

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	2
1.1 Distribution interfaces	2
1.2 Logging interfaces	2
1.3 MDS general purpose interfaces	2
1.4 Measurement object specific interfaces	2
1.5 Receiver interfaces	2
1.6 [Transmitter interfaces]()	2
2 Todo List	2
3 Hierarchical Index	2
3.1 Class Hierarchy	2
4 Data Structure Index	3
4.1 Data Structures	3
5 File Index	4
5.1 File List	4
6 Data Structure Documentation	4
6.1 ConfigurationFactory Struct Reference	4
6.1.1 Member Function Documentation	4
6.2 ConfigurationParser Struct Reference	5
6.2.1 Detailed Description	5
6.2.2 Member Function Documentation	5
6.3 DataDistribution Struct Reference	7
6.3.1 Detailed Description	7
6.3.2 Member Function Documentation	7
6.4 DataDistributionStatistics Struct Reference	8
6.4.1 Member Function Documentation	8
6.5 DataPackage Struct Reference	8
6.5.1 Detailed Description	9
6.5.2 Field Documentation	9
6.6 DataReceiverObject Struct Reference	10
6.6.1 Detailed Description	10
6.6.2 Member Function Documentation	10
6.7 DataSenderObject Struct Reference	11
6.7.1 Detailed Description	11
6.7.2 Member Function Documentation	11
6.8 EngineInit Struct Reference	12
6.8.1 Detailed Description	12
6.8.2 Member Function Documentation	12
6.9 ExtendedMeasurementObject Struct Reference	12
6.9.1 Member Function Documentation	13

6.10 InterfaceAccess Struct Reference	14
6.10.1 Detailed Description	14
6.10.2 Member Function Documentation	14
6.11 LoggingInterface Struct Reference	14
6.11.1 Detailed Description	15
6.11.2 Member Function Documentation	15
6.12 MeasurementObject Struct Reference	16
6.12.1 Detailed Description	17
6.12.2 Member Function Documentation	17
6.13 NotifySubjects Struct Reference	18
6.13.1 Detailed Description	18
6.13.2 Member Function Documentation	18
6.14 ObjectControl Struct Reference	19
6.14.1 Member Function Documentation	19
6.15 ReceiverSinkManager Struct Reference	19
6.15.1 Detailed Description	19
6.15.2 Member Function Documentation	19
6.16 SystemObject Struct Reference	20
6.16.1 Member Function Documentation	20
7 File Documentation	21
7.1 include/defs/Configuration.hpp File Reference	21
7.2 include/defs/DataPackage.hpp File Reference	21
7.2.1 Typedef Documentation	21
7.2.2 Enumeration Type Documentation	22
7.3 include/defs/Distribution.hpp File Reference	22
7.3.1 Typedef Documentation	22
7.4 include/defs/Log.hpp File Reference	23
7.4.1 Enumeration Type Documentation	23
7.5 include/defs/MdsInterface.hpp File Reference	23
7.5.1 Macro Definition Documentation	24
7.5.2 Enumeration Type Documentation	24
7.6 include/defs/MeasurementObjectDefs.hpp File Reference	25
7.6.1 Typedef Documentation	25
7.6.2 Enumeration Type Documentation	26
7.7 include/defs/Receiver.hpp File Reference	26
7.7.1 Typedef Documentation	27
7.8 include/defs/Transmitter.hpp File Reference	27
7.8.1 Typedef Documentation	27
7.9 README.md File Reference	27
Index	29

1 MDS definitions readme

Back

Here are presents defintions and interfaces used in the proect.

1.1 [Distribution interfaces](/include/defs/Distribution.hpp)

1.2 [Logging interfaces](/include/defs/Log.hpp)

1.3 [MDS general purpose interfaces](/include/defs/MdsInterface.hpp)

1.4 [Measurement object specific interfaces](/include/defs/MeasurementObjectDefs.hpp)

1.5 [Receiver interfaces](/include/defs/Receiver.hpp)

1.6 [Transmitter interfaces]()

For more information read the generated doxygen documentation. To generate the doxygen documentation run the doxygen having as [parameter](#) 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:

ConfigurationFactory	4
ConfigurationParser	5
DataDistribution	7
DataDistributionStatistics	8
DataPackage	8
DataReceiverObject	10
DataSenderObject	11

EngineInit	12
InterfaceAccess	14
LoggingInterface	14
MeasurementObject	16
ExtendedMeasurementObject	12
SystemObject	20
NotifySubjects	18
ObjectControl	19
ReceiverSinkManager	19

4 Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

ConfigurationFactory	4
ConfigurationParser	
Interface used to load a configurarion file	5
DataDistribution	
Data distribution interface. Let distribute a package to all linked the receivers	7
DataDistributionStatistics	8
DataPackage	
Data package definition	8
DataReceiverObject	
Interface used by any processor in order to receive packages from the distribution manager	10
DataSenderObject	
Interface used by any data transmitter in order to transmit packages to the distribution manager	11
EngineInit	
Interface used to initialize MDS Engine	12
ExtendedMeasurementObject	12
InterfaceAccess	
Interface helper that facilitate getting other interfaces	14
LoggingInterface	
Logging interface. Managed by the engine and use to report messages	14
MeasurementObject	
Interface for retrieving measurement object information	16

NotifySubjects

Interface used to notify all the subject registered using registerToReceiverSink method to a processor 18

ObjectControl 19**ReceiverSinkManager**

Interface used to register a subject to a processor 19

SystemObject 20

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/defs/ Configuration.hpp	21
include/defs/ DataPackage.hpp	21
include/defs/ Distribution.hpp	22
include/defs/ Log.hpp	23
include/defs/ MdsInterface.hpp	23
include/defs/ MeasurementObjectDefs.hpp	25
include/defs/ Receiver.hpp	26
include/defs/ Transmitter.hpp	27

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

<i>object</i>	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

<i>path</i>	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

<i>name</i>	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

<i>package</i>	Const data package that will be delivered to all the receivers.
----------------	---

Returns

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

Note

Data distribution will be performed sequentially.

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

<i>package</i>	Pointer to a const data package that will be analyzed and processed.
----------------	--

Returns

Return true if all the processors validate the package, false if any processor 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 thread that will distribute data to the data distribution manager
- virtual void [endProcessing](#) ()=0
end the processing thread 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 thread that will distribute data to the data distribution manager

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

start the processing thread 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::get↵
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

<i>interfaceName</i>	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. Unlink 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

<i>message</i>	log message
<i>handle</i>	reporter handle. If the reporter is registered using the subscribe method, then the name will appear, otherwise the handle will appear instead. If no handle is passed then the Engine will be the reporter.
<i>severity</i>	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

<i>name</i>	name of the reporter. By default the name is Engine.
<i>handle</i>	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. Unlink a name with a handle.

Note

This method should be used before logging any messages.

Parameters

<i>name</i>	name of the reporter. By default the name is Engine.
<i>handle</i>	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 retrieving measurement object information.

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

Inheritance diagram for MeasurementObject:

Public Member Functions

- virtual const uint8_t & [getInstanceNumber](#) ()=0
Method to retrieve the instance number of the measurement object.
- virtual const uint64_t & [getHandle](#) ()=0
Method to retrieve the handle of the measurement object.
- virtual const [MeasurementObjectType](#) & [getType](#) ()=0
Method to retrieve the type of the measurement object.
- virtual const std::string & [getName](#) ()=0
Method to retrieve the name of the measurement object.

6.12.1 Detailed Description

Interface for retrieving measurement object information.

6.12.2 Member Function Documentation

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

Method to retrieve 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 retrieve 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 retrieve 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 retrieve 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

<i>pkg</i>	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

<i>subject</i>	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

<i>subject</i>	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 <stdint>
#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 retrieving 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

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

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

- camera
 - DataPackage.hpp, [22](#)
- can
 - DataPackage.hpp, [22](#)
- clearMeasurementObjectList
 - ConfigurationParser, [5](#)
- clearPropertyTable
 - ExtendedMeasurementObject, [13](#)
- CONFIGURATION_MGR_HANDLE
 - MdsInterface.hpp, [24](#)
- ConfigurationFactory, [4](#)
 - getFactoryMap, [4](#)
 - getFactorySize, [5](#)
- ConfigurationParser, [5](#)
 - clearMeasurementObjectList, [5](#)
 - createMeasurementObject, [6](#)
 - getMOsAddedInConfig, [6](#)
 - loadConfiguration, [6](#)
 - removeMeasurementObject, [7](#)
- createMeasurementObject
 - ConfigurationParser, [6](#)
- critical
 - Log.hpp, [23](#)
- cycle_
 - DataPackage, [9](#)
- data_receiver
 - MeasurementObjectDefs.hpp, [26](#)
- data_source
 - MeasurementObjectDefs.hpp, [26](#)
- DataDistribution, [7](#)
 - distributeData, [7](#)
- DataDistributionPtr
 - Distribution.hpp, [22](#)
- DataDistributionStatistics, [8](#)
 - getMaximumProcessedPackagesPerSecond, [8](#)
 - getNumberOfProcessedPackagesPerSecond, [8](#)
- DataPackage, [8](#)
 - cycle_, [9](#)
 - payload, [9](#)
 - size, [9](#)
 - sourceHandle, [9](#)
 - timestamp, [9](#)
 - type, [10](#)
- DataPackage.hpp
 - camera, [22](#)
 - can, [22](#)
 - DataPackageCPtr, [21](#)
 - DataPackagePtr, [22](#)
 - dummy, [22](#)
 - ethernet, [22](#)
 - flexray, [22](#)
 - PackageType, [22](#)
- DataPackageCPtr
 - DataPackage.hpp, [21](#)
- DataPackagePtr
 - DataPackage.hpp, [22](#)
- DataReceiverObject, [10](#)
 - validatePackage, [10](#)
- DataReceiverObjectPtr
 - Receiver.hpp, [27](#)
- DataSenderObject, [11](#)
 - endProcessing, [11](#)
 - startProcessing, [11](#)
- DataSenderObjectPtr
 - Transmitter.hpp, [27](#)
- debug
 - Log.hpp, [23](#)
- distributeData
 - DataDistribution, [7](#)
- Distribution.hpp
 - DataDistributionPtr, [22](#)
- DISTRIBUTION_MGR_HANDLE
 - MdsInterface.hpp, [24](#)
- dummy
 - DataPackage.hpp, [22](#)
- endProcessing
 - DataSenderObject, [11](#)
- ENGINE_HANDLE
 - MdsInterface.hpp, [24](#)
- EngineInit, [12](#)
 - initialize, [12](#)
 - terminate, [12](#)
- EngineInitFlag
 - MdsInterface.hpp, [24](#)
- error
 - Log.hpp, [23](#)
- ethernet
 - DataPackage.hpp, [22](#)
- ExtendedMeasurementObject, [12](#)
 - clearPropertyTable, [13](#)
 - getPropertyEntryValue, [13](#)
 - getPropertyTable, [13](#)
 - hasPropertyTable, [13](#)
 - insertEntry, [13](#)
 - removeProperty, [13](#)
- FACTORY_HANDLE
 - MdsInterface.hpp, [24](#)
- flexray
 - DataPackage.hpp, [22](#)
- getFactoryMap
 - ConfigurationFactory, [4](#)
- getFactorySize
 - ConfigurationFactory, [5](#)
- getHandle
 - MeasurementObject, [17](#)
- getInstanceNumber
 - MeasurementObject, [17](#)

- getInterface
 - InterfaceAccess, 14
- getMaximumProcessedPackagesPerSecond
 - DataDistributionStatistics, 8
- getMOsAddedInConfig
 - ConfigurationParser, 6
- getName
 - MeasurementObject, 17
- getNumberOfProcessedPackagesPerSecond
 - DataDistributionStatistics, 8
- getPropertyEntryValue
 - ExtendedMeasurementObject, 13
- getPropertyTable
 - ExtendedMeasurementObject, 13
- getType
 - MeasurementObject, 17
- hasPropertyTable
 - ExtendedMeasurementObject, 13
- include/defs/Configuration.hpp, 21
- include/defs/DataPackage.hpp, 21
- include/defs/Distribution.hpp, 22
- include/defs/Log.hpp, 23
- include/defs/MdsInterface.hpp, 23
- include/defs/MeasurementObjectDefs.hpp, 25
- include/defs/Receiver.hpp, 26
- include/defs/Transmitter.hpp, 27
- information
 - Log.hpp, 23
- initialize
 - EngineInit, 12
- initializeObject
 - ObjectControl, 19
- initializeSystemObject
 - SystemObject, 20
- insertEntry
 - ExtendedMeasurementObject, 13
- InterfaceAccess, 14
 - getInterface, 14
- INVALID_HANDLE
 - MdsInterface.hpp, 24
- loadConfiguration
 - ConfigurationParser, 6
- log
 - LoggingInterface, 15
- Log.hpp
 - critical, 23
 - debug, 23
 - error, 23
 - information, 23
 - severity, 23
 - warning, 23
- LoggingInterface, 14
 - log, 15
 - save, 15
 - subscribe, 15
 - unsubscribe, 16
- MdsInterface.hpp
 - CONFIGURATION_MGR_HANDLE, 24
 - DISTRIBUTION_MGR_HANDLE, 24
 - ENGINE_HANDLE, 24
 - EngineInitFlag, 24
 - FACTORY_HANDLE, 24
 - INVALID_HANDLE, 24
 - no_exception_handler, 25
 - no_metrics, 25
 - normal, 25
 - performance, 25
 - silent, 25
 - WATCHDOG_HANDLE, 24
- MeasurementObject, 16
 - getHandle, 17
 - getInstanceNumber, 17
 - getName, 17
 - getType, 17
- MeasurementObjectDefs.hpp
 - data_receiver, 26
 - data_source, 26
 - MeasurementObjectList, 25
 - MeasurementObjectPtr, 26
 - MeasurementObjectType, 26
 - player, 26
 - PropertyPair, 26
 - PropertyTable, 26
 - recorder, 26
 - system, 26
 - visualization, 26
- MeasurementObjectList
 - MeasurementObjectDefs.hpp, 25
- MeasurementObjectPtr
 - MeasurementObjectDefs.hpp, 26
- MeasurementObjectType
 - MeasurementObjectDefs.hpp, 26
- no_exception_handler
 - MdsInterface.hpp, 25
- no_metrics
 - MdsInterface.hpp, 25
- normal
 - MdsInterface.hpp, 25
- notify
 - NotifySubjects, 18
- NotifySubjects, 18
 - notify, 18
- ObjectControl, 19
 - initializeObject, 19
 - terminateObject, 19
- PackageType
 - DataPackage.hpp, 22
- payload
 - DataPackage, 9
- performance
 - MdsInterface.hpp, 25
- player

- MeasurementObjectDefs.hpp, [26](#)
- PropertyPair
 - MeasurementObjectDefs.hpp, [26](#)
- PropertyTable
 - MeasurementObjectDefs.hpp, [26](#)
- README.md, [27](#)
- Receiver.hpp
 - DataReceiverObjectPtr, [27](#)
- ReceiverSinkManager, [19](#)
 - registerToReceiverSink, [19](#)
 - unregisterToReceiverSink, [20](#)
- recorder
 - MeasurementObjectDefs.hpp, [26](#)
- registerToReceiverSink
 - ReceiverSinkManager, [19](#)
- removeMeasurementObject
 - ConfigurationParser, [7](#)
- removeProperty
 - ExtendedMeasurementObject, [13](#)
- save
 - LoggingInterface, [15](#)
- severity
 - Log.hpp, [23](#)
- silent
 - MdsInterface.hpp, [25](#)
- size
 - DataPackage, [9](#)
- sourceHandle
 - DataPackage, [9](#)
- startProcessing
 - DataSenderObject, [11](#)
- subscribe
 - LoggingInterface, [15](#)
- system
 - MeasurementObjectDefs.hpp, [26](#)
- SystemObject, [20](#)
 - initializeSystemObject, [20](#)
 - terminateSystemObject, [20](#)
- terminate
 - EngineInit, [12](#)
- terminateObject
 - ObjectControl, [19](#)
- terminateSystemObject
 - SystemObject, [20](#)
- timestamp
 - DataPackage, [9](#)
- Transmitter.hpp
 - DataSenderObjectPtr, [27](#)
- type
 - DataPackage, [10](#)
- unregisterToReceiverSink
 - ReceiverSinkManager, [20](#)
- unsubscribe
 - LoggingInterface, [16](#)
- validatePackage
 - DataReceiverObject, [10](#)
 - visualization
 - MeasurementObjectDefs.hpp, [26](#)
 - warning
 - Log.hpp, [23](#)
 - WATCHDOG_HANDLE
 - MdsInterface.hpp, [24](#)