and interfaces used in the proect.

1.1 <a href="./include/defs/Distribution.hpp">Distribution interfaces</a>

1.2 <a href="./include/defs/Log.hpp">Logging interfaces</a>

1.3 <a href="./include/defs/MdsInterface.hpp">MDS general purpose interfaces</a>

1.4 <a href="./include/defs/MeasurementObjectDefs.hpp">Measurement object specific interfaces</a>

1.5 <a href="./include/defs/Receiver.hpp">Receiver interfaces</a>

1.6 [Transmitter interfaces]()

For more information read the generated doxygen documentation. To generate the doxygen documentation run the doxygen having as `paramter` the doxygen configuration for this project.

2 Todo List

Global LoggingInterface::save ()=0

unimplemented, all the messages will be piped to the out or err

3 Hierarchical Index

3.1 Class Hierarchy

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

4 Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

6 Data Structure Documentation

6.1 ConfigurationFactory Struct Reference

```
#include <Configuration.hpp>
```

Public Member Functions

- virtual const FactoryMap & getFactoryMap ()=0
- virtual size_t getFactorySize ()=0

  Get the factory size. The factory size represents the number of unique measurement objects that can be created. The factory is populated using dlopen methods.

6.1.1 Member Function Documentation

6.1.1.1 getFactoryMap() `virtual const FactoryMap& ConfigurationFactory::getFactoryMap ( ) [pure virtual]`

6.1.1.2 getFactorySize() `virtual size_t ConfigurationFactory::getFactorySize ( ) [pure virtual]`

Get the factory size. The factory size represents the number of unique measurement objects that can be created. The factory is populated using dlopen methods.

Returns

Return the factory size.

Note

For more information about the factory read MeasurementObjectFactory class definitions.

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

- include/defs/Configuration.hpp

6.2 ConfigurationParser Struct Reference

Interface used to load a configurarion file.

```
#include <Configuration.hpp>
```

Public Member Functions

- virtual const MeasurementObjectList & loadConfiguration (std::filesystem::path path)=0

  loads all the objects with their properties into the configuration manager.
- virtual bool createMeasurementObject (const std::string &name, uint8_t instanceNb)=0
- virtual bool createMeasurementObject (MeasurementObjectPtr object)=0

  Method used to introduce an already created measurement object into the configuration manager.
- virtual bool removeMeasurementObject (const std::string &name)=0

  Method used to remove a measurement object from the configuration manager.
- virtual void clearMeasurementObjectList ()=0
- virtual const MeasurementObjectList & getMOsAddedInConfig ()=0

  Retreive the active measurement object lists from the configuration manager.

6.2.1 Detailed Description

Interface used to load a configurarion file.

6.2.2 Member Function Documentation

6.2.2.1 clearMeasurementObjectList() `virtual void ConfigurationParser::clearMeasurementObject↩ List ( ) [pure virtual]`

6.2.2.2 createMeasurementObject() [1/2] `virtual bool ConfigurationParser::createMeasurement↩ Object (`

```
            const std::string & name,
            uint8_t instanceNb )   [pure virtual]
```

6.2.2.3 createMeasurementObject() [2/2] `virtual bool ConfigurationParser::createMeasurement↩ Object (`

```
            MeasurementObjectPtr object )   [pure virtual]
```

Method used to introduce an already created measurement object into the configuration manager.

Parameters

object	Already created measurement object that will be inserted into the configuration manager.

Returns

Return true if the object was inserted correctly, false otherwise.

6.2.2.4 getMOsAddedInConfig() `virtual const MeasurementObjectList& ConfigurationParser::get↩ MOsAddedInConfig ( ) [pure virtual]`

Retreive the active measurement object lists from the configuration manager.

Returns

Return a const reference of the measurement object list.

6.2.2.5 loadConfiguration() `virtual const MeasurementObjectList& ConfigurationParser::load↩ Configuration (`

```
            std::filesystem::path path )   [pure virtual]
```

loads all the objects with their properties into the configuration manager.

Parameters

path	path to the configuration file. The full path must be provided.

Returns

Return a const list of measurement objects.

Note

When loading a configuration, engine will be reseted and reinitialized.

6.2.2.6 removeMeasurementObject() `virtual bool ConfigurationParser::removeMeasurementObject ( const std::string & name ) [pure virtual]`

Method used to remove a measurement object from the configuration manager.

Parameters

name	Measurement object name

Returns

True if the

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

- include/defs/Configuration.hpp

6.3 DataDistribution Struct Reference

Data distribution interface. Let distribute a package to all linked the receivers.

```
#include <Distribution.hpp>
```

Public Member Functions

- virtual bool distributeData (DataPackageCPtr package)=0

  Distribute data to all the available receivers from the receiver pool.

6.3.1 Detailed Description

Data distribution interface. Let distribute a package to all linked the receivers.

6.3.2 Member Function Documentation

6.3.2.1 distributeData() `virtual bool DataDistribution::distributeData ( DataPackageCPtr package ) [pure virtual]`

Distribute data to all the available receivers from the receiver pool.

Parameters

package	Const data package that will be delived to all the receivers.

Returns

true if the distribution succedded, false if any of the receiver fails to interpret the package.

Note

Data distribution will be performed sequencially.

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

- include/defs/Distribution.hpp

6.4 DataDistributionStatistics Struct Reference

```
#include <Distribution.hpp>
```

Public Member Functions

- virtual const uint64_t & getNumberOfProcessedPackagesPerSecond ()=0
- virtual const uint64_t & getMaximumProcessedPackagesPerSecond ()=0

6.4.1 Member Function Documentation

6.4.1.1 getMaximumProcessedPackagesPerSecond() `virtual const uint64_t& DataDistribution↩ Statistics::getMaximumProcessedPackagesPerSecond ( ) [pure virtual]`

6.4.1.2 getNumberOfProcessedPackagesPerSecond() `virtual const uint64_t& DataDistribution↩ Statistics::getNumberOfProcessedPackagesPerSecond ( ) [pure virtual]`

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

- include/defs/Distribution.hpp

6.5 DataPackage Struct Reference

Data package definition.

```
#include <DataPackage.hpp>
```

Data Fields

- uint64_t timestamp

  package timestamp
- uint64_t sourceHandle

  package source handle
- size_t size

  package size
- uint8_t cycle_

  package cycle
- PackageType type

  package type
- void ∗ payload

  pointer to where the package payload starts.

6.5.1 Detailed Description

Data package definition.

6.5.2 Field Documentation

6.5.2.1 cycle_ `uint8_t DataPackage::cycle_`

package cycle

6.5.2.2 payload `void* DataPackage::payload`

pointer to where the package payload starts.

6.5.2.3 size `size_t DataPackage::size`

package size

6.5.2.4 sourceHandle `uint64_t DataPackage::sourceHandle`

package source handle

6.5.2.5 timestamp `uint64_t DataPackage::timestamp`

package timestamp

6.5.2.6 type `PackageType DataPackage::type`

package type

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

- include/defs/DataPackage.hpp

6.6 DataReceiverObject Struct Reference

Interface used by any processor in order to receive packages from the distribution manager.

```
#include <Receiver.hpp>
```

Public Member Functions

- virtual bool validatePackage (DataPackageCPtr package)=0

  Process data package.

6.6.1 Detailed Description

Interface used by any processor in order to receive packages from the distribution manager.

6.6.2 Member Function Documentation

6.6.2.1 validatePackage() `virtual bool DataReceiverObject::validatePackage ( DataPackageCPtr package ) [pure virtual]`

Process data package.

Note

The package cannot be altered, but a new package can be created to be delivered to the subscribers

Parameters

package	Pointer to a const data package that will be analyzed and proccessed.

Returns

Return true if all the processors validate the package, false if any proccesor cannot validate it.

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

- include/defs/Receiver.hpp

6.7 DataSenderObject Struct Reference

Interface used by any data transmitter in order to transmit packages to the distribution manager.

```
#include <Transmitter.hpp>
```

Public Member Functions

- virtual void startProcessing ()=0
  start the processing threah that will distribute data to the data distribution manager
- virtual void endProcessing ()=0
  end the processing threah that will distribute data to the data distribution manager

6.7.1 Detailed Description

Interface used by any data transmitter in order to transmit packages to the distribution manager.

6.7.2 Member Function Documentation

6.7.2.1 endProcessing() `virtual void DataSenderObject::endProcessing ( ) [pure virtual]`

end the processing threah that will distribute data to the data distribution manager

6.7.2.2 startProcessing() `virtual void DataSenderObject::startProcessing ( ) [pure virtual]`

start the processing threah that will distribute data to the data distribution manager

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

- include/defs/Transmitter.hpp

6.8 EngineInit Struct Reference

Interface used to initialize MDS Engine.

```
#include <MdsInterface.hpp>
```

Public Member Functions

- virtual void initialize ()=0

  Initialize the engine with all its components.
- virtual void terminate ()=0

  Destroy the engine with all its dependencies created.

6.8.1 Detailed Description

Interface used to initialize MDS Engine.

6.8.2 Member Function Documentation

6.8.2.1 initialize() `virtual void EngineInit::initialize ( ) [pure virtual]`

Initialize the engine with all its components.

6.8.2.2 terminate() `virtual void EngineInit::terminate ( ) [pure virtual]`

Destroy the engine with all its dependencies created.

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

- include/defs/MdsInterface.hpp

6.9 ExtendedMeasurementObject Struct Reference

```
#include <MeasurementObjectDefs.hpp>
```

Inheritance diagram for ExtendedMeasurementObject:

Collaboration diagram for ExtendedMeasurementObject:

Public Member Functions

- virtual bool hasPropertyTable ()=0
- virtual bool insertEntry (const PropertyPair &entryPair)=0
- virtual bool removeProperty (const std::string &propertyName)=0
- virtual void clearPropertyTable ()=0
- virtual const PropertyTable & getPropertyTable ()=0
- virtual const std::string & getPropertyEntryValue (const std::string &entry)=0

6.9.1 Member Function Documentation

6.9.1.1 clearPropertyTable()

`virtual void ExtendedMeasurementObject::clearPropertyTable ( ) [pure virtual]`

6.9.1.2 getPropertyEntryValue()

```
virtual const std::string& ExtendedMeasurementObject::get←
PropertyEntryValue (
            const std::string & entry )  [pure virtual]
```

6.9.1.3 getPropertyTable()

```
virtual const PropertyTable& ExtendedMeasurementObject::getProperty←
Table ( )  [pure virtual]
```

6.9.1.4 hasPropertyTable()

`virtual bool ExtendedMeasurementObject::hasPropertyTable ( ) [pure virtual]`

6.9.1.5 insertEntry()

```
virtual bool ExtendedMeasurementObject::insertEntry (
            const PropertyPair & entryPair )  [pure virtual]
```

6.9.1.6 removeProperty()

```
virtual bool ExtendedMeasurementObject::removeProperty (
            const std::string & propertyName )  [pure virtual]
```

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

- include/defs/MeasurementObjectDefs.hpp

6.10 InterfaceAccess Struct Reference

Interface helper that facilitate getting other interfaces.

```
#include <MdsInterface.hpp>
```

Public Member Functions

- virtual void ∗ getInterface (const std::string &interfaceName)=0

  Retreive a pointer to the desired interface.

6.10.1 Detailed Description

Interface helper that facilitate getting other interfaces.

6.10.2 Member Function Documentation

6.10.2.1 getInterface()

```
virtual void* InterfaceAccess::getInterface (
        const std::string & interfaceName )  [pure virtual]
```

Retreive a pointer to the desired interface.

Parameters

interfaceName	Name of the interface required.

Returns

a pointer to the requested interface

Warning

check the pointer before using it. If the interface doesn't exist it will return a nullptr.

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

- include/defs/MdsInterface.hpp

6.11 LoggingInterface Struct Reference

Logging interface. Managed by the engine and use to report messages.

```
#include <Log.hpp>
```

Public Member Functions

- virtual void save ()=0

  Save the log under a XML format.
- virtual void log (const char ∗message, const uint64_t handle=0, const severity sev=severity::debug)=0

  logging method.
- virtual bool subscribe (const char ∗name="Engine", const uint64_t handle=0)=0

  subscribe method. Link a name with a handle.
- virtual bool unsubscribe (const uint64_t handle=0)=0

  unsubscribe method. Unink a name with a handle.

6.11.1 Detailed Description

Logging interface. Managed by the engine and use to report messages.

6.11.2 Member Function Documentation

6.11.2.1 log()

```
virtual void LoggingInterface::log (
        const char * message,
        const uint64_t handle = 0,
        const severity sev = severity::debug ) [pure virtual]
```

logging method.

Parameters

message	log message
handle	reporter handle. If the reporter is registered using the subscribe method, then the name will apear, otherwise the handle will appear instead. If no handle is passed then the Engine will be the reporter.
severity	log message severity. By default the severity is set as debug.

6.11.2.2 save()

```
virtual void LoggingInterface::save ( ) [pure virtual]
```

Save the log under a XML format.

Todo unimplemented, all the messages will be piped to the out or err

6.11.2.3 subscribe()

```
virtual bool LoggingInterface::subscribe (
            const char * name = "Engine",
            const uint64_t handle = 0 )  [pure virtual]
```

subscribe method. Link a name with a handle.

Note

This method should be used before logging any messages.

Parameters

name	name of the reporter. By default the name is Engine.
handle	reporter handle. By default the handle is 0 (corresponding to the engine)

Returns

Return true if the link between the name and the handle was made possible, false otherwise.

6.11.2.4 unsubscribe()

```
virtual bool LoggingInterface::unsubscribe (
            const uint64_t handle = 0 )  [pure virtual]
```

unsubscribe method. Unink a name with a handle.

Note

This method should be used before logging any messages.

Parameters

name	name of the reporter. By default the name is Engine.
handle	reporter handle. By default the handle is 0 (corresponding to the engine)

Returns

Return true if the unlink between the name and the handle was made possible, false otherwise.

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

- include/defs/Log.hpp

6.12 MeasurementObject Struct Reference

Interface for retreiving measurement object information.

```
#include <MeasurementObjectDefs.hpp>
```

Inheritance diagram for MeasurementObject:

Public Member Functions

- virtual const uint8_t & getInstanceNumber ()=0

  Method to retreive the instance number of the measurement object.
- virtual const uint64_t & getHandle ()=0

  Method to retreive the handle of the measurement object.
- virtual const MeasurementObjectType & getType ()=0

  Method to retreive the type of the measurement object.
- virtual const std::string & getName ()=0

  Method to retreive the name of the measurement object.

6.12.1 Detailed Description

Interface for retreiving measurement object information.

6.12.2 Member Function Documentation

6.12.2.1 getHandle() `virtual const uint64_t& MeasurementObject::getHandle ( ) [pure virtual]`

Method to retreive the handle of the measurement object.

Returns

Return measurement object handle.

6.12.2.2 getInstanceNumber() `virtual const uint8_t& MeasurementObject::getInstanceNumber ( ) [pure virtual]`

Method to retreive the instance number of the measurement object.

Returns

Return measurement object instance number.

6.12.2.3 getName() `virtual const std::string& MeasurementObject::getName ( ) [pure virtual]`

Method to retreive the name of the measurement object.

Returns

Return measurement object name.

6.12.2.4 getType() `virtual const MeasurementObjectType& MeasurementObject::getType ( ) [pure virtual]`

Method to retreive the type of the measurement object.

Returns

Return measurement object type.

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

- include/defs/MeasurementObjectDefs.hpp

6.13 NotifySubjects Struct Reference

Interface used to notify all the subject registered using registerToReceiverSink method to a processor.

```
#include <Receiver.hpp>
```

Public Member Functions

- virtual bool notify (DataPackageCPtr pkg)=0

  Method use to notify a subscriber when a data package is received.

6.13.1 Detailed Description

Interface used to notify all the subject registered using registerToReceiverSink method to a processor.

6.13.2 Member Function Documentation

6.13.2.1 notify() `virtual bool NotifySubjects::notify ( DataPackageCPtr pkg ) [pure virtual]`

Method use to notify a subscriber when a data package is received.

Parameters

pkg	data package that will be transmitted to the subject

Returns

Return true if the subject was notified.

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

- include/defs/Receiver.hpp

6.14 ObjectControl Struct Reference

`#include <MeasurementObjectDefs.hpp>`

Public Member Functions

- virtual void initializeObject ()=0
- virtual void terminateObject ()=0

6.14.1 Member Function Documentation

6.14.1.1 initializeObject() `virtual void ObjectControl::initializeObject ( ) [pure virtual]`

6.14.1.2 terminateObject() `virtual void ObjectControl::terminateObject ( ) [pure virtual]`

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

- include/defs/MeasurementObjectDefs.hpp

6.15 ReceiverSinkManager Struct Reference

Interface used to register a subject to a processor.

`#include <Receiver.hpp>`

Public Member Functions

- virtual bool registerToReceiverSink (NotifySubjects ∗subject)=0

  Method use to register a subject to a processor.
- virtual bool unregisterToReceiverSink (NotifySubjects ∗subject)=0

  Method use to unregister a subject to a processor.

6.15.1 Detailed Description

Interface used to register a subject to a processor.

6.15.2 Member Function Documentation

6.15.2.1 registerToReceiverSink() `virtual bool ReceiverSinkManager::registerToReceiverSink ( NotifySubjects * subject ) [pure virtual]`

Method use to register a subject to a processor.

Parameters

subject	subject that will be stored in the notification sink of the processor

Returns

Return true if the subject was introduced to the processor sink

6.15.2.2 unregisterToReceiverSink() `virtual bool ReceiverSinkManager::unregisterToReceiverSink ( NotifySubjects * subject ) [pure virtual]`

Method use to unregister a subject to a processor.

Parameters

subject	subject that will be cleared from the notification sink of the processor

Returns

Return true if the subject was cleared from the processor sink

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

- include/defs/Receiver.hpp

6.16 SystemObject Struct Reference

`#include <MeasurementObjectDefs.hpp>`

Inheritance diagram for SystemObject:

Collaboration diagram for SystemObject:

Public Member Functions

- virtual void initializeSystemObject ()=0
- virtual void terminateSystemObject ()=0

6.16.1 Member Function Documentation

6.16.1.1 initializeSystemObject() `virtual void SystemObject::initializeSystemObject ( ) [pure virtual]`

6.16.1.2 terminateSystemObject() `virtual void SystemObject::terminateSystemObject ( ) [pure virtual]`

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

- include/defs/MeasurementObjectDefs.hpp

7 File Documentation

7.1 include/defs/Configuration.hpp File Reference

```
#include <string>
#include <cstdint>
#include <filesystem>
```

Include dependency graph for Configuration.hpp:

7.2 include/defs/DataPackage.hpp File Reference

```
#include <cstdint>
#include <memory>
```

Include dependency graph for DataPackage.hpp: This graph shows which files directly or indirectly include this file:

Data Structures

- struct DataPackage

  Data package definition.

Typedefs

- using DataPackagePtr = DataPackage ∗

  Data package pointer.
- using DataPackageCPtr = const DataPackage ∗

  Const data package pointer.

Enumerations

- enum class PackageType : uint8_t {
  dummy = 0x00 , camera = 0x01 , can = 0x02 , flexray = 0x04 ,
  ethernet = 0x08 }

7.2.1 Typedef Documentation

7.2.1.1 DataPackageCPtr `using DataPackageCPtr = const DataPackage*`

Const data package pointer.

7.2.1.2 DataPackagePtr `using DataPackagePtr = DataPackage*`

Data package pointer.

7.2.2 Enumeration Type Documentation

7.2.2.1 PackageType `enum PackageType : uint8_t [strong]`

Enumerator

dummy	
camera	
can	
flexray	
ethernet	

7.3 include/defs/Distribution.hpp File Reference

```
#include <cstdint>
#include <defs/DataPackage.hpp>
```

Include dependency graph for Distribution.hpp:

Data Structures

- struct DataDistribution

  Data distribution interface. Let distribute a package to all linked the receivers.
- struct DataDistributionStatistics

Typedefs

- using DataDistributionPtr = std::shared_ptr< DataDistribution >

  Data distribution pointer.

7.3.1 Typedef Documentation

7.3.1.1 DataDistributionPtr `using DataDistributionPtr = std::shared_ptr<DataDistribution>`

Data distribution pointer.

7.4 include/defs/Log.hpp File Reference

```
#include <cstdint>
```
Include dependency graph for Log.hpp:

Data Structures

- struct LoggingInterface

  Logging interface. Managed by the engine and use to report messages.

Enumerations

- enum class severity : uint8_t {
  debug = 0x00 , information = 0x01 , warning = 0x02 , error = 0x03 ,
  critical = 0x04 }

  severity of the log message enum

7.4.1 Enumeration Type Documentation

7.4.1.1 severity `enum severity : uint8_t [strong]`

severity of the log message enum

Enumerator

debug	debug message.
information	information message.
warning	warning message.
error	error message.
critical	critical message.

7.5 include/defs/MdsInterface.hpp File Reference

```
#include <defs/MeasurementObjectDefs.hpp>
#include <string>
```
Include dependency graph for MdsInterface.hpp:

Data Structures

- struct InterfaceAccess

Interface helper that facilitate getting other interfaces.
- struct EngineInit

  Interface used to initialize MDS Engine.

Macros

- #define INVALID_HANDLE 0xffffffffffffffff
- #define ENGINE_HANDLE 0x00
- #define CONFIGURATION_MGR_HANDLE 0x01
- #define DISTRIBUTION_MGR_HANDLE 0x02
- #define WATCHDOG_HANDLE 0x03
- #define FACTORY_HANDLE 0x04

Enumerations

- enum class EngineInitFlag : uint8_t {
  normal = 0x00 , silent = 0x01 , no_metrics = 0x02 , no_exception_handler = 0x04 ,
  performance = 0x07 }

  Flag indicating how the engine will be created.

7.5.1 Macro Definition Documentation

7.5.1.1 CONFIGURATION_MGR_HANDLE `#define CONFIGURATION_MGR_HANDLE 0x01`

7.5.1.2 DISTRIBUTION_MGR_HANDLE `#define DISTRIBUTION_MGR_HANDLE 0x02`

7.5.1.3 ENGINE_HANDLE `#define ENGINE_HANDLE 0x00`

7.5.1.4 FACTORY_HANDLE `#define FACTORY_HANDLE 0x04`

7.5.1.5 INVALID_HANDLE `#define INVALID_HANDLE 0xffffffffffffffff`

7.5.1.6 WATCHDOG_HANDLE `#define WATCHDOG_HANDLE 0x03`

7.5.2 Enumeration Type Documentation

7.5.2.1 EngineInitFlag `enum EngineInitFlag : uint8_t [strong]`

Flag indicating how the engine will be created.

Enumerator

normal	engine will start normaly with all its components.
silent	engine will inform the logger to not log debug messages.
no_metrics	engine won't create a watchdog.
no_exception_handler	some safety casts will be ignored.
performance	engine will run with the following flags: silent | no_metrics | no_exception_handler

7.6 include/defs/MeasurementObjectDefs.hpp File Reference

```
#include <map>
#include <list>
#include <memory>
#include <cstdint>
#include <string>
```

Include dependency graph for MeasurementObjectDefs.hpp: This graph shows which files directly or indirectly include this file:

Data Structures

- struct MeasurementObject

  Interface for retreiving measurement object information.
- struct ExtendedMeasurementObject
- struct SystemObject
- struct ObjectControl

Typedefs

- using MeasurementObjectPtr = MeasurementObject ∗

  MO pointer.
- using MeasurementObjectList = std::list< MeasurementObjectPtr >

  List of measurement objects.
- using PropertyTable = std::map< std::string, std::string >
- using PropertyPair = std::pair< std::string, std::string >

Enumerations

- enum class MeasurementObjectType : uint8_t {
  data_source , data_receiver , player , recorder ,
  system , visualization }

  Measurement object type.

7.6.1 Typedef Documentation

7.6.1.1 MeasurementObjectList `using MeasurementObjectList = std::list<MeasurementObjectPtr>`

List of measurement objects.

7.6.1.2 MeasurementObjectPtr `using MeasurementObjectPtr = MeasurementObject*`

MO pointer.

7.6.1.3 PropertyPair `using PropertyPair = std::pair<std::string, std::string>`

7.6.1.4 PropertyTable `using PropertyTable = std::map<std::string, std::string>`

7.6.2 Enumeration Type Documentation

7.6.2.1 MeasurementObjectType `enum MeasurementObjectType : uint8_t [strong]`

Measurement object type.

Enumerator

data_source	object used for transmitting data to processors
data_receiver	object used for processing data and notifying all the subscribers
player	object used for reader a recording file and act as a transmitter
recorder	object used for recording data, can be present in visualizers for recording snapshots
system	system objectr. E.g. engine, configuration manager, factory ...
visualization	visualization objects, receive data from data_receiver

7.7 include/defs/Receiver.hpp File Reference

```
#include <cstdint>
#include <defs/DataPackage.hpp>
```

Include dependency graph for Receiver.hpp:

Data Structures

- struct DataReceiverObject

  Interface used by any processor in order to receive packages from the distribution manager.

- struct NotifySubjects

  Interface used to notify all the subject registered using registerToReceiverSink method to a processor.
- struct ReceiverSinkManager

  Interface used to register a subject to a processor.

Typedefs

- using DataReceiverObjectPtr = DataReceiverObject ∗

  Data processor pointer.

7.7.1 Typedef Documentation

7.7.1.1 DataReceiverObjectPtr `using DataReceiverObjectPtr = DataReceiverObject*`

Data processor pointer.

7.8 include/defs/Transmitter.hpp File Reference

Data Structures

- struct DataSenderObject

  Interface used by any data transmitter in order to transmit packages to the distribution manager.

Typedefs

- using DataSenderObjectPtr = std::shared_ptr< DataSenderObject >

  Data transmitter pointer.

7.8.1 Typedef Documentation

7.8.1.1 DataSenderObjectPtr `using DataSenderObjectPtr = std::shared_ptr<DataSenderObject>`

Data transmitter pointer.

7.9 README.md File Reference

Index