Fast DDS QoS Profiles Manager Documentation

Release 0.1.0

eProsima

Apr 28, 2023

INTRODUCTION

eProsima Fast DDS QoS Profiles Manager is a tool suite for the generation of Fast DDS XML configuration files. The suite provides both a Graphical User Interface (GUI) and a Command Line Interface (CLI).

eProsima Fast DDS QoS Profiles Manager provides a GUI to facilitate users the creation and modification of XML configuration files in an easy way. With an easy to use interface which allows to navigate between the configuration menus, the corresponding features can be configured and the XML to be deployed can be validated and created. Thus, the user can be agnostic of the specific XML format followed by *eProsima Fast DDS*. More information can be found in the *Fast DDS QoS Profiles Manager GUI* section.

CHAPTER

ONE

COMMAND LINE INTERFACE

eProsima Fast DDS QoS Profiles Manager CLI provides a command line interface which can be integrated in any script to automatically generate the deployment XML configuration files. More information about the CLI and how to use it can be found in the *Introduction* section. Also, examples of how to integrate the CLI in a script to generate deployment configuration files can be found in the tool suite repository.

CHAPTER

TWO

FAST DDS QOS PROFILES MANAGER LIBRARY

Both the CLI and GUI tools depends on a library which takes charge of handling and modifying the XML configuration files. The *Introduction* section summarizes the API exposed publicly. Any user can consequently develop any other integration tool adapted to its own development environment, although the most common use would be through the CLI or GUI tools provided by eProsima. The graph below shows the relationship between the different packages provided in the tool suite.

Fig. 1: Fast DDS QoS Profiles Manager Tool Suite

Warning: The tool suite is still in its development stage. For this reason, the API should not be considered stable, as API breaks may occur before the first official release. Furthermore, several features may not be implemented yet.

eProsima Fast DDS QoS Profiles Manager is a tool suite for the generation of Fast DDS XML configuration files. The suite provides both a Graphical User Interface (GUI) and a Command Line Interface (CLI).

2.1 Graphical User Interface

eProsima Fast DDS QoS Profiles Manager provides a GUI to facilitate users the creation and modification of XML configuration files in an easy way. With an easy to use interface which allows to navigate between the configuration menus, the corresponding features can be configured and the XML to be deployed can be validated and created. Thus, the user can be agnostic of the specific XML format followed by *eProsima Fast DDS*. More information can be found in the *Fast DDS QoS Profiles Manager GUI* section.

2.2 Command Line Interface

eProsima Fast DDS QoS Profiles Manager CLI provides a command line interface which can be integrated in any script to automatically generate the deployment XML configuration files. More information about the CLI and how to use it can be found in the *Introduction* section. Also, examples of how to integrate the CLI in a script to generate deployment configuration files can be found in the tool suite repository.

2.3 Fast DDS QoS Profiles Manager Library

Both the CLI and GUI tools depends on a library which takes charge of handling and modifying the XML configuration files. The *Introduction* section summarizes the API exposed publicly. Any user can consequently develop any other integration tool adapted to its own development environment, although the most common use would be through the CLI or GUI tools provided by eProsima. The graph below shows the relationship between the different packages provided in the tool suite.

Fig. 2: Fast DDS QoS Profiles Manager Tool Suite

Warning: The tool suite is still in its development stage. For this reason, the API should not be considered stable, as API breaks may occur before the first official release. Furthermore, several features may not be implemented yet.

2.4 Linux installation from sources

The *eProsima Fast DDS QoS Profiles Manager* tool suite is composed by several packages as described in the *Introduction*. It is organized as follows:

The tool suite can either be built using colcon or CMake. With CMake, the tool suite build and environment source processes must be done manually and following the tool package dependencies, whereas colcon automatizes these two actions with a single command. For that reason, the use of colcon is highly recommended.

2.4.1 Prerequisites

The installation of the tool suite in a Linux environment from sources requires some *Build and development tools* to be installed in the system.

Build and development tools

The following packages are required to build the *eProsima Fast DDS QoS Profiles Manager* tool suite from sources. Install CMake, g++, pip3 and git using the package manager of the appropriate Linux distribution. For example, on Ubuntu use the command:

```
sudo apt update
sudo apt install cmake g++ python3-pip git
```

Recommended build tools (optional)

Despite the fact that the *eProsima Fast DDS QoS Profiles Manager* tool suite packages can be built using CMake, colcon simplifies the task substantially. colcon is a command line tool based on CMake aimed at building sets of software packages. Install colcon by executing the following command:

```
pip3 install -U colcon-common-extensions
```

Note: Mind that under non-root users, `pip3` may install python `colcon` executable in `$HOME/.local/bin`, for instance when running with `--user`. To be able to run this application, make sure that `pip3` binary installation directory is in your `$PATH` (`$HOME/.local/bin` is normally introduced while login on an interactive non-root shell).

2.4.2 Dependencies

This section deals with the required dependencies for each package included in the tool suite.

Important: The *Library* package is required for building the remaining tool packages.

Library tool

The *eProsima Fast DDS QoS Profiles Manager Library* requires Xerces. The package test suite depends on GTest.

Xerces

Xerces C++ is an Apache project that provides XML parsers and related components solutions for development. In Ubuntu, it can be installed running:

```
sudo apt install libxerces-c-dev
```

GTest (optional)

GTest is a unit testing library for C++. By default, *eProsima Fast DDS QoS Profiles Manager* does not compile tests. It is possible to activate them with the opportune CMake configuration options when calling colcon or CMake.

GTest installation can be performed either following the GTest installation instructions, or including the Gtest repository in the workspace directory, running:

```
git clone https://github.com/google/googletest ~/fastdds_qosprofman_ws/src/googletest-
↪distribution
```

Note: If the colcon deployment is followed, and GTest has been installed in the workspace directory, the GTest dependency needs to be included in the *Library* `colcon.pkg` file. Please check the *colcon build procedure* for further information.

CLI tool

The *eProsima Fast DDS QoS Profiles Manager CLI* depends on the *Library tool* and also on the following package.

Docopt

docopt is a CLI description language that defines the interface of the command line applications and generates a parser for it automatically. In Ubuntu, the dependency can be installed running:

```
sudo apt install libdocopt-dev
```

GUI tool

Warning: This tool is still in development process.

Documentation tool

The *eProsima Fast DDS QoS Profiles Manager Documentation* depends on the *Library tool* and also on the following packages.

Doxygen

Doxygen is a C++ documentation generator. It is required to build the *Library API reference*. In Ubuntu, the dependency can be installed running:

```
sudo apt install doxygen
```

Python3 virtual environment (recommended)

It is recommended to install the *Documentation* tool dependencies in a python3 virtual environment. That would avoid polluting the user's installation setup.

Deploy the project in a python3 virtual environment by running:

```
# Python3 virtual environment installation
sudo apt install python3.10-venv

# Virtual environment deployment
cd <path_to_virtual_environment_directory>
python3 -m venv fastdds_qosprofman_venv
source fastdds_qosprofman_venv/bin/activate
```

Note: As colcon performs a recursive search to build packages inside the workspace directory, it is recommended to deploy the virtual environment outside of the workspace directory in order to avoid conflicts.

Python3 package dependencies

The *Documentation* tool has several python3 dependencies described in the `requirements.txt` project file.

```
breathe==4.35.0
doc8==0.10.1
docutils==0.16.0
setuptools==67.6.0
sphinx_rtd_theme==0.5.2
sphinx-tabs==3.2.0
sphinx==4.3.1
sphinxcontrib-imagehelper==1.1.1
sphinxcontrib-mermaid==0.8.1
sphinxcontrib-plantuml==0.22
sphinxcontrib.spelling==7.2.1
```

Important: Once *the repository is downloaded*, the `requirements.txt` dependencies file can be installed by running:

```
pip3 install -r ~/fastdds_qosprofman_ws/src/Fast-DDS-QoS-Profiles-Manager/docs/
↪requirements.txt
```

2.4.3 Tool suite build process

Create a workspace directory and download the project:

```
mkdir -p ~/fastdds_qosprofman_ws/src
cd ~/fastdds_qosprofman_ws/src
git clone https://github.com/eProsima/Fast-DDS-QoS-Profiles-Manager.git
```

Building with colcon

eProsima Fast DDS QoS Profiles Manager complete tool suite is build using colcon by running:

```
cd ~/fastdds_qosprofman_ws
colcon build
```

Note: Being based on CMake it is possible to pass CMake configuration options to the `colcon build` command. These options can be also passed through a colcon.meta file.

For instance, the *Library* and *Documentation* test suites building is enabled by building with the CMake option *EPROSIMA_BUILD_TESTS*. If GTest repository has been included in the workspace directory, add the GTest dependency (`"googletest-distribution"`) in the *Library* colcon.pkg file. Also, colcon provides certain arguments to perform package selection, excluding undesired packages or including specific packages in the build process.

Thus, the following command would build the *Library* and *Documentation* tools with its test suite.

```
colcon build --packages-up-to fastdds_qos_profiles_manager_docs --cmake-args -DEPROSIMA_
↪BUILD_TESTS=ON
```

Building with CMake

eProsima Fast DDS QoS Profiles Manager tool suite is build using CMake following these steps:

Library

```
mkdir -p ~/fastdds_qosprofman_ws/build/lib
cd ~/fastdds_qosprofman_ws/build/lib
cmake ../../src/Fast-DDS-QoS-Profiles-Manager/lib -DCMAKE_INSTALL_PREFIX=../../install/
↪lib
cmake --build . --target install
```

Note: If the *Library* test suite compilation is needed, the enable test build flag *EPROSIMA_BUILD_TESTS* should be included while configuring the building of the *Library* package:

```
cmake ../../src/Fast-DDS-QoS-Profiles-Manager/lib -DCMAKE_INSTALL_PREFIX=../../install/
↪lib -DEPROSIMA_BUILD_TESTS=ON
```

CLI

```
mkdir -p ~/fastdds_qosprofman_ws/build/cli
cd ~/fastdds_qosprofman_ws/build/cli
export CMAKE_PREFIX_PATH=${CMAKE_PREFIX_PATH}:~/fastdds_qosprofman_ws/install/lib
```

(continues on next page)

(continued from previous page)

```
cmake ../../src/Fast-DDS-QoS-Profiles-Manager/cli -DCMAKE_INSTALL_PREFIX=../../install/
↪cli
cmake --build . --target install
```

Note: Note that the already installed *Library* path has been sourced in the *CLI* build process with the command:

```
export CMAKE_PREFIX_PATH=${CMAKE_PREFIX_PATH}:~/fastdds_qosprofman_ws/install/lib
```

Documentation

Important: If the *Library* path has been sourced in the previous step, there is no need to include it again in the environment variable `CMAKE_PREFIX_PATH`. If, on the other hand, it has not been sourced previously, as long as it is required also to build the *Documentation* package, it needs to be sourced.

```
mkdir -p ~/fastdds_qosprofman_ws/build/docs
cd ~/fastdds_qosprofman_ws/build/docs
cmake ../../src/Fast-DDS-QoS-Profiles-Manager/docs -DCMAKE_INSTALL_PREFIX=../../install/
↪docs
cmake --build . --target install
```

2.4.4 Run CLI example

The `fastddsqosprof` executable file is generated in the installation path. In order to run the CLI, please, ensure to source the environment:

Built with colcon

```
source ~/fastdds_qosprofman_ws/install/setup.bash
```

Built with CMake

```
export LD_LIBRARY_PATH=~/fastdds_qosprofman_ws/install/lib/lib
export PATH=$PATH:~/fastdds_qosprofman_ws/install/cli/bin
```

CLI should display its software version by running:

```
fastddsqosprof -v
```

Next steps: please look at the CLI examples.

2.4.5 Local documentation

The *Documentation* generated can be opened in a web browser locally by running:

Built with colcon

```
xdg-open ~/fastdds_qosprofman_ws/install/fastdds_qos_profiles_manager_docs/docs/fastdds_
↪qos_profiles_manager_docs/sphinx/html/index.html
```

Built with CMake

```
xdg-open ~/fastdds_qosprofman_ws/install/docs/docs/fastdds_qos_profiles_manager_docs/
↪sphinx/html/index.html
```

2.5 Fast DDS QoS Profiles Manager

Warning: *eProsima Fast DDS QoS Profiles Manager Graphical User Interface* is still under development.

2.6 Introduction

eProsima Fast DDS QoS Profiles Manager CLI is a Command Line Interface to easily interact with the *Fast DDS QoS Profiles Manager Library*. Once the CLI has been installed and sourced (read the *Installation manual* section for more information), the CLI executable, `fastddsqosprof` can be used.

Note: Currently *eProsima Fast DDS QoS Profiles Manager CLI* is only supported in Linux systems.

2.7 Usage

eProsima Fast DDS QoS Profiles Manager CLI general usage is:

```
fastddsqosprof <file> <verb> <element> [<value>]
```

After the executable, the first parameter expected is the name of the XML configuration file that is going to be read/modified. The `<file>` path can be provided both in an absolute or relative manner.

The CLI supported verbs are explained in the table below:

Verb	Description
`clear`	Remove configuration parameter from XML configuration file.
`compare`	Compare two different configuration files.
`help`	Show CLI usage.
`print`	Print configuration parameter from XML configuration file.
`query`	Extract information related to parameter lists. This might be useful for automated scripts that leverages the CLI to automatically generate the deployment configuration files.
`set`	Add a configuration parameter into the XML configuration file.
`validate`	Validate given XML configuration file against Fast DDS XSD schema.

The next CLI argument, `<element>`, specifies the configuration parameter to be read/modified. The access to the specific configuration parameter is done following the structure explained in *Configuration parameter access* section. If the parameter is being `set`, the corresponding configuration `value` has to be passed to the CLI.

Important: At any time, passing `help` (or the also supported flags `--help` and `-h`) as the last value in the CLI will provide the usage for the specific element/subelement which is being configured. For instance, the following command

```
fastddsqosprof file.xml set participant -h
```

will show the usage corresponding to the DomainParticipant profiles configuration. More information can be found in *help verb*.

The previous usage is the most common among the supported verbs. However, some of the verbs have a specific usage which is shown below.

Warning: Currently only `set` and `help` verbs are supported.

2.7.1 Compare usage

Warning: `compare` verb is not yet supported.

2.7.2 Query usage

Warning: `query` verb is not yet supported.

2.7.3 Validate usage

Warning: `validate` verb is not yet supported.

2.8 Supported verbs

Previous *section* described in a general manner the CLI supported verbs. This section will explain in detail the behavior of each of these verbs.

2.8.1 clear verb

Warning: Fast DDS QoS Profiles Manager `clear` verb not yet supported.

2.8.2 compare verb

Warning: Fast DDS QoS Profiles Manager `compare` verb not yet supported.

2.8.3 help verb

Passing to the CLI the verb `help` as the last argument would display the corresponding subelement CLI usage. Besides `help`, the flag `--help` and the shorthand `-h` are also supported.

The snippet below shows different examples:

```
$ fastddsqosprof --help
  > Generic CLI usage

$ fastddsqosprof file help
  > Generic CLI usage

$ fastddsqosprof file set -h
  > Specific SET command usage

$ fastddsqosprof file set datareader help
  > Specific DataReader element usage

$ fastddsqosprof file set datareader[profile].default_profile -h
  > Specific default profile attribute usage
```

2.8.4 print verb

Warning: Fast DDS QoS Profiles Manager `print` verb not yet supported.

2.8.5 query verb

Warning: Fast DDS QoS Profiles Manager `query` verb not yet supported.

2.8.6 set verb

Passing to the CLI the verb `set` would include the given `value` in the selected XML `element`. The verb is the third argument passed to the CLI (see *Usage* section). The element is given by a dot-separated string accessing the configuration parameter to be set (see *Configuration parameter access* section). The set command requires usually at least the value desired for the specified element.

Note: There are exceptions in which the verb `set` does not need a `value`, as in the `default_profile` attribute cases for `datareader`, `datawriter` and `participant` entities.

The snippet below shows different examples:

```
$ fastddsqosprof file set datareader[dr_profile].qos.durability volatile

$ fastddsqosprof file set datareader[dr_profile].default_profile

$ fastddsqosprof file set datawriter[dw_profile].qos.reliability.duration.max_blocking_
↪time 1 100

$ fastddsqosprof file set participant[p_profile].name "My participant"

$ fastddsqosprof file set participant[p_profile].use_builtin_transports true

$ fastddsqosprof file set participant[p_profile].user_transports[] td_profile

$ fastddsqosprof file set transport_descriptor[td_profile].kind shm
```

The following XML configuration file represents the result of the consecutive CLI commands run.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<dds xmlns="http://www.eprosima.com/XMLSchemas/fastRTPS_Profiles">
    <profiles>
        <data_reader is_default_profile="true" profile_name="dr_profile">
            <qos>
                <durability>
                    <kind>VOLATILE</kind>
                </durability>
            </qos>
        </data_reader>
        <data_writer profile_name="dw_profile">
            <qos>
                <reliability>
                    <max_blocking_time>
                        <sec>1</sec>
                        <nanosec>100</nanosec>
                    </max_blocking_time>
                </reliability>
            </qos>
        </data_writer>
        <participant profile_name="p_profile">
            <rtps>
                <name>My participant</name>
                <useBuiltinTransports>true</useBuiltinTransports>
                <userTransports>
                    <transport_id>td_profile</transport_id>
                </userTransports>
            </rtps>
        </participant>
        <transport_descriptors>
```

(continues on next page)

(continued from previous page)

```
            <transport_descriptor>
                <transport_id>td_profile</transport_id>
                <type>SHM</type>
            </transport_descriptor>
        </transport_descriptors>
    </profiles>
</dds>
```

2.8.7 `validate` verb

Warning: Fast DDS QoS Profiles Manager `validate` verb not yet supported.

2.9 Configuration parameter access

The access to each configuration parameter is done sequentially using a dot-separated entity string. This section explains with detail how to access the different configuration parameters.

Note: **Disclaimer**: This section does not explain the meaning of each configuration parameter. Please, refer to Fast DDS documentation for a more in-depth understanding of the configuration parameters.

The main supported elements that can be configured using the CLI are:

Main element	Description
`participant`	DomainParticipant profile configuration.
`datareader`	DataReader profile configuration.
`datawriter`	DataWriter profile configuration.
`topic`	Topic profile configuration.
`transport_descriptor`	Transport descriptor configuration.
`intraprocess`	Intra-process delivery communication configuration.
`log`	Fast DDS log module configuration.
`types`	Dynamic types configuration.

2.9.1 Configuration parameter classification

Every configurable parameter can be classified using the following concepts:

- **Simple parameters**: these parameters are final configurable elements. Access through the CLI is done by means of a dot-separated string until reaching the final element which is always a simple parameter.
- **Complex parameters**: these parameters contain another configurable parameters. Consequently, complex parameters are not final and another subelement can be configured within them.
- **Parameter lists**: some parameters allows for an undefined number of elements. These collections should be accessed by means of a given integer index between brackets to select the specific collection element. Both positive and negative indexes are allowed to access the collection from both ends. If no index is provided, the complete collection is selected (either to be printed or cleared, for instance). To push a new element into

the collection, no index should be provided to the `set` command. Parameter lists can be simple or complex depending on the contained element.

- **Parameter maps**: these parameters are a specific collection kind whose access is performed using a `key` instead of an integer index. Parameter maps can be simple or complex depending on the contained element.

Note: The verb `query` can only be used with parameter lists or maps. Querying a non-collection parameter would result in an error being displayed.

2.9.2 Main supported elements configurable parameters

The sections below explain the configurable parameters in each of the main supported elements.

Participant profiles configuration access

This section explains how to access the different configurable parameters in a DomainParticipant profile. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The DomainParticipant profile is accessed through the `participant` element. The profile name is mandatory and must be passed within brackets. Whitespaces are allowed in the profile name if the profile name is quoted. Consequently, `participant` is a complex parameter map according to the *Configuration parameter classification* with the profile name as `key`.

Important: The only case when the profile name is not required is when printing or clearing the default profile attribute.

```
fastddsqosprof <file> <verb> participant[profile_name].<subelement>
fastddsqosprof <file> <verb> participant["profile name"].<subelement>
```

The currently supported DomainParticipant configurable `<subelements>` are explained in the table below.

`<subelement>`	CLI access	*Configuration parameter classification*	Valid `set` values
Builtin configuration	`participant[profile_name].builtin`	Complex parameter. Please, refer to *Builtin configuration* section for more information.	N/A
Default external unicast locators configuration	`participant[profile_name].external_locators.default_unicast[(index)]`	Complex parameter list. Please, refer to *External locators configuration* section for more information.	N/A
Default profile attribute	`participant[(profile_name)].default_profile`	Simple parameter	`true` `false` (`profile_name` is only required with verb `set`).
Name element	`participant[profile_name].name`	Simple parameter	`string`
Use builtin transports flag	`participant[profile_name].use_builtin_transports`	Simple parameter	`true` `false`
User transports list	`participant[profile_name].user_transports[(index)]`	Simple parameter list	`string`

Warning: Any other configurable `<subelement>` is not yet supported.

Builtin configuration

The DomainParticipant builtin configuration is accessed through the `builtin` element which is a complex parameter according to the *Configuration parameter classification*. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The currently supported builtin configurable `<subelements>` are explained in the table below.

`<subelement>`	CLI access	*Configuration parameter classification*	Valid `set` value
Discovery configuration	`participant[profile_name].builtin.discovery_config`	Complex parameter. Please, refer to *Discovery configuration* section for more information.	N/A
Initial peers configuration	`participant[profile_name].builtin.locators.initial_peers[(index)]`	Complex parameter list. Please, refer to *Locators configuration* section for more information.	N/A
Metatraffic external unicast locators configuration	`participant[profile_name].builtin.external_locators.metatraffic_unicast[(index)]`	Complex parameter list. Please, refer to *External locators configuration* section for more information.	N/A

Warning: Any other configurable `subelement` is not yet supported.

Discovery configuration

The DomainParticipant builtin discovery configuration is accessed through the `discovery_config` element which is a complex parameter according to the *Configuration parameter classification*. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The currently supported discovery configurable `<subelements>` are explained in the table below.

`<subelement>`	CLI access	*Configuration parameter classification*	Valid `set` value
Lease announcements	`participant[profile_name].builtin.discovery_config.duration.announcements`	Complex parameter. Please, refer to *Duration type configuration* section for more information.	N/A
Lease duration	`participant[profile_name].builtin.discovery_config.duration.lease`	Complex parameter. Please, refer to *Duration type configuration* section for more information.	N/A

Warning: Any other configurable `subelement` is not yet supported.

DataReader profiles configuration access

This section explains how to access the different configurable parameters in a DataReader profile. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The DataReader profile is accessed through the `datareader` element. The profile name is mandatory and must be passed within brackets. Whitespaces are allowed in the profile name if the profile name is quoted. Consequently, `datareader` is a complex parameter map according to the *Configuration parameter classification* with the profile name as `key`.

Important: The only case when the profile name is not required is when printing or clearing the default profile attribute.

```
fastddsqosprof <file> <verb> datareader[profile_name].<subelement>
fastddsqosprof <file> <verb> datareader["profile name"].<subelement>
```

The currently supported DataReader configurable `<subelements>` are explained in the table below.

`<subelement>`	CLI access	*Configuration parameter classification*	Valid `set` values
Default profile attribute	`datareader[(profile_name)].default_profile`	Simple parameter	`true` `false` (`profile_name` is only required with verb `set`).
QoS configuration	`datareader[profile_name].qos`	Complex parameter. Please, refer to *QoS configuration* section for more information.	N/A

Warning: Any other configurable `<subelement>` is not yet supported.

DataWriter profiles configuration access

This section explains how to access the different configurable parameters in a DataWriter profile. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The DataWriter profile is accessed through the `datawriter` element. The profile name is mandatory and must be passed within brackets. Whitespaces are allowed in the profile name if the profile name is quoted. Consequently, `datawriter` is a complex parameter map according to the *Configuration parameter classification* with the profile name as `key`.

Important: The only case when the profile name is not required is when printing or clearing the default profile attribute.

```
fastddsqosprof <file> <verb> datawriter[profile_name].<subelement>
fastddsqosprof <file> <verb> datawriter["profile name"].<subelement>
```

The currently supported DataWriter configurable `<subelements>` are explained in the table below.

<subelement>	CLI access	*Configuration parameter classification*	Valid `set` values
Default profile attribute	`datawriter[(profile_name)].default_profile`	Simple parameter	`true` `false` (`profile_name` is only required with verb `set`).
QoS configuration	`datawriter[profile_name].qos`	Complex parameter. Please, refer to *QoS configuration* section for more information.	N/A

Warning: Any other configurable <subelement> is not yet supported.

Topic profiles configuration access

Warning: Topic profiles configuration not yet supported.

Transport Descriptor configuration access

This section explains how to access the different configurable parameters in a transport descriptor profile. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The transport descriptor profile is accessed through the `transport_descriptor` element. The profile name is mandatory and must be passed within brackets. Whitespaces are allowed in the profile name if the profile name is quoted. The transport descriptor profile name matches the `transport_id` name defined in Fast DDS documentation. Consequently, `transport_descriptor` is a complex parameter map according to the *Configuration parameter classification* with the profile name as `key`.

```
fastddsqosprof <file> <verb> transport_descriptor[profile_name].<subelement>
fastddsqosprof <file> <verb> transport_descriptor["profile name"].<subelement>
```

The currently supported transport descriptor configurable <subelements> are explained in the table below.

<subelement>	CLI access	*Configuration parameter classification*	Valid `set` values
Transport descriptor kind	`transport_descriptor[profile_name].kind`	Simple parameter	`udp_v4` (Default value) `udp_v6` `tcp_v4` `tcp_v6` `shm`
Interface Whitelist	`transport_descriptor[profile_name].interface_whitelist[]`	Simple parameter list	IP or DNS address

Warning: Any other configurable <subelement> is not yet supported.

Intra-process delivery configuration access

Warning: Intra-process delivery configuration not yet supported.

Log module configuration access

Warning: Fast DDS Log module configuration not yet supported.

Dynamic Types configuration access

Warning: Dynamic types configuration not yet supported.

Common configuration elements access

This section explains how to access the different configurable parameters in common subelements shared among different elements. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

Allocations configuration

Warning: Allocations configuration not yet supported.

Duration type configuration

This section explains how to access the different configurable parameters in a duration type configuration. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The duration type configuration is accessed through the `duration` element which is a complex parameter according to the *Configuration parameter classification*. The duration type configurable `<subelements>` are explained in the table below.

`subelement`	CLI access	*Configuration parameter classification*	Valid `set` values
Seconds	`duration.<duration_type>.sec`	Simple parameter	`int32_t`
Nanoseconds	`duration.<duration_type>.nanosec`	Simple parameter	`uint32_t`

Duration type can also be set in the following supported usages:

- Setting an infinite duration: `duration.<duration_type> infinite`
- Setting the duration directly: `duration.<duration_type> <sec_value> <nanosec_value>`

Note: The duration can be directly set also by passing only the `<sec_value>` argument: `duration. <duration_type> <sec_value>`

The `<duration_type>` element is defined for each specific element. The list below includes the currently supported `<duration_type>` elements:

- *Reliability QoS maximum blocking time*.
- *Participant builtin discovery configuration lease duration*.
- *Participant builtin discovery configuration lease announcements*.

Warning: Any other `<duration_type>` is not yet supported.

History Memory Policy configuration

Warning: History Memory Policy configuration not yet supported.

Locators configuration

This section explains how to access the different configurable parameters in a locator list configuration. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The locator list configuration is accessed through the `locators` element which is a complex parameter list according to the *Configuration parameter classification*. The locator list configurable `<subelements>` are explained in the table below.

`subelement`	CLI access	*Configuration parameter classification*	Valid `set` values
Address element	`locators. <locator_type>[(index)].address`	Simple parameter	IP or DNS address
Port element	`locators. <locator_type>[(index)].port`	Simple parameter	`uint16_t`

Note: As explained in the *Configuration parameter classification* section, this collection should be accessed by means of a given integer index between brackets to select the specific collection element. Both positive and negative indexes are allowed to access the collection from both ends. If no index is provided, the complete collection is selected (either to be printed or cleared, for instance). To push a new element into the collection, no index should be provided to the `set` command.

The `<locator_type>` element is defined for each specific element. The list below includes the currently supported `<locator_type>` elements:

- *Participant builtin initial peers*.

Warning: Any other <locator_type> is not yet supported.

External locators configuration

This section explains how to access the different configurable parameters in an external locator list configuration. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

The external locator list configuration is accessed through the `external_locators` element which is a complex parameter list according to the *Configuration parameter classification*. The external locator list configurable <subelements> are explained in the table below.

subelement	CLI access	*Configuration parameter classification*	Valid `set` values
Address element	external_locators. <external_locator_type>[(index)].address	Simple parameter	IP or DNS address
Externality attribute	external_locators. <external_locator_type>[(index)]. externality	Simple parameter	uint8_t
Port element	external_locators. <external_locator_type>[(index)].port	Simple parameter	uint16_t

Note: As explained in the *Configuration parameter classification* section, this collection should be accessed by means of a given integer index between brackets to select the specific collection element. Both positive and negative indexes are allowed to access the collection from both ends. If no index is provided, the complete collection is selected (either to be printed or cleared, for instance). To push a new element into the collection, no index should be provided to the `set` command.

The <external_locator_type> element is defined for each specific element. The list below includes the currently supported <external_locator_type> elements:

- *Participant builtin metatraffic external unicast locators configuration*.
- *Participant default external unicast locators configuration*.

Warning: Any other <external_locator_type> is not yet supported.

Property Policy configuration

Warning: Property Policy configuration not yet supported.

QoS configuration

This section explains how to access the different configurable parameters in a Quality of Service (QoS) profile. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Fast DDS documentation for that information.

Note: The CLI QoS element also includes the Topic Type defined in Fast DDS documentation.

The QoS configuration is accessed through the `qos` element which is a complex parameter according to the *Configuration parameter classification*. The currently supported QoS configurable `<subelements>` are explained in the table below.

`<subelement>`	CLI access	*Configuration parameter classification*	Valid `set` value
Durability QoS Policy	`qos.durability`	Simple parameter	`volatile` `transient_local` `transient` `persistent`
Reliability QoS Policy	`qos.reliability`	Complex parameter. Please, refer to *Reliability QoS Policy configuration* section for more information.	N/A

Warning: Any other configurable `subelement` is not yet supported.

Reliability QoS Policy configuration

This section explains how to access the different configurable parameters in the Reliability QoS Policy. The scope of this section does not include explaining the usage of these specific parameters. Please, refer to Reliability QoS Policy section in Fast DDS documentation for that information.

The Reliability QoS Policy is accessed through the `reliability` element which is a complex parameter according to the *Configuration parameter classification*. The Reliability QoS Policy configurable `<subelements>` are explained in the table below.

`<subelement>`	CLI access	*Configuration parameter classification*	Valid `set` values
Reliability kind	`reliability.kind`	Simple parameter	`best_effort` `reliable`
Reliability maximum blocking time	`reliability.duration.max_blocking_time`	Complex parameter. Please, refer to *Duration type configuration* for more information.	N/A

2.10 Introduction

This section presents the public API provided by *eProsima Fast DDS QoS Profiles Manager Library*.

2.11 QoS Profiles Manager

namespace **eprosima**

namespace **qosprof**

Functions

void **initialize**(const std::string &xml_file)

Initialize eProsima QoS Profiles Manager Library XML workspace. This workspace gets related to the given XML file, and all the Library API calls would be related to that XML file.

Parameters **xml_file** – Absolute/relative path to the XML file.

Throws *Error* – *Exception* if the workspace could not be initialized, or it was already initialized.

void **terminate**()

Terminate the initialized eProsima QoS Profiles Manager Library XML workspace. This function must be called before initializing a new workspace. If changes required to be applied in the XML configuration file, this method would also apply them in filesystem. If error occurred or resultant XML configuration is not valid, the XML configuration will not be modified.

Throws *Error* – *Exception* if the XML file could not be written in filesystem

2.12 Domain Participant

namespace **domain_participant**

Functions

std::string **print**(const std::string &profile_id)

Parse XML file and print specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier. If empty every Domain Participant profile is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

Returns std::string XML section containing the specific Domain Participant profile.

std::string **print_default_profile**()

Parse XML file and print the name of the Domain Participant default profile.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if there is no default Domain Participant profile in the XML file.

Returns std::string Domain Participant default profile.

std::string **print_domain_id**(const std::string &profile_id)
Parse XML file and print the Domain Participant Domain ID.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the Domain ID has not been set in the profile.

Returns std::string Domain Participant Domain ID.

std::string **print_name**(const std::string &profile_id)
Parse XML file and print the Domain Participant name.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the name has not been set in the profile.

Returns std::string Domain Participant name.

std::string **print_ignore_non_matching_locators**(const std::string &profile_id)
Parse XML file and print the Domain Participant ignore non matching locators flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the ignore non matching locators flag has not been set in the profile.

Returns std::string Domain Participant ignore non matching locators flag.

std::string **print_send_socket_buffer_size**(const std::string &profile_id)
Parse XML file and print the Domain Participant send socket buffer size.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the send socket buffer size has not been set in the profile.

Returns std::string Domain Participant send socket buffer size.

std::string **print_listen_socket_buffer_size**(const std::string &profile_id)
Parse XML file and print the Domain Participant listen socket buffer size.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the listen socket buffer size has not been set in the profile.

Returns std::string Domain Participant listen socket buffer size.

std::string **print_participant_id**(const std::string &profile_id)
Parse XML file and print the Domain Participant ID.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the Domain Participant ID has not been set in the profile.

Returns std::string Domain Participant ID.

std::string **print_user_transports**(const std::string &profile_id, const std::string &index)
Parse XML file and print the Domain Participant specific user transport element.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific user transport.

std::string **print_use_builtin_transports**(const std::string &profile_id)
Parse XML file and print the Domain Participant use builtin transports flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the use builtin transports flag has not been set in the profile.

Returns std::string Domain Participant use builtin transports flag.

std::string **print_user_data**(const std::string &profile_id, const std::string &index)
Parse XML file and print the Domain Participant specific user data element.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list does not contain any element in index position.

- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific user data.

std::string **`print_prefix`**(const std::string &profile_id)
Parse XML file and print the Domain Participant GUID prefix.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the GUID prefix has not been set in the profile.

Returns std::string Domain Participant GUID prefix.

uint32_t **`size`**()
Number of Domain Participant profiles contained in the XML file.

Throws `Error` – *Exception* if the workspace was not initialized.

Returns uint32_t Number of Domain Participant profiles in the XML file.

std::vector<std::string> **`keys`**()
List of the identifiers for every Domain Participant profile in the XML file.

Throws `Error` – *Exception* if the workspace was not initialized.

Returns std::vector<std::string> Identifier list. Empty list if there are no Domain Participant profiles.

uint32_t **`user_transports_size`**(const std::string &profile_id)
Number of user transports in the Domain Participant profile.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file.

Returns uint32_t Number of user transports in the list.

uint32_t **`user_data_size`**(const std::string &profile_id)
Number of user data elements in the Domain Participant profile.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file.

Returns uint32_t Number of user data elements in the list.

void **`clear`**(const std::string &profile_id)
Remove specific Domain Participant profile.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier. If empty, every Domain Participant profile is deleted.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_default_profile**()
Remove the is_default_profile attribute from the default Domain Participant profile.

Throws *Error* – *Exception* if the workspace was not initialized.

void **clear_domain_id**(const std::string &profile_id)
Remove Domain ID from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_name**(const std::string &profile_id)
Remove Domain Participant name from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_ignore_non_matching_locators**(const std::string &profile_id)
Remove ignore non matching locators flag from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_send_socket_buffer_size**(const std::string &profile_id)
Remove send socket buffer size from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_listen_socket_buffer_size**(const std::string &profile_id)
Remove listen socket buffer size from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_participant_id**(const std::string &profile_id)
Remove Domain Participant ID from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_user_transports**(const std::string &profile_id, const std::string &index)
Remove specific user transport from specific Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_use_builtin_transports**(const std::string &profile_id)
Remove use builtin transports flag from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_user_data**(const std::string &profile_id, const std::string &index)
Remove specific user data from specific Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_prefix**(const std::string &profile_id)
Remove GUID prefix from specific Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **`set_default_profile`**(const std::string &profile_id)

Set the given Domain Participant profile as the default profile. As only one default profile is allowed, if another default profile exists, it is overridden and its is_default_profile attribute is set to false.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **`set_domain_id`**(const std::string &profile_id, const std::string &domain_id)

Set the Domain Participant domain ID.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`domain_id`** – **[in]** Domain Participant domain ID.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided domain ID is not valid.

void **`set_name`**(const std::string &profile_id, const std::string &name)

Set the Domain Participant name.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`name`** – **[in]** Domain Participant name.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided name is not valid.

void **`set_ignore_non_matching_locators`**(const std::string &profile_id, const std::string &ignore_non_matching_locators)

Set the Domain Participant ignore non matching locators flag.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`ignore_non_matching_locators`** – **[in]** Ignore non matching locators flag.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided flag value is not valid.

void **set_send_socket_buffer_size**(const std::string &profile_id, const std::string &send_socket_buffer_size)

Set the Domain Participant send socket buffer size.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **send_socket_buffer_size** – **[in]** Size of the buffer in the socket used for sending data.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided size is not valid.

void **set_listen_socket_buffer_size**(const std::string &profile_id, const std::string &listen_socket_buffer_size)

Set the Domain Participant listen socket buffer size.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **listen_socket_buffer_size** – **[in]** Size of the buffer in the socket used for listening data.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided size is not valid.

void **set_participant_id**(const std::string &profile_id, const std::string &participant_id)

Set the Domain Participant ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **participant_id** – **[in]** Domain Participant ID.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided ID is not valid.

void **set_use_builtin_transports**(const std::string &profile_id, const std::string &use_builtin_transports)

Set the Domain Participant use builtin transports flag.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **use_builtin_transports** – **[in]** Use builtin transports flag.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided flag value is not valid.

void **set_prefix**(const std::string &profile_id, const std::string &prefix)

Set the Domain Participant GUID prefix.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Domain Participant GUID prefix.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided GUID prefix is not valid.

void **set_user_transports**(const std::string &profile_id, const std::string &transport_id, const std::string &index)

Append a user transport to the collection or update user transport element in the collection.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **transport_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided Transport descriptor profile identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_user_data**(const std::string &profile_id, const std::string &user_data, const std::string &index)

Append user data or update specific user data.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **user_data** – **[in]** User data to be updated.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided user data is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **allocations**

Functions

std::string **print**(const std::string &profile_id)

Parse XML file and print specific Domain Participant allocations configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the allocations configuration does not exist.

Returns std::string XML section containing the allocations configuration.

std::string **print_remote_locators**(const std::string &profile_id)

Parse XML file and print specific Domain Participant remote locators limit.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the remote locators configuration does not exist.

Returns std::string XML section with the remote locators limit configuration.

std::string **print_remote_locators_max_unicast**(const std::string &profile_id)

Parse XML file and print specific Domain Participant remote unicast locators limit.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the remote unicast locators limit does not exist.

Returns std::string Remote unicast locators limit.

std::string **print_remote_locators_max_multicast**(const std::string &profile_id)

Parse XML file and print specific Domain Participant remote multicast locators limit.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the remote multicast locators limit does not exist.

Returns std::string Remote multicast locators limit.

std::string **print_total_participants**(const std::string &profile_id)

Parse XML file and print specific Domain Participant total participants limit.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the total participants configuration does not exist.

Returns std::string XML section with the total participants allocation configuration.

std::string **print_total_participants_initial**(const std::string &profile_id)

Parse XML file and print specific Domain Participant initial participants (preallocated).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the initial participants configuration does not exist.

Returns std::string Initial number of preallocated participants.

std::string **print_total_participants_maximum**(const std::string &profile_id)

Parse XML file and print specific Domain Participant maximum number of participants allowed.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the maximum participants configuration does not exist.

Returns std::string Maximum number of participants.

std::string **print_total_participants_increment**(const std::string &profile_id)

Parse XML file and print specific Domain Participant number of allocated participants when the allocated memory is consumed.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the increment participants configuration does not exist.

Returns std::string Increment of participants.

std::string **print_total_readers**(const std::string &profile_id)

Parse XML file and print specific Domain Participant total readers limit.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the total readers configuration does not exist.

Returns std::string XML section with the total readers allocation configuration.

std::string **print_total_readers_initial**(const std::string &profile_id)

Parse XML file and print specific Domain Participant initial readers (preallocated).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the initial readers configuration does not exist.

Returns std::string Initial number of preallocated readers.

std::string **print_total_readers_maximum**(const std::string &profile_id)

Parse XML file and print specific Domain Participant maximum number of readers allowed.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the maximum readers configuration does not exist.

Returns std::string Maximum number of readers.

std::string **print_total_readers_increment**(const std::string &profile_id)

Parse XML file and print specific Domain Participant number of allocated readers when the allocated memory is consumed.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the increment readers configuration does not exist.

Returns std::string Increment of readers.

std::string **print_total_writers**(const std::string &profile_id)

Parse XML file and print specific Domain Participant total writers limit.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the total writers configuration does not exist.

Returns std::string XML section with the total writers allocation configuration.

std::string **print_total_writers_initial**(const std::string &profile_id)

Parse XML file and print specific Domain Participant initial writers (preallocated).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the initial writers configuration does not exist.

Returns std::string Initial number of preallocated writers.

std::string **print_total_writers_maximum**(const std::string &profile_id)

Parse XML file and print specific Domain Participant maximum number of writers allowed.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the maximum writers configuration does not exist.

Returns std::string Maximum number of writers.

std::string **print_total_writers_increment**(const std::string &profile_id)

Parse XML file and print specific Domain Participant number of allocated writers when the allocated memory is consumed.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the increment writers configuration does not exist.

Returns std::string Increment of writers.

std::string **print_max_partitions**(const std::string &profile_id)

Parse XML file and print specific Domain Participant maximum number of partitions.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the maximum number of partitions is not set.

Returns std::string Maximum number of partitions allowed in the Domain Participant.

std::string **print_max_user_data**(const std::string &profile_id)

Parse XML file and print specific Domain Participant maximum size of user data message.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the maximum size is not set.

Returns std::string Maximum allowed size for user data message in the Domain Participant.

std::string **print_max_properties**(const std::string &profile_id)

Parse XML file and print specific Domain Participant maximum number of properties.

Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the maximum property number is not set.

Returns std::string Maximum allowed number of properties in the Domain Participant.

std::string **print_send_buffers**(const std::string &profile_id)
Parse XML file and print specific Domain Participant send buffers allocations configuration.
Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the send buffers configuration does not exist.

Returns std::string XML section with the send buffers allocation configuration.

std::string **print_send_buffers_preallocated_number**(const std::string &profile_id)
Parse XML file and print specific Domain Participant initially preallocated send buffers.
Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the send buffers preallocations are not set.

Returns std::string Number of send buffers preallocated.

std::string **print_send_buffers_dynamic**(const std::string &profile_id)
Parse XML file and print specific Domain Participant send buffers dynamic flag.
Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the send buffers configuration flag is not set.

Returns std::string Send buffers dynamic flag.

void **clear**(const std::string &profile_id)
Remove allocations configuration in the Domain Participant profile.
Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_remote_locators**(const std::string &profile_id)
Remove remote locators limit in the Domain Participant profile.
Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_remote_locators_max_unicast**(const std::string &profile_id)
Remove remote unicast locators limit in the Domain Participant profile.
Parameters **profile_id** – **[in]** Domain participant profile identifier.
Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_remote_locators_max_multicast**(const std::string &profile_id)

Remove remote multicast locators limit in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_participants**(const std::string &profile_id)

Remove total participants limit in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_participants_initial**(const std::string &profile_id)

Remove initial participants (preallocated) in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_participants_maximum**(const std::string &profile_id)

Remove maximum number of participants allowed in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_participants_increment**(const std::string &profile_id)

Remove participant increment configuration in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_readers**(const std::string &profile_id)

Remove total readers limit in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_readers_initial**(const std::string &profile_id)

Remove initial readers (preallocated) in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_readers_maximum**(const std::string &profile_id)

Remove maximum number of readers allowed in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_readers_increment**(const std::string &profile_id)

Remove readers increment configuration in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_writers**(const std::string &profile_id)

Remove total writers limit in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_writers_initial**(const std::string &profile_id)

Remove initial writers (preallocated) in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_writers_maximum**(const std::string &profile_id)

Remove maximum number of writers allowed in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_total_writers_increment**(const std::string &profile_id)

Remove writers increment configuration in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_max_partitions**(const std::string &profile_id)

Remove maximum number of partitions in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_max_user_data**(const std::string &profile_id)

Remove maximum size of user data message in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_max_properties**(const std::string &profile_id)

Remove maximum number of properties in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_send_buffers**(const std::string &profile_id)

Remove send buffers allocations configuration in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_send_buffers_preallocated_number**(const std::string &profile_id)

Remove initially preallocated send buffers configuration in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_send_buffers_preallocated_dynamic**(const std::string &profile_id)

Remove send buffers dynamic flag in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **set_remote_locators_max_unicast**(const std::string &profile_id, const std::string &max_unicast)

Set the remote unicast locators limit in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **max_unicast** – **[in]** Maximum number of allowed remote unicast locators.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_remote_locators_max_multicast**(const std::string &profile_id, const std::string &max_multicast)

Set the remote multicast locators limit in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **max_multicast** – **[in]** Maximum number of allowed remote multicast locators.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_participants_initial**(const std::string &profile_id, const std::string &initial)

Set the initial participants (preallocated) in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **initial** – **[in]** Initial number of preallocated participants.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_participants_maximum**(const std::string &profile_id, const std::string &maximum)

Set the maximum number of participants allowed in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **maximum** – **[in]** Maximum number of participants.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_participants_increment**(const std::string &profile_id, const std::string &increment)

Set the number of allocated participants when the allocated memory is consumed in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **increment** – **[in]** Increment of participants.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_readers_initial**(const std::string &profile_id, const std::string &initial)

Set the initial readers (preallocated) in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **initial** – **[in]** Initial number of preallocated readers.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_readers_maximum**(const std::string &profile_id, const std::string &maximum)

Set the maximum number of readers allowed in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **maximum** – **[in]** Maximum number of readers.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_readers_increment**(const std::string &profile_id, const std::string &increment)

Set the number of allocated readers when the allocated memory is consumed in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **increment** – **[in]** Increment of readers.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_writers_initial**(const std::string &profile_id, const std::string &initial)

Set the initial writers (preallocated) in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **initial** – **[in]** Initial number of preallocated writers.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_writers_maximum**(const std::string &profile_id, const std::string &maximum)

Set the maximum number of writers allowed in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **maximum** – **[in]** Maximum number of writers.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_total_writers_increment**(const std::string &profile_id, const std::string &increment)

Set the number of allocated writers when the allocated memory is consumed in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **increment** – **[in]** Increment of writers.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_max_partitions**(const std::string &profile_id, const std::string &max_partitions)

Set the maximum number of partitions in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **max_partitions** – **[in]** Maximum number of partitions allowed in the Domain Participant.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_max_user_data**(const std::string &profile_id, const std::string &max_user_data)

Set the maximum size of user data message in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **max_user_data** – **[in]** Maximum allowed size for user data message in the Domain Participant.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_max_properties**(const std::string &profile_id, const std::string &max_properties)

Set the maximum number of properties in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **max_properties** – **[in]** Maximum allowed number of properties in the Domain Participant.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_send_buffers_preallocated_number**(const std::string &profile_id, const std::string &send_buffers_preallocated_number)

Set the initially preallocated send buffers in the Domain Participant profile.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **send_buffers_preallocated_number** – **[in]** Number of send buffers preallocated.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_send_buffers_dynamic**(const std::string &profile_id, const std::string &send_buffers_dynamic)

Set the send buffers dynamic flag in the Domain Participant profile.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **send_buffers_dynamic** – **[in]** Send buffers dynamic flag.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

namespace **builtin**

Functions

std::string **print**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin element does not exist.

Returns std::string XML section containing the specific Domain Participant builtin configuration.

std::string **print_avoid_builtin_multicast**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin avoid_builtin_multicast flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the corresponding builtin flag does not exist.

Returns std::string Domain Participant specific avoid_builtin_multicast flag.

std::string **print_use_writer_liveliness_protocol**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin use writer liveliness protocol flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the corresponding builtin flag does not exist.

Returns std::string Domain Participant specific use writer liveliness protocol flag.

std::string **print_reader_history_memory_policy**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin DataReaders History Memory Policy.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the corresponding builtin History Memory Policy does not exist.

Returns std::string Domain Participant builtin DataReaders History Memory Policy.

std::string **print_writer_history_memory_policy**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin DataWriters History Memory Policy.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the corresponding builtin History Memory Policy does not exist.

Returns std::string Domain Participant builtin DataWriters History Memory Policy.

std::string **print_reader_payload_size**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin DataReaders payload size.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the corresponding builtin payload size element does not exist.

Returns std::string Domain Participant builtin DataReaders payload size.

std::string **print_writer_payload_size**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin DataWriters payload size.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the corresponding builtin payload size element does not exist.

Returns std::string Domain Participant builtin DataWriters payload size.

std::string **print_mutation_tries**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin number of physical ports to try if configured port is already in use (mutation tries).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin mutation tries element does not exist.

Returns std::string Domain Participant builtin mutation tries.

void **clear**(const std::string &profile_id)

Remove specific Domain Participant builtin configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_avoid_builtin_multicast**(const std::string &profile_id)

Remove specific Domain Participant builtin avoid_builtin_multicast flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_use_writer_liveliness_protocol**(const std::string &profile_id)

Remove specific Domain Participant builtin use writer liveliness protocol flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_reader_history_memory_policy**(const std::string &profile_id)

Remove specific Domain Participant builtin DataReaders History Memory Policy.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_writer_history_memory_policy**(const std::string &profile_id)

Remove specific Domain Participant builtin DataWriters History Memory Policy.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_reader_payload_size**(const std::string &profile_id)

Remove specific Domain Participant builtin DataReaders payload size.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_witer_payload_size**(const std::string &profile_id)

Remove specific Domain Participant builtin DataWriters payload size.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_mutation_tries**(const std::string &profile_id)

Remove specific Domain Participant builtin DataWriters mutation tries.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **set_avoid_builtin_multicast**(const std::string &profile_id, const std::string &avoid_builtin_multicast)

Set the Domain Participant builtin avoid_builtin_multicast flag.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **avoid_builtin_multicast** – **[in]** Builtin avoid_builtin_multicast flag.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided flag value is not valid.

void **set_use_writer_liveliness_protocol**(const std::string &profile_id, const std::string &use_writer_liveliness_protocol)

Set the Domain Participant builtin use writer liveliness protocol flag.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **use_writer_liveliness_protocol** – **[in]** Builtin use writer liveliness protocol flag.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided flag value is not valid.

void **set_reader_history_memory_policy**(const std::string &profile_id, const std::string &reader_history_memory_policy)

Set the Domain Participant builtin DataReaders History Memory Policy.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **reader_history_memory_policy** – **[in]** Builtin DataReaders History Memory Policy.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided Memory Policy value is not valid.

void **set_writer_history_memory_policy**(const std::string &profile_id, const std::string &writer_history_memory_policy)

Set the Domain Participant builtin DataWriters History Memory Policy.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **writer_history_memory_policy** – **[in]** Builtin DataWriters History Memory Policy.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided Memory Policy value is not valid.

void **set_reader_payload_size**(const std::string &profile_id, const std::string &reader_payload_size)

Set the Domain Participant builtin DataReaders payload size.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **reader_payload_size** – **[in]** Builtin DataReaders payload size.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided payload size value is not valid.

void **set_writer_payload_size**(const std::string &profile_id, const std::string &writer_payload_size)

Set the Domain Participant builtin DataWriters payload size.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **writer_payload_size** – **[in]** Builtin DataWriters payload size.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided payload size value is not valid.

void **set_mutation_tries**(const std::string &profile_id, const std::string &mutation_tries)

Set the Domain Participant builtin mutation tries.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **mutation_tries** – **[in]** Builtin mutation tries.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided mutation tries value is not valid.

namespace **discovery_config**

Functions

std::string **print**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery configuration section.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery configuration section does not exist.

Returns std::string XML section containing the specific Domain Participant builtin discovery configuration.

std::string **print_discovery_protocol**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery protocol.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery protocol element does not exist.

Returns std::string Domain Participant specific builtin discovery protocol.

std::string **print_ignore_participant_flags**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery ignore participant flags.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery flag does not exist.

Returns std::string Domain Participant specific builtin discovery ignore participant flags.

std::string **print_edp**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery EDP flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery flag does not exist.

Returns std::string Domain Participant specific builtin discovery EDP flag.

std::string **print_simple_edp**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery simple EDP configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws
- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery simple EDP configuration does not exist.

Returns std::string Domain Participant specific builtin discovery simple EDP configuration.

std::string **print_simple_edp_pubwriter_subreader**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery simple EDP configuration flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery simple EDP configuration flag does not exist.

Returns std::string Domain Participant specific builtin discovery simple EDP pubwriter_subreader flag configuration.

std::string **print_simple_edp_pubreader_subwriter**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery simple EDP configuration flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery simple EDP configuration flag does not exist.

Returns std::string Domain Participant specific builtin discovery simple EDP pubreader_subwriter flag configuration.

std::string **print_lease_duration**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease duration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease duration element does not exist.

Returns std::string Domain Participant specific builtin discovery lease duration.

std::string **print_lease_duration_sec**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease duration (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease duration seconds element does not exist.

Returns std::string Domain Participant specific builtin discovery lease duration (seconds member).

std::string **print_lease_duration_nanosec**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease duration (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease duration nanoseconds element does not exist.

Returns std::string Domain Participant specific builtin discovery lease duration (nanoseconds member).

std::string **print_lease_announcement**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease announcement.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease announcement element does not exist.

Returns std::string Domain Participant specific builtin discovery lease announcement.

std::string **print_lease_announcement_sec**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease announcement (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease announcement seconds element does not exist.

Returns std::string Domain Participant specific builtin discovery lease announcement (seconds member).

std::string **print_lease_announcement_nanosec**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease announcement (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease announcement nanoseconds element does not exist.

Returns std::string Domain Participant specific builtin discovery lease announcement (nanoseconds member).

std::string **print_initial_announcements**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery initial announcements configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery initial announcements element does not exist.

Returns std::string Domain Participant specific builtin discovery initial announcements configuration.

std::string **print_initial_announcements_count**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery number of initial announcements.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery initial announcements count element does not exist.

Returns std::string Domain Participant specific builtin discovery number of initial announcements.

std::string **`print_initial_announcements_period`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery initial announcements duration.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery initial announcements duration element does not exist.

Returns std::string Domain Participant specific builtin discovery initial announcements duration.

std::string **`print_initial_announcements_period_sec`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery initial announcement duration (seconds).

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery initial announcement duration seconds element does not exist.

Returns std::string Domain Participant specific builtin discovery initial announcement duration (seconds member).

std::string **`print_initial_announcements_period_nanosec`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery lease announcement duration (nanoseconds).

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery lease announcement duration nanoseconds element does not exist.

Returns std::string Domain Participant specific builtin discovery lease announcement duration (nanoseconds member).

std::string **`print_client_announcement_period`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery client announcement period. Discovery Server configuration.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery client announcement period element does not exist.

Returns std::string Domain Participant specific builtin discovery client announcement period.

std::string **print_client_announcement_period_sec**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery client announcement period (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery client announcement period seconds element does not exist.

Returns std::string Domain Participant specific builtin discovery client announcement period (seconds member).

std::string **print_client_announcement_period_nanosec**(const std::string &profile_id)

Parse XML file and print specific Domain Participant builtin discovery client announcement period (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin discovery client announcement period nanoseconds element does not exist.

Returns std::string Domain Participant specific builtin discovery client announcement period (nanoseconds member).

std::string **print_static_edp_xml_config**(const std::string &profile_id, const std::string &index)

Parse XML file and print specific Domain Participant builtin discovery specific Static EDP XML configuration file.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the Static EDP XML configuration files element is not defined, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin discovery specific Static EDP XML configuration file.

uint32_t **static_edp_xml_config_size**(const std::string &profile_id)

Number of builtin discovery Static EDP XML configuration files in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or there are no Static EDP XML files configured.

Returns uint32_t Number of builtin discovery Static EDP XML configuration files.

void **clear**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not

found in the XML file.

void **clear_discovery_protocol**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery protocol.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_ignore_participant_flags**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery ignore participant flags.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_edp**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery EDP flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_simple_edp**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery simple EDP configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_simple_edp_pubwriter_subreader**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery simple EDP configuration flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_simple_edp_edp_pubreader_subwriter**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery simple EDP configuration flag.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_lease_duration**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery lease duration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_lease_duration_sec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery lease duration (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_lease_duration_nanosec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery lease duration (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_lease_announcement**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery lease announcement.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_lease_announcement_sec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery lease announcement (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_lease_announcement_nanosec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery lease announcement (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_initial_announcements**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery initial announcements configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_initial_announcements_count**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery number of initial announcements.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_initial_announcements_period**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery initial announcements duration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_initial_announcements_period_sec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery initial announcements duration (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_initial_announcements_period_nanosec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery initial announcements duration (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_client_announcement_period**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery client announcement period (Discovery Server specific).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_client_announcement_period_sec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery client announcement period (seconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_client_announcement_period_nanosec**(const std::string &profile_id)

Remove specific Domain Participant builtin discovery client announcement period (nanoseconds).

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_static_edp_xml_config**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin discovery specific Static EDP XML configuration file.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection if an index is provided..
- `BadParameter` – *Exception* if the index is not an integer.

void **set_discovery_protocol**(const std::string &profile_id, const std::string &discovery_protocol)

Set the Domain Participant builtin discovery protocol.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **discovery_protocol** – **[in]** Builtin discovery protocol.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided discovery protocol is not valid.

void **set_ignore_participant_flags**(const std::string &profile_id, const std::string &ignore_participant_flags)

Set the Domain Participant builtin discovery ignore participant flags.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **ignore_participant_flags** – **[in]** Builtin discovery ignore participant flags.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided flag is not valid.

void **set_edp**(const std::string &profile_id, const std::string &edp)

Set the Domain Participant builtin discovery EDP flag.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **edp** – **[in]** Builtin discovery EDP flag.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided flag is not valid.

void **set_simple_edp_pubwriter_subreader**(const std::string &profile_id, const std::string &simple_edp_pubwriter_subreader)

Set the Domain Participant builtin discovery simple EDP configuration flag.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **simple_edp_pubwriter_subreader** – **[in]** Builtin discovery simple EDP configuration flag.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided flag is not valid.

void **set_simple_edp_pubreader_subwriter**(const std::string &profile_id, const std::string &simple_edp_pubreader_subwriter)

Set the Domain Participant builtin discovery simple EDP configuration flag.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **simple_edp_pubreader_subwriter** – **[in]** Builtin discovery simple EDP configuration flag.

Throws

- `Error` – *Exception* if the workspace was not initialized.

- *ElementInvalid* – *Exception* if the provided flag is not valid.

void **set_lease_duration_sec**(const std::string &profile_id, const std::string &duration_sec)

Set the Domain Participant builtin discovery lease duration (seconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **duration_sec** – **[in]** Builtin discovery lease duration seconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided seconds are not valid.

void **set_lease_duration_nanosec**(const std::string &profile_id, const std::string &duration_nanosec)

Set the Domain Participant builtin discovery lease duration (nanoseconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **duration_nanosec** – **[in]** Builtin discovery lease duration nanoseconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided nanoseconds are not valid.

void **set_lease_announcement_sec**(const std::string &profile_id, const std::string &announcement_sec)

Set the Domain Participant builtin discovery lease announcement (seconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **announcement_sec** – **[in]** Builtin discovery lease announcement seconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided seconds are not valid.

void **set_lease_announcement_nanosec**(const std::string &profile_id, const std::string &announcement_nanosec)

Set the Domain Participant builtin discovery lease announcement (nanoseconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **announcement_nanosec** – **[in]** Builtin discovery lease announcement nanoseconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided nanoseconds are not valid.

void **set_initial_announcements_count**(const std::string &profile_id, const std::string &count)

Set the Domain Participant builtin discovery number of initial announcements.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **count** – **[in]** Builtin discovery number of initial announcements.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided number of initial announcements is not valid.

void **set_initial_announcements_period_sec**(const std::string &profile_id, const std::string &period_sec)

Set the Domain Participant builtin discovery initial announcements duration (seconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **period_sec** – **[in]** Builtin discovery initial announcements duration seconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided seconds are not valid.

void **set_initial_announcements_period_nanosec**(const std::string &profile_id, const std::string &period_nanosec)

Set the Domain Participant builtin discovery initial announcements duration (nanoseconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **period_nanosec** – **[in]** Builtin discovery initial announcements duration nanoseconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided nanoseconds are not valid.

void **set_client_announcement_period_sec**(const std::string &profile_id, const std::string &period_sec)

Set the Domain Participant builtin discovery client announcement period (seconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **period_sec** – **[in]** Builtin discovery client announcement period seconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided seconds are not valid.

void **set_client_announcement_period_nanosec**(const std::string &profile_id, const std::string &period_nanosec)

Set the Domain Participant builtin discovery client announcement period (nanoseconds).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **period_nanosec** – **[in]** Builtin discovery client announcement period nanoseconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided nanoseconds are not valid.

void **set_static_edp_xml_config**(const std::string &profile_id, const std::string &static_edp_xml_config, const std::string &index)

Append a builtin discovery Static EDP XML configuration file or update a builtin discovery Static EDP XML configuration file.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **static_edp_xml_config** – **[in]** Builtin discovery Static EDP XML configuration file.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided path is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **discovery_servers**

Functions

std::string **print**(const std::string &profile_id, const std::string &prefix)

Parse XML file and print specific Domain Participant builtin remote discovery server (Discovery Server specific).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** GUID prefix of the remote server to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the builtin remote discovery servers list does not exist, or the list does not contain any element with the provided GUID prefix.

Returns std::string Specific Domain Participant builtin remote discovery server.

uint32_t **size**(const std::string &profile_id)

Number of builtin remote discovery servers in the Domain Participant list.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or there are no remote discovery servers configured.

Returns uint32_t Number of builtin remote discovery servers.

std::vector<std::string> **keys**(const std::string &profile_id)

List of the identifiers for every remote server included in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws *Error* – *Exception* if the workspace was not initialized.

Returns std::vector<std::string> Identifier list. Empty list if there are no remote servers.

void **clear**(const std::string &profile_id, const std::string &prefix)

Remove specific Domain Participant discovery remote server from the list.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** GUID prefix of the remote server to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the specified GUID prefix does not exist either.

namespace **metatraffic_multicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific remote server metatraffic multicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic multicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic multicast locator port.

std::string **print_physical_port**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator physical port. TCP only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic multicast locator TCP physical port.

std::string **print_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic multicast locator IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic multicast locator TCPv4 unique LAN ID.

std::string **print_wan_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic multicast locator WAN IPv4 address. TCPv4 only.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic multicast locator TCPv4 WAN IP address.

uint32_t **size**(const std::string &profile_id, const std::string &prefix)

Number of metatraffic multicast locators in the specific remote server of the specified Domain Participant.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or the list has not been set.

Returns uint32_t Number of metatraffic multicast locators in the remote server list.

void **clear**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic multicast locator.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic multicast locator port.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws
- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_physical_port**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic multicast locator physical port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic multicast locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic multicast locator WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &prefix, const std::string &kind, const std::string &index)

Append a remote server metatraffic multicast locator with specified kind or update the remote server metatraffic multicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **kind** – **[in]** Metatraffic multicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator kind is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &prefix, const std::string &port, const std::string &index)

Append a remote server metatraffic multicast locator with specified port or update the remote server metatraffic multicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **port** – **[in]** Metatraffic multicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &prefix, const std::string &physical_port, const std::string &index)

Append a remote server metatraffic multicast locator with specified physical port or update the remote server metatraffic multicast locator physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **physical_port** – **[in]** Metatraffic multicast locator TCP physical port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant pro-

file/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &prefix, const std::string &address, const std::string &index)

Append a remote server metatraffic multicast locator with specified IP address or update the remote server metatraffic multicast locator IP address.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **address** – **[in]** Metatraffic multicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws
- `ElementInvalid` – *Exception* if the provided locator IP address is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &prefix, const std::string &unique_lan_id, const std::string &index)

Append a remote server metatraffic multicast TCPv4 locator with specified unique LAN ID or update the remote server metatraffic multicast TCPv4 locator unique LAN ID.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **unique_lan_id** – **[in]** Metatraffic multicast TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws
- `ElementInvalid` – *Exception* if the provided locator identifier is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &prefix, const std::string &wan_address, const std::string &index)

Append a remote server metatraffic multicast TCPv4 locator with specified WAN address or update the remote server metatraffic multicast TCPv4 locator WAN address.

Parameters
- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **wan_address** – **[in]** Metatraffic multicast TCPv4 locator WAN address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator IP address is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

namespace **`metatraffic_unicast_locators`**

Functions

std::string **`print`**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`prefix`** – **[in]** Remote server GUID prefix.
- **`index`** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific remote server metatraffic unicast locator.

std::string **`print_kind`**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator kind.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`prefix`** – **[in]** Remote server GUID prefix.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic unicast locator kind.

std::string **`print_port`**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator port.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`prefix`** – **[in]** Remote server GUID prefix.

- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic unicast locator port.

std::string **print_physical_port**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator physical port. TCP only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic unicast locator TCP physical port.

std::string **print_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic unicast locator IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant pro-

file/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic unicast locator TCPv4 unique LAN ID.

std::string **`print_wan_address`**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Parse XML file and print the specific Domain Participant specific remote server metatraffic unicast locator WAN IPv4 address. TCPv4 only.

Parameters
- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`prefix`** – **[in]** Remote server GUID prefix.
- **`index`** – **[in]** Collection element to be printed.

Throws
- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific remote server metatraffic unicast locator TCPv4 WAN IP address.

uint32_t **`size`**(const std::string &profile_id, const std::string &prefix)

Number of metatraffic unicast locators in the specific remote server of the specified Domain Participant.

Parameters
- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`prefix`** – **[in]** Remote server GUID prefix.

Throws
- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or the list has not been set.

Returns uint32_t Number of metatraffic unicast locators in the remote server list.

void **`clear`**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic unicast locator.

Parameters
- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`prefix`** – **[in]** Remote server GUID prefix.
- **`index`** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws
- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **`clear_port`**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic unicast locator port.

Parameters
- **`profile_id`** – **[in]** Domain participant profile identifier.

- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_physical_port**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic unicast locator physical port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic unicast locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &prefix, const std::string &index)

Remove specific Domain Participant remote server metatraffic unicast locator WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &prefix, const std::string &kind, const std::string &index)

Append a remote server metatraffic unicast locator with specified kind or update the remote server metatraffic unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **kind** – **[in]** Metatraffic unicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator kind is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &prefix, const std::string &port, const std::string &index)

Append a remote server metatraffic unicast locator with specified port or update the remote server metatraffic unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **port** – **[in]** Metatraffic unicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &prefix, const std::string &physical_port, const std::string &index)

Append a remote server metatraffic unicast locator with specified physical port or update the remote server metatraffic unicast locator physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **physical_port** – **[in]** Metatraffic unicast locator TCP physical port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &prefix, const std::string &address, const std::string &index)

Append a remote server metatraffic unicast locator with specified IP address or update the remote server metatraffic unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **address** – **[in]** Metatraffic unicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &prefix, const std::string &unique_lan_id, const std::string &index)

Append a remote server metatraffic unicast TCPv4 locator with specified unique LAN ID or update the remote server metatraffic unicast TCPv4 locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **unique_lan_id** – **[in]** Metatraffic unicast TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &prefix, const std::string &wan_address, const std::string &index)

Append a remote server metatraffic unicast TCPv4 locator with specified WAN address or update the remote server metatraffic unicast TCPv4 locator WAN address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **prefix** – **[in]** Remote server GUID prefix.
- **wan_address** – **[in]** Metatraffic unicast TCPv4 locator WAN address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/GUID prefix is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **initial_peers**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific builtin initial peers.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin initial peers kind.

std::string **`print_port`**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers port.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin initial peers port.

std::string **`print_physical_port`**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers physical port. TCP only.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin initial peers TCP physical port.

std::string **`print_address`**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers IP address.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin initial peers IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin initial peers TCPv4 unique LAN ID.

std::string **print_wan_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin initial peers WAN IPv4 address. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin initial peers TCPv4 WAN IP address.

uint32_t **size**(const std::string &profile_id)

Number of builtin initial peers in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of builtin initial peers in the list.

void **clear**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin initial peers.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin initial peers port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_physical_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin initial peers physical port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin initial peers IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin initial peers unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin initial peers WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)

Append a builtin initial peers with specified kind or update the builtin initial peers kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Initial peers kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator kind is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)

Append a builtin initial peers with specified port or update the builtin initial peers port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port** – **[in]** Initial peers port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &physical_port, const std::string &index)

Append a builtin initial peers with specified physical port or update the builtin initial peers physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **physical_port** – **[in]** Initial peers TCP physical port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a builtin initial peers with specified IP address or update the builtin initial peers IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Initial peers IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &unique_lan_id, const std::string &index)

Append a builtin initial peers TCPv4 locator with specified unique LAN ID or update the builtin initial peers TCPv4 locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **unique_lan_id** – **[in]** Initial peers TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &wan_address, const std::string &index)

Append a builtin initial peers TCPv4 locator with specified WAN address or update the builtin initial peers TCPv4 locator WAN address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **wan_address** – **[in]** Initial peers TCPv4 locator WAN address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **metatraffic_external_unicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific metatraffic external unicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific metatraffic external unicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.

- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific metatraffic external unicast locator port.

std::string **`print_address`**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator IP address.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific metatraffic external unicast locator IP address.

std::string **`print_externality`**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator externality.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific metatraffic external unicast locator externality.

std::string **`print_cost`**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator cost.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific metatraffic external unicast locator cost.

std::string **print_mask**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific metatraffic external unicast locator mask.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific metatraffic external unicast locator mask.

uint32_t **size**(const std::string &profile_id)

Number of metatraffic external unicast locators in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of metatraffic external unicast locators in the list.

void **clear**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant metatraffic external unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant metatraffic external unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant metatraffic external unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_externality**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant metatraffic external unicast locator externality.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_cost**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant metatraffic external unicast locator cost.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_mask**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant metatraffic external unicast locator mask.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)

Append a metatraffic external unicast locator with specified kind or update the metatraffic external unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Metatraffic unicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator kind is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)

Append a metatraffic external unicast locator with specified port or update the metatraffic external unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port** – **[in]** Metatraffic unicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a metatraffic external unicast locator with specified IP address or update the metatraffic external unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Metatraffic unicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_externality**(const std::string &profile_id, const std::string &externality, const std::string &index)

Append a metatraffic external unicast locator with specified externality or update the metatraffic external unicast locator externality.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **externality** – **[in]** Metatraffic unicast locator externality.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator externality is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_cost**(const std::string &profile_id, const std::string &cost, const std::string &index)

Append a metatraffic external unicast locator with specified cost or update the metatraffic external unicast locator cost.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **cost** – **[in]** Metatraffic unicast locator cost.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator cost is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_mask**(const std::string &profile_id, const std::string &mask, const std::string &index)

Append a metatraffic external unicast locator with specified mask or update the metatraffic external unicast locator mask.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **mask** – **[in]** Metatraffic unicast locator mask.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator mask is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

namespace **metatraffic_multicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific builtin metatraffic multicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic multicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic multicast locator port.

std::string **print_physical_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator physical port. TCP only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic multicast locator TCP physical port.

std::string **print_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic multicast locator IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic multicast locator TCPv4 unique LAN ID.

std::string **print_wan_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic multicast locator WAN IPv4 address. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.

- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic multicast locator TCPv4 WAN IP address.

uint32_t **`size`**(const std::string &profile_id)

Number of builtin metatraffic multicast locators in the Domain Participant.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of builtin metatraffic multicast locators in the list.

void **`clear`**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic multicast locator.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **`clear_port`**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic multicast locator port.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be modified.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **`clear_physical_port`**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic multicast locator physical port.

Parameters

- **`profile_id`** – **[in]** Domain participant profile identifier.
- **`index`** – **[in]** Collection element to be modified.

Throws

- `Error` – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic multicast locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic multicast locator WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)

Append a builtin metatraffic multicast locator with specified kind or update the builtin metatraffic multicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Metatraffic multicast locator kind.

- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator kind is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)

Append a builtin metatraffic multicast locator with specified port or update the builtin metatraffic multicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port** – **[in]** Metatraffic multicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &physical_port, const std::string &index)

Append a builtin metatraffic multicast locator with specified physical port or update the builtin metatraffic multicast locator physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **physical_port** – **[in]** Metatraffic multicast locator TCP physical port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a builtin metatraffic multicast locator with specified IP address or update the builtin metatraffic multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Metatraffic multicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &unique_lan_id, const std::string &index)

Append a builtin metatraffic multicast TCPv4 locator with specified unique LAN ID or update the builtin metatraffic multicast TCPv4 locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **unique_lan_id** – **[in]** Metatraffic multicast TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &wan_address, const std::string &index)

Append a builtin metatraffic multicast TCPv4 locator with specified WAN address or update the builtin metatraffic multicast TCPv4 locator WAN address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **wan_address** – **[in]** Metatraffic multicast TCPv4 locator WAN address.

- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **metatraffic_unicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific builtin metatraffic unicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic unicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic unicast locator port.

std::string **print_physical_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator physical port. TCP only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic unicast locator TCP physical port.

std::string **print_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.

- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic unicast locator IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic unicast locator TCPv4 unique LAN ID.

std::string **print_wan_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific builtin metatraffic unicast locator WAN IPv4 address. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific builtin metatraffic unicast locator TCPv4 WAN IP address.

uint32_t **size**(const std::string &profile_id)

Number of builtin metatraffic unicast locators in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of builtin metatraffic unicast locators in the list.

void **clear**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_physical_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic unicast locator physical port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant builtin metatraffic unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &index)
Remove specific Domain Participant builtin metatraffic unicast locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &index)
Remove specific Domain Participant builtin metatraffic unicast locator WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)
Append a builtin metatraffic unicast locator with specified kind or update the builtin metatraffic unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Metatraffic unicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator kind is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)
Append a builtin metatraffic unicast locator with specified port or update the builtin metatraffic unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.

- **port** – **[in]** Metatraffic unicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &physical_port, const std::string &index)

Append a builtin metatraffic unicast locator with specified physical port or update the builtin metatraffic unicast locator physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **physical_port** – **[in]** Metatraffic unicast locator TCP physical port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a builtin metatraffic unicast locator with specified IP address or update the builtin metatraffic unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Metatraffic unicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.

- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &unique_lan_id, const std::string &index)

Append a builtin metatraffic unicast TCPv4 locator with specified unique LAN ID or update the builtin metatraffic unicast TCPv4 locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **unique_lan_id** – **[in]** Metatraffic unicast TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &wan_address, const std::string &index)

Append a builtin metatraffic unicast TCPv4 locator with specified WAN address or update the builtin metatraffic unicast TCPv4 locator WAN address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **wan_address** – **[in]** Metatraffic unicast TCPv4 locator WAN address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **default_external_unicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific default external unicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default external unicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default external unicast locator port.

std::string **print_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default external unicast locator IP address.

std::string **print_externality**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator externality.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default external unicast locator externality.

std::string **print_cost**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator cost.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default external unicast locator cost.

std::string **print_mask**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default external unicast locator mask.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default external unicast locator mask.

uint32_t **size**(const std::string &profile_id)

Number of default external unicast locators in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of default external unicast locators in the list.

void **clear**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default external unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default external unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default external unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_externality**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default external unicast locator externality.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_cost**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default external unicast locator cost.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_mask**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default external unicast locator mask.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)

Append a new default external unicast locator with specified kind or update the existing locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Default unicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator kind is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)

Append a default external unicast locator with specified port or update the existing locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port** – **[in]** Default unicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator port is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a default external unicast locator with specified IP address or update the existing locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Default unicast locator IP address.

- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_externality**(const std::string &profile_id, const std::string &externality, const std::string &index)

Append a default external unicast locator with specified externality or update the existing locator externality.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **externality** – **[in]** Default unicast locator externality.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator externality is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_cost**(const std::string &profile_id, const std::string &cost, const std::string &index)

Append a default external unicast locator with specified cost or update the existing locator cost.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **cost** – **[in]** Default unicast locator cost.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator cost is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_mask**(const std::string &profile_id, const std::string &mask, const std::string &index)

Append a default external unicast locator with specified mask or update the existing locator mask.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **mask** – **[in]** Default unicast locator mask.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator mask is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **default_multicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific default multicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.

- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default multicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default multicast locator port.

std::string **print_physical_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator physical port. TCP only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default multicast locator TCP physical port.

std::string **print_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default multicast locator IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default multicast locator TCPv4 unique LAN ID.

std::string **print_wan_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default multicast locator WAN IPv4 address. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default multicast locator TCPv4 WAN IP address.

uint32_t **size**(const std::string &profile_id)

Number of default multicast locators in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of default multicast locators in the list.

void **clear**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default multicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default multicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_physical_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default multicast locator physical port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default multicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &index)
Remove specific Domain Participant default multicast locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &index)
Remove specific Domain Participant default multicast locator WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)
Append a default multicast locator with specified kind or update the existing locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Default multicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator kind is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)
Append a default multicast locator with specified port or update the existing locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port** – **[in]** Default multicast locator port.

- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &physical_port, const std::string &index)

Append a default multicast locator with specified physical port or update the existing locator physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **physical_port** – **[in]** Default multicast locator TCP physical port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a default multicast locator with specified IP address or update the existing locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Default multicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &unique_lan_id, const std::string &index)

Append a default multicast TCPv4 locator with specified unique LAN ID or update the existing locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **unique_lan_id** – **[in]** Default multicast TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &wan_address, const std::string &index)

Append a default multicast TCPv4 locator with specified WAN address or update the existing locator WAN address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **wan_address** – **[in]** Default multicast TCPv4 locator WAN address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **default_unicast_locators**

Functions

std::string **print**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete list is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string XML section containing the Domain Participant specific default unicast locator.

std::string **print_kind**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default unicast locator kind.

std::string **print_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default unicast locator port.

std::string **print_physical_port**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator physical port. TCP only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default unicast locator TCP physical port.

std::string **print_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default unicast locator IP address.

std::string **print_unique_lan_id**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator unique LAN ID. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default unicast locator TCPv4 unique LAN ID.

std::string **print_wan_address**(const std::string &profile_id, const std::string &index)

Parse XML file and print the specific Domain Participant specific default unicast locator WAN IPv4 address. TCPv4 only.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string Domain Participant specific default unicast locator TCPv4 WAN IP address.

uint32_t **size**(const std::string &profile_id)

Number of default unicast locators in the Domain Participant.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the list has not been set.

Returns uint32_t Number of default unicast locators in the list.

void **clear**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default unicast locator.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete list is removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default unicast locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_physical_port**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default unicast locator physical port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default unicast locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_unique_lan_id**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default unicast locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_wan_address**(const std::string &profile_id, const std::string &index)

Remove specific Domain Participant default unicast locator WAN IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **index** – **[in]** Collection element to be modified.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or if the element does not exist in the collection.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_kind**(const std::string &profile_id, const std::string &kind, const std::string &index)

Append a default unicast locator with specified kind or update the existing locator kind.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **kind** – **[in]** Default unicast locator kind.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator kind is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_port**(const std::string &profile_id, const std::string &port, const std::string &index)

Append a default unicast locator with specified port or update the existing locator port.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port** – **[in]** Default unicast locator port.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided locator port is not valid.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **set_physical_port**(const std::string &profile_id, const std::string &physical_port, const std::string &index)

Append a default unicast locator with specified physical port or update the existing locator physical port (TCP only).

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **physical_port** – **[in]** Default unicast locator TCP physical port.

- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator physical port is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_address**(const std::string &profile_id, const std::string &address, const std::string &index)

Append a default unicast locator with specified IP address or update the existing locator IP address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **address** – **[in]** Default unicast locator IP address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_unique_lan_id**(const std::string &profile_id, const std::string &unique_lan_id, const std::string &index)

Append a default unicast TCPv4 locator with specified unique LAN ID or update the existing locator unique LAN ID.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **unique_lan_id** – **[in]** Default unicast TCPv4 locator unique LAN ID.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator identifier is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_wan_address**(const std::string &profile_id, const std::string &wan_address, const std::string &index)

Append a default unicast TCPv4 locator with specified WAN address or update the existing locator WAN address.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **wan_address** – **[in]** Default unicast TCPv4 locator WAN address.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided locator IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

namespace **port**

Functions

std::string **print**(const std::string &profile_id)

Parse XML file and print specific Domain Participant port parameters configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the port parameters configuration element does not exist.

Returns std::string XML section containing the port parameters configuration.

std::string **print_base**(const std::string &profile_id)

Parse XML file and print specific Domain Participant base port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the base port parameter has not been set.

Returns std::string Base port parameter.

std::string **print_domain_id_gain**(const std::string &profile_id)

Parse XML file and print specific Domain Participant domain ID gain port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the domain ID gain port parameter has not been set.

Returns std::string Domain ID gain port parameter.

std::string **`print_participant_id_gain`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant participant ID gain port parameter.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the participant ID gain port parameter has not been set.

Returns std::string Participant ID gain port parameter.

std::string **`print_offset_d0`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant multicast metadata offset port parameter.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the multicast metadata offset port parameter has not been set.

Returns std::string Multicast metadata offset port parameter.

std::string **`print_offset_d1`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant unicast metadata offset port parameter.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the unicast metadata offset port parameter has not been set.

Returns std::string Unicast metadata offset port parameter.

std::string **`print_offset_d2`**(const std::string &profile_id)

Parse XML file and print specific Domain Participant multicast user data offset port parameter.

Parameters **`profile_id`** – **[in]** Domain participant profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified Domain Participant profile is not found in the XML file or the multicast user data offset port parameter has not been set.

Returns std::string Multicast user data offset port parameter.

std::string **print_offset_d3**(const std::string &profile_id)

Parse XML file and print specific Domain Participant unicast user data offset port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file or the user data metadata offset port parameter has not been set.

Returns std::string Unicast user data offset port parameter.

void **clear**(const std::string &profile_id)

Remove specific Domain Participant port parameters configuration.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_base**(const std::string &profile_id)

Remove specific Domain Participant base port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_domain_id_gain**(const std::string &profile_id)

Remove specific Domain Participant domain ID gain port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_participant_id_gain**(const std::string &profile_id)

Remove specific Domain Participant participant ID gain port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_offset_d0**(const std::string &profile_id)

Remove specific Domain Participant multicast metadata offset port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_offset_d1**(const std::string &profile_id)

Remove specific Domain Participant unicast metadata offset port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_offset_d2**(const std::string &profile_id)

Remove specific Domain Participant multicast user data offset port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **clear_offset_d3**(const std::string &profile_id)

Remove specific Domain Participant unicast user data offset port parameter.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

void **set_base**(const std::string &profile_id, const std::string &port_base)

Set the Domain Participant base port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **port_base** – **[in]** Base port parameter.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided parameter value is not valid.

void **set_domain_id_gain**(const std::string &profile_id, const std::string &domain_id_gain)

Set the Domain Participant domain ID gain port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **domain_id_gain** – **[in]** Domain ID gain port parameter.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided parameter value is not valid.

void **set_participant_id_gain**(const std::string &profile_id, const std::string &participant_id_gain)
Set the Domain Participant participant ID gain port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **participant_id_gain** – **[in]** Participant ID gain port parameter.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided parameter value is not valid.

void **set_offset_d0**(const std::string &profile_id, const std::string &offset_d0)
Set the Domain Participant multicast metadata offset port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **offset_d0** – **[in]** Multicast metadata offset port parameter.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided parameter value is not valid.

void **set_offset_d1**(const std::string &profile_id, const std::string &offset_d1)
Set the Domain Participant unicast metadata offset port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **offset_d1** – **[in]** Unicast metadata offset port parameter.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided parameter value is not valid.

void **set_offset_d2**(const std::string &profile_id, const std::string &offset_d2)
Set the Domain Participant multicast user data offset port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **offset_d2** – **[in]** Multicast user data offset port parameter.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided parameter value is not valid.

void **set_offset_d3**(const std::string &profile_id, const std::string &offset_d3)
Set the Domain Participant unicast user data offset port parameter.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **offset_d3** – **[in]** Unicast user data offset port parameter.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided parameter value is not valid.

namespace **properties_policy**

Functions

std::string **print**(const std::string &profile_id, const std::string &property_id)

Parse XML file and print specific Domain Participant property in the policy list.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name to be printed. If empty, print the complete property policy list.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/Property name is not found in the XML file or the properties policy element does not exist.

Returns std::string XML section containing the specific property.

std::string **print_value**(const std::string &profile_id, const std::string &property_id)

Parse XML file and print specific Domain Participant property value in the policy list.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name which value is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/Property name is not found in the XML file or the properties policy element does not exist.

Returns std::string Property value.

std::string **print_propagate**(const std::string &profile_id, const std::string &property_id)

Parse XML file and print specific Domain Participant property propagate attribute in the policy list.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name which propagate attribute is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementNotFound* – *Exception* if the specified Domain Participant profile/Property name is not found in the XML file or the properties policy element does not exist.

Returns std::string Property propagate attribute.

uint32_t **size**(const std::string &profile_id)

Number of properties defined in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

Returns uint32_t Number of properties.

std::vector<std::string> **keys**(const std::string &profile_id)

List of the property names defined in the Domain Participant profile.

Parameters **profile_id** – **[in]** Domain participant profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile is not found in the XML file.

Returns std::vector<std::string> Identifier list. Empty list if there are no properties.

void **clear**(const std::string &profile_id, const std::string &property_id)

Remove specific property in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name to be removed. If empty, every property is removed from the profile.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/Property is not found in the XML file.

void **clear_value**(const std::string &profile_id, const std::string &property_id)

Remove specific property value in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name which value is to be removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/Property is not found in the XML file.

void **clear_propagate**(const std::string &profile_id, const std::string &property_id)

Remove specific property propagate attribute in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name which propagate attribute is to be removed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified Domain Participant profile/Property is not found in the XML file.

void **set_value**(const std::string &profile_id, const std::string &property_id, const std::string &value)

Set the property value in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name which value is being set/updated.
- **value** – **[in]** New property value.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **set_propagate**(const std::string &profile_id, const std::string &property_id, const std::string &propagate)

Set the property propagate attribute in the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name which value is being set/updated.
- **propagate** – **[in]** New propagate value.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided value is not valid.

void **push**(const std::string &profile_id, const std::string &property_id)

Append a new property with empty value to the Domain Participant profile.

Parameters

- **profile_id** – **[in]** Domain participant profile identifier.
- **property_id** – **[in]** Property name.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided property name is not valid.

2.13 DataReader

namespace **data_reader**

Functions

void **set_default_profile**(const std::string &profile_id)

Set the selected DataReader profile as default profile. As only one default profile is allowed, if another default profile exists, it is overridden and its is_default_profile attribute is set to false.

Parameters **profile_id** – **[in]** DataReader profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified profile is not found in the XML file.

namespace **qos**

Functions

void **set_durability_kind**(const std::string &profile_id, const std::string &kind)

Set the DataReader durability QoS kind.

Parameters

- **profile_id** – **[in]** DataReader profile identifier.
- **kind** – **[in]** Durability QoS kind value.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided durability QoS kind value is not valid.

void **set_reliability_kind**(const std::string &profile_id, const std::string &kind)

Set the DataReader reliability QoS kind.

Parameters

- **profile_id** – **[in]** DataReader profile identifier.
- **kind** – **[in]** Reliability QoS kind value.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided reliability QoS kind value is not valid.

void **set_reliability_max_blocking_time_sec**(const std::string &profile_id, const std::string &sec)

Set the DataReader reliability QoS max blocking time seconds.

Parameters

- **profile_id** – **[in]** DataReader profile identifier.
- **sec** – **[in]** Reliability QoS max blocking time seconds to be set

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided reliability QoS max blocking time seconds are not valid.

void **set_reliability_max_blocking_time_nanosec**(const std::string &profile_id, const std::string &nanosec)

Set the DataReader reliability QoS max blocking time nanoseconds.

Parameters

- **profile_id** – **[in]** DataReader profile identifier.
- **nanosec** – **[in]** Reliability QoS max blocking time nanoseconds to be set

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided reliability QoS max blocking time nanoseconds are not valid.

2.14 DataWriter

namespace **data_writer**

Functions

void **set_default_profile**(const std::string &profile_id)

Set the selected DataWriter profile as default profile. As only one default profile is allowed, if another default profile exists, it is overridden and its is_default_profile attribute is set to false.

Parameters **profile_id** – **[in]** DataWriter profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified profile is not found in the XML file.

namespace **qos**

Functions

void **set_durability_kind**(const std::string &profile_id, const std::string &kind)

Set the DataWriter durability QoS kind.

Parameters

- **profile_id** – **[in]** DataWriter profile identifier.
- **kind** – **[in]** Durability QoS kind value.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided durability QoS kind value is not valid.

void **set_reliability_kind**(const std::string &profile_id, const std::string &kind)

Set the DataWriter reliability QoS kind.

Parameters

- **profile_id** – **[in]** DataWriter profile identifier.
- **kind** – **[in]** Reliability QoS kind value.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided reliability QoS kind value is not valid.

void **set_reliability_max_blocking_time_sec**(const std::string &profile_id, const std::string &sec)

Set the DataWriter reliability QoS max blocking time seconds.

Parameters

- **profile_id** – **[in]** DataWriter profile identifier.
- **sec** – **[in]** Reliability QoS max blocking time seconds to be set

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided reliability QoS max blocking time seconds are not valid.

void **set_reliability_max_blocking_time_nanosec**(const std::string &profile_id, const std::string &nanosec)

Set the DataWriter reliability QoS max blocking time nanoseconds.

Parameters

- **profile_id** – **[in]** DataWriter profile identifier.
- **nanosec** – **[in]** Reliability QoS max blocking time nanoseconds to be set

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided reliability QoS max blocking time nanoseconds are not valid.

2.15 Transport Descriptor

namespace **transport_descriptor**

Functions

std::string **print**(const std::string &transport_descriptor_id)

Parse XML file and print specific transport descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier. If empty, every transport descriptor profile is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns std::string XML section containing the specific transport descriptor profile.

std::string **print_kind**(const std::string &transport_descriptor_id)

Parse XML file and print the specific Transport Descriptor kind.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns std::string Transport Descriptor kind

std::string **print_send_buffer_size**(const std::string &transport_descriptor_id)

Parse XML file and print the specific Transport Descriptor send buffer size.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the send buffer size element has not been set in the profile.

Returns std::string Transport descriptor send buffer size.

std::string **print_receive_buffer_size**(const std::string &transport_descriptor_id)

Parse XML file and print the specific Transport Descriptor receive buffer size.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the receive buffer size element has not been set in the profile.

Returns std::string Transport descriptor receive buffer size.

std::string **print_max_message_size**(const std::string &transport_descriptor_id)

Parse XML file and print the specific Transport Descriptor maximum message size.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the maximum message size element has not been set in the profile.

Returns std::string Transport descriptor maximum message size.

std::string **`print_max_initial_peers_range`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific Transport Descriptor maximum initial peers range.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the maximum initial peers range element has not been set in the profile.

Returns std::string Transport descriptor maximum initial peers range.

std::string **`print_interface_whitelist`**(const std::string &transport_descriptor_id, const std::string &index)

Parse XML file and print the Transport Descriptor specific whitelisted network interface element.

Parameters

- `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.
- `index` – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string Transport descriptor specific network interface whitelisted.

std::string **`print_ttl`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific Transport Descriptor TTL (Time to live).

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the TTL element has not been set in the profile.

Returns std::string Transport descriptor TTL in number of hops.

std::string **`print_non_blocking_send`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific UDP Transport Descriptor non blocking send flag.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the non blocking send flag has not been set in the profile.

Returns std::string Transport descriptor non blocking send flag.

std::string **`print_output_port`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific UDP Transport Descriptor output port.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the output port has not been set in the profile.

Returns std::string UDP Transport descriptor output port.

std::string **`print_wan_addr`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCPv4 Transport Descriptor WAN address.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the WAN address has not been set in the profile.

Returns std::String TCPv4 Transport descriptor WAN address.

std::string **`print_keep_alive_frequency_ms`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific frequency of TCP Transport Descriptor keep alive requests.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the keep alive frequency has not been set in the profile.

Returns std::string Frequency of TCP keep alive requests (in milliseconds).

std::string **`print_keep_alive_timeout_ms`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific timeout (in milliseconds) to consider a TCP connection broken if no keep alive is received.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the keep alive timeout has not been set in the profile.

Returns std::string TCP keep alive timeout (in milliseconds).

std::string **`print_max_logical_port`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific maximum number of logical ports to try during TCP negotiation.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the maximum number of logical ports has not been set in the profile.

Returns std::string Maximum number of logical ports to try during TCP negotiation.

std::string **`print_logical_port_range`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific maximum number of logical ports per request to try during TCP negotiation.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the maximum number of logical ports per request has not been set in the profile.

Returns std::string Maximum number of logical ports per request to try during TCP negotiation.

std::string **`print_logical_port_increment`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific increment between logical ports to try during TCP negotiation.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the logical port increment has not been set in the profile.

Returns std::string Increment between logical ports to try during TCP negotiation.

std::string **`print_listening_ports`**(const std::string &transport_descriptor_id, const std::string &index)

Parse XML file and print the TCP Transport Descriptor specific listening port element.

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.
- **`index`** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the list does not contain any element in index position.
- `BadParameter` – *Exception* if the index is not an integer.

Returns std::string TCP Transport descriptor specific listening port.

std::string **`print_tls`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS configuration.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.

- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS configuration element has not been set in the profile.

Returns std::string XML section containing the specific TCP Transport Descriptor TLS configuration.

std::string **`print_tls_password`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS password.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS password element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor TLS password.

std::string **`print_tls_private_key_file`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS private key certificate path.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS private key certificate path element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor path to the TLS private key certificate file.

std::string **`print_tls_rsa_private_key_file`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS private key RSA certificate path.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS private key RSA certificate path element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor path to the TLS private key RSA certificate file.

std::string **`print_tls_cert_chain_file`**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS public certificate chain file path.

Parameters **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS public certificate chain file path element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor path to the TLS public certificate chain file.

std::string **print_tls_tmp_dh_file**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS Diffie-Hellman parameters file path.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS Diffie-Hellman parameters file path element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor path to the TLS Diffie-Hellman parameters file.

std::string **print_tls_verify_file**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS CA (Certification Authority) file path.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS CA file path element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor path to the CA file.

std::string **print_tls_verify_mode**(const std::string &transport_descriptor_id, const std::string &index)

Parse XML file and print the TCP Transport Descriptor specific TLS verification mode element.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string TCP Transport descriptor specific TLS verification mode.

std::string **print_tls_options**(const std::string &transport_descriptor_id, const std::string &index)

Parse XML file and print the TCP Transport Descriptor specific TLS supported feature.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the list does not contain any element in index position.

- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string TCP Transport descriptor specific TLS supported feature.

std::string **print_tls_verify_paths**(const std::string &transport_descriptor_id, const std::string &index)

Parse XML file and print the TCP Transport Descriptor specific TLS path to verification files.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be printed. If empty, the complete collection is printed.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the list does not contain any element in index position.
- *BadParameter* – *Exception* if the index is not an integer.

Returns std::string TCP Transport descriptor specific TLS verification path.

std::string **print_tls_verify_depth**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS maximum verification depth.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS verify depth element has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor maximum allowed depth for verifying intermediate certificates.

std::string **print_tls_default_verify_path**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS verify default paths flag.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS verify default paths flag has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor verify default paths flag.

std::string **print_tls_handshake_role**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS handshake role.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS handshake role has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor handshake role.

std::string **print_tls_server_name**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TLS SNI server name.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TLS SNI server name has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor SNI server name.

std::string **print_calculate_crc**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor calculate CRC flag.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the calculate CRC flag has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor calculate CRC flag.

std::string **print_check_crc**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor check CRC flag.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the check CRC flag has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor check CRC flag.

std::string **print_enable_tcp_nodelay**(const std::string &transport_descriptor_id)

Parse XML file and print the specific TCP Transport Descriptor TCP_NODELAY socket option flag.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the TCP_NODELAY socket option flag has not been set in the profile.

Returns std::string Specific TCP Transport Descriptor TCP_NODELAY socket option flag.

std::string **print_segment_size**(const std::string &transport_descriptor_id)

Parse XML file and print the specific SHM Transport Descriptor shared memory segment size.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the shared memory segment size has not been set in the profile.

Returns std::string Specific SHM Transport Descriptor shared memory segment size.

std::string **print_port_queue_capacity**(const std::string &transport_descriptor_id)

Parse XML file and print the specific SHM Transport Descriptor listening port capacity.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the listening port capacity has not been set in the profile.

Returns std::string Specific SHM Transport Descriptor listening port capacity in number of messages.

std::string **print_healthy_check_timeout_ms**(const std::string &transport_descriptor_id)

Parse XML file and print the specific SHM Transport Descriptor listening port liveliness timeout.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the listening port liveliness timeout has not been set in the profile.

Returns std::string Specific SHM Transport Descriptor listening port liveliness timeout (in milliseconds).

std::string **print_rtps_dump_file**(const std::string &transport_descriptor_id)

Parse XML file and print the specific SHM Transport Descriptor debugging file path.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or the debugging file path has not been set in the profile.

Returns std::string Specific SHM Transport Descriptor debugging file path.

uint32_t **size**()

Number of transport descriptor profiles contained in the XML file.

Throws *Error* – *Exception* if the workspace was not initialized.

Returns uint32_t Number of transport descriptor profiles in the XML file.

std::vector<std::string> **keys**()

List of the identifiers for every transport descriptor profile in the XML file.

Throws *Error* – *Exception* if the workspace was not initialized.

Returns std::vector<std::string> Identifier list. Empty list if there are no transport descriptor profiles.

uint32_t **interface_whitelist_size**(const std::string &transport_descriptor_id)

Number of whitelisted network interfaces in the Transport Descriptor list.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns uint32_t Number of network interfaces whitelisted in the list.

uint32_t **listening_ports_size**(const std::string &transport_descriptor_id)

Number of listening ports in the TCP Transport Descriptor list.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns uint32_t Number of listening ports in the list.

uint32_t **tls_verify_mode_size**(const std::string &transport_descriptor_id)

Number of TLS verification modes enabled in the mask in the TCP Transport Descriptor.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns uint32_t Number of TLS verification modes enabled in the mask.

uint32_t **tls_options_size**(const std::string &transport_descriptor_id)

Number of TLS supported features enabled in the mask in the TCP Transport Descriptor.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns uint32_t Number of TLS supported features enabled in the mask.

uint32_t **tls_verify_paths_size**(const std::string &transport_descriptor_id)

Number of TLS verification paths included in the TCP Transport Descriptor.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

Returns uint32_t Number of TLS paths to look for verification files.

void **clear**(const std::string &transport_descriptor_id)

Remove specific transport descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier. If empty, every transport descriptor profile is deleted.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_kind**(const std::string &transport_descriptor_id)

Remove transport descriptor profile kind.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_send_buffer_size**(const std::string &transport_descriptor_id)

Remove send buffer size element from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_receive_buffer_size**(const std::string &transport_descriptor_id)

Remove receive buffer size element from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_max_message_size**(const std::string &transport_descriptor_id)

Remove maximum message size element from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_max_initial_peers_range**(const std::string &transport_descriptor_id)

Remove maximum initial peers range element from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_interface_whitelist**(const std::string &transport_descriptor_id, const std::string &index)

Remove specific whitelisted network interface from specific Transport Descriptor profile.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- *BadParameter* – *Exception* if the index is not an integer.

void **clear_ttl**(const std::string &transport_descriptor_id)

Remove TTL (Time to live) element from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_non_blocking_send**(const std::string &transport_descriptor_id)

Remove non blocking send flag from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_output_port**(const std::string &transport_descriptor_id)

Remove output port from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_wan_addr**(const std::string &transport_descriptor_id)

Remove WAN address from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_keep_alive_frequency_ms**(const std::string &transport_descriptor_id)

Remove TCP keep alive frequency from specific Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_keep_alive_timeout_ms**(const std::string &transport_descriptor_id)

Remove TCP keep alive timeout from specific Transport Descriptor profile.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file.

void `clear_max_logical_port`(const std::string &transport_descriptor_id)

Remove maximum number of logical ports from specific TCP Transport Descriptor profile.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file.

void `clear_logical_port_range`(const std::string &transport_descriptor_id)

Remove maximum number of logical ports per request from specific TCP Transport Descriptor profile.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file.

void `clear_logical_port_increment`(const std::string &transport_descriptor_id)

Remove logical port increment element from specific TCP Transport Descriptor profile.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file.

void `clear_listening_ports`(const std::string &transport_descriptor_id, const std::string &index)

Remove specific listening port from specific Transport Descriptor profile.

Parameters

- `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.
- `index` – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- `BadParameter` – *Exception* if the index is not an integer.

void `clear_tls`(const std::string &transport_descriptor_id)

Remove TLS configuration from TCP Transport Descriptor profile.

Parameters `transport_descriptor_id` – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_password**(const std::string &transport_descriptor_id)

Remove TLS password from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_private_key_file**(const std::string &transport_descriptor_id)

Remove TLS private key certificate path from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_rsa_private_key_file**(const std::string &transport_descriptor_id)

Remove TLS private key RSA certificate path from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_cert_chain_file**(const std::string &transport_descriptor_id)

Remove TLS public certificate chain file path from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_tmp_dh_file**(const std::string &transport_descriptor_id)

Remove TLS Diffie-Hellman parameters file path from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_verify_file**(const std::string &transport_descriptor_id)

Remove TLS CA (Certification Authority) file path from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_verify_mode**(const std::string &transport_descriptor_id, const std::string &index)

Remove specific TLS verification mode from specific Transport Descriptor profile mask.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- `BadParameter` – *Exception* if the index is not an integer.

void **clear_tls_options**(const std::string &transport_descriptor_id, const std::string &index)

Remove specific TLS supported feature from specific Transport Descriptor profile mask.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- `BadParameter` – *Exception* if the index is not an integer.

void **clear_tls_verify_paths**(const std::string &transport_descriptor_id, const std::string &index)

Remove specific TLS verification path from specific Transport Descriptor profile mask.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **index** – **[in]** Collection element to be removed. If empty, the complete collection is erased.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementNotFound` – *Exception* if the specified transport descriptor profile is not found in the XML file or if the element does not exist in the collection if an index is provided.
- `BadParameter` – *Exception* if the index is not an integer.

void **clear_tls_verify_depth**(const std::string &transport_descriptor_id)

Remove TLS maximum verification depth from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_default_verify_path**(const std::string &transport_descriptor_id)

Remove TLS verify default paths flag from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_handshake_role**(const std::string &transport_descriptor_id)

Remove TLS handshake role from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_tls_server_name**(const std::string &transport_descriptor_id)

Remove TLS SNI server name from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_calculate_crc**(const std::string &transport_descriptor_id)

Remove calculate CRC flag from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_check_crc**(const std::string &transport_descriptor_id)

Remove check CRC flag from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_enable_tcp_nodelay**(const std::string &transport_descriptor_id)

Remove TCP_NODELAY socket option flag from TCP Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_segment_size**(const std::string &transport_descriptor_id)

Remove shared memory segment size from SHM Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_port_queue_capacity**(const std::string &transport_descriptor_id)

Remove shared memory listening port capacity from SHM Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_healthy_check_timeout_ms**(const std::string &transport_descriptor_id)

Remove shared memory listening port liveliness timeout from SHM Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **clear_rtps_dump_file**(const std::string &transport_descriptor_id)

Remove shared memory debugging file path from SHM Transport Descriptor profile.

Parameters **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementNotFound* – *Exception* if the specified transport descriptor profile is not found in the XML file.

void **set_kind**(const std::string &transport_descriptor_id, const std::string &kind)

Set the Transport Descriptor kind.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **kind** – **[in]** Transport descriptor kind.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided kind is not valid.

void **set_send_buffer_size**(const std::string &transport_descriptor_id, const std::string &send_buffer_size)

Set the Transport Descriptor send buffer size.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **send_buffer_size** – **[in]** Transport descriptor send buffer size.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided send buffer size is not valid.

void **set_receive_buffer_size**(const std::string &transport_descriptor_id, const std::string &receive_buffer_size)

Set the Transport Descriptor receive buffer size.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **receive_buffer_size** – **[in]** Transport descriptor receive buffer size.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided receive buffer size is not valid.

void **set_max_message_size**(const std::string &transport_descriptor_id, const std::string &max_message_size)

Set the Transport Descriptor maximum message size.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **max_message_size** – **[in]** Transport descriptor maximum message size.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided maximum message size is not valid.

void **set_max_initial_peers_range**(const std::string &transport_descriptor_id, const std::string &max_initial_peers_range)

Set the Transport Descriptor maximum initial peers range.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **max_initial_peers_range** – **[in]** Transport descriptor maximum initial peers range.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided maximum initial peers range is not valid.

void **set_ttl**(const std::string &transport_descriptor_id, const std::string &ttl)

Set the Transport Descriptor TTL (Time to live).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **ttl** – **[in]** Transport descriptor TTL (Time to live) in number of hops.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided TTL is not valid.

void **set_non_blocking_send**(const std::string &transport_descriptor_id, const std::string &non_blocking_send)

Set the Transport Descriptor non blocking send flag (UDP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **non_blocking_send** – **[in]** Enable/disable this flag in the Transport Descriptor.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the flag does not apply to the Transport Descriptor.

void **set_output_port**(const std::string &transport_descriptor_id, const std::string &output_port)

Set the Transport Descriptor output port (UDP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **output_port** – **[in]** UDP Transport output port.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the output port provided is not valid.

void **set_wan_addr**(const std::string &transport_descriptor_id, const std::string &wan_addr)

Set the Transport Descriptor WAN address (TCPv4 Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **wan_addr** – **[in]** TCPv4 Transport WAN address.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the output port provided is not valid.

void **set_keep_alive_frequency_ms**(const std::string &transport_descriptor_id, const std::string &keep_alive_frequency_ms)

Set the TCP Transport Descriptor frequency for keep alive requests in milliseconds (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **keep_alive_frequency_ms** – **[in]** Frequency of TCP keep alive requests (in ms).

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided keep alive frequency is not valid.

void **`set_keep_alive_timeout_ms`**(const std::string &transport_descriptor_id, const std::string &keep_alive_timeout_ms)

Set the TCP Transport Descriptor timeout to consider a connection broken if no keep alive request is received in milliseconds (TCP Transport specific).

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.
- **`keep_alive_timeout_ms`** – **[in]** Time since the last keep alive request to consider the connection broken (in ms).

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided keep alive timeout is not valid.

void **`set_max_logical_port`**(const std::string &transport_descriptor_id, const std::string &max_logical_port)

Set the TCP Transport Descriptor maximum number of logical ports to try during TCP negotiation (TCP Transport specific).

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.
- **`max_logical_port`** – **[in]** Maximum number of logical ports to try during TCP negotiation.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided maximum number of logical ports is not valid.

void **`set_logical_port_range`**(const std::string &transport_descriptor_id, const std::string &logical_port_range)

Set the TCP Transport Descriptor maximum number of logical ports per request to try during TCP negotiation (TCP Transport specific).

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.
- **`logical_port_range`** – **[in]** Maximum number of logical ports per request to try during TCP negotiation.

Throws

- `Error` – *Exception* if the workspace was not initialized.
- `ElementInvalid` – *Exception* if the provided maximum number of logical ports per request is not valid.

void **`set_logical_port_increment`**(const std::string &transport_descriptor_id, const std::string &logical_port_increment)

Set the TCP Transport Descriptor increment between logical ports to try during TCP negotiation (TCP Transport specific).

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.

- **logical_port_increment** – **[in]** Increment between logical ports to try during TCP negotiation.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided maximum number of logical ports per request is not valid.

void **set_tls_password**(const std::string &transport_descriptor_id, const std::string &tls_password)

Set the TCP Transport Descriptor TLS password (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_password** – **[in]** TCP Transport Descriptor TLS password.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided TLS password is not valid.

void **set_tls_private_key_file**(const std::string &transport_descriptor_id, const std::string &tls_private_key_file)

Set the TCP Transport Descriptor TLS private key certificate path (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_private_key_file** – **[in]** Path to the private key certificate file.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided path is not valid.

void **set_tls_rsa_private_key_file**(const std::string &transport_descriptor_id, const std::string &tls_rsa_private_key_file)

Set the TCP Transport Descriptor TLS private key RSA certificate path (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_rsa_private_key_file** – **[in]** Path to the private key RSA certificate file.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided path is not valid.

void **set_tls_cert_chain_file**(const std::string &transport_descriptor_id, const std::string &tls_cert_chain_file)

Set the TCP Transport Descriptor TLS public certificate chain file path (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_cert_chain_file** – **[in]** Path to the public certificate chain file.

Throws

- *Error* – *Exception* if the workspace was not initialized.

- *ElementInvalid* – *Exception* if the provided path is not valid.

void **set_tls_tmp_dh_file**(const std::string &transport_descriptor_id, const std::string &tls_tmp_dh_file)

Set the TCP Transport Descriptor TLS Diffie-Hellman parameters file path (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_tmp_dh_file** – **[in]** Path to the Diffie-Hellman parameters file.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided path is not valid.

void **set_tls_verify_file**(const std::string &transport_descriptor_id, const std::string &tls_verify_file)

Set the TCP Transport Descriptor TLS CA (Certification Authority) file path (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_verify_file** – **[in]** Path to the CA file.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided path is not valid.

void **set_tls_verify_depth**(const std::string &transport_descriptor_id, const std::string &tls_verify_depth)

Set the TCP Transport Descriptor TLS maximum allowed depth for verifying intermediate certificates (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_verify_depth** – **[in]** Maximum allowed depth for verifying intermediate certificates.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided depth is not valid.

void **set_tls_default_verify_path**(const std::string &transport_descriptor_id, const std::string &tls_default_verify_path)

Set the TCP Transport Descriptor TLS verify default paths flag (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_default_verify_path** – **[in]** Flag that enables looking for verification files on the default paths.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided flag value is not valid.

void **set_tls_handshake_role**(const std::string &transport_descriptor_id, const std::string &tls_handshake_role)

Set the TCP Transport Descriptor TLS handshake role (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_handshake_role** – **[in]** Role that the Transport will take on handshaking.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided handshake role is not valid.

void **set_tls_server_name**(const std::string &transport_descriptor_id, const std::string &tls_server_name)

Set the TCP Transport Descriptor TLS SNI server name (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_server_name** – **[in]** SNI (Server Name Indication) server name.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided SNI server name is not valid.

void **set_calculate_crc**(const std::string &transport_descriptor_id, const std::string &calculate_crc)

Set the TCP Transport Descriptor calculate CRC flag (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **calculate_crc** – **[in]** Flag to enable CRC calculation and sending.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided CRC calculation flag is not valid.

void **set_check_crc**(const std::string &transport_descriptor_id, const std::string &check_crc)

Set the TCP Transport Descriptor check CRC flag (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **check_crc** – **[in]** Flag to enable CRC checking of incoming messages.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided check CRC flag is not valid.

void **set_enable_tcp_nodelay**(const std::string &transport_descriptor_id, const std::string &enable_tcp_nodelay)

Enable the TCP Transport Descriptor TCP_NODELAY socket option (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **enable_tcp_nodelay** – **[in]** Flag to enable TCP_NODELAY socket option.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the TCP_NODELAY flag is not valid.

void **set_segment_size**(const std::string &transport_descriptor_id, const std::string &segment_size)

Set the SHM Transport Descriptor shared memory segment size (SHM Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **segment_size** – **[in]** Shared memory segment size.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided segment size is not valid.

void **set_port_queue_capacity**(const std::string &transport_descriptor_id, const std::string &port_queue_capacity)

Set the SHM Transport Descriptor listening port capacity (SHM Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **port_queue_capacity** – **[in]** Listening port capacity in number of messages.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided listening port capacity is not valid.

void **set_healthy_check_timeout_ms**(const std::string &transport_descriptor_id, const std::string &healthy_check_timeout_ms)

Set the SHM Transport Descriptor listening port liveliness timeout (SHM Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **healthy_check_timeout_ms** – **[in]** Listening port liveliness timeout in milliseconds.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided listening port liveliness timeout is not valid.

void **set_rtps_dump_file**(const std::string &transport_descriptor_id, const std::string &rtps_dump_file)

Set the SHM Transport Descriptor path to the debug file (SHM Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **rtps_dump_file** – **[in]** Path to the file where RTPS messages will be stored for debugging purposes.

Throws

- *Error* – *Exception* if the workspace was not initialized.
- *ElementInvalid* – *Exception* if the provided path is not valid.

void **set_interface_whitelist**(const std::string &transport_descriptor_id, const std::string &ip_address, const std::string &index)

Append an IP address to the whitelisted network interfaces collection or update IP address in the whitelisted network interfaces collection.

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **ip_address** – **[in]** IP address to be updated in the whitelist collection.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided IP address is not valid.
- *ElementNotFound* – *Exception* if the specified Transport descriptor profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_listening_ports**(const std::string &transport_descriptor_id, const std::string &port, const std::string &index)

Append a listening port to the TCP Transport Descriptor collection or update a listening port to the TCP Transport Descriptor collection (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **port** – **[in]** Updated port to listen as server.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided port is not valid.
- *ElementNotFound* – *Exception* if the specified Transport descriptor profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.
- *BadParameter* – *Exception* if the index is not an integer.

void **set_tls_verify_mode**(const std::string &transport_descriptor_id, const std::string &tls_verify_mode, const std::string &index)

Append a TLS verification mode to the TCP Transport Descriptor mask or update a TLS verification mode to the TCP Transport Descriptor mask (TCP Transport specific).

Parameters

- **transport_descriptor_id** – **[in]** Transport descriptor profile identifier.
- **tls_verify_mode** – **[in]** TLS verification mode to update in the mask.
- **index** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- *ElementInvalid* – *Exception* if the provided TLS verification mode is not valid.
- *ElementNotFound* – *Exception* if the specified Transport descriptor profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- *Error* – *Exception* if the workspace was not initialized.

- `BadParameter` – *Exception* if the index is not an integer.

void **`set_tls_options`**(const std::string &transport_descriptor_id, const std::string &tls_options, const std::string &index)

Append TLS supported features to the TCP Transport Descriptor mask or update a TLS supported feature to the TCP Transport Descriptor mask (TCP Transport specific).

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.
- **`tls_options`** – **[in]** TLS supported feature to update in the mask.
- **`index`** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided TLS supported feature is not valid.
- `ElementNotFound` – *Exception* if the specified Transport descriptor profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

void **`set_tls_verify_path`**(const std::string &transport_descriptor_id, const std::string &tls_verify_path, const std::string &index)

Append TLS verification path to the TCP Transport Descriptor or update a TLS verification path to the TCP Transport Descriptor (TCP Transport specific).

Parameters

- **`transport_descriptor_id`** – **[in]** Transport descriptor profile identifier.
- **`tls_verify_path`** – **[in]** TLS verification path to be updated in the list.
- **`index`** – **[in]** Collection element to be changed. If empty, a new element is added to the list.

Throws

- `ElementInvalid` – *Exception* if the provided path is not valid.
- `ElementNotFound` – *Exception* if the specified Transport descriptor profile is not found in the XML file, the list element does not exist, or the list does not contain any element in index position.
- `Error` – *Exception* if the workspace was not initialized.
- `BadParameter` – *Exception* if the index is not an integer.

2.16 Exceptions

class **`Exception`** : public exception

Base class for all exceptions thrown by the eProsima Fast DDS QoS Profiles Manager.

Subclassed by *eprosima::qosprof::BadParameter*, *eprosima::qosprof::ElementInvalid*, *eprosima::qosprof::ElementNotFound*, *eprosima::qosprof::Error*, *eprosima::qosprof::Unsupported*

class **BadParameter** : public *eprosima*::*qosprof*::*Exception*
Exception to signal that the provided parameter is not correct.

class **ElementInvalid** : public *eprosima*::*qosprof*::*Exception*
Exception to signal that the provided XML element is not valid according to the XSD Schema.

Query the `what` in order to have more information about the validation failure

class **ElementNotFound** : public *eprosima*::*qosprof*::*Exception*
Exception to signal that the queried XML element does not exist.

class **Error** : public *eprosima*::*qosprof*::*Exception*
Exception to signal unexpected error in library dependencies.

class **Unsupported** : public *eprosima*::*qosprof*::*Exception*
Exception to signal that an operation is not supported.

Query the `what` in order to have more information about the validation failure

2.17 Version 0.1.0

This first release includes the following features:

2.17.1 Fast DDS QoS Profiles Manager Library

1. Library infrastructure.
2. **Implementation of several *set* functions:**
    - DomainParticipant name.
    - DomainParticipant external locators (default and metatraffic): externality, address, and port.
    - DomainParticipant initial peers: address and port.
    - DomainParticipant lease duration and lease announcements.
    - DomainParticipant use of custom transports disabling builtin transports.
    - Reliability and Durability QoS for DataReader and DataWriter profiles.
    - Default profile attribute for DomainParticipant, DataReader and DataWriter.
    - Transport descriptor kind.
    - Transport descriptor whitelist interfaces.

2.17.2 Fast DDS QoS Profiles Manager CLI

- Support for the library features.
- External locators example

2.17.3 Fast DDS QoS Profiles Manager Documentation

- *Installation manual*
- *Library API Reference*
- *CLI documentation for supported features*
- Automatic CI

INDEX

E

(*C++ function*), 116
eprosima::qosprof::domain_participant::default_unicast_locators::set_wan_address
(*C++ function*), 117
eprosima::qosprof::domain_participant::default_unicast_locators::size
(*C++ function*), 113
eprosima::qosprof::domain_participant::keys
(*C++ function*), 28
eprosima::qosprof::domain_participant::port
(*C++ type*), 117
eprosima::qosprof::domain_participant::port::clear
(*C++ function*), 119
eprosima::qosprof::domain_participant::port::clear_base
(*C++ function*), 119
eprosima::qosprof::domain_participant::port::clear_domain_id_gain
(*C++ function*), 119
eprosima::qosprof::domain_participant::port::clear_offset_d0
(*C++ function*), 119
eprosima::qosprof::domain_participant::port::clear_offset_d1
(*C++ function*), 120
eprosima::qosprof::domain_participant::port::clear_offset_d2
(*C++ function*), 120
eprosima::qosprof::domain_participant::port::clear_offset_d3
(*C++ function*), 120
eprosima::qosprof::domain_participant::port::clear_participant_id_gain
(*C++ function*), 119
eprosima::qosprof::domain_participant::port::print
(*C++ function*), 117
eprosima::qosprof::domain_participant::port::print_base
(*C++ function*), 117
eprosima::qosprof::domain_participant::port::print_domain_id_gain
(*C++ function*), 117
eprosima::qosprof::domain_participant::port::print_offset_d0
(*C++ function*), 118
eprosima::qosprof::domain_participant::port::print_offset_d1
(*C++ function*), 118
eprosima::qosprof::domain_participant::port::print_offset_d2
(*C++ function*), 118
eprosima::qosprof::domain_participant::port::print_offset_d3
(*C++ function*), 119
eprosima::qosprof::domain_participant::port::print_participant_id_gain
(*C++ function*), 118
eprosima::qosprof::domain_participant::port::set_base
(*C++ function*), 120
eprosima::qosprof::domain_participant::port::set_domain_id_gain
(*C++ function*), 120
eprosima::qosprof::domain_participant::port::set_offset_d0
(*C++ function*), 121
eprosima::qosprof::domain_participant::port::set_offset_d1
(*C++ function*), 121
eprosima::qosprof::domain_participant::port::set_offset_d2
(*C++ function*), 121
eprosima::qosprof::domain_participant::port::set_offset_d3
(*C++ function*), 121
eprosima::qosprof::domain_participant::port::set_participant_id_gain
(*C++ function*), 121
eprosima::qosprof::domain_participant::print
(*C++ function*), 25
eprosima::qosprof::domain_participant::print_default_profi
(*C++ function*), 25
eprosima::qosprof::domain_participant::print_domain_id
(*C++ function*), 26
eprosima::qosprof::domain_participant::print_ignore_non_ma
(*C++ function*), 26
eprosima::qosprof::domain_participant::print_listen_socket
(*C++ function*), 26
eprosima::qosprof::domain_participant::print_name
(*C++ function*), 26
eprosima::qosprof::domain_participant::print_participant_i
(*C++ function*), 27
eprosima::qosprof::domain_participant::print_prefix
(*C++ function*), 28
eprosima::qosprof::domain_participant::print_send_socket_b
(*C++ function*), 26
eprosima::qosprof::domain_participant::print_use_builtin_t
(*C++ function*), 27
eprosima::qosprof::domain_participant::print_user_data
(*C++ function*), 27
eprosima::qosprof::domain_participant::print_user_transpor
(*C++ function*), 27
eprosima::qosprof::domain_participant::properties_policy
(*C++ type*), 122
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 123
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 124
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 123
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 123
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 122
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 122
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 122
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 124
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 124
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 124
eprosima::qosprof::domain_participant::properties_policy::
(*C++ function*), 123
eprosima::qosprof::domain_participant::set_default_profile
(*C++ function*), 31
eprosima::qosprof::domain_participant::set_domain_id
(*C++ function*), 31
eprosima::qosprof::domain_participant::set_ignore_non_matc