

artdaq_utilities
1.09.00

Generated by Doxygen 1.8.5

Thu Sep 5 2024 10:46:11

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	artdaq Namespace Reference	9
5.1.1	Detailed Description	10
5.1.2	Typedef Documentation	10
5.1.2.1	makeFunc_t	10
5.1.3	Enumeration Type Documentation	11
5.1.3.1	MetricMode	11
5.1.3.2	MetricType	11
5.1.4	Function Documentation	11
5.1.4.1	makeMetricPlugin	11
5.1.4.2	operator&	12
5.1.4.3	operator 	12
5.2	artdaqutilities Namespace Reference	12
5.2.1	Detailed Description	13
6	Class Documentation	15
6.1	artdaq::MetricManager::Config Struct Reference	15
6.1.1	Detailed Description	15

6.1.2	Member Data Documentation	15
6.1.2.1	metric_queue_notify_size	15
6.1.2.2	metric_queue_size	16
6.1.2.3	metric_send_maximum_delay_ms	16
6.2	artdaq::MetricPlugin::Config Struct Reference	16
6.2.1	Detailed Description	17
6.3	artdaq::FileMetric Class Reference	17
6.3.1	Detailed Description	18
6.3.2	Constructor & Destructor Documentation	18
6.3.2.1	FileMetric	18
6.3.3	Member Function Documentation	19
6.3.3.1	getLibName	19
6.3.3.2	sendMetric_	19
6.3.3.3	sendMetric_	19
6.3.3.4	sendMetric_	19
6.3.3.5	sendMetric_	20
6.3.3.6	sendMetric_	20
6.4	artdaqutilities::GetPackageBuildInfo Struct Reference	20
6.4.1	Detailed Description	21
6.4.2	Member Function Documentation	21
6.4.2.1	getPackageBuildInfo	21
6.5	artdaq::GraphiteMetric Class Reference	21
6.5.1	Detailed Description	22
6.5.2	Constructor & Destructor Documentation	22
6.5.2.1	GraphiteMetric	22
6.5.3	Member Function Documentation	22
6.5.3.1	getLibName	22
6.5.3.2	sendMetric_	23
6.5.3.3	sendMetric_	23
6.5.3.4	sendMetric_	23
6.5.3.5	sendMetric_	24
6.5.3.6	sendMetric_	25
6.6	artdaq::MetricData Struct Reference	25
6.6.1	Detailed Description	27
6.6.2	Constructor & Destructor Documentation	27
6.6.2.1	MetricData	27
6.6.2.2	MetricData	27

6.6.2.3	MetricData	27
6.6.2.4	MetricData	28
6.6.2.5	MetricData	28
6.6.2.6	MetricData	28
6.6.2.7	MetricData	29
6.6.2.8	MetricData	29
6.6.3	Member Function Documentation	29
6.6.3.1	Add	29
6.6.3.2	AddPoint	29
6.6.3.3	AddPoint	30
6.6.3.4	AddPoint	30
6.6.3.5	AddPoint	30
6.6.3.6	operator=	30
6.6.3.7	operator=	30
6.6.3.8	Reset	31
6.6.4	Member Data Documentation	31
6.6.4.1	DataPointCount	31
6.6.4.2	Level	31
6.6.4.3	MetricPrefix	31
6.6.4.4	Mode	31
6.6.4.5	Name	31
6.6.4.6	StringValue	32
6.6.4.7	Type	32
6.6.4.8	Unit	32
6.6.4.9	UseNameOverride	32
6.6.4.10	Value	32
6.7	artdaq::MetricData::MetricDataValue Union Reference	32
6.7.1	Detailed Description	33
6.7.2	Constructor & Destructor Documentation	33
6.7.2.1	MetricDataValue	33
6.7.2.2	MetricDataValue	33
6.7.2.3	MetricDataValue	33
6.7.2.4	MetricDataValue	33
6.7.2.5	MetricDataValue	34
6.8	artdaq::MetricManager Class Reference	34
6.8.1	Detailed Description	36
6.8.2	Constructor & Destructor Documentation	36

6.8.2.1	<code>~MetricManager</code>	36
6.8.3	Member Function Documentation	36
6.8.3.1	<code>Active</code>	36
6.8.3.2	<code>initialize</code>	36
6.8.3.3	<code>Initialized</code>	36
6.8.3.4	<code>metricManagerBusy</code>	37
6.8.3.5	<code>metricQueueEmpty</code>	37
6.8.3.6	<code>metricQueueSize</code>	37
6.8.3.7	<code>operator=</code>	37
6.8.3.8	<code>operator=</code>	37
6.8.3.9	<code>reinitialize</code>	38
6.8.3.10	<code>Running</code>	38
6.8.3.11	<code>sendMetric</code>	38
6.8.3.12	<code>sendMetric</code>	38
6.8.3.13	<code>sendMetric</code>	39
6.8.3.14	<code>sendMetric</code>	39
6.8.3.15	<code>sendMetric</code>	40
6.8.3.16	<code>setPrefix</code>	40
6.9	<code>artdaq::MetricPlugin</code> Class Reference	40
6.9.1	Detailed Description	42
6.9.2	Constructor & Destructor Documentation	42
6.9.2.1	<code>MetricPlugin</code>	42
6.9.3	Member Function Documentation	43
6.9.3.1	<code>addMetricData</code>	43
6.9.3.2	<code>IsLevelEnabled</code>	43
6.9.3.3	<code>metricsPending</code>	43
6.9.3.4	<code>sendMetric_</code>	43
6.9.3.5	<code>sendMetric_</code>	44
6.9.3.6	<code>sendMetric_</code>	45
6.9.3.7	<code>sendMetric_</code>	45
6.9.3.8	<code>sendMetric_</code>	45
6.9.3.9	<code>sendMetrics</code>	46
6.9.3.10	<code>startMetrics_</code>	46
6.9.3.11	<code>stopMetrics_</code>	46
6.10	<code>artdaqtest::MetricPluginTestAdapter</code> Class Reference	46
6.10.1	Detailed Description	48
6.10.2	Constructor & Destructor Documentation	48

6.10.2.1	MetricPluginTestAdapter	48
6.10.3	Member Function Documentation	48
6.10.3.1	get_accumulationTime_	48
6.10.3.2	get_app_name_	48
6.10.3.3	get_inhibit_	49
6.10.3.4	get_level_mask_	49
6.10.3.5	get_pset	49
6.11	artdaq::TestMetric::MetricPoint Struct Reference	49
6.11.1	Detailed Description	50
6.12	artdaq::MsgFacilityMetric Class Reference	50
6.12.1	Detailed Description	51
6.12.2	Constructor & Destructor Documentation	51
6.12.2.1	MsgFacilityMetric	51
6.12.3	Member Function Documentation	51
6.12.3.1	getLibName	51
6.12.3.2	sendMetric_	51
6.12.3.3	sendMetric_	52
6.12.3.4	sendMetric_	52
6.12.3.5	sendMetric_	52
6.12.3.6	sendMetric_	53
6.13	artdaq::PackageBuildInfo Class Reference	54
6.13.1	Detailed Description	54
6.13.2	Member Function Documentation	54
6.13.2.1	getBuildTimestamp	54
6.13.2.2	getPackageName	55
6.13.2.3	getPackageVersion	55
6.13.2.4	setBuildTimestamp	55
6.13.2.5	setPackageName	55
6.13.2.6	setPackageVersion	55
6.14	artdaq::PeriodicReportMetric Class Reference	56
6.14.1	Detailed Description	57
6.14.2	Constructor & Destructor Documentation	57
6.14.2.1	PeriodicReportMetric	57
6.14.3	Member Function Documentation	57
6.14.3.1	getLibName	57
6.14.3.2	sendMetric_	57
6.14.3.3	sendMetric_	58

6.14.3.4	sendMetric_	58
6.14.3.5	sendMetric_	58
6.14.3.6	sendMetric_	59
6.15	artdaq::ProcFileMetric Class Reference	60
6.15.1	Detailed Description	61
6.15.2	Constructor & Destructor Documentation	61
6.15.2.1	ProcFileMetric	61
6.15.3	Member Function Documentation	61
6.15.3.1	getLibName	61
6.15.3.2	sendMetric_	62
6.15.3.3	sendMetric_	63
6.15.3.4	sendMetric_	63
6.15.3.5	sendMetric_	63
6.15.3.6	sendMetric_	64
6.16	artdaq::SystemMetricCollector Class Reference	64
6.16.1	Detailed Description	65
6.16.2	Constructor & Destructor Documentation	65
6.16.2.1	SystemMetricCollector	65
6.16.3	Member Function Documentation	65
6.16.3.1	GetAvailableRAM	65
6.16.3.2	GetAvailableRAMPercent	66
6.16.3.3	GetBufferedRAM	66
6.16.3.4	GetNetworkInterfaceNames	66
6.16.3.5	GetNetworkReceiveBytes	66
6.16.3.6	GetNetworkReceiveErrors	66
6.16.3.7	GetNetworkSendBytes	67
6.16.3.8	GetNetworkSendErrors	67
6.16.3.9	GetNetworkTCPRetransSegs	67
6.16.3.10	GetProcessCPUUsagePercent	67
6.16.3.11	GetProcessMemUsage	68
6.16.3.12	GetProcessMemUsagePercent	68
6.16.3.13	GetSystemCPUUsage	68
6.16.3.14	GetTotalRAM	68
6.16.3.15	SendMetrics	68
6.17	artdaq::TestMetric Class Reference	69
6.17.1	Detailed Description	69
6.17.2	Member Function Documentation	69

6.17.2.1	LockReceivedMetricMutex	69
6.17.2.2	UnlockReceivedMetricMutex	69
6.18	artdaq::TestMetricImpl Class Reference	70
6.18.1	Detailed Description	70
6.18.2	Constructor & Destructor Documentation	71
6.18.2.1	TestMetricImpl	71
6.18.3	Member Function Documentation	72
6.18.3.1	getLibName	72
6.18.3.2	sendMetric_	72
6.18.3.3	sendMetric_	72
6.18.3.4	sendMetric_	73
6.18.3.5	sendMetric_	74
6.18.3.6	sendMetric_	74
7	File Documentation	75
7.1	artdaq_utilities/artdaq-utilities/Plugins/TestMetric.hh File Reference	75
7.1.1	Detailed Description	75
Index		76

Chapter 1

Namespace Index

1.1 Namespace List

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

artdaq	The artdaq namespace	9
artdaqutilities	Namespace used to differentiate the artdaq_utilities version of GetPackageBuildInfo from other versions present in the system	12

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

artdaq::MetricManager::Config	15
artdaq::MetricPlugin::Config	16
artdaqutilities::GetPackageBuildInfo	20
artdaq::MetricData	25
artdaq::MetricData::MetricDataValue	32
artdaq::MetricManager	34
artdaq::MetricPlugin	40
artdaq::FileMetric	17
artdaq::GraphiteMetric	21
artdaq::MsgFacilityMetric	50
artdaq::PeriodicReportMetric	56
artdaq::ProcFileMetric	60
artdaq::TestMetricImpl	70
artdaqtest::MetricPluginTestAdapter	46
artdaq::TestMetric::MetricPoint	49
artdaq::PackageBuildInfo	54
artdaq::SystemMetricCollector	64
artdaq::TestMetric	69

Chapter 3

Class Index

3.1 Class List

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

artdaq::MetricManager::Config	
The Config struct defines the accepted configuration parameters for this class	15
artdaq::MetricPlugin::Config	
The Config struct defines the accepted configuration parameters for this class	16
artdaq::FileMetric	
FileMetric writes metric data to a file on disk	17
artdaqutilities::GetPackageBuildInfo	
Wrapper around the artdaqutilities::GetPackageBuildInfo::getPackageBuildInfo function	20
artdaq::GraphiteMetric	
Send a metric to Graphite	21
artdaq::MetricData	
Small structure used to hold a metric data point before sending to the metric plugins	25
artdaq::MetricData::MetricDataValue	
This union holds the values for all other metric types	32
artdaq::MetricManager	
Handles loading metric plugins and asynchronously sending metric data to them. It is designed to be a "black hole" for metrics, taking as little time as possible so that metrics do not impact the data-taking performance	34
artdaq::MetricPlugin	
Defines the interface that MetricManager uses to send metric data to the various metric plugins	40
artdaqtest::MetricPluginTestAdapter	
Metric plugin which stores metric call counts for testing	46
artdaq::TestMetric::MetricPoint	
Describes a single metric point	49
artdaq::MsgFacilityMetric	
A MetricPlugin class which sends metric data to MessageFacility	50
artdaq::PackageBuildInfo	
Class holding information about the <i>artdaq</i> package build	54
artdaq::PeriodicReportMetric	
PeriodicReportMetric writes metric data to a file on disk	56
artdaq::ProcFileMetric	
A MetricPlugin which writes a long unsigned int metric with a given name to a given pipe	60
artdaq::SystemMetricCollector	
Collects metrics from the system, using proc filesystem or kernel API calls	64

artdaq::TestMetric	
Provides in-memory storage of metric data for testing	69
artdaq::TestMetricImpl	
TestMetric writes metric data to a statically-allocated memory block	70

Chapter 4

File Index

4.1 File List

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

artdaq_utilities/artdaq-utilities/BuildInfo/ GetPackageBuildInfo.hh	??
artdaq_utilities/artdaq-utilities/BuildInfo/ PackageBuildInfo.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ file_metric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ graphite_metric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ makeMetricPlugin.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ makeMetricPlugin.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ MetricData.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ MetricMacros.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ MetricManager.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ MetricManager.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ MetricPlugin.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ msgFacility_metric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ procFile_metric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ report_metric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ SystemMetricCollector.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ SystemMetricCollector.hh	??
artdaq_utilities/artdaq-utilities/Plugins/ test_metric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ TestMetric.cc	??
artdaq_utilities/artdaq-utilities/Plugins/ TestMetric.hh	75
artdaq_utilities/for_jenkins/ parse_product_deps.py	??
artdaq_utilities/test/Plugins/ MetricManager_t.cc	??
artdaq_utilities/test/Plugins/ MetricPlugin_t.cc	??

Chapter 5

Namespace Documentation

5.1 artdaq Namespace Reference

The artdaq namespace.

Classes

- class [PackageBuildInfo](#)
Class holding information about the artdaq package build.
- class [FileMetric](#)
FileMetric writes metric data to a file on disk.
- class [GraphiteMetric](#)
Send a metric to Graphite.
- struct [MetricData](#)
Small structure used to hold a metric data point before sending to the metric plugins
- class [MetricManager](#)
The MetricManager class handles loading metric plugins and asynchronously sending metric data to them. It is designed to be a "black hole" for metrics, taking as little time as possible so that metrics do not impact the data-taking performance.
- class [MetricPlugin](#)
The MetricPlugin class defines the interface that MetricManager uses to send metric data to the various metric plugins.
- class [MsgFacilityMetric](#)
A MetricPlugin class which sends metric data to MessageFacility.
- class [ProcFileMetric](#)
A MetricPlugin which writes a long unsigned int metric with a given name to a given pipe.
- class [PeriodicReportMetric](#)
PeriodicReportMetric writes metric data to a file on disk.
- class [SystemMetricCollector](#)
Collects metrics from the system, using proc filesystem or kernel API calls
- class [TestMetricImpl](#)
TestMetric writes metric data to a statically-allocated memory block.
- class [TestMetric](#)
Provides in-memory storage of metric data for testing

Typedefs

- typedef std::unique_ptr< [artdaq::MetricPlugin](#) > [makeFunc_t](#) (fhicl::ParameterSet const &ps, std::string const &app_name)
Make a [MetricPlugin](#) instance, loading the plugin if necessary.

Enumerations

- enum [MetricType](#) {
[MetricType::InvalidMetric](#), [MetricType::StringMetric](#), [MetricType::IntMetric](#), [MetricType::DoubleMetric](#),
[MetricType::FloatMetric](#), [MetricType::UnsignedMetric](#) }
This enumeration is used to identify the type of the metric instance (which value should be extracted from the union)
- enum [MetricMode](#) : uint32_t {
None = 0x0, [MetricMode::LastPoint](#) = 0x1, [MetricMode::Accumulate](#) = 0x2, [MetricMode::Average](#) = 0x4,
[MetricMode::Rate](#) = 0x8, [MetricMode::Minimum](#) = 0x10, [MetricMode::Maximum](#) = 0x20, [MetricMode::Persist](#) = 0x40 }
The Mode of the metric indicates how multiple metric values should be combined within a reporting interval

Functions

- std::unique_ptr< [MetricPlugin](#) > [makeMetricPlugin](#) (std::string const &generator_plugin_spec, fhicl::ParameterSet const &ps, std::string const &app_name, std::string const &metric_name)
Load a given [MetricPlugin](#) and return a pointer to it.
- constexpr [MetricMode](#) operator| ([MetricMode](#) a, [MetricMode](#) b)
Bitwise OR operator for [MetricMode](#)
- constexpr [MetricMode](#) operator& ([MetricMode](#) a, [MetricMode](#) b)
Bitwise AND operator for [MetricMode](#)

5.1.1 Detailed Description

The artdaq namespace.

5.1.2 Typedef Documentation

5.1.2.1 typedef std::unique_ptr<[artdaq::MetricPlugin](#)> [artdaq::makeFunc_t](#)(fhicl::ParameterSet const &ps, std::string const &app_name)

Make a [MetricPlugin](#) instance, loading the plugin if necessary.

Parameters

<i>ps</i>	ParameterSet used to configure the MetricPlugin instance
<i>app_name</i>	Name of the application sending metrics

Returns

A std::unique_ptr<[artdaq::MetricPlugin](#)> to the new instance

Definition at line 19 of file MetricMacros.hh.

5.1.3 Enumeration Type Documentation

5.1.3.1 enum artdaq::MetricMode : uint32_t [strong]

The Mode of the metric indicates how multiple metric values should be combined within a reporting interval

Enumerator

LastPoint Report only the last value recorded. Useful for event counters, run numbers, etc.

Accumulate Report the sum of all values. Use for counters to report accurate results.

Average Report the average of all values. Use for rates to report accurate results.

Rate over. Use to create rates from counters. Reports the sum of all values, divided by the length of the time interval they were accumulated

Minimum Reports the minimum value recorded.

Maximum Repots the maximum value recorded.

Persist Keep previous metric value in memory.

Definition at line 27 of file MetricData.hh.

5.1.3.2 enum artdaq::MetricType [strong]

This enumeration is used to identify the type of the metric instance (which value should be extraced from the union)

Enumerator

InvalidMetric Default, invalid value.

StringMetric Metric is a std::string (not in union)

IntMetric Metric is an int.

DoubleMetric Metric is a double.

FloatMetric Metric is a float.

UnsignedMetric Metric is a long unsigned int.

Definition at line 14 of file MetricData.hh.

5.1.4 Function Documentation

5.1.4.1 std::unique_ptr< artdaq::MetricPlugin > artdaq::makeMetricPlugin (std::string const & generator_plugin_spec, fhicl::ParameterSet const & ps, std::string const & app_name, std::string const & metric_name)

Load a given [MetricPlugin](#) and return a pointer to it.

Parameters

<i>generator_plugin_spec</i>	Name of the MetricPlugin
------------------------------	--

<i>ps</i>	ParameterSet with which to configure the MetricPlugin
<i>app_name</i>	Application name of the calling application
<i>metric_name</i>	Name of this MetricPlugin instance

Returns

std::unique_ptr to the new [MetricPlugin](#) instance

Definition at line 8 of file makeMetricPlugin.cc.

5.1.4.2 constexpr MetricMode artdaq::operator& (MetricMode *a*, MetricMode *b*)

Bitwise AND operator for MetricMode

Parameters

<i>a</i>	LHS of AND
<i>b</i>	RHS of AND

Returns

Logical AND of two MetricMode instances

Definition at line 55 of file MetricData.hh.

5.1.4.3 constexpr MetricMode artdaq::operator| (MetricMode *a*, MetricMode *b*)

Bitwise OR operator for MetricMode

Parameters

<i>a</i>	LHS of OR
<i>b</i>	RHS of OR

Returns

Logical OR of two MetricMode instances

Definition at line 45 of file MetricData.hh.

5.2 artdaqutilities Namespace Reference

Namespace used to differentiate the artdaq_utilities version of [GetPackageBuildInfo](#) from other versions present in the system.

Classes

- struct [GetPackageBuildInfo](#)

Wrapper around the [artdaqutilities::GetPackageBuildInfo::getPackageBuildInfo](#) function.

5.2.1 Detailed Description

Namespace used to differentiate the artdaq_utilities version of [GetPackageBuildInfo](#) from other versions present in the system.

Chapter 6

Class Documentation

6.1 artdaq::MetricManager::Config Struct Reference

The [Config](#) struct defines the accepted configuration parameters for this class.

```
#include <artdaq-utilities/Plugins/MetricManager.hh>
```

Public Attributes

- fhicl::Atom< size_t > [metric_queue_size](#)
- fhicl::Atom< size_t > [metric_queue_notify_size](#)
- fhicl::Atom< int > [metric_send_maximum_delay_ms](#)
- fhicl::Atom< bool > [send_system_metrics](#) {fhicl::Name{"send_system_metrics"}, fhicl::Comment{"Whether to collect and send system metrics such as CPU usage, Memory usage and network activity."}, false}

"send_system_metrics": (Default: false): Whether to collect and send system metrics such as CPU usage, Memory usage and network activity.

- fhicl::Atom< bool > [send_process_metrics](#) {fhicl::Name{"send_process_metrics"}, fhicl::Comment{"Whether to collect and send process CPU usage and Memory usage"}, false}

"send_process_metrics" (Default: false): Whether to collect and send process CPU usage and Memory usage

- fhicl::OptionalTable
< [artdaq::MetricPlugin::Config](#) > [metricConfig](#) {fhicl::Name{"metricConfig"}}

Example [MetricPlugin](#) Configuration.

6.1.1 Detailed Description

The [Config](#) struct defines the accepted configuration parameters for this class.

Definition at line 47 of file MetricManager.hh.

6.1.2 Member Data Documentation

6.1.2.1 fhicl::Atom<size_t> artdaq::MetricManager::Config::metric_queue_notify_size

Initial value:

```
{
    fhicl::Name{"metric_queue_notify_size"},
    fhicl::Comment{
        "The number of metric entries in the list which will cause reports of the queue size to be
        printed."},
    10}
}
```

"metric_queue_notify_size": (Default: 10): The number of metric entries in the list which will cause reports of the queue size to be printed.

Definition at line 56 of file MetricManager.hh.

6.1.2.2 fhicl::Atom<size_t> artdaq::MetricManager::Config::metric_queue_size

Initial value:

```
{
    fhicl::Name{"metric_queue_size"},
    fhicl::Comment{"The maximum number of metric entries which can be stored in the metric queue."}
    , 1000}
}
```

"metric_queue_size": (Default: 1000): The maximum number of metric entries which can be stored in the metric queue.

Definition at line 51 of file MetricManager.hh.

6.1.2.3 fhicl::Atom<int> artdaq::MetricManager::Config::metric_send_maximum_delay_ms

Initial value:

```
{
    fhicl::Name{"metric_send_maximum_delay_ms"},
    fhicl::Comment{"The maximum amount of time between metric send calls (will send 0s for metrics
    which have not "
        "reported in this interval)"},
    15000}
}
```

"metric_send_maximum_delay_ms": (Default: 15000): The maximum amount of time between metric send calls (will send 0s for metrics which have not reported in this interval)

Definition at line 63 of file MetricManager.hh.

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

- artdaq_utilities/artdaq-utilities/Plugins/MetricManager.hh

6.2 artdaq::MetricPlugin::Config Struct Reference

The [Config](#) struct defines the accepted configuration parameters for this class.

```
#include <artdaq-utilities/Plugins/MetricPlugin.hh>
```

Public Attributes

- fhicl::Atom< std::string > [metricPluginType](#) {fhicl::Name{"metricPluginType"}, fhicl::Comment{"The name of the metric plugin to load (may have additional configuration parameters)"}}

The name of the metric plugin to load (may have additional configuration parameters).

- fhicl::Atom< size_t > [level](#) {fhicl::Name{"level"}, fhicl::Comment{"The verbosity level threshold for this plugin. sendMetric calls with verbosity level greater than this will not be sent to the plugin. OPTIONAL"}, 0}
"level" (OPTIONAL): The verbosity level threshold for this plugin. sendMetric calls with verbosity level greater than this will not be sent to the plugin.
- fhicl::Sequence< size_t > [metric_levels](#) {fhicl::Name{"metric_levels"}, fhicl::Comment{"A list of levels that should be enabled for this plugin. OPTIONAL"}, std::vector<size_t>{}}
"metric_levels" (OPTIONAL): A list of levels that should be enabled for this plugin.
- fhicl::Atom< std::string > [level_string](#) {fhicl::Name{"level_string"}, fhicl::Comment{"A string containing a comma-separated list of levels to enable. Ranges are supported. Example: \"1,2,4-10,11\" OPTIONAL"}, ""}
"level_string" (OPTIONAL): A string containing a comma-separated list of levels to enable. Ranges are supported. Example: "1,2,4-10,11"
- fhicl::Atom< double > [reporting_interval](#) {fhicl::Name{"reporting_interval"}, fhicl::Comment{"How often recorded metrics are sent to the underlying metric storage"}, 15.0}
"reporting_interval" (Default: 15.0): The interval, in seconds, which the metric plugin will accumulate values for.
- fhicl::Atom< bool > [send_zeros](#) {fhicl::Name{"send_zeros"}, fhicl::Comment{"Whether zeros should be sent to the metric back-end when metrics are not reported in an interval and during shutdown"}, true}
"send_zeros" (Default: true): Whether zeros should be sent to the metric back-end when metrics are not reported in an interval and during shutdown

6.2.1 Detailed Description

The [Config](#) struct defines the accepted configuration parameters for this class.

Definition at line 47 of file MetricPlugin.hh.

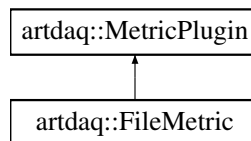
The documentation for this struct was generated from the following file:

- artdaq_utilities/artdaq-utilities/Plugins/MetricPlugin.hh

6.3 artdaq::FileMetric Class Reference

[FileMetric](#) writes metric data to a file on disk.

Inheritance diagram for artdaq::FileMetric:



Public Member Functions

- [FileMetric](#) (fhicl::ParameterSet const &config, std::string const &app_name, std::string const &metric_name)
[FileMetric](#) Constructor. Opens the file and starts the metric.
- [~FileMetric](#) () override
[FileMetric](#) Destructor. Calls stopMetrics and then closes the file.
- std::string [getLibName](#) () const override
Get the library name for the File metric.

- void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [startMetrics_](#) () override
Perform startup actions. Writes start message to output file.
- void [stopMetrics_](#) () override
Perform shutdown actions. Writes stop message to output file.

Additional Inherited Members

6.3.1 Detailed Description

[FileMetric](#) writes metric data to a file on disk.

Definition at line 27 of file `file_metric.cc`.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `artdaq::FileMetric::FileMetric (fhicl::ParameterSet const & config, std::string const & app_name, std::string const & metric_name) [inline], [explicit]`

[FileMetric](#) Constructor. Opens the file and starts the metric.

Parameters

<i>config</i>	ParameterSet used to configure FileMetric
<i>app_name</i>	Name of the application sending metrics
<i>metric_name</i>	Name of this MetricPlugin instance

```
* FileMetric accepts the following Parameters:
* "fileName" (Default: "FileMetric.out"): Name of the output file
* "absolute_file_path" (Default: true): Whether the fileName should be treated as an absolute path (default), or
* "relative_directory_env_var" (Default: ARTDAQ_LOG_ROOT): If fileName is not an absolute path (absolute_file_path
* "uniquify" (Default: false): If true, will replace %UID% with the PID of the current process, or append %UID%
* "time_format" (Default: "%c"): Format to use for time printout (see std::put_time)
* "fileMode" (Default: "append"): Set to "Overwrite" to create a new file instead of appending
```

Definition at line 73 of file `file_metric.cc`.

6.3.3 Member Function Documentation

6.3.3.1 `std::string artdaq::FileMetric::getLibName () const` `[inline]`, `[override]`, `[virtual]`

Get the library name for the File metric.

Returns

The library name for the File metric, "file"

Reimplemented from [artdaq::MetricPlugin](#).

Definition at line 131 of file `file_metric.cc`.

6.3.3.2 `void artdaq::FileMetric::sendMetric_ (const std::string & name, const std::string & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline]`, `[override]`, `[virtual]`

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 140 of file `file_metric.cc`.

6.3.3.3 `void artdaq::FileMetric::sendMetric_ (const std::string & name, const int & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline]`, `[override]`, `[virtual]`

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 155 of file `file_metric.cc`.

6.3.3.4 `void artdaq::FileMetric::sendMetric_ (const std::string & name, const double & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline]`, `[override]`, `[virtual]`

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 167 of file file_metric.cc.

6.3.3.5 void artdaq::FileMetric::sendMetric_ (const std::string & *name*, const float & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline],[override],[virtual]

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 179 of file file_metric.cc.

6.3.3.6 void artdaq::FileMetric::sendMetric_ (const std::string & *name*, const uint64_t & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline],[override],[virtual]

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 191 of file file_metric.cc.

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

- artdaq_utilities/artdaq-utilities/Plugins/file_metric.cc

6.4 artdaquilities::GetPackageBuildInfo Struct Reference

Wrapper around the [artdaqutilities::GetPackageBuildInfo::getPackageBuildInfo](#) function.

```
#include <artdaq-utilities/BuildInfo/GetPackageBuildInfo.hh>
```

Static Public Member Functions

- static [artdaq::PackageBuildInfo](#) getPackageBuildInfo ()
Gets the version number and build timestmap for artdaq_utilities.

6.4.1 Detailed Description

Wrapper around the [artdaqutilities::GetPackageBuildInfo::getPackageBuildInfo](#) function.

Definition at line 16 of file GetPackageBuildInfo.hh.

6.4.2 Member Function Documentation

6.4.2.1 static artdaq::PackageBuildInfo artdaqutilities::GetPackageBuildInfo::getPackageBuildInfo () [static]

Gets the version number and build timestmap for artdaq_utilities.

Returns

An [artdaq::PackageBuildInfo](#) object containing the version number and build timestamp for artdaq_utilities

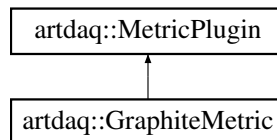
The documentation for this struct was generated from the following file:

- artdaq_utilities/artdaq-utilities/BuildInfo/GetPackageBuildInfo.hh

6.5 artdaq::GraphiteMetric Class Reference

Send a metric to Graphite.

Inheritance diagram for artdaq::GraphiteMetric:



Public Member Functions

- [GraphiteMetric](#) (fhicl::ParameterSet const &config, std::string const &app_name, std::string const &metric_name)
GraphiteMetric Constructor.
- [~GraphiteMetric](#) () override
GraphiteMetric Destructor. Calls [stopMetrics\(\)](#)
- std::string [getLibName](#) () const override
Get the library name for the Graphite metric.
- void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &, const std::chrono::system_clock::time_point &time) override
Send a metric to Graphite.
- void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to Graphite.
- void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to Graphite.

- void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to Graphite.
- void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to Graphite.
- void [startMetrics_](#) () override
Perform startup actions. For Graphite, this means reconnecting the socket.
- void [stopMetrics_](#) () override
Perform shutdown actions. This shuts down the socket and closes it.

Additional Inherited Members

6.5.1 Detailed Description

Send a metric to Graphite.

Graphite accepts metrics in a tree hierarchy, using '.' as a delimiter. Therefore, the metric artdaq.BoardReader.Fragment_Rate will appear in Graphite as: artdaq/ BoardReader/ Fragment_Rate

This plugin sends TCP messages with the following content: [name] [value] [timestamp], units are discarded

Definition at line 34 of file graphite_metric.cc.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `artdaq::GraphiteMetric::GraphiteMetric (fhicl::ParameterSet const & config, std::string const & app_name, std::string const & metric_name)` `[inline]`, `[explicit]`

[GraphiteMetric](#) Constructor.

Parameters

<i>config</i>	ParameterSet used to configure GraphiteMetric
<i>app_name</i>	Name of the application sending metrics
<i>metric_name</i>	Name of this MetricPlugin instance

```
* GraphiteMetric accepts the following Parameters:
* "host" (Default: "localhost"): Destination host
* "port" (Default: 2003): Destination port
* "namespace" (Default: "artdaq."): Directory name to prepend to all metrics. Should include the trailing '.'
*
```

Definition at line 60 of file graphite_metric.cc.

6.5.3 Member Function Documentation

6.5.3.1 `std::string artdaq::GraphiteMetric::getLibName () const` `[inline]`, `[override]`, `[virtual]`

Get the library name for the Graphite metric.

Returns

The library name for the Graphite metric, "graphite"

Reimplemented from [artdaq::MetricPlugin](#).

Definition at line 83 of file graphite_metric.cc.

6.5.3.2 `void artdaq::GraphiteMetric::sendMetric_ (const std::string & name, const std::string & value, const std::string & , const std::chrono::system_clock::time_point & time)` `[inline]`, `[override]`, `[virtual]`

Send a metric to Graphite.

Parameters

<i>name</i>	Name of the metric. Will have the namespace prepended
<i>value</i>	Value of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 91 of file graphite_metric.cc.

6.5.3.3 `void artdaq::GraphiteMetric::sendMetric_ (const std::string & name, const int & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline]`, `[override]`, `[virtual]`

Send a metric to Graphite.

Parameters

<i>name</i>	Name of the metric. Will have the namespace prepended
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric (Not used)
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 120 of file graphite_metric.cc.

6.5.3.4 `void artdaq::GraphiteMetric::sendMetric_ (const std::string & name, const double & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline]`, `[override]`, `[virtual]`

Send a metric to Graphite.

Parameters

<i>name</i>	Name of the metric. Will have the namespace prepended
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric (Not used)
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 132 of file graphite_metric.cc.

6.5.3.5 void artdaq::GraphiteMetric::sendMetric_ (const std::string & *name*, const float & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline],[override],[virtual]

Send a metric to Graphite.

Parameters

<i>name</i>	Name of the metric. Will have the namespace prepended
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric (Not used)
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 144 of file graphite_metric.cc.

```
6.5.3.6 void artdaq::GraphiteMetric::sendMetric_( const std::string & name, const uint64_t & value, const std::string & unit, const
std::chrono::system_clock::time_point & time ) [inline],[override],[virtual]
```

Send a metric to Graphite.

Parameters

<i>name</i>	Name of the metric. Will have the namespace prepended
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric (Not used)
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 156 of file graphite_metric.cc.

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

- artdaq_utilities/artdaq-utilities/Plugins/graphite_metric.cc

6.6 artdaq::MetricData Struct Reference

Small structure used to hold a metric data point before sending to the metric plugins

```
#include <artdaq-utilities/Plugins/MetricData.hh>
```

Classes

- union [MetricDataValue](#)

This union holds the values for all other metric types

Public Member Functions

- [MetricData](#) (const [MetricData](#) &r)=default
Default copy constructor
- [MetricData](#) ([MetricData](#) &&r) noexcept=default
Default move constructor
- [MetricData](#) & operator= (const [MetricData](#) &r)=default
Default copy assignment operator
- [MetricData](#) & operator= ([MetricData](#) &&r) noexcept=default
Default move assignment operator

- [MetricData](#) (std::string const &name, std::string const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix, bool useNameOverride)
Construct a [MetricData](#) point using a string value
- [MetricData](#) (std::string const &name, int const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix, bool useNameOverride)
Construct a [MetricData](#) point using a int value
- [MetricData](#) (std::string const &name, double const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix, bool useNameOverride)
Construct a [MetricData](#) point using a double value
- [MetricData](#) (std::string const &name, float const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix, bool useNameOverride)
Construct a [MetricData](#) point using a float value
- [MetricData](#) (std::string const &name, uint64_t const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix, bool useNameOverride)
Construct a [MetricData](#) point using a uint64_t value
- [MetricData](#) ()
Default constructor, constructs an [MetricType::InvalidMetric](#)
- bool [Add](#) ([MetricData](#) other)
Add two [MetricData](#) instances together
- void [AddPoint](#) (int point)
Add an integer point to this [MetricData](#)
- void [AddPoint](#) (double point)
Add a double point to this [MetricData](#)
- void [AddPoint](#) (float point)
Add a float point to this [MetricData](#)
- void [AddPoint](#) (uint64_t point)
Add an uint64_t point to this [MetricData](#)
- void [Reset](#) ()
Reset this [MetricData](#) instance to the initial state

Public Attributes

- std::string [Name](#)
Name of the metric
- std::string [StringValue](#)
Value of the metric, if it is a [MetricType::StringMetric](#)
- [MetricDataValue](#) [Value](#)
Accumulated value of this [MetricData](#)
- [MetricDataValue](#) [Last](#)
Last value of this [MetricData](#).
- [MetricDataValue](#) [Min](#)
Minimum recorded value of this [MetricData](#).
- [MetricDataValue](#) [Max](#)
Maximum recorded vaule of this [MetricData](#).
- [MetricType](#) [Type](#) {[MetricType::InvalidMetric](#)}
Type of the metric
- std::string [Unit](#)

Units of the metric

- int [Level](#)

Reporting level of the metric

- [MetricMode Mode](#)

Accumulation mode of the metric

- std::string [MetricPrefix](#)

Name prefix for the metric

- bool [UseNameOverride](#)

Whether to override the default naming convention for this metric

- size_t [DataPointCount](#) {0}

Number of data points accumulated in this [MetricData](#)

6.6.1 Detailed Description

Small structure used to hold a metric data point before sending to the metric plugins

Definition at line 63 of file MetricData.hh.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `artdaq::MetricData::MetricData (const MetricData & r)` `[default]`

Default copy constructor

Parameters

<i>r</i>	MetricData to copy
----------	------------------------------------

6.6.2.2 `artdaq::MetricData::MetricData (MetricData && r)` `[default],[noexcept]`

Default move constructor

Parameters

<i>r</i>	MetricData to move
----------	------------------------------------

6.6.2.3 `artdaq::MetricData::MetricData (std::string const & name, std::string const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix, bool useNameOverride)` `[inline]`

Construct a [MetricData](#) point using a string value

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric

<i>level</i>	Reporting level of the metric
<i>mode</i>	Accumulation mode of the metric
<i>metricPrefix</i>	Name prefix for the metric
<i>useName-Override</i>	Whether to override the default name

Definition at line 188 of file MetricData.hh.

6.6.2.4 `artdaq::MetricData::MetricData (std::string const & name, int const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix, bool useNameOverride) [inline]`

Construct a [MetricData](#) point using a int value

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>level</i>	Reporting level of the metric
<i>mode</i>	Accumulation mode of the metric
<i>metricPrefix</i>	Name prefix for the metric
<i>useName-Override</i>	Whether to override the default name

Definition at line 202 of file MetricData.hh.

6.6.2.5 `artdaq::MetricData::MetricData (std::string const & name, double const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix, bool useNameOverride) [inline]`

Construct a [MetricData](#) point using a double value

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>level</i>	Reporting level of the metric
<i>mode</i>	Accumulation mode of the metric
<i>metricPrefix</i>	Name prefix for the metric
<i>useName-Override</i>	Whether to override the default name

Definition at line 216 of file MetricData.hh.

6.6.2.6 `artdaq::MetricData::MetricData (std::string const & name, float const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix, bool useNameOverride) [inline]`

Construct a [MetricData](#) point using a float value

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>level</i>	Reporting level of the metric
<i>mode</i>	Accumulation mode of the metric
<i>metricPrefix</i>	Name prefix for the metric
<i>useName-Override</i>	Whether to override the default name

Definition at line 230 of file MetricData.hh.

6.6.2.7 `artdaq::MetricData::MetricData (std::string const & name, uint64_t const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix, bool useNameOverride) [inline]`

Construct a [MetricData](#) point using a uint64_t value

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>level</i>	Reporting level of the metric
<i>mode</i>	Accumulation mode of the metric
<i>metricPrefix</i>	Name prefix for the metric
<i>useName-Override</i>	Whether to override the default name

Definition at line 244 of file MetricData.hh.

6.6.2.8 `artdaq::MetricData::MetricData () [inline]`

Default constructor, constructs an [MetricType::InvalidMetric](#)

Definition at line 251 of file MetricData.hh.

6.6.3 Member Function Documentation

6.6.3.1 `bool artdaq::MetricData::Add (MetricData other) [inline]`

Add two [MetricData](#) instances together

Parameters

<i>other</i>	MetricData to add to this one
--------------	---

Returns

True if the other [MetricData](#) is compatible and was added, false otherwise

Definition at line 259 of file MetricData.hh.

6.6.3.2 `void artdaq::MetricData::AddPoint (int point) [inline]`

Add an integer point to this [MetricData](#)

Parameters

<i>point</i>	Int value to add
--------------	------------------

Definition at line 346 of file MetricData.hh.

6.6.3.3 void artdaq::MetricData::AddPoint (double *point*) [inline]

Add a double point to this [MetricData](#)

Parameters

<i>point</i>	Double value to add
--------------	---------------------

Definition at line 358 of file MetricData.hh.

6.6.3.4 void artdaq::MetricData::AddPoint (float *point*) [inline]

Add a float point to this [MetricData](#)

Parameters

<i>point</i>	Float value to add
--------------	--------------------

Definition at line 370 of file MetricData.hh.

6.6.3.5 void artdaq::MetricData::AddPoint (uint64_t *point*) [inline]

Add an uint64_t point to this [MetricData](#)

Parameters

<i>point</i>	uint64_t value to add
--------------	-----------------------

Definition at line 382 of file MetricData.hh.

6.6.3.6 MetricData& artdaq::MetricData::operator= (const MetricData & *r*) [default]

Default copy assignment operator

Parameters

<i>r</i>	MetricData to copy
----------	------------------------------------

Returns

[MetricData](#) copy

6.6.3.7 MetricData& artdaq::MetricData::operator= (MetricData && *r*) [default], [noexcept]

Default move assignment operator

Parameters

<i>r</i>	MetricData to move
----------	------------------------------------

Returns

[MetricData](#) reference

6.6.3.8 void artdaq::MetricData::Reset () [inline]

Reset this [MetricData](#) instance to the initial state

Sets the value, last and count fields to 0, the min to the maximum for the datatype and the max to the minimum for the datatype

Definition at line 396 of file MetricData.hh.

6.6.4 Member Data Documentation**6.6.4.1 size_t artdaq::MetricData::DataPointCount {0}**

Number of data points accumulated in this [MetricData](#)

Definition at line 176 of file MetricData.hh.

6.6.4.2 int artdaq::MetricData::Level

Reporting level of the metric

Definition at line 160 of file MetricData.hh.

6.6.4.3 std::string artdaq::MetricData::MetricPrefix

Name prefix for the metric

Definition at line 168 of file MetricData.hh.

6.6.4.4 MetricMode artdaq::MetricData::Mode

Accumulation mode of the metric

Definition at line 164 of file MetricData.hh.

6.6.4.5 std::string artdaq::MetricData::Name

Name of the metric

Definition at line 94 of file MetricData.hh.

6.6.4.6 `std::string artdaq::MetricData::StringValue`

Value of the metric, if it is a [MetricType::StringMetric](#)

Definition at line 98 of file MetricData.hh.

6.6.4.7 `MetricType artdaq::MetricData::Type {MetricType::InvalidMetric}`

Type of the metric

Definition at line 152 of file MetricData.hh.

6.6.4.8 `std::string artdaq::MetricData::Unit`

Units of the metric

Definition at line 156 of file MetricData.hh.

6.6.4.9 `bool artdaq::MetricData::UseNameOverride`

Whether to override the default naming convention for this metric

Definition at line 172 of file MetricData.hh.

6.6.4.10 `MetricDataValue artdaq::MetricData::Value`

Accumulated value of this [MetricData](#)

Definition at line 144 of file MetricData.hh.

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

- `artdaq_utilities/artdaq-utilities/Plugins/MetricData.hh`

6.7 `artdaq::MetricData::MetricDataValue` Union Reference

This union holds the values for all other metric types

```
#include <artdaq-utilities/Plugins/MetricData.hh>
```

Public Member Functions

- [MetricDataValue](#) ()
Construct a [MetricDataValue](#)
- [MetricDataValue](#) (int v)
Construct a [MetricDataValue](#) as integer
- [MetricDataValue](#) (double v)
Construct a [MetricDataValue](#) as double
- [MetricDataValue](#) (float v)
Construct a [MetricDataValue](#) as float
- [MetricDataValue](#) (uint64_t v)
Construct a [MetricDataValue](#) as unsigned int

Public Attributes

- int [i](#)
Value of the metric, if it is a [MetricType::IntMetric](#).
- double [d](#)
Value of the metric, if it is a [MetricType::DoubleMetric](#).
- float [f](#)
Value of the metric, if it is a [MetricType::FloatMetric](#).
- uint64_t [u](#)
Value of the metric, if it is a [MetricType::UnsignedMetric](#).

6.7.1 Detailed Description

This union holds the values for all other metric types

Definition at line 103 of file MetricData.hh.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 artdaq::MetricData::MetricDataValue::MetricDataValue () [\[inline\]](#)

Construct a [MetricDataValue](#)

Definition at line 113 of file MetricData.hh.

6.7.2.2 artdaq::MetricData::MetricDataValue::MetricDataValue (int v) [\[inline\]](#)

Construct a [MetricDataValue](#) as integer

Parameters

v	Integer to store
-------------------	------------------

Definition at line 119 of file MetricData.hh.

6.7.2.3 artdaq::MetricData::MetricDataValue::MetricDataValue (double v) [\[inline\]](#)

Construct a [MetricDataValue](#) as double

Parameters

v	Double to store
-------------------	-----------------

Definition at line 125 of file MetricData.hh.

6.7.2.4 artdaq::MetricData::MetricDataValue::MetricDataValue (float v) [\[inline\]](#)

Construct a [MetricDataValue](#) as float

Parameters

v	Float to store
-------------------	----------------

Definition at line 131 of file MetricData.hh.

6.7.2.5 `artdaq::MetricData::MetricDataValue::MetricDataValue (uint64_t v) [inline]`

Construct a [MetricDataValue](#) as unsigned int

Parameters

v	Unsigned int to store
-------------------	-----------------------

Definition at line 137 of file MetricData.hh.

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

- `artdaq_utilities/artdaq-utilities/Plugins/MetricData.hh`

6.8 `artdaq::MetricManager` Class Reference

The [MetricManager](#) class handles loading metric plugins and asynchronously sending metric data to them. It is designed to be a "black hole" for metrics, taking as little time as possible so that metrics do not impact the data-taking performance.

```
#include <artdaq-utilities/Plugins/MetricManager.hh>
```

Classes

- struct [Config](#)
The [Config](#) struct defines the accepted configuration parameters for this class.

Public Types

- using [Parameters](#) = `fhicl::WrappedTable< Config >`
Used for [ParameterSet](#) validation (if desired)

Public Member Functions

- [MetricManager](#) ()
Construct an instance of the [MetricManager](#) class.
- [MetricManager](#) ([MetricManager](#) const &)=delete
Copy Constructor is deleted.
- virtual [~MetricManager](#) () noexcept
[MetricManager](#) destructor.
- [MetricManager](#) & [operator=](#) ([MetricManager](#) const &)=delete
Copy Assignment operator is deleted.
- [MetricManager](#) ([MetricManager](#) &&)=delete
Move Constructor is deleted.
- [MetricManager](#) & [operator=](#) ([MetricManager](#) &&)=delete

Move assignment operator is deleted.

- void [initialize](#) (fhicl::ParameterSet const &pset, std::string const &prefix="")
Initialize the [MetricPlugin](#) instances.
- void [do_start](#) ()
Perform startup actions for each configured [MetricPlugin](#).
- void [do_stop](#) ()
Stop sending metrics to the [MetricPlugin](#) instances.
- void [do_pause](#) ()
Pause metric sending. Currently a No-Op.
- void [do_resume](#) ()
Resume metric sending. Currently a No-Op.
- void [reinitialize](#) (fhicl::ParameterSet const &pset, std::string const &prefix="")
Reinitialize all [MetricPlugin](#) Instances.
- void [shutdown](#) ()
Call the destructors for all configured [MetricPlugin](#) instances.
- void [sendMetric](#) (std::string const &name, std::string const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix="", bool useNameOverride=false)
Send a metric with the given parameters to any MetricPlugins with a threshold level \geq to level.
- void [sendMetric](#) (std::string const &name, int const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix="", bool useNameOverride=false)
Send a metric with the given parameters to any MetricPlugins with a threshold level \geq to level.
- void [sendMetric](#) (std::string const &name, double const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix="", bool useNameOverride=false)
Send a metric with the given parameters to any MetricPlugins with a threshold level \geq to level.
- void [sendMetric](#) (std::string const &name, float const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix="", bool useNameOverride=false)
Send a metric with the given parameters to any MetricPlugins with a threshold level \geq to level.
- void [sendMetric](#) (std::string const &name, uint64_t const &value, std::string const &unit, int level, [MetricMode](#) mode, std::string const &metricPrefix="", bool useNameOverride=false)
Send a metric with the given parameters to any MetricPlugins with a threshold level \geq to level.
- void [setPrefix](#) (std::string const &prefix)
Sets the prefix prepended to all metrics without useNameOverride set.
- bool [Initialized](#) ()
Returns whether the [MetricManager](#) has been initialized (configured)
- bool [Running](#) ()
Returns whether the [MetricManager](#) is running (accepting metric calls)
- bool [Active](#) ()
Returns whether any Metric Plugins are defined and configured
- bool [metricQueueEmpty](#) ()
Returns whether the metric queue is completely empty
- bool [metricManagerBusy](#) ()
Determine whether the [MetricManager](#) or any of its plugins are currently processing metrics. Used for MetricManager_t
- size_t [metricQueueSize](#) (std::string const &name="")
Return the size of the named metric queue

6.8.1 Detailed Description

The [MetricManager](#) class handles loading metric plugins and asynchronously sending metric data to them. It is designed to be a "black hole" for metrics, taking as little time as possible so that metrics do not impact the data-taking performance.

Definition at line 41 of file MetricManager.hh.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `artdaq::MetricManager::~MetricManager () [virtual], [noexcept]`

[MetricManager](#) destructor.

Calls [shutdown\(\)](#)

Definition at line 34 of file MetricManager.cc.

6.8.3 Member Function Documentation

6.8.3.1 `bool artdaq::MetricManager::Active () [inline]`

Returns whether any Metric Plugins are defined and configured

Returns

True if a Metric Plugin can accept metrics

Definition at line 262 of file MetricManager.hh.

6.8.3.2 `void artdaq::MetricManager::initialize (fhicl::ParameterSet const & pset, std::string const & prefix = " ")`

Initialize the [MetricPlugin](#) instances.

Parameters

<i>pset</i>	The ParameterSet used to configure the MetricPlugin instances
<i>prefix</i>	The prefix to prepend to all metric names, unless <code>useNameOverride</code> is set to true

The ParameterSet should be a collection of tables, each configuring a [MetricPlugin](#). See the [MetricPlugin](#) documentation for how to configure a [MetricPlugin](#). "metric_queue_size": (Default: 1000): The maximum number of metric entries which can be stored in the metric queue. If the queue is above this size, new metric entries will be dropped until the plugins catch up. "metric_queue_notify_size": (Default: 10): The number of metric entries in the list which will cause reports of the queue size to be printed. "metric_send_maximum_delay_ms": (Default: 15000): The maximum amount of time between metric send calls (will send 0s for metrics which have not reported in this interval) "metric_holdoff_us": (Default: 1000): Amount of time, in microseconds, to delay sending metrics after each `sendMetric` call (to ensure that multiple associated calls are in the same metrics interval)

Definition at line 36 of file MetricManager.cc.

6.8.3.3 `bool artdaq::MetricManager::Initialized () [inline]`

Returns whether the [MetricManager](#) has been initialized (configured)

Returns

True if [MetricManager](#) is initialized

Definition at line 250 of file MetricManager.hh.

6.8.3.4 bool artdaq::MetricManager::metricManagerBusy ()

Determine whether the [MetricManager](#) or any of its plugins are currently processing metrics. Used for MetricManager_t

Returns

True if [MetricManager](#) or one of its MetricPlugins are currently processing metrics.

Definition at line 539 of file MetricManager.cc.

6.8.3.5 bool artdaq::MetricManager::metricQueueEmpty ()

Returns whether the metric queue is completely empty

Returns

True if the metric queue is empty

Definition at line 525 of file MetricManager.cc.

6.8.3.6 size_t artdaq::MetricManager::metricQueueSize (std::string const & name = " ")

Return the size of the named metric queue

Parameters

<i>name</i>	Name of the metric queue to query. "" returns size of all queues (default)
-------------	--

Returns

Size of selected metric queue

Definition at line 556 of file MetricManager.cc.

6.8.3.7 MetricManager& artdaq::MetricManager::operator= (MetricManager const &) [delete]

Copy Assignment operator is deleted.

Returns

[MetricManager](#) copy

6.8.3.8 MetricManager& artdaq::MetricManager::operator= (MetricManager &&) [delete]

Move assignment operator is deleted.

Returns

Moved [MetricManager](#)

6.8.3.9 `void artdaq::MetricManager::reinitialize (fhicl::ParameterSet const & pset, std::string const & prefix = " ")`

Reinitialize all [MetricPlugin](#) Instances.

Parameters

<i>pset</i>	ParameterSet used to configure the MetricPlugin instances
<i>prefix</i>	Prefix to apply to Metric names

Calls [shutdown\(\)](#), then `initialize(pset, prefix)`.

Definition at line 185 of file `MetricManager.cc`.

6.8.3.10 `bool artdaq::MetricManager::Running () [inline]`

Returns whether the [MetricManager](#) is running (accepting metric calls)

Returns

True if [MetricManager](#) is running

Definition at line 256 of file `MetricManager.hh`.

6.8.3.11 `void artdaq::MetricManager::sendMetric (std::string const & name, std::string const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix = " ", bool useNameOverride = false)`

Send a metric with the given parameters to any [MetricPlugins](#) with a threshold level \geq to level.

Parameters

<i>name</i>	The Name of the metric
<i>value</i>	The value of the metric
<i>unit</i>	The units of the metric
<i>level</i>	The verbosity level of the metric. Higher number == more verbose
<i>mode</i>	The MetricMode that the metric should operate in. Options are: LastPoint: Every reporting_ _interval, the latest metric value is sent (For run/event numbers, etc) Accumulate: Every reporting_ _interval, the sum of all metric values since the last report is sent (for counters) Average: Every reporting_ _interval, the average of all metric values since the last report is sent (for rates)
<i>metricPrefix</i>	An additional prefix to prepend to the metric name
<i>useName- Override</i>	Whether to use name verbatim and not apply prefixes

Definition at line 219 of file `MetricManager.cc`.

6.8.3.12 `void artdaq::MetricManager::sendMetric (std::string const & name, int const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix = " ", bool useNameOverride = false)`

Send a metric with the given parameters to any [MetricPlugins](#) with a threshold level \geq to level.

Parameters

<i>name</i>	The Name of the metric
-------------	------------------------

<i>value</i>	The value of the metric
<i>unit</i>	The units of the metric
<i>level</i>	The verbosity level of the metric. Higher number == more verbose
<i>mode</i>	The MetricMode that the metric should operate in. Options are: LastPoint: Every reporting_ _interval, the latest metric value is sent (For run/event numbers, etc) Accumulate: Every reporting_ _interval, the sum of all metric values since the last report is sent (for counters) Average: Every reporting_ _interval, the average of all metric values since the last report is sent (for rates)
<i>metricPrefix</i>	An additional prefix to prepend to the metric name
<i>useName- Override</i>	Whether to use name verbatim and not apply prefixes

Definition at line 282 of file MetricManager.cc.

6.8.3.13 void artdaq::MetricManager::sendMetric (std::string const & *name*, double const & *value*, std::string const & *unit*, int *level*, MetricMode *mode*, std::string const & *metricPrefix* = " ", bool *useNameOverride* = false)

Send a metric with the given parameters to any MetricPlugins with a threshold level >= to level.

Parameters

<i>name</i>	The Name of the metric
<i>value</i>	The value of the metric
<i>unit</i>	The units of the metric
<i>level</i>	The verbosity level of the metric. Higher number == more verbose
<i>mode</i>	The MetricMode that the metric should operate in. Options are: LastPoint: Every reporting_ _interval, the latest metric value is sent (For run/event numbers, etc) Accumulate: Every reporting_ _interval, the sum of all metric values since the last report is sent (for counters) Average: Every reporting_ _interval, the average of all metric values since the last report is sent (for rates)
<i>metricPrefix</i>	An additional prefix to prepend to the metric name
<i>useName- Override</i>	Whether to use name verbatim and not apply prefixes

Definition at line 335 of file MetricManager.cc.

6.8.3.14 void artdaq::MetricManager::sendMetric (std::string const & *name*, float const & *value*, std::string const & *unit*, int *level*, MetricMode *mode*, std::string const & *metricPrefix* = " ", bool *useNameOverride* = false)

Send a metric with the given parameters to any MetricPlugins with a threshold level >= to level.

Parameters

<i>name</i>	The Name of the metric
<i>value</i>	The value of the metric
<i>unit</i>	The units of the metric
<i>level</i>	The verbosity level of the metric. Higher number == more verbose
<i>mode</i>	The MetricMode that the metric should operate in. Options are: LastPoint: Every reporting_ _interval, the latest metric value is sent (For run/event numbers, etc) Accumulate: Every reporting_ _interval, the sum of all metric values since the last report is sent (for counters) Average: Every reporting_ _interval, the average of all metric values since the last report is sent (for rates)

<i>metricPrefix</i>	An additional prefix to prepend to the metric name
<i>useName-Override</i>	Whether to use name verbatim and not apply prefixes

Definition at line 388 of file MetricManager.cc.

6.8.3.15 `void artdaq::MetricManager::sendMetric (std::string const & name, uint64_t const & value, std::string const & unit, int level, MetricMode mode, std::string const & metricPrefix = " ", bool useNameOverride = false)`

Send a metric with the given parameters to any MetricPlugins with a threshold level \geq to level.

Parameters

<i>name</i>	The Name of the metric
<i>value</i>	The value of the metric
<i>unit</i>	The units of the metric
<i>level</i>	The verbosity level of the metric. Higher number == more verbose
<i>mode</i>	The MetricMode that the metric should operate in. Options are: LastPoint: Every reporting_interval, the latest metric value is sent (For run/event numbers, etc) Accumulate: Every reporting_interval, the sum of all metric values since the last report is sent (for counters) Average: Every reporting_interval, the average of all metric values since the last report is sent (for rates)
<i>metricPrefix</i>	An additional prefix to prepend to the metric name
<i>useName-Override</i>	Whether to use name verbatim and not apply prefixes

Definition at line 441 of file MetricManager.cc.

6.8.3.16 `void artdaq::MetricManager::setPrefix (std::string const & prefix) [inline]`

Sets the prefix prepended to all metrics without useNameOverride set.

Parameters

<i>prefix</i>	The prefix to prepend. Delimiter character in names is "."
---------------	--

Definition at line 244 of file MetricManager.hh.

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

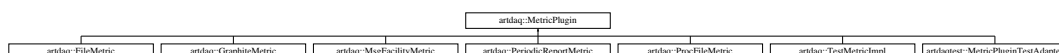
- artdaq_utilities/artdaq-utilities/Plugins/MetricManager.hh
- artdaq_utilities/artdaq-utilities/Plugins/MetricManager.cc

6.9 artdaq::MetricPlugin Class Reference

The [MetricPlugin](#) class defines the interface that [MetricManager](#) uses to send metric data to the various metric plugins.

```
#include <artdaq-utilities/Plugins/MetricPlugin.hh>
```

Inheritance diagram for artdaq::MetricPlugin:



Classes

- struct [Config](#)

The [Config](#) struct defines the accepted configuration parameters for this class.

Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >

Used for [ParameterSet](#) validation (if desired)

Public Member Functions

- [MetricPlugin](#) (fhicl::ParameterSet const &ps, std::string const &app_name, std::string const &metric_name)
MetricPlugin Constructor.
- virtual [~MetricPlugin](#) ()=default
Default virtual Destructor.
- virtual std::string [getLibName](#) () const
Return the name of the current MetricPlugin instance.
- void [addMetricData](#) (std::unique_ptr< [MetricData](#) > const &data)
Send a metric value to the MetricPlugin.
- void [sendMetrics](#) (bool forceSend=false, std::chrono::steady_clock::time_point interval_end=std::chrono::steady_clock::now())
For each known metric, determine whether the reporting interval has elapsed, and if so, report a value to the underlying metric storage.
- void [startMetrics](#) ()
Perform startup actions. Simply calls the virtual startMetrics_ function.
- void [stopMetrics](#) ()
Perform shutdown actions. Zeroes out all accumulators, and sends zeros for each metric. Calls stopMetrics_() for any plugin-defined shutdown actions.
- bool [IsLevelEnabled](#) (int level)
Determine if the given level is enabled for this MetricPlugin instance.
- bool [metricsPending](#) ()
Determine if metrics are waiting to be sent.

Protected Member Functions

- virtual void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &unit, const std::chrono::system_clock::time_point &interval_end)=0
Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)
- virtual void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &interval_end)=0
Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)
- virtual void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &interval_end)=0
Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)
- virtual void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &interval_end)=0
Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)

- virtual void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &interval_end)=0
Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)
- virtual void [startMetrics_](#) ()=0
Perform any start-up actions necessary for the metric plugin.
- virtual void [stopMetrics_](#) ()=0
Perform any shutdown actions necessary for the metric plugin.

Protected Attributes

- fhicl::ParameterSet [pset](#)
The ParameterSet used to configure the [MetricPlugin](#).
- double [accumulationTime_](#)
The amount of time to average metric values; except for accumulate=false metrics, will be the interval at which each metric is sent.
- std::string [app_name_](#)
Name of the application which is sending metrics to this plugin.
- std::string [metric_name_](#)
Name of this [MetricPlugin](#) instance.
- bool [inhibit_](#)
Flag to indicate that the [MetricPlugin](#) is being stopped, and any metric back-ends which do not have a persistent state (i.e. file) should not report further metrics.
- std::bitset< 64 > [level_mask_](#)
Bitset indicating for each possible metric level, whether this plugin will receive those metrics.
- bool [sendZeros_](#)
Whether zeros should be sent to this metric backend when metric instances are missing or at the end of the run.

6.9.1 Detailed Description

The [MetricPlugin](#) class defines the interface that [MetricManager](#) uses to send metric data to the various metric plugins. Definition at line 41 of file MetricPlugin.hh.

6.9.2 Constructor & Destructor Documentation

- 6.9.2.1 `artdaq::MetricPlugin::MetricPlugin (fhicl::ParameterSet const & ps, std::string const & app_name, std::string const & metric_name) [inline], [explicit]`

[MetricPlugin](#) Constructor.

Parameters

<i>ps</i>	The ParameterSet used to configure this MetricPlugin instance
<i>app_name</i>	The Application name which can be used by the Metric Plugin for identification
<i>metric_name</i>	Name for this MetricPlugin instance

Calling sendMetric with the accumulate parameter set to false will bypass this accumulation and directly send the metric. String metrics cannot be accumulated.

Definition at line 74 of file MetricPlugin.hh.

6.9.3 Member Function Documentation

6.9.3.1 void artdaq::MetricPlugin::addMetricData (std::unique_ptr< MetricData > const & data) [inline]

Send a metric value to the [MetricPlugin](#).

Parameters

<i>data</i>	A MetricData struct containing the metric value
-------------	---

Definition at line 235 of file MetricPlugin.hh.

6.9.3.2 bool artdaq::MetricPlugin::IsLevelEnabled (int level) [inline]

Determine if the given level is enabled for this [MetricPlugin](#) instance.

Parameters

<i>level</i>	Level to check
--------------	----------------

Returns

True if level is enabled, false otherwise

Definition at line 406 of file MetricPlugin.hh.

6.9.3.3 bool artdaq::MetricPlugin::metricsPending () [inline]

Determine if metrics are waiting to be sent.

Returns

True if metrics have been queued for sending by this [MetricPlugin](#) instance

Definition at line 417 of file MetricPlugin.hh.

6.9.3.4 virtual void artdaq::MetricPlugin::sendMetric_ (const std::string & name, const std::string & value, const std::string & unit, const std::chrono::system_clock::time_point & interval_end) [protected],[pure virtual]

Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units for the metric
<i>interval_end</i>	End point of the aggregation interval

Note this is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::GraphiteMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

6.9.3.5 `virtual void artdaq::MetricPlugin::sendMetric_ (const std::string & name, const int & value, const std::string & unit, const std::chrono::system_clock::time_point & interval_end)` `[protected], [pure virtual]`

Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units for the metric
<i>interval_end</i>	End point of the aggregation interval

Note this is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::GraphiteMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

6.9.3.6 `virtual void artdaq::MetricPlugin::sendMetric_ (const std::string & name, const double & value, const std::string & unit, const std::chrono::system_clock::time_point & interval_end) [protected], [pure virtual]`

Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units for the metric
<i>interval_end</i>	End point of the aggregation interval

Note this is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::GraphiteMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

6.9.3.7 `virtual void artdaq::MetricPlugin::sendMetric_ (const std::string & name, const float & value, const std::string & unit, const std::chrono::system_clock::time_point & interval_end) [protected], [pure virtual]`

Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units for the metric
<i>interval_end</i>	End point of the aggregation interval

Note this is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::GraphiteMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

6.9.3.8 `virtual void artdaq::MetricPlugin::sendMetric_ (const std::string & name, const uint64_t & value, const std::string & unit, const std::chrono::system_clock::time_point & interval_end) [protected], [pure virtual]`

Send a metric to the underlying metric storage (file, Graphite, Ganglia, etc.)

Parameters

<i>name</i>	Name of the metric
-------------	--------------------

<i>value</i>	Value of the metric
<i>unit</i>	Units for the metric
<i>interval_end</i>	End point of the aggregation interval

Note this is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::GraphiteMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

```
6.9.3.9 void artdaq::MetricPlugin::sendMetrics ( bool forceSend = false, std::chrono::steady_clock::time_point interval_end =
std::chrono::steady_clock::now() ) [inline]
```

For each known metric, determine whether the reporting interval has elapsed, and if so, report a value to the underlying metric storage.

Parameters

<i>forceSend</i>	(Default = false): Force sending metrics, even if reporting interval has not elapsed
<i>interval_end</i>	(Default = now): For calculating rates, when the current reporting interval ended (interval began at last value of <i>interval_end</i>)

Definition at line 262 of file MetricPlugin.hh.

```
6.9.3.10 virtual void artdaq::MetricPlugin::startMetrics_( ) [protected],[pure virtual]
```

Perform any start-up actions necessary for the metric plugin.

This is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::GraphiteMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

```
6.9.3.11 virtual void artdaq::MetricPlugin::stopMetrics_( ) [protected],[pure virtual]
```

Perform any shutdown actions necessary for the metric plugin.

This is a pure virtual function, it should be overridden by implementation plugins

Implemented in [artdaq::FileMetric](#), [artdaq::GraphiteMetric](#), [artdaq::MsgFacilityMetric](#), [artdaq::ProcFileMetric](#), [artdaq::PeriodicReportMetric](#), [artdaq::TestMetricImpl](#), and [artdaqtest::MetricPluginTestAdapter](#).

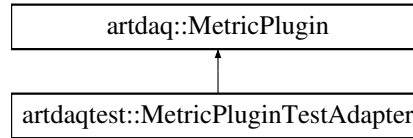
The documentation for this class was generated from the following file:

- [artdaq_utilities/artdaq_utilities/Plugins/MetricPlugin.hh](#)

6.10 artdaqtest::MetricPluginTestAdapter Class Reference

Metric plugin which stores metric call counts for testing

Inheritance diagram for [artdaqtest::MetricPluginTestAdapter](#):



Public Member Functions

- [MetricPluginTestAdapter](#) (fhicl::ParameterSet ps)
Constructor
- virtual void [sendMetric_](#) (const std::string &, const std::string &, const std::string &, const std::chrono::system_clock::time_point &) override
Send a String metric, record the call and discard the metric's data.
- virtual void [sendMetric_](#) (const std::string &, const int &, const std::string &, const std::chrono::system_clock::time_point &) override
Send an int metric, record the call and discard the metric's data.
- virtual void [sendMetric_](#) (const std::string &, const double &, const std::string &, const std::chrono::system_clock::time_point &) override
Send a double metric, record the call and discard the metric's data.
- virtual void [sendMetric_](#) (const std::string &, const float &, const std::string &, const std::chrono::system_clock::time_point &) override
Send a float metric, record the call and discard the metric's data.
- virtual void [sendMetric_](#) (const std::string &, const uint64_t &, const std::string &, const std::chrono::system_clock::time_point &) override
Send an unsigned metric, record the call and discard the metric's data.
- void [startMetrics_](#) () override
Record that a startMetrics call was received.
- void [stopMetrics_](#) () override
Record that a stopMetrics call was received.
- fhicl::ParameterSet [get_pset](#) ()
Get the ParameterSet used to configure the plugin
- double [get_accumulationTime_](#) ()
Get the MetricPlugin's accumulationTime variable
- std::string [get_app_name_](#) ()
Get the MetricPlugin's app_name_ variable
- bool [get_inhibit_](#) ()
Get the MetricPlugin's inhibit_ variable
- std::bitset< 64 > [get_level_mask_](#) ()
Get the MetricPlugin's level_mask_ variable

Public Attributes

- size_t [sendMetric_string_calls](#)
The number of string metric calls received.
- size_t [sendMetric_int_calls](#)
The number of int metric calls received.
- size_t [sendMetric_double_calls](#)

The number of double metric calls received.

- size_t [sendMetric_float_calls](#)

The number of float metric calls received.

- size_t [sendMetric_unsigned_calls](#)

The number of unsigned metric calls received.

- size_t [startMetrics_calls](#)

The number of startMetrics_ calls received.

- size_t [stopMetrics_calls](#)

The number of stopMetrics_ calls received.

Additional Inherited Members

6.10.1 Detailed Description

Metric plugin which stores metric call counts for testing

Definition at line 14 of file MetricPlugin_t.cc.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `artdaqtest::MetricPluginTestAdapter::MetricPluginTestAdapter (fhicl::ParameterSet ps)` `[inline]`, `[explicit]`

Constructor

Parameters

<i>ps</i>	fhicl::ParameterSet used to configure MetricPLugin
-----------	--

Definition at line 21 of file MetricPlugin_t.cc.

6.10.3 Member Function Documentation

6.10.3.1 `double artdaqtest::MetricPluginTestAdapter::get_accumulationTime_ ()` `[inline]`

Get the MetricPlugin's accumulationTime variable

Returns

The accumulationTime variable from the MetricPlugin

Definition at line 80 of file MetricPlugin_t.cc.

6.10.3.2 `std::string artdaqtest::MetricPluginTestAdapter::get_app_name_ ()` `[inline]`

Get the MetricPlugin's app_name_ variable

Returns

The app_name_ variable from the MetricPlugin

Definition at line 85 of file MetricPlugin_t.cc.

6.10.3.3 `bool artdaqtest::MetricPluginTestAdapter::get_inhibit_ () [inline]`

Get the MetricPlugin's inhibit_ variable

Returns

The inhibit_ variable from the MetricPlugin

Definition at line 90 of file MetricPlugin_t.cc.

6.10.3.4 `std::bitset<64> artdaqtest::MetricPluginTestAdapter::get_level_mask_ () [inline]`

Get the MetricPlugin's level_mask_ variable

Returns

The level_mask_ variable from the MetricPlugin

Definition at line 95 of file MetricPlugin_t.cc.

6.10.3.5 `fhicl::ParameterSet artdaqtest::MetricPluginTestAdapter::get_pset () [inline]`

Get the ParameterSet used to configure the plugin

Returns

The ParameterSet used to configure the plugin

Definition at line 75 of file MetricPlugin_t.cc.

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

- artdaq_utilities/test/Plugins/MetricPlugin_t.cc

6.11 artdaq::TestMetric::MetricPoint Struct Reference

Describes a single metric point

```
#include <artdaq_utilities/Plugins/TestMetric.hh>
```

Public Attributes

- `std::chrono::system_clock::time_point` [sent_time](#)
When the metric was received.
- `std::string` [metric](#)
Name of the metric.
- `std::string` [value](#)
Value of the metric.
- `std::string` [unit](#)
Units for the metric.

6.11.1 Detailed Description

Describes a single metric point

Definition at line 25 of file TestMetric.hh.

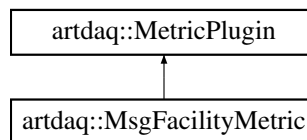
The documentation for this struct was generated from the following file:

- [artdaq_utilities/artdaq-utilities/Plugins/TestMetric.hh](#)

6.12 artdaq::MsgFacilityMetric Class Reference

A [MetricPlugin](#) class which sends metric data to MessageFacility.

Inheritance diagram for artdaq::MsgFacilityMetric:



Public Member Functions

- [MsgFacilityMetric](#) (fhicl::ParameterSet const &config, std::string const &app_name, std::string const &metric_name)
MsgFacilityMetric Constructor.
- [~MsgFacilityMetric](#) () override
MsgFacilityMetric Destructor. Calls stopMetrics()
- std::string [getLibName](#) () const override
Return the library name of the MetricPlugin.
- void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &unit, const std::chrono::system_clock::time_point &) override
Send a metric to MessageFacility. Format is: "name: value unit."
- void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to MessageFacility. All metrics are converted to strings.
- void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to MessageFacility. All metrics are converted to strings.
- void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to MessageFacility. All metrics are converted to strings.
- void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Send a metric to MessageFacility. All metrics are converted to strings.
- void [startMetrics_](#) () override
Perform startup actions. No-Op.
- void [stopMetrics_](#) () override
Perform shutdown actions. No-Op.

Additional Inherited Members

6.12.1 Detailed Description

A [MetricPlugin](#) class which sends metric data to MessageFacility.

Definition at line 22 of file msgFacility_metric.cc.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `artdaq::MsgFacilityMetric::MsgFacilityMetric (fhicl::ParameterSet const & config, std::string const & app_name, std::string const & metric_name) [inline], [explicit]`

[MsgFacilityMetric](#) Constructor.

Parameters

<i>config</i>	ParameterSet used to configure MsgFacilityMetric
<i>app_name</i>	Name of the application sending metrics
<i>metric_name</i>	Name of this MetricPlugin instance

MsgFacilityMetric accepts the following Parameters:

"output_message_category_name" (Default: "ARTDAQ Metric"): Name of the "category" (for filtering) in MessageFacility.
 "output_message_severity" (Default: 0): Severity which messages should be sent with. This parameter may also be specified as the string name of the severity.
 0: Info, 1: Debug, 2: Warning, 3: Error
 *

Definition at line 48 of file msgFacility_metric.cc.

6.12.3 Member Function Documentation

6.12.3.1 `std::string artdaq::MsgFacilityMetric::getLibName () const [inline], [override], [virtual]`

Return the library name of the [MetricPlugin](#).

Returns

The library name of [MsgFacilityMetric](#): "msgFacility"

Reimplemented from [artdaq::MetricPlugin](#).

Definition at line 88 of file msgFacility_metric.cc.

6.12.3.2 `void artdaq::MsgFacilityMetric::sendMetric_ (const std::string & name, const std::string & value, const std::string & unit, const std::chrono::system_clock::time_point &) [inline], [override], [virtual]`

Send a metric to MessageFacility. Format is: "name: value unit."

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units for the metric

Not using the time field, as MessageFacility will put its own timestamp on

Implements [artdaq::MetricPlugin](#).

Definition at line 98 of file msgFacility_metric.cc.

6.12.3.3 `void artdaq::MsgFacilityMetric::sendMetric_(const std::string & name, const int & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline],[override],[virtual]`

Send a metric to MessageFacility. All metrics are converted to strings.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 127 of file msgFacility_metric.cc.

6.12.3.4 `void artdaq::MsgFacilityMetric::sendMetric_(const std::string & name, const double & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline],[override],[virtual]`

Send a metric to MessageFacility. All metrics are converted to strings.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 139 of file msgFacility_metric.cc.

6.12.3.5 `void artdaq::MsgFacilityMetric::sendMetric_(const std::string & name, const float & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline],[override],[virtual]`

Send a metric to MessageFacility. All metrics are converted to strings.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 151 of file msgFacility_metric.cc.

6.12.3.6 void artdaq::MsgFacilityMetric::sendMetric_(const std::string & *name*, const uint64_t & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline],[override],[virtual]

Send a metric to MessageFacility. All metrics are converted to strings.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 163 of file msgFacility_metric.cc.

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

- artdaq_utilities/artdaq-utilities/Plugins/msgFacility_metric.cc

6.13 artdaq::PackageBuildInfo Class Reference

Class holding information about the *artdaq* package build.

```
#include <artdaq-utilities/BuildInfo/PackageBuildInfo.hh>
```

Public Member Functions

- [PackageBuildInfo](#) ()
Default Constructor.
- std::string [getPackageName](#) () const
Gets the package name.
- std::string [getPackageVersion](#) () const
Gets the package version.
- std::string [getBuildTimestamp](#) () const
Gets the build timestamp.
- void [setPackageName](#) (std::string str)
Sets the package name.
- void [setPackageVersion](#) (std::string str)
Sets the package version.
- void [setBuildTimestamp](#) (std::string str)
Sets the build timestamp.

6.13.1 Detailed Description

Class holding information about the *artdaq* package build.

The [PackageBuildInfo](#) class contains the name of the package, the version, and the timestamp of the build. *artdaq* stores this information in each data file.

Definition at line 17 of file PackageBuildInfo.hh.

6.13.2 Member Function Documentation

6.13.2.1 std::string artdaq::PackageBuildInfo::getBuildTimestamp () const `[inline]`

Gets the build timestamp.

Returns

The timestamp of the build

Definition at line 41 of file PackageBuildInfo.hh.

6.13.2.2 `std::string artdaq::PackageBuildInfo::getPackageName () const [inline]`

Gets the package name.

Returns

The package name

Definition at line 29 of file PackageBuildInfo.hh.

6.13.2.3 `std::string artdaq::PackageBuildInfo::getPackageVersion () const [inline]`

Gets the package version.

Returns

The package version

Definition at line 35 of file PackageBuildInfo.hh.

6.13.2.4 `void artdaq::PackageBuildInfo::setBuildTimestamp (std::string str) [inline]`

Sets the build timestamp.

Parameters

<i>str</i>	The timestamp of the build
------------	----------------------------

Definition at line 59 of file PackageBuildInfo.hh.

6.13.2.5 `void artdaq::PackageBuildInfo::setPackageName (std::string str) [inline]`

Sets the package name.

Parameters

<i>str</i>	The package name
------------	------------------

Definition at line 47 of file PackageBuildInfo.hh.

6.13.2.6 `void artdaq::PackageBuildInfo::setPackageVersion (std::string str) [inline]`

Sets the package version.

Parameters

<i>str</i>	The package version
------------	---------------------

Definition at line 53 of file PackageBuildInfo.hh.

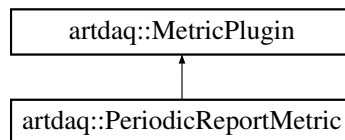
The documentation for this class was generated from the following file:

- artdaq_utilities/artdaq-utilities/BuildInfo/PackageBuildInfo.hh

6.14 artdaq::PeriodicReportMetric Class Reference

[PeriodicReportMetric](#) writes metric data to a file on disk.

Inheritance diagram for artdaq::PeriodicReportMetric:



Public Member Functions

- [PeriodicReportMetric](#) (fhicl::ParameterSet const &config, std::string const &app_name, std::string const &metric_name)
PeriodicReportMetric Constructor.
- [~PeriodicReportMetric](#) () override
PeriodicReportMetric Destructor. Calls stopMetrics and then closes the file.
- std::string [getLibName](#) () const override
Get the library name for the PeriodicReport metric.
- void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &unit, const std::chrono::system_clock::time_point &) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to a file.
- void [startMetrics_](#) () override
Perform startup actions.
- void [stopMetrics_](#) () override
Perform shutdown actions.

Additional Inherited Members

6.14.1 Detailed Description

[PeriodicReportMetric](#) writes metric data to a file on disk.

Definition at line 25 of file report_metric.cc.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `artdaq::PeriodicReportMetric::PeriodicReportMetric (fhicl::ParameterSet const & config, std::string const & app_name, std::string const & metric_name) [inline], [explicit]`

[PeriodicReportMetric](#) Constructor.

Parameters

<i>config</i>	ParameterSet used to configure PeriodicReportMetric
<i>app_name</i>	Name of the application sending metrics
<i>metric_name</i>	Name of this MetricPlugin instance

* `PeriodicReportMetric` accepts no parameters.
*

Definition at line 45 of file report_metric.cc.

6.14.3 Member Function Documentation

6.14.3.1 `std::string artdaq::PeriodicReportMetric::getLibName () const [inline], [override], [virtual]`

Get the library name for the PeriodicReport metric.

Returns

The library name for the PeriodicReport metric, "report"

Reimplemented from [artdaq::MetricPlugin](#).

Definition at line 65 of file report_metric.cc.

6.14.3.2 `void artdaq::PeriodicReportMetric::sendMetric_ (const std::string & name, const std::string & value, const std::string & unit, const std::chrono::system_clock::time_point &) [inline], [override], [virtual]`

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric

<i>unit</i>	Units of the metric
-------------	---------------------

Not using the time field, as the report will have its own timestamp from TRACE

Implements [artdaq::MetricPlugin](#).

Definition at line 75 of file report_metric.cc.

```
6.14.3.3 void artdaq::PeriodicReportMetric::sendMetric_ ( const std::string & name, const int & value, const std::string & unit,
const std::chrono::system_clock::time_point & time ) [inline],[override],[virtual]
```

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 91 of file report_metric.cc.

```
6.14.3.4 void artdaq::PeriodicReportMetric::sendMetric_ ( const std::string & name, const double & value, const std::string & unit,
const std::chrono::system_clock::time_point & time ) [inline],[override],[virtual]
```

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 103 of file report_metric.cc.

```
6.14.3.5 void artdaq::PeriodicReportMetric::sendMetric_ ( const std::string & name, const float & value, const std::string & unit,
const std::chrono::system_clock::time_point & time ) [inline],[override],[virtual]
```

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 115 of file report_metric.cc.

6.14.3.6 void artdaq::PeriodicReportMetric::sendMetric_(const std::string & *name*, const uint64_t & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline],[override],[virtual]

Write metric data to a file.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 127 of file report_metric.cc.

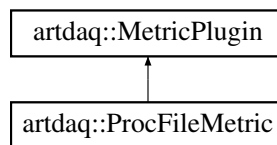
The documentation for this class was generated from the following file:

- artdaq_utilities/artdaq-utilities/Plugins/report_metric.cc

6.15 artdaq::ProcFileMetric Class Reference

A [MetricPlugin](#) which writes a long unsigned int metric with a given name to a given pipe.

Inheritance diagram for artdaq::ProcFileMetric:



Public Member Functions

- [ProcFileMetric](#) (fhicl::ParameterSet const &config, std::string const &app_name, std::string const &metric_name)
ProcFileMetric Constructor.
- [~ProcFileMetric](#) () override
ProcFileMetric Destructor.
- std::string [getLibName](#) () const override
Get the "library name" of this Metric.
- void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &, const std::chrono::system_clock::time_point &) override
Set the value to be written to the pipe when it is opened by a reader.
- void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Set the value to be written to the pipe when it is opened by a reader.
- void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Set the value to be written to the pipe when it is opened by a reader.
- void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Set the value to be written to the pipe when it is opened by a reader.
- void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Set the value to be written to the pipe when it is opened by a reader.

- void [startMetrics_](#) () override
Start the metric-sending thread.
- void [stopMetrics_](#) () override
Open the pipe for reading to allow the metric-sending thread to end gracefully.
- void [writePipe](#) ()
Wait for the pipe to be opened and then write the current value to it.

Additional Inherited Members

6.15.1 Detailed Description

A [MetricPlugin](#) which writes a long unsigned int metric with a given name to a given pipe.

This [MetricPlugin](#) emulates the function of the /proc file system, where the kernel provides access to various counters and parameters.

Definition at line 23 of file `procFile_metric.cc`.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 `artdaq::ProcFileMetric::ProcFileMetric (fhicl::ParameterSet const & config, std::string const & app_name, std::string const & metric_name) [inline],[explicit]`

[ProcFileMetric](#) Constructor.

Parameters

<i>config</i>	FHiCL ParameterSet used to configure the ProcFileMetric
<i>app_name</i>	Name of the application sending metrics
<i>metric_name</i>	Name of this MetricPlugin instance

```
* ProcFileMetric accepts the following Parameters (in addition to those accepted by MetricPlugin):
* "pipe": Name of pipe virtual file to write to
* "name": Name of the metric to write to pipe
*
```

Definition at line 49 of file `procFile_metric.cc`.

6.15.3 Member Function Documentation

6.15.3.1 `std::string artdaq::ProcFileMetric::getLibName () const [inline],[override],[virtual]`

Get the "library name" of this Metric.

Returns

The library name of this metric, "procFile"

Reimplemented from [artdaq::MetricPlugin](#).

Definition at line 81 of file `procFile_metric.cc`.

6.15.3.2 `void artdaq::ProcFileMetric::sendMetric_(const std::string & name, const std::string & value, const std::string & , const std::chrono::system_clock::time_point &) [inline],[override],[virtual]`

Set the value to be written to the pipe when it is opened by a reader.

Parameters

<i>name</i>	Name of the metric. Must match configred name for value to be updated (This MetricPlugin should be used with the <code>useNameOverride</code> parameter!)
<i>value</i>	Value of the metric.

Implements [artdaq::MetricPlugin](#).

Definition at line 88 of file `procFile_metric.cc`.

6.15.3.3 `void artdaq::ProcFileMetric::sendMetric_ (const std::string & name, const int & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline], [override], [virtual]`

Set the value to be written to the pipe when it is opened by a reader.

Parameters

<i>name</i>	Name of the metric. Must match configred name for value to be updated (This MetricPlugin should be used with the <code>useNameOverride</code> parameter!)
<i>value</i>	Value of the metric.
<i>unit</i>	Units of the metric.
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 104 of file `procFile_metric.cc`.

6.15.3.4 `void artdaq::ProcFileMetric::sendMetric_ (const std::string & name, const double & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline], [override], [virtual]`

Set the value to be written to the pipe when it is opened by a reader.

Parameters

<i>name</i>	Name of the metric. Must match configred name for value to be updated (This MetricPlugin should be used with the <code>useNameOverride</code> parameter!)
<i>value</i>	Value of the metric.
<i>unit</i>	Units of the metric.
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 116 of file `procFile_metric.cc`.

6.15.3.5 `void artdaq::ProcFileMetric::sendMetric_ (const std::string & name, const float & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline], [override], [virtual]`

Set the value to be written to the pipe when it is opened by a reader.

Parameters

<i>name</i>	Name of the metric. Must match configred name for value to be updated (This MetricPlugin should be used with the <code>useNameOverride</code> parameter!)
-------------	---

<i>value</i>	Value of the metric.
<i>unit</i>	Units of the metric.
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 128 of file procFile_metric.cc.

6.15.3.6 `void artdaq::ProcFileMetric::sendMetric_(const std::string & name, const uint64_t & value, const std::string & unit, const std::chrono::system_clock::time_point & time) [inline], [override], [virtual]`

Set the value to be written to the pipe when it is opened by a reader.

Parameters

<i>name</i>	Name of the metric. Must match configed name for value to be updated (This MetricPlugin should be used with the <code>useNameOverride</code> parameter!)
<i>value</i>	Value of the metric.
<i>unit</i>	Units of the metric.
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 140 of file procFile_metric.cc.

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

- `artdaq_utilities/artdaq-utilities/Plugins/procFile_metric.cc`

6.16 artdaq::SystemMetricCollector Class Reference

Collects metrics from the system, using proc filesystem or kernel API calls

```
#include <artdaq-utilities/Plugins/SystemMetricCollector.hh>
```

Public Member Functions

- [SystemMetricCollector](#) (bool processMetrics, bool systemMetrics)
SystemMetricCollector Constructor
- void [GetSystemCPUUsage](#) ()
Calculate the system CPU usage percentages
- double [GetProcessCPUUsagePercent](#) ()
Return the current amount of CPU usage for the current process, %
- uint64_t [GetNetworkReceiveBytes](#) (std::string ifname)
Get the amount of data received from the network in the last network collection interval (1.0 s)
- uint64_t [GetNetworkSendBytes](#) (std::string ifname)
Get the amount of data sent to the network in the last network collection interval (1.0 s)
- uint64_t [GetNetworkReceiveErrors](#) (std::string ifname)
Get the number of network receive errors in the last network collection interval (1.0 s)
- uint64_t [GetNetworkSendErrors](#) (std::string ifname)
Get the number of network send errors in the last network collection interval (1.0 s)
- std::list< std::string > [GetNetworkInterfaceNames](#) ()

Get the names of the local network interfaces.

- `std::list< std::unique_ptr< MetricData > > SendMetrics ()`

Send the configured metrics

Static Public Member Functions

- `static uint64_t GetAvailableRAM ()`
Get the amount of available RAM in the system
- `static uint64_t GetBufferedRAM ()`
Get the amount of RAM currently being used for cache
- `static uint64_t GetTotalRAM ()`
Get the total amount of RAM in the system
- `static double GetAvailableRAMPercent (bool buffers)`
Get the percentage of available RAM
- `static uint64_t GetProcessMemUsage ()`
Get the amount of RAM being used by this process
- `static double GetProcessMemUsagePercent ()`
Get the amount of RAM being used by this process
- `static uint64_t GetNetworkTCPRetransSegs ()`
Return the current number of TCP (total) segments retransmitted, segments

6.16.1 Detailed Description

Collects metrics from the system, using proc filesystem or kernel API calls

Definition at line 12 of file SystemMetricCollector.hh.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 `artdaq::SystemMetricCollector::SystemMetricCollector (bool processMetrics, bool systemMetrics)`

[SystemMetricCollector](#) Constructor

Parameters

<i>processMetrics</i>	Whether to collect process-level metrics (i.e. process CPU/RAM)
<i>systemMetrics</i>	Whether to collect system-level metrics (i.e. System CPU/RAM/Network)

Definition at line 16 of file SystemMetricCollector.cc.

6.16.3 Member Function Documentation

6.16.3.1 `uint64_t artdaq::SystemMetricCollector::GetAvailableRAM () [static]`

Get the amount of available RAM in the system

Returns

The amount of available (free) RAM in bytes

Definition at line 83 of file SystemMetricCollector.cc.

6.16.3.2 `double artdaq::SystemMetricCollector::GetAvailableRAMPercent (bool buffers) [static]`

Get the percentage of available RAM

Parameters

<i>buffers</i>	Whether cache RAM should be counted as available
----------------	--

Returns

The amount of available RAM, in %

Definition at line 116 of file SystemMetricCollector.cc.

6.16.3.3 `uint64_t artdaq::SystemMetricCollector::GetBufferedRAM () [static]`

Get the amount of RAM currently being used for cache

Returns

The amount of RAM used in cache in bytes

Definition at line 94 of file SystemMetricCollector.cc.

6.16.3.4 `std::list< std::string > artdaq::SystemMetricCollector::GetNetworkInterfaceNames ()`

Get the names of the local network interfaces.

Returns

The names of the local network interfaces (e.g. {"ens0","enp3s1f1"})

Definition at line 201 of file SystemMetricCollector.cc.

6.16.3.5 `uint64_t artdaq::SystemMetricCollector::GetNetworkReceiveBytes (std::string ifname)`

Get the amount of data received from the network in the last network collection interval (1.0 s)

Parameters

<i>ifname</i>	Name of the interface to collect
---------------	----------------------------------

Returns

The number of bytes received from the network in the last second

Definition at line 145 of file SystemMetricCollector.cc.

6.16.3.6 `uint64_t artdaq::SystemMetricCollector::GetNetworkReceiveErrors (std::string ifname)`

Get the number of network receive errors in the last network collection interval (1.0 s)

Parameters

<i>ifname</i>	Name of the interface to collect
---------------	----------------------------------

Returns

The number of network receive errors in the last second

Definition at line 157 of file SystemMetricCollector.cc.

6.16.3.7 `uint64_t artdaq::SystemMetricCollector::GetNetworkSendBytes (std::string ifname)`

Get the amount of data sent to the network in the last network collection interval (1.0 s)

Parameters

<i>ifname</i>	Name of the interface to collect
---------------	----------------------------------

Returns

The number of bytes sent to the network in the last second

Definition at line 151 of file SystemMetricCollector.cc.

6.16.3.8 `uint64_t artdaq::SystemMetricCollector::GetNetworkSendErrors (std::string ifname)`

Get the number of network send errors in the last network collection interval (1.0 s)

Parameters

<i>ifname</i>	Name of the interface to collect
---------------	----------------------------------

Returns

The number of network send errors in the last second

Definition at line 195 of file SystemMetricCollector.cc.

6.16.3.9 `uint64_t artdaq::SystemMetricCollector::GetNetworkTCPRetransSegs () [static]`

Return the current number of TCP (total) segments retransmitted, segments

Returns

the current number of TCP (total) segments retransmitted, segments

Definition at line 163 of file SystemMetricCollector.cc.

6.16.3.10 `double artdaq::SystemMetricCollector::GetProcessCPUUsagePercent ()`

Return the current amount of CPU usage for the current process, %

Returns

The current amount of CPU usage for the current process, %

Definition at line 62 of file SystemMetricCollector.cc.

6.16.3.11 `uint64_t artdaq::SystemMetricCollector::GetProcessMemUsage () [static]`

Get the amount of RAM being used by this process

Returns

The amount of RAM being used by this process, in bytes

Definition at line 128 of file SystemMetricCollector.cc.

6.16.3.12 `double artdaq::SystemMetricCollector::GetProcessMemUsagePercent () [static]`

Get the amount of RAM being used by this process

Returns

The amount of RAM used by this process, as a percentage of the total RAM in the system

Definition at line 137 of file SystemMetricCollector.cc.

6.16.3.13 `void artdaq::SystemMetricCollector::GetSystemCPUUsage ()`

Calculate the system CPU usage percentages

Definition at line 36 of file SystemMetricCollector.cc.

6.16.3.14 `uint64_t artdaq::SystemMetricCollector::GetTotalRAM () [static]`

Get the total amount of RAM in the system

Returns

The total amount of RAM in the system, in bytes

Definition at line 105 of file SystemMetricCollector.cc.

6.16.3.15 `std::list< std::unique_ptr< artdaq::MetricData > > artdaq::SystemMetricCollector::SendMetrics ()`

Send the configured metrics

Returns

A list of [MetricData](#) pointers for direct injection into [MetricManager](#)

Definition at line 211 of file SystemMetricCollector.cc.

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

- `artdaq_utilities/artdaq-utilities/Plugins/SystemMetricCollector.hh`
- `artdaq_utilities/artdaq-utilities/Plugins/SystemMetricCollector.cc`

6.17 artdaq::TestMetric Class Reference

Provides in-memory storage of metric data for testing

```
#include <artdaq-utilities/Plugins/TestMetric.hh>
```

Classes

- struct [MetricPoint](#)
Describes a single metric point

Static Public Member Functions

- static void [LockReceivedMetricMutex](#) ()
Lock the ReceivedMetricMutex
- static void [UnlockReceivedMetricMutex](#) ()
Unlock the ReceivedMetricMutex

Static Public Attributes

- static std::mutex [received_metrics_mutex](#)
Mutex to protect the received_metrics list.
- static std::list< [MetricPoint](#) > [received_metrics](#) = std::list<[TestMetric::MetricPoint](#)>()
List of received metric data.

6.17.1 Detailed Description

Provides in-memory storage of metric data for testing

Definition at line 19 of file TestMetric.hh.

6.17.2 Member Function Documentation

6.17.2.1 static void artdaq::TestMetric::LockReceivedMetricMutex () [inline],[static]

Lock the ReceivedMetricMutex

Definition at line 36 of file TestMetric.hh.

6.17.2.2 static void artdaq::TestMetric::UnlockReceivedMetricMutex () [inline],[static]

Unlock the ReceivedMetricMutex

Definition at line 46 of file TestMetric.hh.

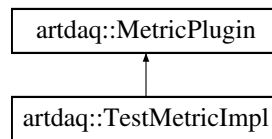
The documentation for this class was generated from the following files:

- artdaq_utilities/artdaq-utilities/Plugins/[TestMetric.hh](#)
- artdaq_utilities/artdaq-utilities/Plugins/TestMetric.cc

6.18 artdaq::TestMetricImpl Class Reference

[TestMetric](#) writes metric data to a statically-allocated memory block.

Inheritance diagram for artdaq::TestMetricImpl:



Public Member Functions

- [TestMetricImpl](#) (fhicl::ParameterSet const &config, std::string const &app_name, std::string const &metric_name)
TestMetric Constructor.
- [~TestMetricImpl](#) () override
TestMetricImpl Destructor. Calls stopMetrics.
- std::string [getLibName](#) () const override
Get the library name for the Test metric.
- void [sendMetric_](#) (const std::string &name, const std::string &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to memory.
- void [sendMetric_](#) (const std::string &name, const int &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to memory.
- void [sendMetric_](#) (const std::string &name, const double &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to memory.
- void [sendMetric_](#) (const std::string &name, const float &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to memory.
- void [sendMetric_](#) (const std::string &name, const uint64_t &value, const std::string &unit, const std::chrono::system_clock::time_point &time) override
Write metric data to memory.
- void [startMetrics_](#) () override
Perform startup actions.
- void [stopMetrics_](#) () override
Perform shutdown actions.

Additional Inherited Members

6.18.1 Detailed Description

[TestMetric](#) writes metric data to a statically-allocated memory block.

Definition at line 22 of file test_metric.cc.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 artdaq::TestMetricImpl::TestMetricImpl (fhicl::ParameterSet const & *config*, std::string const & *app_name*, std::string const & *metric_name*) [inline], [explicit]

[TestMetric](#) Constructor.

Parameters

<i>config</i>	ParameterSet used to configure TestMetric
<i>app_name</i>	Name of the application sending metrics
<i>metric_name</i>	Name of this MetricPlugin instance

Definition at line 31 of file test_metric.cc.

6.18.3 Member Function Documentation

6.18.3.1 `std::string artdaq::TestMetricImpl::getLibName() const` `[inline],[override],[virtual]`

Get the library name for the Test metric.

Returns

The library name for the Test metric, "test"

Reimplemented from [artdaq::MetricPlugin](#).

Definition at line 49 of file test_metric.cc.

6.18.3.2 `void artdaq::TestMetricImpl::sendMetric_(const std::string & name, const std::string & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline],[override],[virtual]`

Write metric data to memory.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 58 of file test_metric.cc.

6.18.3.3 `void artdaq::TestMetricImpl::sendMetric_(const std::string & name, const int & value, const std::string & unit, const std::chrono::system_clock::time_point & time)` `[inline],[override],[virtual]`

Write metric data to memory.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 76 of file test_metric.cc.

6.18.3.4 void artdaq::TestMetricImpl::sendMetric_ (const std::string & *name*, const double & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline], [override], [virtual]

Write metric data to memory.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 88 of file test_metric.cc.

6.18.3.5 void artdaq::TestMetricImpl::sendMetric_ (const std::string & *name*, const float & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline], [override], [virtual]

Write metric data to memory.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 100 of file test_metric.cc.

6.18.3.6 void artdaq::TestMetricImpl::sendMetric_ (const std::string & *name*, const uint64_t & *value*, const std::string & *unit*, const std::chrono::system_clock::time_point & *time*) [inline], [override], [virtual]

Write metric data to memory.

Parameters

<i>name</i>	Name of the metric
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric
<i>time</i>	Time the metric was sent

Implements [artdaq::MetricPlugin](#).

Definition at line 112 of file test_metric.cc.

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

- artdaq_utilities/artdaq-utilities/Plugins/test_metric.cc

Chapter 7

File Documentation

7.1 artdaq_utilities/artdaq-utilities/Plugins/TestMetric.hh File Reference

```
#include <chrono>
#include <list>
#include <mutex>
#include <string>
#include "TRACE/trace.h"
```

Classes

- class [artdaq::TestMetric](#)
Provides in-memory storage of metric data for testing
- struct [artdaq::TestMetric::MetricPoint](#)
Describes a single metric point

Namespaces

- [artdaq](#)
The artdaq namespace.

7.1.1 Detailed Description

: Static storage for testing Metric system

Definition in file [TestMetric.hh](#).

Index

- ~MetricManager
 - artdaq::MetricManager, [36](#)
- Accumulate
 - artdaq, [11](#)
- Active
 - artdaq::MetricManager, [36](#)
- Add
 - artdaq::MetricData, [29](#)
- addMetricData
 - artdaq::MetricPlugin, [43](#)
- AddPoint
 - artdaq::MetricData, [29](#), [30](#)
- artdaq, [9](#)
 - Accumulate, [11](#)
 - Average, [11](#)
 - DoubleMetric, [11](#)
 - FloatMetric, [11](#)
 - IntMetric, [11](#)
 - InvalidMetric, [11](#)
 - LastPoint, [11](#)
 - makeFunc_t, [10](#)
 - makeMetricPlugin, [11](#)
 - Maximum, [11](#)
 - MetricMode, [11](#)
 - MetricType, [11](#)
 - Minimum, [11](#)
 - operator&, [12](#)
 - Persist, [11](#)
 - Rate, [11](#)
 - StringMetric, [11](#)
 - UnsignedMetric, [11](#)
- artdaq::FileMetric, [17](#)
 - FileMetric, [18](#)
 - getLibName, [19](#)
 - sendMetric_, [19](#), [20](#)
- artdaq::GraphiteMetric, [21](#)
 - getLibName, [22](#)
 - GraphiteMetric, [22](#)
 - sendMetric_, [23](#), [25](#)
- artdaq::MetricData, [25](#)
 - Add, [29](#)
 - AddPoint, [29](#), [30](#)
 - DataPointCount, [31](#)
 - Level, [31](#)
 - MetricData, [27–29](#)
 - MetricPrefix, [31](#)
 - Mode, [31](#)
 - Name, [31](#)
 - operator=, [30](#)
 - Reset, [31](#)
 - StringValue, [31](#)
 - Type, [32](#)
 - Unit, [32](#)
 - UseNameOverride, [32](#)
 - Value, [32](#)
- artdaq::MetricData::MetricDataValue, [32](#)
 - MetricDataValue, [33](#), [34](#)
- artdaq::MetricManager, [34](#)
 - ~MetricManager, [36](#)
 - Active, [36](#)
 - initialize, [36](#)
 - Initialized, [36](#)
 - metricManagerBusy, [37](#)
 - metricQueueEmpty, [37](#)
 - metricQueueSize, [37](#)
 - operator=, [37](#)
 - reinitialize, [37](#)
 - Running, [38](#)
 - sendMetric, [38–40](#)
 - setPrefix, [40](#)
- artdaq::MetricManager::Config, [15](#)
 - metric_queue_notify_size, [15](#)
 - metric_queue_size, [16](#)
 - metric_send_maximum_delay_ms, [16](#)
- artdaq::MetricPlugin, [40](#)
 - addMetricData, [43](#)
 - IsLevelEnabled, [43](#)
 - MetricPlugin, [42](#)
 - metricsPending, [43](#)
 - sendMetric_, [43](#), [45](#)
 - sendMetrics, [46](#)
 - startMetrics_, [46](#)
 - stopMetrics_, [46](#)
- artdaq::MetricPlugin::Config, [16](#)
- artdaq::MsgFacilityMetric, [50](#)
 - getLibName, [51](#)
 - MsgFacilityMetric, [51](#)
 - sendMetric_, [51](#), [52](#)
- artdaq::PackageBuildInfo, [54](#)
 - getBuildTimestamp, [54](#)

- getPackageName, 55
 - getPackageVersion, 55
 - setBuildTimestamp, 55
 - setPackageName, 55
 - setPackageVersion, 55
- artdaq::PeriodicReportMetric, 56
 - getLibName, 57
 - PeriodicReportMetric, 57
 - sendMetric_, 57, 58
- artdaq::ProcFileMetric, 60
 - getLibName, 61
 - ProcFileMetric, 61
 - sendMetric_, 61, 63, 64
- artdaq::SystemMetricCollector, 64
 - GetAvailableRAM, 65
 - GetAvailableRAMPercent, 65
 - GetBufferedRAM, 66
 - GetNetworkInterfaceNames, 66
 - GetNetworkReceiveBytes, 66
 - GetNetworkReceiveErrors, 66
 - GetNetworkSendBytes, 67
 - GetNetworkSendErrors, 67
 - GetNetworkTCPReTransSegs, 67
 - GetProcessCPUUsagePercent, 67
 - GetProcessMemUsage, 67
 - GetProcessMemUsagePercent, 68
 - GetSystemCPUUsage, 68
 - GetTotalRAM, 68
 - SendMetrics, 68
 - SystemMetricCollector, 65
- artdaq::TestMetric, 69
 - LockReceivedMetricMutex, 69
 - UnlockReceivedMetricMutex, 69
- artdaq::TestMetric::MetricPoint, 49
- artdaq::TestMetricImpl, 70
 - getLibName, 72
 - sendMetric_, 72, 74
 - TestMetricImpl, 71
- artdaq_utilities/artdaq-utilities/Plugins/TestMetric.hh, 75
- artdaqtest::MetricPluginTestAdapter, 46
 - get_accumulationTime_, 48
 - get_app_name_, 48
 - get_inhibit_, 48
 - get_level_mask_, 49
 - get_pset, 49
 - MetricPluginTestAdapter, 48
- artdaqutilities, 12
- artdaqutilities::GetPackageBuildInfo, 20
 - getPackageBuildInfo, 21
- Average
 - artdaq, 11
- DataPointCount
 - artdaq::MetricData, 31
- DoubleMetric
 - artdaq, 11
- FileMetric
 - artdaq::FileMetric, 18
- FloatMetric
 - artdaq, 11
- get_accumulationTime_
 - artdaqtest::MetricPluginTestAdapter, 48
- get_app_name_
 - artdaqtest::MetricPluginTestAdapter, 48
- get_inhibit_
 - artdaqtest::MetricPluginTestAdapter, 48
- get_level_mask_
 - artdaqtest::MetricPluginTestAdapter, 49
- get_pset
 - artdaqtest::MetricPluginTestAdapter, 49
- GetAvailableRAM
 - artdaq::SystemMetricCollector, 65
- GetAvailableRAMPercent
 - artdaq::SystemMetricCollector, 65
- GetBufferedRAM
 - artdaq::SystemMetricCollector, 66
- getBuildTimestamp
 - artdaq::PackageBuildInfo, 54
- getLibName
 - artdaq::FileMetric, 19
 - artdaq::GraphiteMetric, 22
 - artdaq::MsgFacilityMetric, 51
 - artdaq::PeriodicReportMetric, 57
 - artdaq::ProcFileMetric, 61
 - artdaq::TestMetricImpl, 72
- GetNetworkInterfaceNames
 - artdaq::SystemMetricCollector, 66
- GetNetworkReceiveBytes
 - artdaq::SystemMetricCollector, 66
- GetNetworkReceiveErrors
 - artdaq::SystemMetricCollector, 66
- GetNetworkSendBytes
 - artdaq::SystemMetricCollector, 67
- GetNetworkSendErrors
 - artdaq::SystemMetricCollector, 67
- GetNetworkTCPReTransSegs
 - artdaq::SystemMetricCollector, 67
- getPackageBuildInfo
 - artdaqutilities::GetPackageBuildInfo, 21
- getPackageName
 - artdaq::PackageBuildInfo, 55
- getPackageVersion
 - artdaq::PackageBuildInfo, 55
- GetProcessCPUUsagePercent
 - artdaq::SystemMetricCollector, 67
- GetProcessMemUsage
 - artdaq::SystemMetricCollector, 67

- GetProcessMemUsagePercent
 - artdaq::SystemMetricCollector, 68
- GetSystemCPUUsage
 - artdaq::SystemMetricCollector, 68
- GetTotalRAM
 - artdaq::SystemMetricCollector, 68
- GraphiteMetric
 - artdaq::GraphiteMetric, 22
- initialize
 - artdaq::MetricManager, 36
- Initialized
 - artdaq::MetricManager, 36
- IntMetric
 - artdaq, 11
- InvalidMetric
 - artdaq, 11
- IsLevelEnabled
 - artdaq::MetricPlugin, 43
- LastPoint
 - artdaq, 11
- Level
 - artdaq::MetricData, 31
- LockReceivedMetricMutex
 - artdaq::TestMetric, 69
- makeFunc_t
 - artdaq, 10
- makeMetricPlugin
 - artdaq, 11
- Maximum
 - artdaq, 11
- metric_queue_notify_size
 - artdaq::MetricManager::Config, 15
- metric_queue_size
 - artdaq::MetricManager::Config, 16
- metric_send_maximum_delay_ms
 - artdaq::MetricManager::Config, 16
- MetricData
 - artdaq::MetricData, 27–29
- MetricDataValue
 - artdaq::MetricData::MetricDataValue, 33, 34
- metricManagerBusy
 - artdaq::MetricManager, 37
- MetricMode
 - artdaq, 11
- MetricPlugin
 - artdaq::MetricPlugin, 42
- MetricPluginTestAdapter
 - artdaqtest::MetricPluginTestAdapter, 48
- MetricPrefix
 - artdaq::MetricData, 31
- metricQueueEmpty
 - artdaq::MetricManager, 37
- metricQueueSize
 - artdaq::MetricManager, 37
- MetricType
 - artdaq, 11
- metricsPending
 - artdaq::MetricPlugin, 43
- Minimum
 - artdaq, 11
- Mode
 - artdaq::MetricData, 31
- MsgFacilityMetric
 - artdaq::MsgFacilityMetric, 51
- Name
 - artdaq::MetricData, 31
- operator=
 - artdaq::MetricData, 30
 - artdaq::MetricManager, 37
- operator&
 - artdaq, 12
- PeriodicReportMetric
 - artdaq::PeriodicReportMetric, 57
- Persist
 - artdaq, 11
- ProcFileMetric
 - artdaq::ProcFileMetric, 61
- Rate
 - artdaq, 11
- reinitialize
 - artdaq::MetricManager, 37
- Reset
 - artdaq::MetricData, 31
- Running
 - artdaq::MetricManager, 38
- sendMetric
 - artdaq::MetricManager, 38–40
- sendMetric_
 - artdaq::FileMetric, 19, 20
 - artdaq::GraphiteMetric, 23, 25
 - artdaq::MetricPlugin, 43, 45
 - artdaq::MsgFacilityMetric, 51, 52
 - artdaq::PeriodicReportMetric, 57, 58
 - artdaq::ProcFileMetric, 61, 63, 64
 - artdaq::TestMetricImpl, 72, 74
- SendMetrics
 - artdaq::SystemMetricCollector, 68
- sendMetrics
 - artdaq::MetricPlugin, 46
- setBuildTimestamp
 - artdaq::PackageBuildInfo, 55
- setPackageName

- artdaq::PackageBuildInfo, [55](#)
- setPackageVersion
 - artdaq::PackageBuildInfo, [55](#)
- setPrefix
 - artdaq::MetricManager, [40](#)
- startMetrics_
 - artdaq::MetricPlugin, [46](#)
- stopMetrics_
 - artdaq::MetricPlugin, [46](#)
- StringMetric
 - artdaq, [11](#)
- StringValue
 - artdaq::MetricData, [31](#)
- SystemMetricCollector
 - artdaq::SystemMetricCollector, [65](#)
- TestMetricImpl
 - artdaq::TestMetricImpl, [71](#)
- Type
 - artdaq::MetricData, [32](#)
- Unit
 - artdaq::MetricData, [32](#)
- UnlockReceivedMetricMutex
 - artdaq::TestMetric, [69](#)
- UnsignedMetric
 - artdaq, [11](#)
- UseNameOverride
 - artdaq::MetricData, [32](#)
- Value
 - artdaq::MetricData, [32](#)