cavecanem

0.2.0

Generated by Doxygen 1.7.1

Mon Jun 27 2011 16:03:48

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 cc_general_properties Class Reference

```
#include <xml_parser.hpp>
```

Public Attributes

- int **publishing_period**
- int **domain_id**
- std::string **qos_file**
- std::string **qos_library**
- std::string **qos_profile**
- std::map< std::string, std::list< std::string > > **plugin_list_map**

3.1.1 Detailed Description

This structure stores the general properties of Cave Canem got from a XML configuration file.

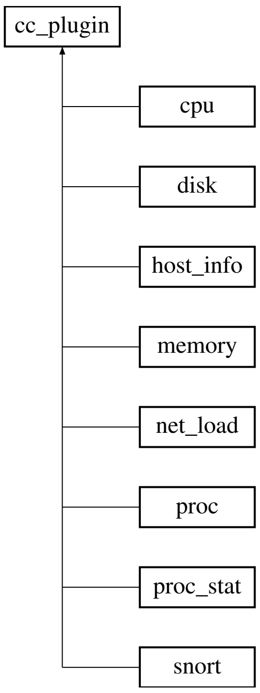
Definition at line 87 of file xml_parser.hpp.

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

- main/xml_parser.hpp

3.2 cc_plugin Class Reference

Inheritance diagram for cc_plugin:



Public Member Functions

- virtual std::string plugin_class ()=0

  Returns the name of the plugin.

- virtual bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_-DynamicData ∗data)=0

  The plugin gathers the information and publishes it.

- virtual bool publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Plugins must use this method to publish the information in generate_and_publish_information().

- void destroy_plugin ()

  Deletes the plugin.

3.2.1 Detailed Description

Definition at line 29 of file plugin.hpp.

3.2.2 Member Function Documentation

3.2.2.1 void cc_plugin::destroy_plugin ( ) `[inline]`

Deletes the plugin.

Deletes the plugin using the C++ function `delete()`.

Definition at line 82 of file plugin.hpp.

3.2.2.2 virtual bool cc_plugin::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer,* DDS_DynamicData ∗ *data*) `[pure virtual]`

The plugin gathers the information and publishes it.

Parameters

writer A pointer to the DDS DynamicDataWriter.

data A pointer to the DDS Dynamic Data to fill.

Returns

Returns true if everything was correct and false if not.

Implemented in cpu, disk, host_info, memory, net_load, proc, proc_stat, and snort.

3.2.2.3 virtual std::string cc_plugin::plugin_class ( ) `[pure virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implemented in cpu, disk, host_info, memory, net_load, proc, proc_stat, and snort.

3.2.2.4 virtual bool cc_plugin::publish_information (DDSDynamicDataWriter ∗ *writer,* DDS_DynamicData ∗ *data*) `[inline, virtual]`

Plugins must use this method to publish the information in generate_and_publish_information().

Abstracts plugin developers from the publication of the gathered information using DDS. Therefore, each plugin must call this method whithin generate_and_publish_information() to publish.

Parameters

writer A pointer to the DDS DynamicDataWriter.

data A pointer to the filled DDS Dynamic Data.

Returns

Definition at line 65 of file plugin.hpp.

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

- main/plugin.hpp

3.3 cc_plugin_properties Class Reference

```
#include <xml_parser.hpp>
```

Public Attributes

- std::string **dll**
- std::string **create_function**
- int **publishing_period**
- std::string **qos_profile**
- std::string **qos_library**
- std::string **topic_name**
- std::map< std::string, std::string > **plugin_config**
- struct DDS_DataWriterQos ∗ **datawriter_qos**
- struct DDS_TypeCode ∗ **type_code**

3.3.1 Detailed Description

This structure stores the properties of a plugin got from a XML configuration file.

Definition at line 102 of file xml_parser.hpp.

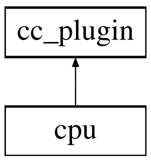
The documentation for this class was generated from the following file:

- main/xml_parser.hpp

3.4 cpu Class Reference

`#include <cpu.hpp>`

Inheritance diagram for cpu:



Public Member Functions

- cpu (std::string plugin_id, std::map< std::string, std::string > properties)

  Constructor of the cpu class.

- virtual ∼cpu ()
- bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Gets some information related to the cpu status and publishes it.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.4.1 Detailed Description

This class defines the cpu plugin. The objective of this plugin is to get and publish some information related to the use and load of the CPU. To achieve this objective it uses the Hyperic Sigar library.

Returns

Definition at line 40 of file cpu.hpp.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 cpu::cpu (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the cpu class.

Constructor of the cpu class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 33 of file cpu.cpp.

3.4.2.2 cpu::~cpu (void) `[virtual]`

Destructor of the cpu class.

Definition at line 47 of file cpu.cpp.

3.4.3 Member Function Documentation

3.4.3.1 bool cpu::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer*, DDS_DynamicData ∗ *data*) `[virtual]`

Gets some information related to the cpu status and publishes it.

Gets some information related to the cpu--CPU usage, load average, etc.-- and publishes it using the method `publish_information` -- defined in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 84 of file cpu.cpp.

3.4.3.2 virtual std::string cpu::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 48 of file cpu.hpp.

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

- plugins/cpu/cpu.hpp
- plugins/cpu/cpu.cpp

3.5 date Class Reference

`#include <snort.hpp>`

Public Attributes

- int **year**
- int **mday**
- int **mon**
- int **hour**
- int **min**
- int **sec**

3.5.1 Detailed Description

Represents a date using integers to store a year, month day, month, hour, minute and second.

Definition at line 80 of file snort.hpp.

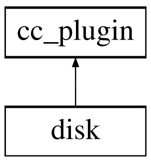
The documentation for this class was generated from the following file:

- plugins/snort/snort.hpp

3.6 disk Class Reference

`#include <disk.hpp>`

Inheritance diagram for disk:



Public Member Functions

- disk (std::string plugin_id, std::map< std::string, std::string > properties)

  Constructor of the disk class.

- virtual ~disk (void)
- virtual bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Gets the list of the filesystems of a machine and publishes their status.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.6.1 Detailed Description

This class defines the disk plugin. The objective of this plugin is to get and publish the status of the filesystems of a machine. To achieve this objetive it uses the Hyperic Sigar library.

Definition at line 40 of file disk.hpp.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 disk::disk (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the disk class.

Constructor of the disk class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 32 of file disk.cpp.

3.6.2.2 disk::~disk (void) `[virtual]`

Destructor of the disk class.

Definition at line 44 of file disk.cpp.

3.6.3 Member Function Documentation

3.6.3.1 bool disk::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer*, DDS_DynamicData ∗ *data*) `[virtual]`

Gets the list of the filesystems of a machine and publishes their status.

Gests the list of filesystems of a machine using Hyperic Sigar and publishes their status using the method `publish_information`--defined and implemented in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 81 of file disk.cpp.

3.6.3.2 virtual std::string disk::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 48 of file disk.hpp.

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

- plugins/disk/disk.hpp
- plugins/disk/disk.cpp

3.7 dynamicdata_info Class Reference

`#include <plugin_manager.hpp>`

Public Attributes

- DDSDynamicDataWriter ∗ **writer**
- DDS_DynamicData ∗ **data**

3.7.1 Detailed Description

Stores the information related to the dynamic data used to publish the plugin information in the DDS Global Data Space, that is, a DDS Dynamic DataWriter and the DDS Dynamic Data.

Definition at line 40 of file plugin_manager.hpp.

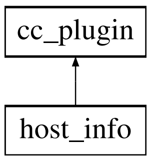
The documentation for this class was generated from the following file:

- main/plugin_manager.hpp

3.8 host_info Class Reference

```
#include <host_info.hpp>
```

Inheritance diagram for host_info:



Public Member Functions

- host_info (std::string plugin_id, std::map< std::string, std::string > properties)

  Constructor of the host_info class.

- virtual ~host_info ()
- bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Gets some information related to the host status and publishes it.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.8.1 Detailed Description

This class defines the host_info plugin. The objective of this plugin is to get and publish some general information related to the host. To achieve this objective it uses the Hyperic Sigar library.

Returns

Definition at line 40 of file host_info.hpp.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 host_info::host_info (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the host_info class.

Constructor of the host_info class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 33 of file host_info.cpp.

3.8.2.2 host_info::∼host_info (void) `[virtual]`

Destructor of the host_info class.

Definition at line 47 of file host_info.cpp.

3.8.3 Member Function Documentation

3.8.3.1 bool host_info::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer*, DDS_DynamicData ∗ *data*) `[virtual]`

Gets some information related to the host status and publishes it.

Gets some information related to the host status and publishes it using the method `publish_-information` -- defined in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 84 of file host_info.cpp.

3.8.3.2 virtual std::string host_info::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 48 of file host_info.hpp.

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

- plugins/host_info/host_info.hpp
- plugins/host_info/host_info.cpp

3.9 ids_alert_without_timestamp Class Reference

```
#include <snort.hpp>
```

Public Attributes

- std::string **sig_generator**
- std::string **sig_id**
- std::string **sig_rev**
- std::string **msg**
- std::string **proto**
- std::string **src**
- std::string **srcport**
- std::string **dst**
- std::string **dstport**
- std::string **ethsrc**
- std::string **ethdst**
- std::string **ethlen**
- std::string **tcpflags**
- std::string **tcpseq**
- std::string **tcpack**
- std::string **tcplen**
- std::string **tcpwindow**
- std::string **ttl**
- std::string **tos**
- std::string **id**
- std::string **dgmlen**
- std::string **iplen**
- std::string **icmptype**
- std::string **icmpcode**
- std::string **icmpid**

3.9.1 Detailed Description

Stores the fiels of an IDS alert read from a Snort's CSV log file--all the field except the timestamp.

Definition at line 46 of file snort.hpp.

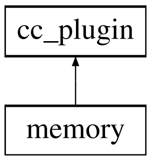
The documentation for this class was generated from the following file:

- plugins/snort/snort.hpp

3.10 memory Class Reference

`#include <memory.hpp>`

Inheritance diagram for memory:



Public Member Functions

- memory (std::string plugin_id, std::map< std::string, std::string > properties)

  Constructor of the memory class.

- virtual ∼memory ()
- bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Gets some information related to physical and swap memory and publishes it.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.10.1 Detailed Description

This class defines the memory plugin. The objective of this plugin is to get and publish information related to the use of the physical and swap memory. To achieve this objective it uses the Hyperic Sigar library.

Returns

Definition at line 39 of file memory.hpp.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 memory::memory (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the memory class.

Constructor of the memory class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 33 of file memory.cpp.

3.10.2.2 memory::∼memory (void) `[virtual]`

Destructor of the memory class.

Definition at line 47 of file memory.cpp.

3.10.3 Member Function Documentation

3.10.3.1 bool memory::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer*, DDS_DynamicData ∗ *data*) `[virtual]`

Gets some information related to physical and swap memory and publishes it.

Gets some information related to physical and swap memory and publishes it using the method `publish_information` -- defined in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 84 of file memory.cpp.

3.10.3.2 virtual std::string memory::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 48 of file memory.hpp.

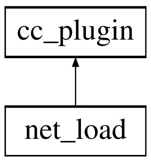
The documentation for this class was generated from the following files:

- plugins/memory/memory.hpp
- plugins/memory/memory.cpp

3.11 net_load Class Reference

`#include <net_load.hpp>`

Inheritance diagram for net_load:



Public Member Functions

- net_load (std::string plugin_id, std::map< std::string, std::string > properties)

  Constructor of the net_load class.

- virtual ∼net_load ()

Destructor of the net_load class.

- bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Gets the list of the network interfaces of a machine and publishes their status.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.11.1 Detailed Description

This class defines the net_load plugin. The objective of this plugin is to get and publish the status of the network interfaces of a machine. To achieve this objetive it uses the Hyperic Sigar library.

Definition at line 38 of file net_load.hpp.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 net_load::net_load (std::string *plugin_id,* std::map< std::string, std::string > *properties*)

Constructor of the net_load class.

Constructor of the net_load class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 33 of file net_load.cpp.

3.11.2.2 net_load::∼net_load ( ) `[virtual]`

Destructor of the net_load class.

Destructor of the memory class.

Definition at line 47 of file net_load.cpp.

3.11.3 Member Function Documentation

3.11.3.1 bool net_load::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer,* DDS_DynamicData ∗ *data*) `[virtual]`

Gets the list of the network interfaces of a machine and publishes their status.

Gets the network interfaces of a machine using Hyperic Sigar and publishes the status of them using the method `publish_information` -- defined and implemented in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 85 of file net_load.cpp.

3.11.3.2 virtual std::string net_load::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 48 of file net_load.hpp.

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

- plugins/net_load/net_load.hpp
- plugins/net_load/net_load.cpp

3.12 plugin_manager Class Reference

```
#include <plugin_manager.hpp>
```

Public Member Functions

- plugin_manager (std::string cfgfile)

  Constructor of the plugin_manager class.

- ∼plugin_manager ()

  Destructor of the class plugin_manager.

- bool initialize_dds (int domain_id, std::string qos_file, std::string qos_library, std::string qos_-profile)

  Creates all the DDS entities that plugins need.

- bool load_plugins ()

  Loads all the cc_plugins specified in the general configuration file.

- void publish_plugins_information ()

  Calls all the loaded plugins to publish.

- void unload_plugins ()

Unloads all the plugins loaded in load_plugins().

- bool shutdown_dds ()

  Deletes all the DDS Entities initialized in initialize_dds().

3.12.1 Detailed Description

Addresses the load and unload of plugins. This process involves the initialization and destruction of DDS entities used within the plugins.

Definition at line 51 of file plugin_manager.hpp.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 plugin_manager::plugin_manager (std::string *cfgfile*)

Constructor of the plugin_manager class.

The constructor of the plugin_manager class parses the general configuration file by XML_parser. Then, it loads the plugins indicated in the file by using load_plugins(), and finally it creates all the DDS entities trough initialize_dds().

Parameters

cfgfile General XML configuration file.

Definition at line 36 of file plugin_manager.cpp.

3.12.2.2 plugin_manager::∼plugin_manager ( )

Destructor of the class plugin_manager.

The destructor of the class plugin_manager shutdowns all the DDS entities and unloads all the plugins allocated by load_plugins() by using shutdown_dds() and unload_plugins().

Definition at line 68 of file plugin_manager.cpp.

3.12.3 Member Function Documentation

3.12.3.1 bool plugin_manager::initialize_dds (int *domain_id,* std::string *qos_file,* std::string *qos_library,* std::string *qos_profile*)

Creates all the DDS entities that plugins need.

It creates the DDS Domain Participant and DDS Publisher--shared by all the plugins-- using the method create_dds_participant_and_publisher() and creates the DDS Topic and DDS DataWriter for each plugin calling create_dds_topic_and_datawriter().

Parameters

domain_id DDS Domain ID

qos_configuration_file XML configuration file for the QoS.

qos_library Name of the QoS library.

qos_profile Name of the QoS profile (if "default" it loads the default RTI DDS QoS settings).

Returns

Returns true if everything was initialized correctly and false if it was not.

Definition at line 179 of file plugin_manager.cpp.

3.12.3.2 bool plugin_manager::load_plugins ( )

Loads all the cc_plugins specified in the general configuration file.

Loads the plugins specified in the general configuration file for each plugin library -- directory containing plugin definitions.

Returns

True if plugins were loaded correctly and False if they were not.

Definition at line 83 of file plugin_manager.cpp.

3.12.3.3 void plugin_manager::publish_plugins_information ( )

Calls all the loaded plugins to publish.

Calls the publishing method of all the plugins loaded. It also controls the publishing rate of each function.

Definition at line 435 of file plugin_manager.cpp.

3.12.3.4 bool plugin_manager::shutdown_dds ( )

Deletes all the DDS Entities initialized in initialize_dds().

Deletes all the DDS entities initialized by `initialize_dds()`, including the DDS Domain Participant.

Returns

It indicates wether the plugins were shutdowned correctly or not.

Definition at line 407 of file plugin_manager.cpp.

3.12.3.5 void plugin_manager::unload_plugins ( )

Unloads all the plugins loaded in load_plugins().

This method iterates both through the plugin_map_ and the libraries_map_ to clean up plugins and libraries.

Definition at line 101 of file plugin_manager.cpp.

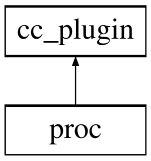
The documentation for this class was generated from the following files:

- main/plugin_manager.hpp
- main/plugin_manager.cpp

3.13 proc Class Reference

```
#include <proc.hpp>
```

Inheritance diagram for proc:



Public Member Functions

- proc (std::string plugin_id, std::map< std::string, std::string > properties)

  Constructor of the proc class.

- virtual ∼proc ()

  Destructor of the proc class.

- bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

  Gets the list of the processes of a machine and publishes their status.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.13.1 Detailed Description

This class defines the proc plugin. The objective of this plugin is to get and publish the status of the processes running on a machine. To achieve this objetive it uses the Hyperic Sigar library.

Definition at line 39 of file proc.hpp.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 proc::proc (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the proc class.

Constructor of the proc class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 33 of file proc.cpp.

3.13.2.2 proc::∼proc ( ) `[virtual]`

Destructor of the proc class.

Destructor of the memory class.

Definition at line 47 of file proc.cpp.

3.13.3 Member Function Documentation

3.13.3.1 bool proc::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer,* DDS_DynamicData ∗ *data*) `[virtual]`

Gets the list of the processes of a machine and publishes their status.

Gets the list of processes of a machine using Hyperic Sigar and publishes the status of them using the method `publish_information` -- defined and implemented in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 85 of file proc.cpp.

3.13.3.2 virtual std::string proc::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 49 of file proc.hpp.

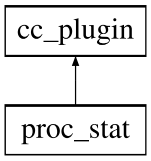
The documentation for this class was generated from the following files:

- plugins/proc/proc.hpp
- plugins/proc/proc.cpp

3.14 proc_stat Class Reference

`#include <proc_stat.hpp>`

Inheritance diagram for proc_stat:



Public Member Functions

- proc_stat (std::string plugin_id, std::map< std::string, std::string > properties)

    Constructor of the proc_stat class.

- virtual ∼proc_stat ()
- bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_DynamicData ∗data)

    Gets some information related to the processes running and publishes it.

- virtual std::string plugin_class ()

    Returns the name of the plugin.

3.14.1 Detailed Description

This class defines the proc_stat plugin. The objective of this plugin is to get and publish a overall view of the processes running on the machine. To achieve this objective it uses the Hyperic Sigar library.

Returns

Definition at line 40 of file proc_stat.hpp.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 proc_stat::proc_stat (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the proc_stat class.

Constructor of the proc_stat class.

Parameters

plugin_id Name of the plugin.

properties Map of properties (will be empty in this plugin).

Definition at line 33 of file proc_stat.cpp.

3.14.2.2 proc_stat::∼proc_stat (void) `[virtual]`

Destructor of the proc_stat class.

Definition at line 47 of file proc_stat.cpp.

3.14.3 Member Function Documentation

3.14.3.1 bool proc_stat::generate_and_publish_information (DDSDynamicDataWriter $*$ *writer*, DDS_DynamicData $*$ *data*) `[virtual]`

Gets some information related to the processes running and publishes it.

Gets some information related to the processes running on the host and publishes it using the method `publish_information` -- defined in the base class.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methods.

Returns

True if everything was right.

Implements cc_plugin.

Definition at line 84 of file proc_stat.cpp.

3.14.3.2 virtual std::string proc_stat::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 48 of file proc_stat.hpp.

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

- plugins/proc_stat/proc_stat.hpp
- plugins/proc_stat/proc_stat.cpp

3.15 RTIXMLCaveCanemExtensionObject Class Reference

```
#include <xml_parser.hpp>
```

Public Attributes

- char $**$ **attr**
- int **attr_length**
- int **current_element_index**
- struct RTIXMLCaveCanemExtensionObjectElement $*$ **tag_elements** [XML_CAVECANEM_-MAX_NUMBER_OF_NON_EXTENSION_TAGS]

3.15.1 Detailed Description

The extension classes are used to create XML objects. Each extension class is associated to a XML tag and they must be registered with the parser

Definition at line 69 of file xml_parser.hpp.

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

- main/xml_parser.hpp

3.16 RTIXMLCaveCanemExtensionObjectElement Class Reference

```
#include <xml_parser.hpp>
```

Public Attributes

- char ∗∗ **attr**
- int **attr_length**
- char ∗ **element_text**
- char ∗ **tag_name**

3.16.1 Detailed Description

Elements (object attribute tags) of the extension class object.

Definition at line 52 of file xml_parser.hpp.

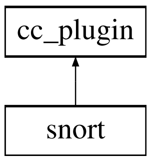
The documentation for this class was generated from the following file:

- main/xml_parser.hpp

3.17 snort Class Reference

```
#include <snort.hpp>
```

Inheritance diagram for snort:



Public Member Functions

- snort (std::string plugin_id, std::map< std::string, std::string > properties)

    Constructor of the class snort.

- virtual ∼snort (void)
- virtual bool generate_and_publish_information (DDSDynamicDataWriter ∗writer, DDS_-DynamicData ∗data)

  Gets new alerts from Snort's logfile using update_alertsmap() and publishes them.

- virtual std::string plugin_class ()

  Returns the name of the plugin.

3.17.1 Detailed Description

This class defines the snort plugin. The objective of this plugin is to get and publish alerts generated from the IDS Snort, by reading its output from a CSV log file.

Definition at line 96 of file snort.hpp.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 snort::snort (std::string *plugin_id*, std::map< std::string, std::string > *properties*)

Constructor of the class snort.

The constructor calls initialize_plugin() to load the properties.

Parameters

plugin_id Name of the plugin

properties Map of properties.

Definition at line 34 of file snort.cpp.

3.17.2.2 snort::∼snort (void) `[virtual]`

Destructor of the class snort.

Definition at line 47 of file snort.cpp.

3.17.3 Member Function Documentation

3.17.3.1 bool snort::generate_and_publish_information (DDSDynamicDataWriter ∗ *writer*, DDS_DynamicData ∗ *data*) `[virtual]`

Gets new alerts from Snort's logfile using update_alertsmap() and publishes them.

Gets new alerts from Snort's logfile using update_alertsmap() and publishes them if they are found.

Parameters

writer DDS Dynamic DataWriter.

data DDS Dynamic DataWriter to fill--using DDS Dynamic Data methdos.

Returns

True if everything was good.

Implements cc_plugin.

Definition at line 292 of file snort.cpp.

3.17.3.2 virtual std::string snort::plugin_class ( ) `[inline, virtual]`

Returns the name of the plugin.

Must be defined by each plugin and must be unique across plugins

Returns

Returns the name of the plugin.

Implements cc_plugin.

Definition at line 103 of file snort.hpp.

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

- plugins/snort/snort.hpp
- plugins/snort/snort.cpp

3.18 XML_parser Class Reference

This class parses all XML configurations files needed in Cave Canem.

`#include <xml_parser.hpp>`

Public Member Functions

- ~XML_parser ()

  Destructor of the class XML_parser. Destructor of the class XML_parser.

- bool parse_general_configuration_file (std::string cfg_file)

  Parses a general configuration file.

- bool parse_plugin_configuration_file (std::string cfg_file)

  Parses a plugin configuration file.

- void set_publishing_period (int publishing_period)

  Sets the publishing period of the general configuration.

- void set_domain_id (int domain_id)

  Sets the DDS Domain.

- void set_qos_file (std::string qos_file)

  Sets the file from which all the plugins will load their QoS.

- void set_qos_default_library (std::string qos_library)

  Sets the QoS default library for DDS Domain Participant and DDS Publisher.

- void set_qos_default_profile (std::string qos_profile)

  Sets the QoS default library for DDS Domain Participant and DDS Publisher.

- void set_plugin_library (std::string dir, std::list< std::string > plugin_list)

  Stores a new plugin library in the plugin_library_map_.

- void add_tmp_plugin (std::string tmp_plugin)

  Stores temporally the name of a plugin in a list, that will be included whithin a plugin library afterwards.

- std::list< std::string > get_tmp_plugin_list ()

  Returns the temporal list of plugins.

- void clear_tmp_plugin_list ()
- void set_plugin_properties (std::string plugin_name)
- void set_tmp_plugin_properties_dll (std::string dll)

  Sets the name of the dynamic library of a plugin.

- void set_tmp_plugin_properties_create_function (std::string create_function)

  Sets the create function of a plugin.

- void set_tmp_plugin_properties_qos_library (std::string qos_library)

  Sets the QoS library for a plugin.

- void set_tmp_plugin_properties_qos_profile (std::string qos_profile)

  Sets the QoS profile for a plugin.

- void set_tmp_plugin_properties_topic_name (std::string topic_name)
- void set_tmp_plugin_properties_add_element (std::string name, std::string value)

  Adds an element to the plugin elements list.

- void set_tmp_plugin_properties_type_code (struct DDS_TypeCode ∗type_code)

  Sets the typecode of the plugin.

- void set_tmp_plugin_properties_datawriter_qos (const struct DDS_DataWriterQos ∗datawriter_-qos)

  Sets the QoS for the DDS DataWriter if defined (the QoS).

- void set_tmp_plugin_properties_publishing_period (int publishing_period)
- struct DDS_TypeCode ∗ get_type_code_from_XML (struct DDS_XMLObject ∗xml, const char ∗type_name, struct DDS_XMLContext ∗context1)

  Returns a DDS Type Code given a DDS XML Object.

- cc_general_properties get_general_properties ()

  Returns the general properties of Cave Canem.

- cc_plugin_properties get_plugin_properties (std::string plugin_name)

  This method returns the properties of a plugin.

Static Public Member Functions

- static XML_parser ∗ get_singleton ()
  Provides access to the XML_parser singleton class.

3.18.1 Detailed Description

This class parses all XML configurations files needed in Cave Canem.

Definition at line 157 of file xml_parser.hpp.

3.18.2 Member Function Documentation

3.18.2.1 void XML_parser::add_tmp_plugin (std::string *tmp_plugin*)

Stores temporally the name of a plugin in a list, that will be included whithin a plugin library afterwards.

This method stores a plugin name in a temporal list that corresponds to the plugins contained whithin the directory of a plugin library.

Parameters

tmp_plugin The name of the plugin to be stored.

Definition at line 997 of file xml_parser.cpp.

3.18.2.2 void XML_parser::clear_tmp_plugin_list ( )

Cleans the temporal list of plugins.

Definition at line 1020 of file xml_parser.cpp.

3.18.2.3 cc_general_properties XML_parser::get_general_properties ( )

Returns the general properties of Cave Canem.

Returns a structure with the general properties of Cave Canem once the XML general configuration file has been parsed and read.

Returns

Definition at line 984 of file xml_parser.cpp.

3.18.2.4 cc_plugin_properties XML_parser::get_plugin_properties (std::string *plugin_name*)

This method returns the properties of a plugin.

It returns the properties of a plugin after a correct parsing.

Parameters

plugin_name Name of the plugin.

Returns

The properties of the plugin stored in a cc_plugin_properties structure.

Definition at line 1183 of file xml_parser.cpp.

3.18.2.5 static XML_parser∗ XML_parser::get_singleton ( ) `[inline, static]`

Provides access to the XML_parser singleton class.

Provides access to the XML_parser singleton class.

Returns

Returns a reference to the object.

Definition at line 181 of file xml_parser.hpp.

3.18.2.6 list< string > XML_parser::get_tmp_plugin_list ( )

Returns the temporal list of plugins.

Returns the temporal list of plugins to be loaded from a plugin library.

Returns

The list of plugins to be from a plugin library.

Definition at line 1009 of file xml_parser.cpp.

3.18.2.7 struct DDS_TypeCode ∗ XML_parser::get_type_code_from_XML (struct DDS_XMLObject ∗ *xml*, const char ∗ *type_name*, struct DDS_XMLContext ∗ *context1*) `[read]`

Returns a DDS Type Code given a DDS XML Object.

Returns a DDS Type Code looking for the XML definition of the data types contained whithin the tag <type_definition>.

Parameters

xml XML Object containing the data types.

type_name Name of the data type associated to the topic.

context1 XML Contect.

Returns

Definition at line 1707 of file xml_parser.cpp.

3.18.2.8 bool XML_parser::parse_general_configuration_file (std::string *cfg_file*)

Parses a general configuration file.

Defines the DTD of the general configuration file. Registers our custom extensions by using the register_-general_extensions() method and finally it parses the file.

Parameters

cfg_file General configuration file to be parsed.

Returns

Returns true if the configuration file was parsed correctly and false if it was not.

Definition at line 51 of file xml_parser.cpp.

3.18.2.9 bool XML_parser::parse_plugin_configuration_file (std::string *cfg_file*)

Parses a plugin configuration file.

Defines the DTD for the XML configuration file of a plugin. Registers our custom extensions by using the register_plugin_extensions() method and finally parses the file.

Parameters

cfg_file The plugin configuration file--including its route--that we want to parse.

Returns

Returns true if the parsing was right and false if it was not.

Definition at line 115 of file xml_parser.cpp.

3.18.2.10 void XML_parser::set_domain_id (int *domain_id*)

Sets the DDS Domain.

Sets the DDS Domain in which the plugins will publish information.

Parameters

domain_id DDS Domain ID.

Definition at line 920 of file xml_parser.cpp.

3.18.2.11 void XML_parser::set_plugin_library (std::string *dir*, std::list< std::string > *plugin_list*)

Stores a new plugin library in the plugin_library_map_.

Given a dir and a plugin list, this method stores them in the plugin_library_map_.

Parameters

dir Plugin library directory.

plugin_list List of plugins whithin the plugin library directory.

Definition at line 970 of file xml_parser.cpp.

3.18.2.12 void XML_parser::set_plugin_properties (std::string *plugin_name*)

Sets the plugin properties store in the temporal cc_plugin_properties structure.

Stores the plugin properties in the plugin_properties_map_ taking the contents of the tmp_plugin_-properties_ structure.

Parameters

plugin_name Name of the plugin.

Definition at line 1153 of file xml_parser.cpp.

3.18.2.13 void XML_parser::set_publishing_period (int *publishing_period*)

Sets the publishing period of the general configuration.

Sets the rate for calling the plugins to publish.

Parameters

publishing_period This parameter indicates the rate of calling the plugins to publish.

Definition at line 903 of file xml_parser.cpp.

3.18.2.14 void XML_parser::set_qos_default_library (std::string *qos_library*)

Sets the QoS default library for DDS Domain Participant and DDS Publisher.

Sets the QoS library for the DDS Domain Participant and the DDS Publisher.

Parameters

qos_library Name of the QoS library.

Definition at line 944 of file xml_parser.cpp.

3.18.2.15 void XML_parser::set_qos_default_profile (std::string *qos_profile*)

Sets the QoS default library for DDS Domain Participant and DDS Publisher.

Sets the QoS profile for the DDS Domain Participant and the DDS Publisher.

Parameters

qos_profile Name of the QoS profile.

Definition at line 956 of file xml_parser.cpp.

3.18.2.16 void XML_parser::set_qos_file (std::string *qos_file*)

Sets the file from which all the plugins will load their QoS.

Parameters

qos_file QoS definitions file.

Definition at line 931 of file xml_parser.cpp.

3.18.2.17 void XML_parser::set_tmp_plugin_properties_add_element (std::string *name*, std::string *value*)

Adds an element to the plugin elements list.

Adds a new element for the plugin list of elements in the temporal structure that stores the information of a plugin while it is being created.

Parameters

name	Name of the plugin element.
value	Value of the plugin element.

Definition at line 1086 of file xml_parser.cpp.

3.18.2.18 void XML_parser::set_tmp_plugin_properties_create_function (std::string *create_function*)

Sets the create function of a plugin.

Sets the name of the create function of the plugin in the temporal structure that stores the information of a plugin while it is being extrated.

Parameters

create_function	Name of the create function

Definition at line 1046 of file xml_parser.cpp.

3.18.2.19 void XML_parser::set_tmp_plugin_properties_datawriter_qos (const struct DDS_DataWriterQos $*$ *datawriter_qos*)

Sets the QoS for the DDS DataWriter if defined (the QoS).

Sets the QoS for the DDS DataWriter that the plugin will use if it is defined. If it is not defined, the qos_library and qos_profile parameters will decide the QoS of it.

Parameters

datawriter_qos	DDS DataWriter QoS for the plugin's datawriter.

Definition at line 1113 of file xml_parser.cpp.

3.18.2.20 void XML_parser::set_tmp_plugin_properties_dll (std::string *dll*)

Sets the name of the dynamic library of a plugin.

Sets the name of a dynamic library in the temporal structure that stores the information of the plugin while it is being extracted.

Parameters

dll	The dynamic library name.

Definition at line 1033 of file xml_parser.cpp.

3.18.2.21 void XML_parser::set_tmp_plugin_properties_publishing_period (int *publishing_period*)

Sets the publishing rate of the plugin.

Sets the publishing rate of the plugin in the temporal structure that stores the information of a plugin while it is being created.

Parameters

publishing_period Publishing rate of the plugin.

Definition at line 1126 of file xml_parser.cpp.

3.18.2.22 void XML_parser::set_tmp_plugin_properties_qos_library (std::string *qos_library*)

Sets the QoS library for a plugin.

Sets the name of the QoS library for a plugin in the temporal structure that stores the information of a plugin while it is being created.

Parameters

qos_library DDS QoS library for the plugin's DataWriter.

Definition at line 1059 of file xml_parser.cpp.

3.18.2.23 void XML_parser::set_tmp_plugin_properties_qos_profile (std::string *qos_profile*)

Sets the QoS profile for a plugin.

Sets the name of the QoS profile for a plugin in the temporal structure that stores the information of a plugin while it is being created.

Parameters

qos_profile DDS QoS profile for the plugin's DataWriter.

Definition at line 1072 of file xml_parser.cpp.

3.18.2.24 void XML_parser::set_tmp_plugin_properties_topic_name (std::string *topic_name*)

Sets the topic name--if defined--of the plugin.

Sets the topic name in the temporal structure that stores the information of a plugin while it is being created.

Parameters

topic_name Name of the topic.

Definition at line 1141 of file xml_parser.cpp.

3.18.2.25 void XML_parser::set_tmp_plugin_properties_type_code (struct DDS_TypeCode ∗ *type_code*)

Sets the typecode of the plugin.

Sets the typecode of the plugin in the temporal structure that stores the information of a plugin while it is being created.

Parameters

type_code Type Code of the plugin's data type.

Definition at line 1099 of file xml_parser.cpp.

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

- main/xml_parser.hpp
- main/xml_parser.cpp

Index