Functional description of the DEEP framework

The DEEP framework contains only one DCPS application, called the deepParticipant. This application can be started multiple times an every instance of this application can be configured individually. The complete set of deepParticipants that run in a test, together with their configuration settings, is called a scenario. The DEEP framework will be a success only if the configuration possibilities are sufficient and easy to use. This document describes the current DEEP scenario design. It is accompanied by a UML diagram that shows the relations of the different entities in the scenario.

The entities in the DEEP scenario and their relations

Conceptually, every scenario is identified by its unique name. This name is used to identify the resource where the scenario details are stored. It is up to the final implementation to decide how a scenario name is translated into the actual resource and how this resource is looked up.

Every scenario defines an unlimited number of participants. Every participant has a name that is unique within its scenario. Every participant in a scenario maps onto the deepParticipant application. The syntax for starting a deepParticipant with name *participantId*, defined in scenario with name *scenarioId* is therefore:

```
deepParticipant scenarioId participantId
```

The deepParticipant will try to find the scenario and lookup the participant in the scenario description. The actual starting of the deepParticipants is outside of the scope of this document. This might be done using a simple script or by using another framework that reads the scenario file and decides what participants to start at what location.

Every participant defines a number of actors that it owns. Any of these actors is either a source, a reflector or a sink. Every actor is associated with a single topic, so an actor can act on one topic only. Actors are also linked to a reader or a writer, or both. A source actor is linked to a writer only. A sink actor is linked to a reader only. A reflector actor is linked to both a writer and a reader.

Detailed descriptions of the configuration of every entity

The table below gives an overview of all configuration settings per entity and their meaning.

Entity	Setting	Type	Description
Scenario	name	string	Name that uniquely identifies the scenario
	participants	Participant[]	Set of references to Participants that will participate in this scenario

Entity	**Setting**	**Type**	**Description**
Participant	name	string	Name that uniquely identifies the participant with the scenario
	domainId	string	String representation of the DDS domainId to connect to
	actors	Source[], Reflector[], Sink[]	Set of references to Actors that are managed and run by the participant
Source	name	string	Name that uniquely identifies the Source configuration within the Scenario
	topicName	string	Name of the topic that will be used by this Source
	typeName	string	Name of the type that will be linked to the topic. This is restricted by a set of predefined strings, corresponding to the different types that are supported by the framework
	writerPartitionExpression	string	The partition expression the Source will write into
	priority	natural	OS priority used for the thread the Source runs in
	burstSize	natural	Number of samples the Source will write within one burst
	nofBursts	natural	Number of bursts that the Source will write in total
	sleepTime	natural	The number of milliseconds the Source will sleep between two bursts
	topic	Topic	Reference to the Topic configuration that will be used by the Source
	writer	Writer	
Reflector	name	string	Name that uniquely identifies the Reflector configuration within the Scenario
	topicName	string	Name of the topic that will be used by this Reflector
	typeName	string	Name of the type that will be linked to the topic. This is restricted by a set of predefined strings, corresponding to the different types that are supported by the framework
	writerPartitionExpression	string	The partition expression the Reflector will write into

Entity	Setting	Type	Description
	readerPartitionExpression	string	The partition expression the Reflector will read from
	doTiming	boolean	A flag indicating whether this Reflector should be calculating latencies for every sample received
	timeoutPeriod	natural	The number of milliseconds to wait for newly arrived data. The Reflector will terminate if no new data has arrived
	topic	Topic	Reference to the Topic configuration that will be used by the Reflector
	writer	Writer	Reference to the Writer configuration that will be used by the Reflector
	reader	Reader	Reference to the Reader configuration that will be used by the Reflector
Sink	name	string	Name that uniquely identifies the Sink configuration within the Scenario
	topicName	string	Name of the topic that will be used by this Sink
	typeName	string	Name of the type that will be linked to the topic. This is restricted by a set of predefined strings, corresponding to the different types that are supported by the framework
	readerPartitionExpression	string	The partition expression the Sink will read from
	doTiming	boolean	A flag indicating whether this Sink should be calculating latencies for every sample received
	timeoutPeriod	natural	The number of milliseconds to wait for newly arrived data. The Sink will terminate if no new data has arrived
	topic	Topic	Reference to the Topic configuration that will be used by the Sink
	reader	Reader	Reference to the Reader configuration that will be used by the Sink
Topic	name	string	Name that uniquely identifies the Topic configuration within the Scenario
	reliable	boolean	Reliability quality of service value used for the Topic.
Writer	name	string	Name that uniquely identifies the Writer configuration within the scenario

Entity	**Setting**	**Type**	**Description**
	urgency	natural	An indication of the urgency of the data written by the Writer. This maps onto the DCPS latency budget quality of service
	importance	natural	An indication of the importance of the data written by the Writer. This maps onto the DCPS transport priority quality of service
Reader	name	string	Name that uniquely identifies the Reader configuration within the senario

As the table shows, every entity has its unique name. This makes it possible to use settings by reference, meaning that a single setting can be reused by several entities. For example, a Scenario containing several Sources with the same characteristics, requires only one Source configuration item. Every Source will be configured by referring to this single configuration item.