DDS Participants and Topics

ParticipantSingleton.hpp

https://github.com/embedded-software-laboratory/cpm_lab/blob/master/cpm_lib/include/cpm/ParticipantSingleton.hpp

Most communication takes place within a single DDS domain. As - per application - only one participant should be created per domain, the most commonly used domain participant is provided via a Singleton. It is used in the cpm library as well. Whenever you need to communicate with the vehicles, parts of the LCC or other parts within the lab that do not use their own participant (like Middleware < - > HLC), use this participant. Most of our constructors give you an option to leave out a participant e.g. when creating a Writer - in that case, **the default choice is the ParticipantSingleton**.

Participant.hpp

https://git.rwth-aachen.de/CPM/Project/Lab/software/-/blob/master/cpm_lib/include/cpm/Participant.hpp

A simple participant abstraction. You can pass a domain number and an optional QoS file. Use this whenever you need to create a participant that is not part of the common DDS domain given by our ParticipantSingleton. This participant **needs to be passed explicitly** to e.g. a Writer so that it does not use the default participant.

get_topic.hpp

https://github.com/embedded-software-laboratory/cpm_lab/blob/master/cpm_lib/include/cpm/get_topic.hpp

For each domain participant, a topic can only be created if a topic with the same name has not been created before. This can lead to some trouble, e.g. if different classes depend on the same topic object but sharing it would be too complicated or would result in ugly code / a lack of data abstraction. RTI DDS provides a `find` method to find already created topics. As the user should, in our opinion, not be forced to decide which object should be created first or to find out which topics have already been created by the lib, get_topic uses both the create and the find method. Independent from the order in which the objects are created, get_topic returns the required topic object without throwing an error.

If only the topic name is passed as a parameter, a topic is created / looked up for the participant singleton instance. Else, the participant must be passed as well.

Most of our other wrappers, e.g. the Writer, do not require you to specify a topic - a topic string for the topic name suffices. The topic is then looked up internally.

VehicleIDFilteredTopic.hpp

https://github.com/embedded-software-laboratory/cpm_lab/blob/master/cpm_lib/include/cpm/VehicleIDFilteredTopic.hpp

If not all data that is received by a DDS Reader is of interest, a filter can be used to only process data that matches the filter expression. The most commonly used filter in this project filters by the vehicle id. The VehicleIDFilteredTopic thus provides a wrapper that uses such a filter. The vehicle id and the topic can be passed as a parameter, and the returned object can be used instead of the topic object. This allows e.g. the Reader to only process information that is relevant for it.