OpenSplice DDS

Version 5.x

Deployment Guide

PRISMTECH

OpenSplice DDS

DEPLOYMENT GUIDE

Part Number: OS-DG

Doc Issue 33, 9 June 2010

Copyright Notice

Contents

Table of Contents

Preface

About the Deployment Guide . . . . . **xi**
Contacts . . . . . **xii**

Deploying OpenSplice DDS

Chapter 1 **Overview** **3**

1.1 **The OpenSplice DDS Architecture** **3**
1.2 **The OpenSplice DDS Services** **4**
1.2.1 The Domain Service 4
1.2.2 The Durability Service. 5
1.2.3 The Networking Service 5
1.2.4 The Tuner Service 6
1.2.5 The DBMS Service 6
1.3 **OpenSplice DDS Usage.** **6**
1.3.1 Starting. 7
1.3.2 Monitoring 7
1.3.2.1 Diagnostic Messages 7
1.3.2.2 OpenSplice Tuner 7
1.3.2.3 OpenSplice Memory Management Statistics Monitor 8
1.3.2.4 OpenSplice Shared Memory Dump Tool 8
1.3.2.5 OpenSplice Configurator (Beta) 8
1.3.3 Stopping 8
1.3.3.1 Stopping OSPL by using signals 8
1.3.3.2 Stopping Applications 9
1.3.4 Deploying OpenSplice DDS on VxWorks 6.x 10
1.3.5 Deploying OpenSplice DDS on Integrity 10
1.3.6 Installing/Uninstalling the OpenSplice DDS C# Assembly to the Global Assembly Cache 10
1.3.6.1 Installing the C# Assembly to the Global Assembly Cache. 10
1.3.6.2 Uninstalling the C# Assembly from the Global Assembly Cache 11
1.4 **OpenSplice DDS Configuration.** **12**
1.4.1 Configuration Files 12
1.4.2 Environment Variables 13
1.4.3 Temporary Files. 13
1.5 **The DDSI Networking Service** **14**
1.6 **Applications which operate in multiple domains.** **14**
1.6.1 Interaction with a Networking Service 15

Chapter 2 **DbmsConnect** **17**
2.1 **Introduction** **17**
2.2 **Usage** **18**
2.2.1 DDS and DBMS Concepts and Types Mapping 18
2.2.2 General DbmsConnect Concepts. 20
2.2.3 DDS to DBMS Use Case. 20
2.2.4 DBMS to DDS Use Case. 22
2.2.5 Replication Use Case. 23
Chapter 3 **Service Configuration** **25**
3.1 **Introduction** **25**
3.2 **The Domain Service** **26**
3.2.1 Element name 26
3.2.2 Element ***Role*** ***27***
3.2.3 Element Lease 28
3.2.3.1 Element ExpiryTime. 28
3.2.4 Element ServiceTerminatePeriod 29
3.2.5 Element Database 29
3.2.5.1 Element Size. 30
3.2.5.2 Element Address. 30
3.2.5.3 Element Locking. 31
3.2.6 Element Service. 31
3.2.6.1 Attribute name 32
3.2.6.2 Attribute enabled 32
3.2.6.3 Element Command. 33
3.2.6.4 Element MemoryPoolSize 33
3.2.6.5 Element HeapSize. 34
3.2.6.6 Element StackSize 34
3.2.6.7 Element Configuration 35
3.2.6.8 Element Scheduling 36
3.2.6.9 Element Locking. 37
3.2.6.10 Element FailureAction 38
3.2.7 Element Listeners 38
3.2.7.1 Element StackSize 39
3.2.8 Element BuiltinTopics. 39
3.2.8.1 Attribute enabled 39
3.2.9 Element Statistics 40
3.2.9.1 Element Category 40
3.2.10 Element ***ReportPlugin*** ***41***
3.2.10.1 Element ***Library*** ***41***
3.2.10.2 Element ***Initialize*** ***42***
3.2.10.3 Attribute symbol_name 42

3.2.10.4 Attribute argument 43
3.2.10.5 Element ***Report*** ***43***
3.2.10.6 Attribute symbol_name 44
3.2.10.7 Element ***Finalize*** ***45***
3.2.10.8 Attribute symbol_name 45
3.2.11 Element ***PartitionAccess*** ***45***
3.2.11.1 Attribute partition_expression 46
3.2.11.2 Attribute access_mode 46
3.2.12 Element ***TopicAccess*** ***47***
3.2.12.1 Attribute topic_expression 48
3.2.12.2 Attribute access_mode 48
3.3 **The Daemon Service** **49**
3.3.1 Element Locking 49
3.3.2 Element KernelManager 50
3.3.2.1 Element Scheduling 51
3.3.3 Element GarbageCollector 52
3.3.3.1 Element Scheduling 53
3.4 **The Durability Service** **54**
3.4.1 Attribute name 55
3.4.2 Element Network 55
3.4.2.1 Attribute latency_budget 55
3.4.2.2 Attribute transport_priority 56
3.4.2.3 Element Heartbeat 56
3.4.2.4 Element InitialDiscoveryPeriod 60
3.4.2.5 Element Alignment 61
3.4.2.6 Element WaitForAttachment 68
3.4.3 Element Persistent 69
3.4.3.1 Element StoreDirectory 70
3.4.3.2 Element StoreMode 70
3.4.3.3 Element StoreSessionTime 71
3.4.3.4 Element StoreSleepTime 71
3.4.3.5 Element StoreOptimizeInterval 72
3.4.3.6 Element Scheduling 72
3.4.4 Element *NameSpaces* *74*
3.4.4.1 Element *NameSpace* *75*
3.4.4.2 Element *Policy* *75*
3.4.5 Element Watchdog 78
3.4.5.1 Element Scheduling 79
3.4.6 Element EntityNames 80
3.4.6.1 Element Publisher 81
3.4.6.2 Element Subscriber 81
3.4.6.3 Element Partition 81

3.4.7 Element Tracing 82
3.4.7.1 Element OutputFile. 82
3.4.7.2 Element Timestamps 83
3.4.7.3 Element Verbosity 83
***3.5* The Networking Service. 84**
3.5.1 Attribute name 85
3.5.2 Element General 85
3.5.2.1 Element NetworkInterfaceAddress. 85
3.5.2.2* Element *Reconnection* *86
3.5.3 Element Partitioning 87
3.5.3.1 Element GlobalPartition 88
3.5.3.2 Element NetworkPartitions. 88
3.5.3.3 Element IgnoredPartitions 91
3.5.3.4 Element PartitionMappings 92
3.5.4 Element Channels 94
3.5.4.1 Element Channel. 94
3.5.5 Element Discovery 110
3.5.5.1* Attribute *Scope* *111
3.5.5.2* Element *ProbeList* *111
3.5.5.3 Element Active 112
3.5.5.4 Element PortNr 112
3.5.5.5 Element Sending. 113
3.5.5.6 Element Receiving 116
3.5.6 Element Tracing 119
3.5.6.1 Element OutputFile. 120
3.5.6.2 Element Timestamps 120
3.5.6.3 Element Categories. 121
***3.6* The Tuner Service 126**
3.6.1 Attribute name 127
3.6.2 Element Client 127
3.6.2.1 Element LeasePeriod 127
3.6.2.2 Element MaxClients 128
3.6.2.3 Element MaxThreadsPerClient. 128
3.6.2.4 Element Scheduling 129
3.6.3 Element Server 131
3.6.3.1 Element Backlog. 131
3.6.3.2 Element PortNr 131
3.6.3.3 Element Verbosity 132
3.6.4 Element GarbageCollector 132
3.6.4.1 Element Scheduling 132
3.6.5 Element LeaseManagement. 134
3.6.5.1 Element Scheduling 134

3.7 **The DbmsConnect Service 136**
3.7.1 Attribute name 136
3.7.2 Element DdsToDbms. 137
3.7.2.1 Attribute replication_mode 137
3.7.2.2 Element NameSpace 138
3.7.3 Element DbmsToDds. 145
3.7.3.1 Attribute event_table_policy 145
3.7.3.2 Attribute publish_initial_data 146
3.7.3.3 Attribute replication_user 146
3.7.3.4 Attribute trigger_policy 147
3.7.3.5 Element NameSpace 148
3.7.4 Element Tracing. 159
3.7.4.1 Element OutputFile. 159
3.7.4.2 Element Timestamps 160
3.7.4.3 Element Verbosity 161
3.8 **The UserClock Service 161**
3.8.1 Attribute name 162
3.8.2 Element UserClockModule 162
3.8.3 Element UserClockStart 162
3.8.3.1 Attribute name 163
3.8.4 Element UserClockStop. 163
3.8.4.1 Attribute name 163
3.8.5 Element UserClockQuery 164
3.8.5.1 Attribute name 164
3.9 **The DDSI Networking Service 165**
3.9.1 Attribute name 165
3.9.2 Element General 165
3.9.2.1 Element NetworkInterfaceAddress. 166
3.9.3 Element Partitioning 166
3.9.3.1 Element GlobalPartition 167
3.9.4 Element Channels 168
3.9.4.1 Element Channel. 168
3.9.4.2 Element FragmentSize 170
3.9.4.3 Element GroupQueueSize 170
3.9.4.4 Element PortNr 171
3.9.5 Element Discovery. 171
3.9.5.1 Attribute enabled 171
3.9.5.2 Element FragmentSize 172
3.9.5.3 Element PortNr 172
3.9.6 Element Tracing. 173
3.9.6.1 Attribute enabled 173
3.9.6.2 Element OutputFile. 173

3.9.7 Example Configuration 174
3.10 **Example Reference Systems 175**
3.10.1 Zero Configuration System 175
3.10.2 Single Node System. 175
3.10.3 Medium Size Static (Near) Real-time System. 175
3.10.3.1 High Volumes. 176
3.10.3.2 Low Latencies. 176
3.10.3.3 Responsiveness. 177
3.10.3.4 Discovery 177

Preface

About the Deployment Guide

The OpenSplice DDS *Deployment Guide* is intended to provide a complete reference on how to configure the OpenSplice service and tune it as required.

The *Deployment Guide* is included with the OpenSplice DDS *Documentation Set*. The *Deployment Guide* is intended to be used after reading and following the instructions in the OpenSplice *Getting Started. Guide*.

Intended Audience

The *Deployment Guide* is intended to be used by anyone who wishes to use and configure the *OpenSplice DDS* service.

Organisation

Chapter 1, *Overview*, gives a general description of the OpenSplice architecture.

Chapter 2, *DbmsConnect* describes how OpenSplice provides integration of real-time DDS and the non-/near-real-time enterprise DBMS domains.

Chapter 3, *Service Configuration*. describes how to configure the OpenSplice daemons.

Conventions

The conventions listed below are used to guide and assist the reader in understanding the Deployment Guide.


Item of special significance or where caution needs to be taken.

i
Item contains helpful hint or special information.

WIN
Information applies to Windows (*e.g.* XP, 2003, Windows 7) only.

UNIX
Information applies to Unix based systems (*e.g.* Solaris) only.

C
C language specific

C++
C++ language specific

Java
Java language specific

Hypertext links are shown as *blue italic underlined.*

On-Line (PDF) versions of this document: Items shown as cross references, e.g. *Contacts* on page xii, are as hypertext links: click on the reference to go to the item.

```
%  Commands or input which the user enters on the
   command line of their computer terminal
```

`Courier` fonts indicate programming code and file names.

Extended code fragments are shown in shaded boxes:

```
NameComponent newName[] = new NameComponent[1];

// set id field to "example" and kind field to an empty string
newName[0] = new NameComponent ("example", "");
```

Italics and ***Italic Bold*** are used to indicate new terms, or emphasise an item.

Arial Bold is used to indicate user related actions, e.g. **File | Save** from a menu.

Step 1: One of several steps required to complete a task.

Contacts

PrismTech can be reached at the following contact points for information and technical support.

USA Corporate Headquarters

PrismTech Corporation
400 TradeCenter
Suite 5900
Woburn, MA
01801
USA

Tel: +1 781 569 5819

European Head Office

PrismTech Limited
PrismTech House
5th Avenue Business Park
Gateshead
NE11 0NG
UK

Tel: +44 (0)191 497 9900
Fax: +44 (0)191 497 9901

Web: *http://www.prismtech.com*
Technical questions: *crc@prismtech.com* (Customer Response Center)
Sales enquiries: *sales@prismtech.com*

Deploying OpenSplice DDS

CHAPTER

1 Overview

This chapter explains the OpenSplice DDS middleware from a configuration perspective. It shows the different components running on a single node and briefly explains the role of each entity. Furthermore, it defines a reference system that will be used throughout the rest of the document as an example.

1.1 The OpenSplice DDS Architecture

OpenSplice DDS utilizes a shared-memory architecture where data is physically present only once on any machine and where smart administration still provides each subscriber with his own private view on this data. This architecture enables a subscriber's data cache to be seen as an individual database. This database can have it content filtered, queried, etc. by using the OpenSplice's *content-subscription profile*. This shared-memory architecture results in an extremely low footprint, excellent scalability and optimal performance when compared to implementations where each reader/writer are communication-end points each with its own storage (i.e. historical data both at reader and writer) and where the data itself still has to be moved, even within the same platform.

Shared-memory is not only used to interconnect all applications that reside within one computing node, but also for a configurable and extensible set of services. These services provide pluggable functionality such as:

- *networking* - providing QoS-driven real-time networking based on multiple reliable multicast 'channels'
- *durability* - providing fault-tolerant storage for both real-time state data as well as persistent settings
- *remote control and monitoring SOAP service* - providing remote web-based access using the SOAP protocol from the OpenSplice Tuner tool
- *dbms service* - providing a connection between the real-time and the enterprise domain by bridging data from DDS to DBMS and *vice versa*

The OpenSplice DDS middleware can be easily configured, on the fly, using its pluggable architecture: the services that are needed can be specified together with their optimum configuration for the particular application domain, including networking parameters, and durability levels for example). *Figure 1* shows an overview of the pluggable service architecture of OpenSplice DDS on one computing node. Typically, there are many nodes within a system.

The OpenSplice DDS middleware and its services can be configured using XML files, which are easy to maintain. These services and the XML configuration syntax are described below.

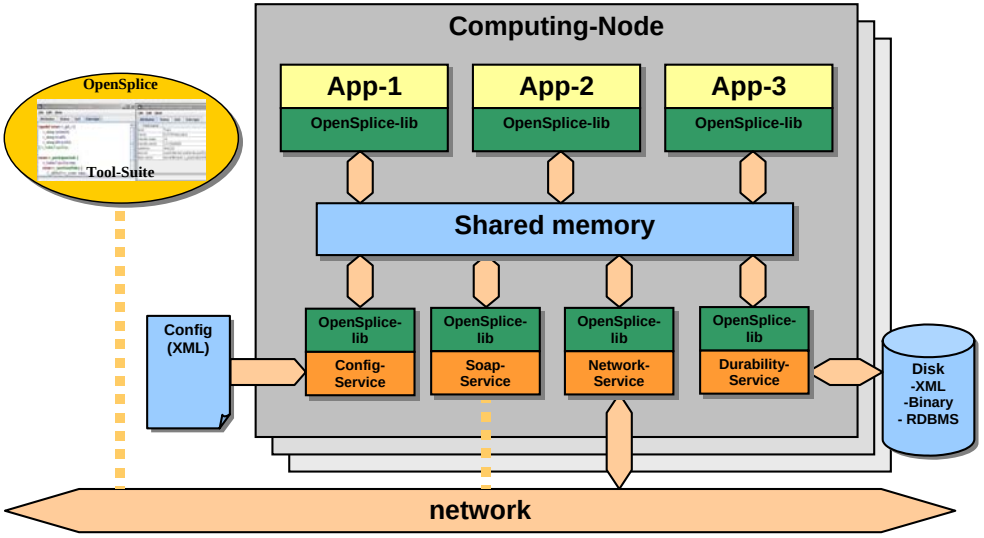

Figure 1 The OpenSplice Single Node Architecture

1.2 The OpenSplice DDS Services

The OpenSplice DDS middleware includes several services: each service has a particular responsibility. *Figure 1* shows the services included with OpenSplice DDS. Each service can be enabled or disabled. The services can be configured or tuned to meet the optimum requirements of a particular application domain (noting that detailed knowledge of the requirement is needed for effective tuning).

The following sections briefly explain each of the services and their responsibilities.

1.2.1 The Domain Service

The Domain Service is responsible for creating and initialising a shared nodal administration, in shared memory, for a specific DDS Domain on a computing node. Without this administration, no other service or application is able to participate in a DDS Domain.

Once the administration has been initialised, the Domain Service starts the set of pluggable services. The lifecycle of the started services is under control of the Domain Service, which means it will monitor the health of all started services, take corrective actions if needed and stop the services when it is terminated.

When a shutdown of the OpenSplice Domain Service is requested, it will react by announcing the shutdown using the shared administration. Applications will not be able to use DDS functionality anymore and services are requested to terminate elegantly. Once this has succeeded, the Domain Service will destroy the shared administration and finally terminate itself.

The exact fulfilment of these responsibilities is determined by the configuration of the Domain Service. An overview of the available configuration parameters and their purpose is presented in Section 3.2, *The Domain Service*, on page 26.

1.2.2 The Durability Service

The OpenSplice DDS middleware supports the concept of so called *durable data*. Durable data produced by applications must stay available for late joining DataReaders. This means that DataReaders joining the system at a later stage will be able to receive (durable) data that has been produced before they joined. The durability of data can be either transient or persistent and is determined by the Quality of Service of the Topic. If a specific Topic is marked to be transient, the corresponding data instances remain available in the system during the complete lifecycle of the system. If a specific Topic is marked to be persistent, the corresponding data instances even survive the shutdown of a system because they are written to a permanent storage e.g. a hard disk.

The Durability Service is responsible for the realisation of these durable properties of the data in the system. The exact fulfilment of these responsibilities depends on the application domain, because scalability of durable data is an issue in large systems. Keeping all durable data on each computing node may not be feasible. Often, computing nodes are interested in a small part of the total system data, on one hand driven by application interest, on the other hand by fault-tolerance (the need for replicates).

The exact fulfilment of the durability responsibilities is determined by the configuration of the Durability Service. An overview of the available configuration parameters and their purpose is presented in Section 3.4, *The Durability Service*, on page 54.

1.2.3 The Networking Service

When communication endpoints are located on different computing nodes, the data produced using the local Domain Service must be communicated to the remote Domain Services and the other way around. The Networking Service provides a bridge between the local Domain Service and a network interface. Multiple Networking Services can exist next to each other; each serving one or more physical network interfaces. The Networking Service is responsible for forwarding data to the network and for receiving data from the network. It can be configured to distinguish multiple communication channels with different QoS policies. These

policies will be used to schedule individual messages to specific channels, which may be configured to provide optimal performance for a specific application domain.

The exact fulfilment of these responsibilities is determined by the configuration of the Networking Service. An overview of the available configuration parameters and their purpose is presented in Section 3.5, *The Networking Service*, on page 84.

1.2.4 The Tuner Service

The Tuner Service provides a remote interface to the monitor and control facilities of OpenSplice by means of the SOAP protocol. This enables the OpenSplice Tuner to remotely monitor and control, from any *reachable* location, OpenSplice services as well as the applications that use OpenSplice for the distribution of their data.

The exact fulfilment of these responsibilities is determined by the configuration of the Tuner Service. An overview of the available configuration parameters and their purpose is presented in Section 3.5.6, *Element Tracing*, on page 119.

1.2.5 The DBMS Service

The DBMS Service provides a bridge between the real-time and enterprise domain. The DBMS Service is capable of bridging data from DDS to any ODBC-enabled DBMS and *vice-versa*. This provides many new capabilities, such as the logging of DDS data in a DBMS as well as QoS-enabled replication of a DBMS using DDS.

The precise implementation of these features is determined by the configuration of the DBMS Service. An overview of the available configuration parameters and their puspose is presented in Section 3.7, *The DbmsConnect Service*, on page 136.

1.3 OpenSplice DDS Usage

It is not possible to do any DDS activities with applications before the OpenSplice Domain Service has created and initialised the nodal shared administration. The OpenSplice environment has to be set up to instruct the node where executables and libraries can be found in order to be able to start the Domain Service.

On UNIX-like platforms this can be realized by starting a shell and sourcing the release.com file located in the root directory of the OpenSplice installation:

```
%  . ./release.com
```

On the Windows platform the environment must be set up by running `release.bat`, or else the OpenSpliceDDS Command Prompt must be used.

1.3.1 Starting

Once the OpenSplice environment has been set up, the Domain Service can be started by means of the `ospl` tool by typing:

```
%  ospl start
```

This will start the Domain Service using the default configuration.

NOTE: The **Integrity** version of OpenSplice DDS does not include the `ospl` program. Instead there is a project generator, `ospl_projgen`, which generates projects containing the required address spaces which will auto-start when loaded. Please refer to the *Getting Started Guide* for details.

NOTE: The **VxWorks** version of OpenSplice DDS does not include the `ospl` program. Please refer to the *Getting StartedGuide* for details of how to use VxWorks projects and Real Time Processes to deploy OpenSplice DDS applications.

1.3.2 Monitoring

The OpenSplice Domain Service can be monitored and tuned in numerous ways after it has been started. The monitoring and tuning capabilities are described in the following subsections.

1.3.2.1 Diagnostic Messages

OpenSplice outputs diagnostic information. This information is written to the *`ospl-info.log`* file located in the start-up directory, by default. Error messages are written to the *`ospl-error.log`* file, by default. The state of the system can be determined from the information written into these files.

The location where the information and error messages are stored can be overridden by setting the `OSPL_LOGPATH` environment variable to a location on disk (by specifying a path), to standard out (by specifying <*stdout*>) or to standard error (by specifying <*stderr*>). The names of these log files can also be changed by setting the `OSPL_INFOFILE` and `OSPL_ERRORFILE` variables.

1.3.2.2 OpenSplice Tuner

The intention of OpenSplice Tuner, *`osplt un`*, is to provide facilities for monitoring and controlling OpenSplice, as well as the applications that use OpenSplice for the distribution of data. The *OpenSplice Tuner User Guide* specifies the capabilities of OpenSplice Tuner and describes how they should be used.

1.3.2.3 OpenSplice Memory Management Statistics Monitor

The OpenSplice Memory Management Statistics Tool, *mmstat*, provides a command line interface that allows monitoring the status of the nodal shared administration (shared memory) used by the middleware and the applications. Use the help switch (*mmstat -h*) for usage information.

1.3.2.4 OpenSplice Shared Memory Dump Tool

The OpenSplice Shared Memory Dump Tool, *shmdump*, provides facilities to make a snapshot of the current status of the nodal shared administration to file and also allows restoring the snapshot from file back to shared memory. Use the help switch (*shmdump -h*) for usage information.

Please note that *shmdump* is suitable for diagnostic purposes only.

1.3.2.5 OpenSplice Configurator (Beta)

The OpenSplice configurator, *osplconf*, provides facilities to create, modify and save OpenSplice configuration files. There is no usage information for the tool at the time of this release.

1.3.3 Stopping

The OpenSplice Domain Service can be stopped by issuing the following command on the command-line.

```
%  ospl stop
```

The OpenSplice Domain Service will react by announcing the shutdown using the shared administration. Applications will not be able to use DDS functionality anymore and services will terminate elegantly. Once this has succeeded, the Domain Service will destroy the shared administration and finally terminate itself.

1.3.3.1 Stopping OSPL by using signals

Alternatively the OpenSplice DDS domain service can also be stopped by sending a signal to the ospl process, assuming the process was started using the `-f` option. For example, on Unix you could use the following command to send a termination signal to the `ospl` tool, where `pid` identifies the `ospl` tool pid:.

```
%  kill -SIGTERM <pid>
```

Sending such a signal will cause the `ospl` tool to exit gracefully, properly terminating all services and exiting with returncode `0`.

The following table shows a list of all POSIX signals and what the behavior of OSPL is when that signal is sent to the `ospl` tool.

Signal	Default action	OSPL action	Description
SIGHUP	Term.	Graceful exit	Hang up on controlling process
SIGINT	Term.	Ignore	Interrupt from keyboard
SIGQUIT	Core	Ignore	Quit from keyboard
SIGILL	Core	Graceful exit	Illegal instruction
SIGABRT	Core	Graceful exit	Abort signal from abort function
SIGFPE	Core	Graceful exit	Floating point exception
SIGKILL	Term.	Term.	Kill signal (can't catch, block, ignore)
SIGSEGV	Core	Graceful exit	Invalid memory reference
SIGPIPE	Term.	Graceful exit	Broken pipe: write to pipe with no readers
SIGALRM	Term.	Graceful exit	Timer signal from alarm function
SIGTERM	Term.	Graceful exit	Termination signal
SIGUSR1	Term.	Graceful exit	User defined signal 1
SIGUSR2	Term.	Graceful exit	User defined signal 2
SIGCHLD	Ignore	Ignore	A child process has terminated or stopped
SIGCONT	Ignore	Ignore	Continue if stopped
SIGSTOP	Stop	Stop	Stop process (can't catch, block, ignore)
SIGTSTOP	Stop	Graceful exit	Stop typed at tty
SIGTTIN	Stop	Graceful exit	Tty input for background process
SIGTTOUT	Stop	Graceful exit	Tty output for background process

1.3.3.2 Stopping Applications

Applications that are connected to and use OpenSplice DDS must *not* be terminated with a `KILL` signal. This will ensure that OpenSplice DDS shared memory always remains in a valid, functional state.

When OpenSplice applications terminate naturally, a cleanup mechanism is executed that releases any references held to the shared memory within OpenSplice which that application was using. This mechanism will be executed even when an application is terminated by other means, *e.g.* by terminating with **Ctrl+C**, or even if the application crashes in the user code.

The cleanup mechanisms are *not* executed when an application is terminated with a `KILL` signal. For this reason a user must not terminate an application with a `kill -9` command (or, on Windows, must not use TaskManager's **End Process** option) because the process will be forcibly removed and the cleanup mechanisms

will be prevented from executing. If an application is killed in this manner, the shared memory regions of OpenSplice will become inconsistent and no recovery will then be possible other than re-starting OpenSplice and all applications on the node.

1.3.4 Deploying OpenSplice DDS on VxWorks 6.x

The **VxWorks** version of OpenSplice DDS does not include the `ospl` program. Please refer to Chapter 7 of the *Getting StartedGuide* for details of how to use VxWorks projects and Real Time Processes to deploy OpenSplice DDS applications.

1.3.5 Deploying OpenSplice DDS on Integrity

The **Integrity** version of OpenSplice DDS does not include the `ospl` program. Instead there is a project generator, `ospl_projgen`, which generates projects containing the required address spaces which will auto-start when loaded. Please refer to Chapter 8 of the *Getting Started Guide* for detailed information about OpenSplice deployment on Integrity.

1.3.6 Installing/Uninstalling the OpenSplice DDS C# Assembly to the Global Assembly Cache

The installer for the commercial distribution of OpenSplice DDS includes the option to install the C# Assembly to the Global Assembly Cache during the installation process. If you chose to omit this step, or you are an open source user, then you should follow the instructions in the next few paragraphs, which describe how to manually install and uninstall the assembly to the Global Assembly Cache.

1.3.6.1 Installing the C# Assembly to the Global Assembly Cache

To install an assembly to the Global Assembly Cache, you need to use the `gacutil.exe` tool. Start a Visual Studio command prompt and type:

```
%  gacutil /i <OpenSpliceDDS installation
path>\lib\dcpssacsAssembly.dll
```

where `<OpenSpliceDDS installation path>` is the installation path of the OpenSplice DDS distribution. If you are successful you will see a message similar to the following:

```
%  C:\Program Files\Microsoft Visual Studio 9.0\VC>gacutil.exe /i
"C:\Program Files \PrismTech\OpenSpliceDDS\V5.1.0\HDE\x86.win32\
lib\dcpssacsAssembly.dll"

%
```

```
%  Microsoft (R) .NET Global Assembly Cache Utility.  Version
3.5.30729.1

%  Copyright (c) Microsoft Corporation.  All rights reserved.

%

%  Assembly successfully added to the cache

%

%  C:\Program Files\Microsoft Visual Studio 9.0\VC>
```

1.3.6.2 Uninstalling the C# Assembly from the Global Assembly Cache

To uninstall an assembly from the Global Assembly Cache, you need to use the `gacutil.exe` tool. Start a Visual Studio command prompt and type:

```
%  gacutil /u dcpssacsAssembly,Version=<version_number_goes_here>
```

The version number of the assembly is defined in the `<OpenSpliceDDS installation path>\etc\RELEASEINFO` file, in the `CS_DLL_VERSION` variable.

If you are successful you will see a message similar to the following:

```
%  C:\Program Files\Microsoft Visual Studio 9.0\VC>gacutil /u
dcpssacsAssembly,Version=5.1.0.14734

%  Microsoft (R) .NET Global Assembly Cache Utility.  Version
3.5.30729.1

%  Copyright (c) Microsoft Corporation.  All rights reserved.

%

%  Assembly: dcpssacsAssembly, Version=5.1.0.14734,
Culture=neutral, PublicKeyToken=5b9310ab51310fa9,
processorArchitecture=MSIL

%  Uninstalled: dcpssacsAssembly, Version=5.1.0.14734,
Culture=neutral, PublicKeyToken=5b9310ab51310fa9,
processorArchitecture=MSIL

%  Number of assemblies uninstalled = 1
```

```
%   Number of failures = 0
%
%   C:\Program Files\Microsoft Visual Studio 9.0\VC>
```

If you do not specify a version to the uninstall option, then all installed OpenSplice DDS C# Assemblies in the GAC, called `dcpssacsAssembly`, will be removed from the GAC, so take care with this option as it can adversely affect any deployed applications that rely on other versions of these assemblies.

We strongly recommend that every time you uninstall an OpenSplice DDS C# Assembly you specify the version you want to uninstall.

1.4 OpenSplice DDS Configuration

Each application domain has its own characteristics. Therefore OpenSplice allows configuring a wide range of parameters that influence its behaviour to be able to achieve optimal performance in every situation. This section describes generally how to instruct OpenSplice to use a configuration that is different from the default. This requires the creation of a custom configuration file and an instruction to the middleware to use this custom configuration file.

1.4.1 Configuration Files

OpenSplice expects the configuration to be defined in the XML format. The expected syntax and semantics of the configuration parameters will be discussed further on in this document. Within the context of OpenSplice, a reference to a configuration is expressed in the form of a Uniform Resource Identifier (URI). Currently, only file URI's are support (for example *`file:///opt/ospl/config/ospl.xml`*).

When OpenSplice is started, the Domain Service parses the configuration file using the provided URI. According to this configuration, it creates a shared administration and initialises it. After that, the Domain Service starts the configured services. The Domain Service passes on its own URI to all services it starts, so they will also be able to resolve their configuration from this resource as well. (Of course, it is also possible to configure a different URI for each of the services, but usually one configuration file for all services will be the most convenient option.) The services will use default values for the parameters that have not been specified in the configuration.

In order to have OpenSplice start with the custom configuration file, use

```
%   ospl start <URI>
```

where `URI` denotes the URI of the Domain Service configuration file. In order to stop a specific OpenSplice instance, the same mechanism holds. Use:

```
%  ospl stop <URI>
```

Several instances of OpenSplice can run simultaneously, as long as their configurations specify different domain names. Typically, only one instance of the middleware is needed. Multiple instances of the middleware are only required when one or more applications on the computing node participate in different or multiple DDS Domains. At any time, the system can be queried for all running OpenSplice instances by using the command:

```
%  ospl list
```

To stop all active OpenSplice Domains, use

```
%  ospl -a stop
```

1.4.2 Environment Variables

The OpenSplice middleware will read several environment variables for different purposes. These variables are mentioned in this document at several places.

To some extent, the user can customize the OpenSplice middleware by adapting the environment variables. For example, instead of using the mechanism described in the previous section, the environment variable `$OSPL_URI` can be used to pass the configuration file URI to the Domain Service. When running *`ospl start`* without the URI parameter, the `ospl` tool will read the `$OSPL_URI` variable and pass this value to the Domain Service.

Also, when specifying configuration parameter values in a configuration file, environment variables can be referenced using the notation *`${VARIABLE}`*. When parsing the XML configuration, the Domain Service will substitute the symbol by the variable value found in the environment.

1.4.3 Temporary Files

Please note that OpenSplice uses temporary files that are used to describe the shared memory that has been created. The exact nature of these files varies according to the operating system; however, the user does not need to manipulate these files directly.

On Linux systems the location of the temp files is `/tmp` by default, while on Windows the location is the value of the `TEMP` (or `TMP` if `TEMP` is not set) environment variable. These location can be over-ridden, if required, by setting the `OSPL_TEMP` variable to a location on disk by specifying a path. Please note, however, that this *must* be consistent for *all* environments on a particular node.

1.5 The DDSI Networking Service

The purpose and scope of the “Data-Distribution Service Interoperability Wire Protocol” is to ensure that applications based on different vendors’ implementations of DDS can interoperate. The protocol was standardized by the OMG in 2008, and was designed to meet the specific requirements of data-distribution systems.

Features of the DDSI protocol include:

- **Performance and Quality-of-Service** properties to enable best-effort and reliable publish-subscribe communications for real-time applications over standard IP networks.
- **Fault tolerance** to allow the creation of networks without single points of failure.
- **Plug-and-play Connectivity** so that new applications and services are automatically discovered and applications can join and leave the network at any time without the need for reconfiguration.
- **Configurability** to allow balancing the requirements for reliability and timeliness for each data delivery.
- **Scalability** to enable systems to potentially scale to very large networks.

Chapter 3, *Service Configuration*, explains how to configure PrismTech’s OpenSplice DDSI implementation. An example configuration is listed in section *3.9.7* on page 174.

1.6 Applications which operate in multiple domains

OpenSplice can be configured to allow a single application to operate in multiple domains.

In order to achieve multi-domain operation, the host node for the application must run OpenSplice instances for every domain in which applications on that node will interact. For example, if an application A wants to operate in domains X, Y and Z then the node on which application A operates must run appropriate services for X, Y and Z.

OpenSplice utilises shared memory regions for intra-node communication. Each domain running on a node must have its own shared memory region, and subsequently the shared memory region for each domain that an application wants to operate within must be mapped into that application's virtual address space. The mapping must occur at a virtual address in memory that is common to both the OpenSplice daemon (and any services) for that domain and the application itself.

This requires some thought when configuring multiple OpenSplice domains on a single node. Care must be taken to ensure that the XML configuration files contain unique and non-overlapping addresses for the shared memory mapping (see the XML element "`OpenSplice/Domain/Database/Address`" in section 3.2.5.2, *Element Address*, on page 30).

When designing and coding applications, care must also be taken with regard to usage of the default domain. If a domain is not explicitly identified in the application code, then appropriate steps must be taken at deployment in order to ensure that applications operate in the domain they were intended to.

1.6.1 Interaction with a Networking Service

Where multiple domains are running on a single node, each domain must run its own instance of a networking service if that domain is to participate in remote communication.

- Each domain should have its own pre-determined port numbers configured in the XML for that domain.
- These port numbers must be common for that domain across the system.

CHAPTER

2 DbmsConnect

2.1 Introduction

The OpenSplice DbmsConnect Module is a pluggable service of OpenSplice that provides a seamless integration of the real-time DDS and the non-/near-real-time enterprise DBMS domains. It complements the advanced distributed information storage features of the OpenSplice Persistence Module (and vice versa).

Where (relational) databases play an essential role to maintain and deliver typically non- or near-real-time 'enterprise' information in mission systems, OpenSplice targets the real-time edge of the spectrum of distributing and providing 'the right information at the right place at the right time' by providing a Quality-Of-Service (QoS) aware 'real-time information backbone'.

Changing expectations about the accessibility of information from remote/non-real-time information-stores and local/real-time sources lead to the challenge of lifting the boundaries between both domains. The DbmsConnect module of OpenSplice answers this challenge in the following ways:

- Transparently 'connects' the real-time DDS 'information backbone' to one or more 'enterprise' databases
- Allows both enterprise as well as embedded/real-time applications to access and share data in the most 'natural' way
- Allows OpenSplice to fault-tolerantly replicate enterprise information persisted in multiple relational databases in real-time
- Provides a pure ODBC/JDBC SQL interface towards real-time information via its transparent DbmsConnection
- Overcomes the lack of communication-control (QoS features controlling real-time behavior) of 'talking' to a remote DBMS
- Overcomes the lack of traditional 3GL and 4GL tools and features in processing information directly from a DDS backbone
- Allows for data-logging and analysis of real-time data persisted in a DBMS
- Aligns multiple and dispersed heterogeneous databases within a distributed system using the QoS-enabled data-distribution features of OpenSplice

The DbmsConnect module is unique in its dynamic configurability to achieve maximum performance:

- Dynamic DDS Partition/Topic selection and configurable content-filters to exchange exactly ‘the right’ information critical for performance and resource-challenged users
- Dynamic creation and mapping of DBMS database-tables and DDS topics to allow seamless data-exchange, even with legacy data models
- Selectable update-triggering per table/topic allowing for both real-time responsiveness as well as high-volume 'batch transfers'
- Works with ANY 3rd party SQL:1999 compatible DBMS system with an ODBC interface

The DbmsConnect module thus effectively eliminates traditional ‘barriers’ of the standalone technologies by facilitating seamless data-exchange between any ODBC compliant (SQL)database and the OpenSplice™ real-time DDS "information-backbone". Applications in traditionally separated mission-system domains can now exploit and leverage each other's information in a highly efficient (based upon ‘current interest’ as supported by the publish/subscribe paradigm of DDS), non-disruptive (obeying the QoS demands as expressed for data-items in DDS) and distributed service-oriented paradigm.

2.2 Usage

In order to understand the configuration and working of the DbmsConnect service, some basic concepts and use-cases will be covered in this chapter.

2.2.1 DDS and DBMS Concepts and Types Mapping

The concepts within DDS and DBMS are related to each other as listed in *Table 1*.

Table 1 DDS <> DBMS mapping: concepts

DDS	DBMS
Topic	Table
Type	Table structure
Instance	Primary key
Sample	Row
DataWriter.write()	`INSERT or UPDATE`
DataWriter.dispose()	`DELETE`

The primitive types available in both domains map onto each other as listed in *Table 2*

Table 2 DDS <> DBMS mapping: primitive types

DDS IDL type	DBMS column type (SQL:1999)
`boolean`	`BOOLEAN/TINYINT`
`short`	`SMALLINT`
`unsigned short`	`SMALLINT`
`long`	`INTEGER`
`unsigned long`	`INTEGER`
`long long`	`BIGINT`
`unsigned long long`	`BIGINT`
`float`	`REAL`
`double`	`DOUBLE`
`octet`	`BINARY(1)`
`char`	`CHAR(1)`
`wchar`	`CHAR(1)`
`string<length>`	`VARCHAR(<length>)`
`wstring<length>`	`VARCHAR(<length>)`

The mapping of complex (composite) types is as follows:

- Struct
  - Flattened out
  - Each member maps to a column with fully scoped name
- Union
  - Flattened out
  - Additional '`#DISCRIMINATOR#`' column
- Enumeration
  - An '`INTEGER`' typed column with fully scoped name
- Array and bounded sequence
  - Flattened out
  - '`[index]`' appended to fully scoped column name

2.2.2 General DbmsConnect Concepts

The DbmsConnect service can bridge data from the DDS domain to the DBMS domain and vice versa. In DDS, data is represented by topics, while in DBMS data is represented by tables. With DbmsConnect, a mapping between a topic and a table can be defined.

Because not all topic-table mappings have to be defined explicitly (DbmsConnect can do matching when the names are the same), namespaces can be defined. A namespace specifies or limits the context of interest and allows for easy configuration of all mappings falling (or defined in) a namespace. The context of interest for bridging data from DDS to DBMS, consists of a partition- and topicname expression. When bridging data from DBMS to DDS, the context of interest consists of a table-name expression.

A mapping thus defines the relationship of a table in DBMS with a topic in DDS and can be used to explicitly map a topic and table with different names, or define settings for a specific mapping only.

2.2.3 DDS to DBMS Use Case

When data in the DDS domain has to be available in the DBMS domain, the DbmsConnect service can be configured to facilitate that functionality. A topic in DDS will be mapped to a table in DBMS.

Scenario

In the DDS domain, we have topics `DbmsTopic` and `DbmsDdsTopic` that we want to make available to a database application. The database application expects the data from topic `DbmsTopic` to be available in an existing table with name `DbmsTable`. Data from the `DbmsDdsTopic` topic can be just forwarded to a table (that not yet exists) with the same name. The scenario is shown in *Figure 2*.

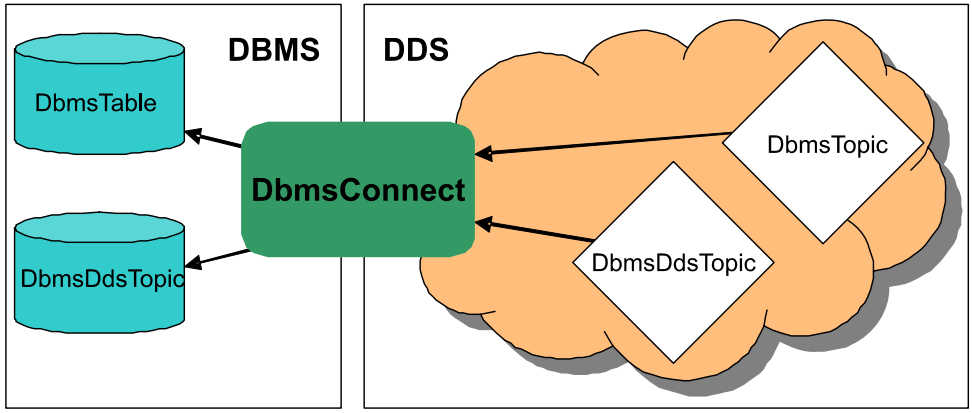

Figure 2 DDS to DBMS scenario

Configuration

The configuration for the DbmsConnect service that fulfils the needs of the scenario is given in the listing below.

```
1   ...
2   <DbmsConnectService name="dbmsconnect">
3      <DdsToDbms>
4         <NameSpace partition="*" topic="Dbms*"
5            dsn="DSN" usr="USR" pwd="PWD" odbc="ODBC">
6            <Mapping topic="DbmsTopic" table="DbmsTable"/>
7         </NameSpace>
8      </DdsToDbms>
9   </DbmsConnectService
10  ...
```

Explanation

On line *3* a `DdsToDbms` element is specified in order to configure data bridging from DDS to DBMS. On line *4*, a `NameSpace` is defined that has interest in all topics starting with `"Dbms"` in all partitions. Both the partition- and topic-expression make use of the `*`-wildcard (matching any sequence of characters). These wildcards match both topics described in the scenario, but will possibly match more. If the mapping should be explicitly limited to both topics, the topic-expression can be changed to `"DbmsTopic,DbmsDdsTopic"`.

The DbmsConnect service will implicitly map all matching topics to an equally named table in the DBMS. While this is exactly what we want for the `DbmsDdsTopic`, the database application expects the data from the `DbmsTopic` topic to be mapped to table `DbmsTable`. This is explicitly configured in the `Mapping` on line *6*. If the tables already exist and the table-definition matches the

topic definition, the service will use that table. If a table does not exist, it will be created by the service. If a table exists, but doesn't match the topic definition, the mapping fails.

2.2.4 DBMS to DDS Use Case

When data in the DBMS domain has to become available in the DDS domain, this can be achieved by configuring the DbmsConnect service to map a table to a topic.

Scenario

In the DBMS, we have tables `DbmsTable` and `DbmsDdsTopic` that we want to make available in the `dbmsPartition` partition in DDS. The database application writes the data we want available in topic `DbmsTopic` to the table named `DbmsTable`. Data from the `DbmsDdsTopic` table can be just forwarded to the equally named topic.

When the DbmsConnect service is started, mapped tables may already contain data. For the `DbmsDdsTopic` table, we are not interested in that data. For the `DbmsTable` table however, we would like all data available to the database application to be available to the DDS applications too. This scenario is the reverse (all arrows reversed) situation of the scenario shown in *Figure 2* on page 21.

Configuration

The configuration for the DbmsConnect service that fulfils the needs of the scenario is given in the listing below.

```
11 ...
12 <DbmsConnectService name="dbmsconnect">
13     <DbmsToDds publish_initial_data="false">
14         <NameSpace partition="dbmsPartition" table="Dbms*"
15             dsn="DSN" usr="USR" pwd="PWD" odbc="ODBC">
16             <Mapping topic="DbmsTopic" table="DbmsTable"
17                 publish_initial_data="true"/>
18         </NameSpace>
19     </DbmsToDds>
20 </DbmsConnectService
21 ...
```

Explanation

On line *13* a `DdsToDbms` element is specified in order to configure data bridging from DBMS to DDS. On line *14*, a `NameSpace` is defined that has interest in all tables starting with `"Dbms"`. The table-expression makes use of the `*`-wildcard (matching any sequence of characters). For this scenario, a single target partition is specified. If needed, a partition expression containing multiple partitions or

wildcards can be used. For example when the DDS system is divided in two partitions (to support applications running in a ‘simulation’- and a ‘real’ world) and applications in both partition need access to the data from the DBMS.

The default setting for the `publish_initial_data` attribute is `true`. Because we only want initially available data to be published for the `DbmsTable-DbmsTopic` mapping, we define the default for all namespaces to be `false` on line *13*. That setting will be inherited by all underlying elements, but can be overridden. The explicit `Mapping` specified on line *16* maps the table to the differently named topic. On line *17*, the `publish_initial_data` attribute is explicitly set to `true`, overriding the setting set at line *13*.

2.2.5 Replication Use Case

A specific application of data bridging from DDS to DBMS and DBMS to DDS is replication of database (tables). Replication requires some specific configuration. The basic configuration is covered in this use case.

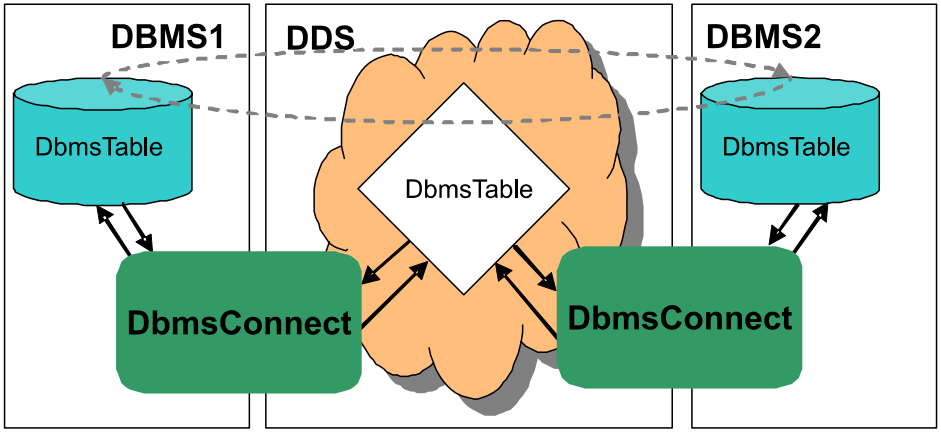

Figure 3 Replication scenario

Scenario

We have a two database servers running on different hosts. The table `DbmsTable` should be available on both database-servers and changes should be sent both ways. This scenario is shown in *Figure 3*, where the dashed arrows show the transparant role of DDS in this scenario.

Configuration

The configuration for the DbmsConnect service for both hosts, that fulfils the needs of the scenario, is given in the listing below.

```
22 ...
```

```
23 <DbmsConnectService name="dbmsconnect">
24     <DdsToDbms replication_mode="true">
25         <NameSpace partition="replication" topic="DbmsTable"
26             dsn="DSN" usr="REPLUSR" pwd="PWD" odbc="ODBC">
27         </NameSpace>
28     </DdsToDbms>
29     <DbmsToDds replication_user="REPLUSR">
30         <NameSpace partition="replication" table="DbmsTable"
31             dsn="DSN" usr="USR" pwd="PWD" odbc="ODBC">
32         </NameSpace>
33     </DbmsToDds>
34 </DbmsConnectService
35 ...
```

Explanation

The configuration for the replication scenario is symmetric in that it can be the same for both hosts. The basic idea is simple: configure a mapping from DDS to DBMS and from DBMS to DDS for the same table-topic pair within a partition (analogue to the *DDS to DBMS Use Case* on page 20 and the *DBMS to DDS Use Case* on page 22). While this (intuitive) cyclic definition would work, more configuration is needed to support this use case properly. In order to prevent modifications from keeping to cause (cyclic) updates, the DbmsConnect service needs to be able to distinguish between data that is modified as part of a replication scenario and data that is changed by other sources.

For the DDS to DBMS mapping, replication data is identified by identification information of the sending node. The `DdsToDbms` part of the service is put to replication mode in line *24*, which lets the service ignore all data transmitted by the node on which the service itself runs.

For the DBMS to DDS mapping, a special database user has to be used, that is only used by the DbmsConnect service, in order to be able to distinguish data from different sources. The database user that is used in the `DdsToDbms` mapping has to be excluded from update-cascading. This is specified on line *29* in the `replication_user` attribute. This means that all data that is inserted in the table by the user with the username specified in the `replication_user` attribute will be ignored. So the user specified at line *26* has to be the same as the user specified on line *29*.

CHAPTER

3 Service Configuration

3.1 Introduction

This chapter provides a more in depth description of the OpenSplice DDS configuration by describing the most important configuration parameters for all available services. Each configuration parameter will be explained by means of an extensive description together with the tabular summary that contains the following information:

- *Full path* - Describes the location of the item within a complete configuration. Because the configuration is in the XML format, an XPath expression is used to point out the name and location of the configuration item.
- *Format* - Describes the format of the value of the configuration item.
- *Dimension* - Describes the unit for the configuration item (e.g. seconds or bytes).
- *Default value* - Describes the default value that is used by service when the configuration item is not in the configuration.
- *Valid values* - Describes the valid values for the configuration item. This can be a range or a set of values.

In case the configuration parameter is an XML attribute, the table also contains the following information:

- *Required* - Describes whether the attribute must be specified explicitly or is optional.

In case the configuration parameter is an XML element, the table also contains the following information:

- *Occurrences* - Describes the range of the possible number of occurrences of the element in the configuration by specifying the minimum and maximum number of occurrences.
- *Child-elements* - Describes the child-elements supported by the current element.
- *Required attributes* - Describes the required attributes, i.e. the attributes that must be specified inside the current element.
- Optional attributes - Describes the optional attributes, i.e. the attributes that may, but do not need to be specified inside the current element.

3.2 The Domain Service

The Domain Service is responsible for creating and initialising a shared nodal administration (in shared memory) for a specific DDS Domain on a computing node. Without this administration, no other service or application is able to participate in a DDS Domain. Once the administration has been initialised, the Domain service starts the set of pluggable services. The lifecycle of the started services is under control of the Domain service, which means it will monitor the health of all started services, take corrective actions if needed and stop the services when it is terminated.

When a shutdown of the OpenSplice Domain service is requested, it will react by announcing the shutdown using the shared administration. Applications will not be able to use DDS functionality anymore and services are requested to terminate elegantly. Once this has succeeded, the Domain service will destroy the shared administration and finally terminate itself.

Please refer to section *1.6* on page 14 for notes about applications operating in multiple domains.

`Full path`	OpenSplice/Domain
`Occurrences (min-max)`	1 - 1
`Child-elements`	*Element name* *Element Lease* *Element ServiceTerminatePeriod* *Element Database* *Element Service* *Element Listeners* *Element BuiltinTopics* *Element Statistics* *Element ReportPlugin*
`Required attributes`	<none>
`Optional attributes`	<none>

3.2.1 Element *name*

This element specifies the name of the instantiated DDS domain. In general, it is recommended to change this name to a name that uniquely identifies the domain. If several different DDS domains are required to run simultaneously, then they all need to have their own domain name.

Full path	OpenSplice/Domain/Name
Format	string
Dimension	n/a
Default value	“The default Domain”
Valid values	any string
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.2 Element *Role*

This (optional) element specifies the role of the instantiated domain. For dynamic discovery, this role will be used to define the communication scope of instantiated domains on other nodes in the system. The purpose of specifying roles within the system is to ‘overlay’ the underlying physical network with a node’s scope-of-interest that allows to “bound” topology discovery effort and related overhead in large scale (WAN) systems.

Full path	OpenSplice/Domain/Role
Format	string
Dimension	n/a
Default value	DefaultRole
Valid values	any string
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.3 Element *Lease*

The Lease parameters specify how the Domain Service as well as the services started by the Domain Service must announce their liveliness in the shared administration.

Full path	OpenSplice/Domain/Lease
Occurrences (min-max)	0 - 1
Child-elements	*Element ExpiryTime*
Required attributes	<none>
Optional attributes	<none>

3.2.3.1 Element *ExpiryTime*

This element specifies the interval in which services have to announce their liveliness.

Every OpenSplice DDS service, including the Domain Service itself, has to announce its liveliness regularly. This allows corrective actions to be taken when one of the services becomes non-responsive. This element specifies the required interval. Decreasing the interval decreases the time in which non-responsiveness of a service is detected, but leads to more processing. Increasing it has the opposite effect.

Full path	OpenSplice/Domain/Lease/ExpiryTime
Format	float
Dimension	seconds
Default value	20.0
Valid values	0.2 - maxFloat
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	*Attribute update_factor*
Optional attributes	<none>

3.2.3.1.1 Attribute *update_factor*

In case of a temporary high CPU load, the scheduling behaviour of the operating system might affect the capability of a service to assert its liveliness ‘on time’. The *update_factor* attribute introduces some elasticity in this mechanism by making the services assert their liveliness more often than required by the *ExpiryTime*. Services will report their liveliness every *ExpiryTime* multiplied by this *update_factor.*

Full path	OpenSplice/Domain/Lease/ExpiryTime[@update_factor]
Format	float
Dimension	n/a
Default value	0.1
Valid values	0.01 - 1.0
Required	yes

3.2.4 Element *ServiceTerminatePeriod*

This element specifies the amount of time the Domain Service, when instructed to terminate, should wait for the other configured Services to terminate.

Full path	OpenSplice/Domain/ServiceTerminatePeriod
Format	float
Dimension	seconds
Default value	10.0
Valid values	1.0 - 60.0
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.5 Element *Database*

The Database element contains information about the nodal administration (shared memory) to be used.

Full path	OpenSplice/Domain/Database
Occurrences (min-max)	1 - 1
Child-elements	*Element Size* *Element Address* *Element Locking*
Required attributes	<none>
Optional attributes	<none>

3.2.5.1 Element *Size*

This element specifies the size of the shared memory segment holding the database. Change this value if your system requires more memory than the default. Please note that the operating system should be configured support the requested size. On most platforms you need ‘root’ privileges to set large sizes.

Full path	OpenSplice/Domain/Database/Size
Format	unsigned integer
Dimension	bytes
Default value	10485670
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.5.2 Element *Address*

This element specifies the start address where the nodal shared administration is mapped into the virtual memory of each process that attaches to the current Domain. The possible values are platform dependent.

Full path	OpenSplice/Domain/Database/Address
Format	(hexadecimal) memory address
Dimension	shared memory address
Default value	0x20000000 (Linux2.6 on x86) 0x40000000 (Windows on x86) 0xA0000000 (Solaris on SPARC) 0xA0000000 (AIX5.3 on POWER5+) 0x0 (VxWorks 5.5.1 on PowerPC604) 0x60000000 (VxWorks 6.x on PowerPC604) 0x20000000 (Integrity on mvme5100)
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

Change this value if the default address is already in use, for example by another Domain Service or another product.

3.2.5.3 Element *Locking*

This element specifies the locking policy of the Database, indicating whether to lock pages in physical memory or not.

With the virtual memory architecture, the operating system decides when to swap memory pages from internal memory to disc. This results in execution delays for the corresponding code because it has to be paged back into main memory. The element *Locking* can be used to avoid such swapping for the shared memory where the database resides. The user needs the appropriate privileges from the underlying operating system to be able to use this option.

The possible values are:

- **True**: lock the pages in memory.
- **False**: don't lock the pages in memory.
- **Default**: use the platform-dependent default value.

`Full path`	OpenSplice/Domain/Database/Locking
`Format`	enumeration
`Dimension`	n/a
`Default value`	Default
`Valid values`	True, False, Default
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.2.6 Element *Service*

The Domain Service is responsible for starting, monitoring and stopping the pluggable services. One Service element must be specified for every service that needs to be started by the Domain Service.

Full path	OpenSplice/Domain/Service
Occurrences (min-max)	1 - *
Child-elements	*Element Command* *Element Configuration* *Element Scheduling* *Element Locking* *Element FailureAction*
Required attributes	*Attribute name*
Optional attributes	*Attribute enabled*

3.2.6.1 Attribute *name*

This attribute specifies the name by which the the corresponding service is identified in the rest of the configuration file.

In the OpenSplice DDS configuration file, services and their settings are identified by a name. When the Domain Service starts a particular service, its corresponding name is passed. The service in question uses this name in order to find its own configuration settings in the rest of the configuration file. The name specified here must match the *name* attribute of the main element of the corresponding service.

Full path	OpenSplice/Domain/Service[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	yes

3.2.6.2 Attribute *enabled*

This attribute indicates whether the service is actually started or not.

Full path	OpenSplice/Domain/Service[@enabled]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	no

Toggling a service between enabled and disabled is a quick alternative for commenting out the corresponding lines in the configuration file.

3.2.6.3 Element *Command*

This element specifies the command to be executed in order to start the service.

OpenSplice DDS comes with a set of pluggable services. The Command element specifies the name of the actual service executable (possibly including its path, but always including its extension, e.g. '.exe' on the Windows platform). When no path is included, the Domain Service will search the *PATH* environment variable for the corresponding executable. Once located, it will be started as a separate process.

Full path	OpenSplice/Domain /Service/Command
Format	string
Dimension	executable file
Default value	none
Valid values	The name of a service executable.
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.4 Element *MemoryPoolSize*

CAUTION: This element should only be used on the GHS Integrity platform.

This element maps directly into the integrate file for the address space for this service. Consult the GHS Integrate documentation for further information on this setting. Valid values are decimal or hexadecimal numbers and they express the number of bytes. The default setting for this element is dependent on the Service for which it is configured.

Full path	OpenSplice/Domain/Service/MemoryPoolSize
Format	string
Dimension	decimal or hexadecimal number of bytes.
Default value	0xa00000 for spliced 0x280000 for durability 0x280000 for networking 0x100000 for cmsoap
Valid values	dependent on underlying platform

Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.5 Element *HeapSize*

CAUTION: This element should only be used on the GHS Integrity platform.

This element maps directly into the integrate file for the address space for this service. Consult the GHS Integrate documentation for further information on this setting. Valid values are decimal or hexadecimal numbers and they express the number of bytes. The default setting for this element is dependent on the Service for which it is configured.

Full path	OpenSplice/Domain/Service/MemoryPoolSize
Format	string
Dimension	decimal or hexadecimal number of bytes.
Default value	0x800000 for spliced 0x240000 for durability 0x240000 for networking 0x200000 for cmsoap
Valid values	dependent on underlying platform
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.6 Element *StackSize*

CAUTION: This element should only be used on the GHS Integrity platform.

This element maps directly into the integrate file for the address space for this service. Consult the GHS Integrate documentation for further information on this setting. Valid values are decimal or hexadecimal numbers and they express the number of bytes. The default setting for this element is dependent on the Service for which it is configured.

Full path	OpenSplice/Domain/Service/StackSize
Format	string
Dimension	decimal or hexadecimal number of bytes.

Default value	0x10000 for spliced 0x10000 for durability 0x10000 for networking 0x10000 for cmsoap
Valid values	dependent on underlying platform
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.7 Element *Configuration*

This element allows overriding of the default URI (specified in the *OSPL_URI* environment variable, or passed explicitly as command-line parameter to the *ospl* executable) with the configuration resource specified here.

When the Domain Service is started by the *ospl* executbale, by default it passes on its own URI to the services that it starts. This is valid when the configuration of the service is located in the same resource file as the configuration of the Domain Service itself. (This is a convenient situation in most cases).

If the configuration of the current service is located in a separate resource file, a separate URI identifying that particular resource file must be specified in this element.

Full path	OpenSplice/Domain /Service/Configuration
Format	string
Dimension	URI
Default value	${OSPL_URI}
Valid values	Any URI to a valid resource file.
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.8 Element *Scheduling*

This element specifies the scheduling parameters used to control the current Service.

Full path	OpenSplice/Domain/Service/Scheduling
Occurrences (min-max)	0 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.2.6.8.1 Element *Class*

This element specifies the thread scheduling class that the Domain Service will assign to the current Service when it is started. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/Domain /Service/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.8.2 Element *Priority*

This element specifies the thread priority that the Domain Service will assign to the current Service when it is started. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/Domain /Service/Scheduling/Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system

Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.2.6.8.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/Domain/Service/Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.2.6.9 Element *Locking*

This element specifies the locking policy of the current Service process, indicating whether pages should be locked in physical memory or not.

On platforms with a virtual memory architecture, the operating system decides when to swap memory pages from internal memory to disk. This results in execution delays for the corresponding code because it has to be paged back into main memory. The element *Locking* can be used to avoid such swapping for the current Service. The user needs the appropriate privileges from the underlying operating system to be able to use this option.

Full path	OpenSplice/Domain/Service/Locking
Format	boolean
Dimension	n/a
Default value	depends on operating system
Valid values	true, false
Occurrences (min-max)	0 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.6.10 Element *FailureAction*

This element specifies what action to take at the moment that the service seems to have become non-responsive.

Each service reports its liveliness regularly using the shared administration. If the service fails to do so, the Domain Service will assume the service has become non-responsive. This element determines what action is taken by the Domain Service in case this happens.

The following actions are available:

- **skip**: Ignore the non-responsiveness and continue.
- **kill**: End the service process by force.
- **restart**: End the service process by force and restart it.
- **systemhalt**: End all OpenSplice services including the Domain Service (for the current DDS Domain on this computing node).

Full path	OpenSplice/Domain /Service/FailureAction
Format	enumeration
Dimension	n/a
Default value	skip
Valid values	skip, kill, restart, systemhalt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.7 Element *Listeners*

This element specifies policies for the thread that services the listeners that the application specifies on the API-level.

Full path	OpenSplice/Domain/Listeners
Occurrences (min-max)	0 - 1

Child-elements	*Element StackSize*
Required attributes	<none>
Optional attributes	<none>

3.2.7.1 Element *StackSize*

This element specifies stack size of the listener thread.

Full path	OpenSplice/Domain/Service/Listeners/StackSize
Format	unsigned integer
Dimension	bytes
Default value	64000
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.2.8 Element *BuiltinTopics*

This element specifies the granularity of the builtin topics.

Full path	OpenSplice/Domain/BuiltinTopics
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	*Attribute enabled*
Optional attributes	<none>

3.2.8.1 Attribute enabled

This attribute enables or disables the publication of builtin topics for the existence of individual Participants/DataWriters/DataReaders. The existence of Topics will always be communicated by means of builtin topics, regardless of the value specified here.

Full path	OpenSplice/Domain/BuiltinTopics[@enabled]
Format	boolean
Dimension	n/a

Default value	true
Valid values	true, false
Required	true

3.2.9 Element *Statistics*

This element specifies the policies regarding statistics. Various statistics can be generated by OpenSplice DDS to help you analyze and tune application behaviour during application development. Since this introduces extra overhead, it is generally turned off in a runtime system.

Full path	OpenSplice/Domain/Statistics
Occurrences (min-max)	0 - 1
Child-elements	*Element Category*
Required attributes	<none>
Optional attributes	<none>

3.2.9.1 Element *Category*

This element specifies the properties for a particular category of statistics.

Full path	OpenSplice/Domain/Statistics/Category
Occurrences (min-max)	0 - *
Child-elements	<none>
Required attributes	*Attribute name*
Optional attributes	*Attribute enable*

3.2.9.1.1 Attribute *name*

This attribute specifies the name of a particular category of statistics.

Full path	OpenSplice/Domain/Statistics/Category[@name]
Format	string
Dimension	name of a statistics category
Default value	reader
Valid values	durability, reader, writer
Required	true

3.2.9.1.2 Attribute *enable*

This attribute enables or disables the generation of statistics for the specified category.

Full path	OpenSplice/Domain/Statistics/Category[@enabled]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	false

3.2.10 Element *ReportPlugin*

This element allows the user to plugin a custom report facility to be used by the domain. The Domain Service is responsible for registering and unregistering any report plugins. All services and applications within the domain will use any registered report plugins.

Full path	OpenSplice/Domain/ReportPlugin
Occurrences (min-max)	0 – 10
Child-elements	*Element Library* *Element Initialize* *Element Report* *Element Finalize*
Required attributes	<none>
Optional Attributes	<none>

3.2.10.1 Element *Library*

This element specifies the library to be loaded.

Full path	OpenSplice/Domain/ReportPlugin/Library
Occurences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute file_name*

3.2.10.1.1 Attribute file_name

This attribute specifes the name of the library to be loaded. The attribute is optional, if no file name is specified then no library will be loaded. If a name is specified but the library cannot be loaded a warning message is generated; the service will, however, still attempt to look up the other elements.

`Full path`	OpenSplice/Domain/ReportPlugin/ Library[@file_name]
`Format`	String
`Dimension`	N/A
`Default value`	None
`Valid values`	Any valid string
`Required`	No

3.2.10.2 Element *Initialize*

The initialize element holds the name of the function to be called when initializing the plugin.

`Full path`	OpenSplice/Domain/ReportPlugin/Initialize
`Occurences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	*Attribute symbol_name* *Attribute argument*

3.2.10.3 Attribute symbol_name

This attribute specifies the name of the function to be called to initialize the report plugin. The symbol_name is optional, if it is not specified then the service will assume that the report plugin does not require initialization. If a symbol_name is provided but cannot be resolved a warning message will be generated, however, the service will continue to attempt to resolve other symbol_names for the report plugin.

The implementation of this function must have the following signature:

```
int symbol_name (const char *argument, void **context)
```

The result value is used to return the status of the call. If `0` then the operation was successful. If not `0` then there was an error and details of the error and the result value are reported to the OpenSplice DDS default report service.

The context parameter is an out reference that can be set to plugin-specific data that will subsequently be passed to any of the other plugin functions. The value of the parameter is meaningless to the service.

Full path	OpenSplice/Domain/ReportPlugin/ Initialize[@symbol_name]
Format	String
Dimension	N/A
Default value	None
Valid values	Any valid string
Required	No

3.2.10.4 Attribute argument

The argument attribute is a string value that is passed to the function specified by the symbol_name. The string value has no meaning to the service and is used to pass any context-specific information that may be required. The argument is optional; if it is not provided then a NULL pointer is passed to the initalize function.

Full path	OpenSplice/Domain/ReportPlugin/ Initialize[@argument]
Format	String
Dimension	N/A
Default value	None
Valid values	Any valid string
Required	No

3.2.10.5 Element *Report*

The report element holds the name of the function to be called when reporting to the plugin.

Full path	OpenSplice/Domain/ReportPlugin/Report
Occurences (min-max)	0-1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute symbol_name*

3.2.10.6 Attribute symbol_name

This attribute specifies the name of the function to be called to pass report data to the report plugin. The symbol_name is optional; if it is not specified then the service will assume that the report plugin does not require reporting. If a symbol_name is provided but cannot be resolved, a warning message will be generated; however, the service will continue to attempt to resolve other symbol_names for the report plugin.

The implementation of this function must have the following signature:

```
int symbol_name (void *context, const char *report)
```

The result value is used to return the status of the call. If `0` then the operation was successful. If not `0` then there was an error and details of the error and the result value are reported to the OpenSplice DDS default report service.

The context parameter is a reference to the plugin-specific data retrieved from the initialize operation.

The report parameter is an XML string representation of the report data.

Below is an example of the mapping that the XML string representation will use:

```
<WARNING>
   <DESCRIPTION>The object “my_topic” not
found</DESCRIPTION>
   <CONTEXT>c_base::resolve</CONTEXT>
   <FILE>c_base.c</FILE>
   <LINE>1234</LINE>
   <CODE>0</CODE>
</WARNING>
```

Full path	OpenSplice/Domain/ReportPlugin/ Report[@symbol_name]
Format	String
Dimension	N/A
Default value	None
Valid values	Any valid string
Required	No

3.2.10.7 Element *Finalize*

The Finalize element holds the name of the function to be called when finalizing the plugin.

Full path	OpenSplice/Domain/ReportPlugin/Finalize
Occurences (min-max)	0-1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute symbol_name*

3.2.10.8 Attribute symbol_name

This attribute specifies the name of the function to be called to finalize the report plugin, when the domain unregisters any registered plugins. The symbol_name is optional; if it is not specified then the service will assume that the report plugin does not require finalization. If a symbol_name is provided but cannot be resolved, a warning message will be generated.

The implementation of this function must have the following signature:

```
int symbol_name (void *context)
```

The result value is used to return the status of the call. If `0` then the operation was successful. If not `0` then there was an error and details of the error and the result value are reported to the OpenSplice DDS default report service.

The context parameter is a reference to the plugin-specific data retrieved from the `initialize` operation.

Full path	OpenSplice/Domain/ReportPlugin/ Report[@symbol_name]
Format	String
Dimension	N/A
Default value	None
Valid values	Any valid string
Required	No

3.2.11 Element *PartitionAccess*

This element is used to configure the access rights for a specific partition on a node. By default all partitions have read and write access, which means subscribers and publishers may be created for all partitions. However by changing the access level of specific partitions it is possible to prevent publishers and/or subscribers from attaching to these partitions.

The `PartitionAccess` element facilitates the configuration of such behavior. This is done by allowing the definition of a partition expression along with a specific access mode for the matched partitions. The `PartitionAccess` element resides as a child element within the `Domain` element. The exact defintion of the `PartitionAccess` element is as follows:

`Full path`	OpenSplice/Domain/PartitionAccess
`Occurrences (min-max)`	*
`Child-elements`	<none>
`Required attributes`	*`Attribute partition_expression`* *`Attribute access_mode`*
Optional attributes	`<none>`

An example demonstrates the usage of this element:

```
<OpenSplice>
   <Domain>
    <PartitionAccess partition_expression=
              "/remote/*" access_mode="readwrite"/>
   </Domain>
</OpenSplice>
```

3.2.11.1 Attribute partition_expression

This attribute identifies an expression that specifies the partitions for which access is being defined. The expression may use wildcards (*i.e.* the '*' and '`?`' tokens) to indicates multiple partitions that match the expression.

`Full path`	OpenSplice/Domain/PartitionAccess/ partition_expression
`Format`	string
`Dimension`	n.a.
`Default value`	n.a.
`Valid values`	Any string of format [A..Z,a..z,0..9,_,/,*,?]*
`Required`	true

3.2.11.2 Attribute access_mode

This attribute identifies the access level for partitions specified by the `partition_expression` attribute. The following values are allowed:

`read`	Indicates applications can only read from this partition
`write`	Indicates applications can only write to this partition
`readwrite`	Indicates applications can read from and write to this partition

none Indicates that applications on this node have no access on partitions matching the partition_expression.

When multiple expressions overlap each other, the following rules are applied:

Access mode 1	Access mode 2	Resulting access mode
read	write	readwrite
read	readwrite	readwrite
read	none	none
write	readwrite	readwrite
write	none	none
readwrite	none	none

Full path	OpenSplice/Domain/PartitionAccess/access_mode
Format	string
Dimension	n.a.
Default value	n.a.
Valid values	read, write, readwrite, none
Required	true

3.2.12 Element *TopicAccess*

This element is used to configure the access rights for a specific topic on a node. By default all topics have read and write access (builtin topics have a default access mode of read), which means datareaders and datawriters may be created for all topics. However by changing the access level of specific topics it is possible to prevent datawriters and/or datareaders from being created for these topics.

The `TopicAccess` element facilitates the configuration of such behavior. This is done by allowing the definition of a topic expression along with a specific access mode for the matched topics.

The `TopicAccess` element resides as a child element within the `Domain` element. The exact definition of the `TopicAccess` element is as follows:

Full path	OpenSplice/Domain/TopicAccess
Occurrences (min-max)	*

Child-elements	<none>
Required attributes	*Attribute topic_expression* *Attribute access_mode*
Optional attributes	<none>

An example demonstrates the usage of this element:

```
<OpenSplice>
   <Domain>
      <TopicAccess topic_expression=
             "/remote/*" access_mode="read"/>
   </Domain>
</OpenSplice>
```

3.2.12.1 Attribute topic_expression

This attribute identifies an expression that specifies the topics for which access is being defined. The topic may use wildcards (*i.e.* the '`*`' and '`?`' tokens) to indicate multiple topics that match the expression.

Full path	OpenSplice/Domain/TopicAccess/topic_expression
Format	string
Dimension	n.a.
Default value	n.a.
Valid values	Any string
Required	true

3.2.12.2 Attribute access_mode

This attribute identifies the access level for topics defined by the `topic_expression` attribute. The following values are allowed:

`read` Indicates that applications on this node have only read access on Topics matching the `topic_expression`.

`write` Indicates that applications on this node have only write access on Topics matching the `topic_expression`.

`readwrite` Indicates that applications on this node have read and write access on Topics matching the `topic_expression`.

`none` Indicates that applications on this node have no access to Topics matching the `topic_expression`.

When multiple expressions overlap each other, the following rules are applied:

Access mode 1	Access mode 2	Resulting access mode
read	write	readwrite
read	readwrite	readwrite
read	none	none
write	readwrite	readwrite
write	none	none
readwrite	none	none

Full path	OpenSplice/Domain/TopicAccess/access_mode
Format	string
Dimension	n.a.
Default value	n.a.
Valid values	read, write, readwrite, none
Required	true

3.3 The Daemon Service

Every domain is controlled by exactly one daemon: the Splice Daemon. The Splice Daemon configuration expects a root element named *OpenSplice/Daemon*. Within this root element, the Splice Daemon will look for several child-elements. Each of these child elements is listed and explained in the following sections.

Full path	OpenSplice/Daemon
Occurrences (min-max)	0 - 1
Child-elements	*Element Locking* *Element KernelManager* *Element GarbageCollector*
Required attributes	<none>
Optional attributes	<none>

3.3.1 Element *Locking*

This element specifies the locking policy for the Splice Deamon process, indicating whether its pages should be locked in physical memory or not.

On platforms with a virtual memory architecture, the operating system decides when to swap memory pages from internal memory to disk. This results in execution delays for the corresponding code because it has to be paged back into main memory. The element *Locking* can be used to avoid such swapping for the Splice Daemon. The user needs the appropriate privileges from the underlying operating system to be able to use this option.

Full path	OpenSplice/Daemon/Locking
Format	boolean
Dimension	n/a
Default value	false
Valid values	true, false
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.3.2 Element *KernelManager*

This element specifies the behaviour of the KernelManager.

The Kernel Manager actively monitors the OpenSplice DDS kernel and executes the following administrative tasks:

- check topic consistency,
- determine liveliness status of readers and writers,
- notify readers and writers on incompatible QoS.

Full path	OpenSplice/Daemon/KernelManager
Occurrences (min-max)	0 - 1
Child-elements	*Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.3.2.1 Element *Scheduling*

This element specifies the scheduling policies used to control the KernelManager thread.

Full path	OpenSplice/Daemon/KernelManager/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.3.2.1.1 Element Class

This element specifies the thread scheduling class that will be used by the KernelManager thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/Daemon/KernelManager/ Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.3.2.1.2 Element Priority

This element specifies the thread priority that will be used by the KernelManager thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/Daemon/KernelManager/ Scheduling/Priority
Format	integer
Dimension	n/a

Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.3.2.1.2.1 Attribute priority_kind

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/Daemon/KernelManager/ Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.3.3 Element *GarbageCollector*

This element specifies the behaviour of the GarbageCollector.

The Garbage Collector is a safety mechanism and is responsible for reclaiming resources in case an application or remote node does not terminate properly.

Full path	OpenSplice/Daemon/GarbageCollector
Occurrences (min-max)	0 - 1
Child-elements	*Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.3.3.1 Element *Scheduling*

This element specifies the scheduling policies used to control the GarbageCollector thread.

Full path	OpenSplice/Daemon/GarbageCollector/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.3.3.1.1 Element Class

This element specifies the thread scheduling class that will be used by the GarbageCollector thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/Daemon/GarbageCollector/ Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.3.3.1.2 Element Priority

This element specifies the thread priority that will be used by the GarbageCollector thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/Daemon/GarbageCollector/ Scheduling/Priority
Format	integer
Dimension	n/a

Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.3.3.1.2.1 Attribute priority_kind

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/Daemon/GarbageCollector/ Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.4 The Durability Service

The responsibilities of the durability service are to realize the durable properties of data in an OpenSplice system. The Durability Service looks for its configuration within the 'OpenSplice/DurabilityService' element. The configuration parameters that the Durability Service will look for within this element are listed and explained in the following subsections.

Full path	OpenSplice/DurabilityService/
Occurrences (min-max)	1 - 1
Child-elements	*Element Network* *Element Persistent* *Element NameSpaces* *Element Watchdog* *Element EntityNames* *Element Tracing*
Required attributes	*Attribute name*
Optional attributes	<none>

3.4.1 Attribute *name*

This attribute identifies the configuration for the Durability Service. Multiple Durability Service configurations can be specified in one single resource. The actual applicable configuration is determined by the value of the *name* attribute, which must match the one specified under the *OpenSplice/Domain/Service[@name]* in the configuration of the Domain Service.

Full path	OpenSplice/DurabilityService[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.4.2 Element *Network*

Applications need to be able to gain access to historical data in a system. When the local DDS service gets connected to a remote DDS service by means of the Networking Service, (parts of) the historical data might not be consistent between the local and remote Durability Services. The Durability Service needs to be able to detect the other available Durability Services and exchange historical data with them to keep and/or restore consistency in historical data between them.

The *Network* element provides handles to fine-tune the behaviour of the communication between Durability Services on different computing nodes on network level. These settings only apply when the Networking Service is active.

Full path	OpenSplice/DurabilityService/Network
Occurrences (min-max)	0 - 1
Child-elements	*Element Heartbeat* *Element InitialDiscoveryPeriod* *Element Alignment* *Element WaitForAttachment*
Required attributes	<none>
Optional attributes	*Attribute latency_budget* *Attribute transport_priority*

3.4.2.1 Attribute *latency_budget*

This attribute controls the latency budget QoS setting that is used by the Durability Service for its communication with other Durability Services.

It specifies the maximum acceptable delay (in seconds) from the time the data is written until the data is inserted in the cache of the receiving Durability Service(s) and the receiver is notified of the fact. The value is used by the Networking Service to make communication as efficient as possible. The default value is zero, indicating the delay should be minimized.

`Full path`	OpenSplice/DurabilityService/ Network[@latency_budget]
`Format`	float
`Dimension`	seconds
`Default value`	0.0
`Valid values`	0.0 - maxFloat
`Required`	false

3.4.2.2 Attribute *transport_priority*

This attribute controls the transport priority QoS setting that is used by the Durability Service for its communication with other Durability Services.

It indicates the importance of the communication of the Durability Service with other Durability Services in the system. The transport priority specified here will be interpreted by the Networking Service and should be used to differentiate the priority between communication of user applications and communication of the Durability Service.

`Full path`	OpenSplice/DurabilityService/ Network[@transport_priority]
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	0
`Valid values`	0 - maxInt
`Required`	false

3.4.2.3 Element *Heartbeat*

During startup and at runtime, the network topology can change dynamically. This happens when OpenSplice services are started/stopped or when a network cable is plugged in/out. The Durability Services need to keep data consistency in that environment. To detect newly joining services as well as detecting nodes that are leaving, the Durability Service uses a heartbeat mechanism. This element allows fine-tuning of this mechanism.

Please note this heartbeat mechanism is similar to but not the same as the service liveliness assertion.

Full path	OpenSplice/DurabilityService/Network/Heartbeat
Occurrences (min-max)	0 - 1
Child-elements	*Element ExpiryTime* *Element Scheduling*
Required attributes	0
Optional attributes	*Attribute latency_budget* *Attribute transport_priority*

3.4.2.3.1 Attribute *latency_budget*

This attribute controls the latency budget QoS setting (in seconds) that is only used by the Durability Service for sending its heartbeats. It overrules the value of the *DurabilityService/Network[@latency_budget]*.

Full path	OpenSplice/DurabilityService/Network/ Heartbeat[@latency_budget]
Format	float
Dimension	seconds
Default value	0.0
Valid values	0.0 - maxFloat
Required	false

3.4.2.3.2 Attribute *transport_priority*

This attribute controls the transport priority QoS setting that is only used by the Durability Service for sending its heartbeats. It overrules the value of the *DurabilityService/Network[@transport_priorrity]*.

Full path	OpenSplice/DurabilityService/Network/ Heartbeat[@transport_priority]
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - maxInt
Required	false

3.4.2.3.3 Element *ExpiryTime*

This element specifies the maximum amount of time in which the Durability Service expects a new heartbeat of other Durability Services. This is obviously also the same amount of time in which the Durability Service must send a heartbeat itself.

Increasing this value will lead to less networking traffic and overhead but also to less responsiveness with respect to the liveliness of a Durability Service. Change this value according to the need of your system with respect to these aspects.

`Full path`	OpenSplice/DurabilityService/Network/ Heartbeat/ExpiryTime
`Format`	float
`Dimension`	seconds
`Default value`	10.0
`Valid values`	0.2 - 20.0
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	*Attribute update_factor*
`Optional attributes`	<none>

3.4.2.3.3.1 Attribute update_factor

In case of a (temporary) high CPU load, the scheduling behaviour of the operating system might affect the capability of the Durability Service to send its heartbeat ‘on time’. This attribute introduces some elasticity in this mechanism by making the service send its heartbeat more often then required by the *ExpiryTime*.

The Durability Service will report its liveliness every *ExpiryTime* multiplied by this *update_factor*.

`Full path`	OpenSplice/DurabilityService/Network/ Heartbeat/ExpiryTime[@update_factor]
`Format`	float
`Dimension`	n/a
`Default value`	0.2
`Valid values`	0.1 - 0.9
`Required`	true

3.4.2.3.4 Element *Scheduling*

This element specifies the scheduling parameters used by the thread that periodically sends the heartbeats.

Full path	OpenSplice/DurabilityService/Network/Heartbeat/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.4.2.3.4.1 Element *Class*

This element specifies the thread scheduling class that will be used by the thread that periodically sends the heartbeats. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/DurabilityService/Network/Heartbeat/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.3.4.2 Element *Priority*

This element specifies the thread priority that will be used by the thread that periodically sends the heartbeats. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

`Full path`	OpenSplice/DurabilityService/Network/Heartbeat/Scheduling/Priority
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	depends on operating system
`Valid values`	depends on operating system
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	*Attribute priority_kind*

3.4.2.3.4.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

`Full path`	OpenSplice/DurabilityService/Network/Heartbeat/Scheduling/Priority[@priority_kind]
`Format`	enum
`Dimension`	n/a
`Default value`	Relative
`Valid values`	Relative, Absolute
`Required`	false

3.4.2.4 Element *InitialDiscoveryPeriod*

To be able to ensure data consistency of historical data, the Durability Service needs to know which other Durability Services are available in the system. The value of this element determines the amount of time the Durability Service takes at startup to get acquainted with all other Durability Services in the system.

Increasing the value will increase the startup time of the Durability Service, but is required in larger domains where a lot of network bandwidth is used.

Full path	OpenSplice/DurabilityService/Network/InitialDiscoveryPeriod
Format	float
Dimension	seconds
Default value	3.0
Valid values	0.1 - 10.0
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.5 Element *Alignment*

The Durability Service is responsible for keeping its local cache consistent with the other available Durability caches in the system. To do this, it needs to exchange data to recover from inconsistencies. The exchange of durable data to restore consistency is called alignment. This element allows fine-tuning alignment behaviour of the Durability Service.

Full path	OpenSplice/DurabilityService/Network/Alignment
Occurrences (min-max)	0 - 1
Child-elements	*Element TimeAlignment* *Element AlignerScheduling* *Element AligneeScheduling* *Element RequestCombinePeriod*
Required attributes	<none>
Optional attributes	*Attribute latency_budget* *Attribute transport_priority*

3.4.2.5.1 Attribute *latency_budget*

This attribute specifies the latency budget QoS setting (in seconds) that is only used by the Durability Service for the alignment of data. It overrules the value of the *DurabilityService/Network[@latency_budget]*.

Full path	OpenSplice/DurabilityService/Network/Alignment[@latency_budget]
Format	float
Dimension	seconds
Default value	0.0
Valid values	0.0 - maxFloat
Required	false

3.4.2.5.2 Attribute *transport_priority*

This attribute specifies the transport priority QoS setting that is only used by the Durability Service for the alignment of data. It overrules the value of the *DurabilityService/Network[@transport_priorrity]*.

Full path	OpenSplice/DurabilityService/Network/Alignment[@transport_priority]
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - maxInt
Required	false

3.4.2.5.3 Element *TimeAlignment*

This attribute specifies whether time on all nodes in the domain can be considered aligned or not. This setting needs to be consistent for all durability services in the domain. In case there is no time alignment, the durability service needs to align more data to compensate possible timing gaps between different nodes in the domain.

Full path	OpenSplice/DurabilityService/Network/Alignment/TimeAlignment
Format	boolean
Dimension	n/a
Default value	true

Valid values	true, false
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.4 Element *AlignerScheduling*

This element specifies the scheduling parameters used to control the thread that aligns other durability services.

Full path	OpenSplice/DurabilityService/Network/ Alignment/AlignerScheduling
Occurrences (min-max)	0 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.4.1 Element *Class*

This element specifies the thread scheduling class that will be used by the aligner thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/DurabilityService/Network/ Alignment/AlignerScheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.4.2 Element *Priority*

This element specifies the thread priority that will be used by the aligner thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/DurabilityService/Network/Alignment/AlignerScheduling/Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.4.2.5.4.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/DurabilityService/Network/Alignment/AlignerScheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.4.2.5.5 Element *AligneeScheduling*

This element specifies the scheduling parameters used to control the thread that makes sure the local node becomes and stays aligned.

Full path	OpenSplice/DurabilityService/Network/Alignment/AligneeScheduling
Occurrences (min-max)	1 - 1

Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.5.1 Element *Class*

This element specifies the thread scheduling class that will be used by the alignee thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/DurabilityService/Network/ Alignment/AligneeScheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.5.2 Element *Priority*

This element specifies the thread priority that will be used by the alignee thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/DurabilityService/Network/ Alignment/AligneeScheduling/Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.4.2.5.5.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/DurabilityService/Network/Alignment/AligneeScheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.4.2.5.6 Element *RequestCombinePeriod*

When the Durability Service detects an inconsistency with another Durability Service, it requests that service to align it. The service that receives this request will restore consistency by sending the requested information. In some cases, the Durability Service may receive alignment requests from multiple Durability Services for the same information around the same moment in time. To reduce the processing and networking load in that case, the Durability Service is capable of aligning multiple Durability Services concurrently.

The RequestCombinePeriod has 2 child-elements: a setting that is used when the current Durability Service is not yet aligned with all others (*Initial*) and one for the period after that (*Operational*). These values specify the maximum amount of time the Durability service is allowed to wait with alignment after an alignment request has been received.

Increasing the value will increase the amount of time in which the Durability Service restores from inconsistencies, but will decrease the processing and network load in case multiple Durability Services need to resolve the same data around the same time. Increasing the value is useful in case OpenSplice is started at the same time with more than two computing nodes.

Full path	OpenSplice/DurabilityService/Network/Alignment/RequestCombinePeriod
Occurrences (min-max)	0 - 1
Child-elements	*Element Initial* *Element Operational*
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.6.1 Element *Initial*

This element specifies the maximum amount of time the Durability Service is allowed to wait with alignment after an alignment request has been received and the service itself is not yet considered operational because it has not yet aligned itself with all other Durability Services.

Full path	OpenSplice/DurabilityService/Network/Alignment/RequestCombinePeriod/Initial
Format	float
Dimension	seconds
Default value	0.5
Valid values	0.01 - 5.0
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.5.6.2 Element *Operational*

This element specifies the maximum amount of time the Durability Service is allowed to wait with alignment after an alignment request has been received and the service itself is already considered operational.

Full path	OpenSplice/DurabilityService/Network/Alignment/RequestCombinePeriod/Operational
Format	float
Dimension	seconds
Default value	0.01
Valid values	0.01 - 5.0
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.2.6 Element *WaitForAttachment*

The Durability Service depends on the Networking Service for its communication with other Durability Services. Before it starts communicating, it must make sure the Networking service is ready to send the data. This element specifies what services must be available and how long the Durability Service must wait for them to become available before sending any data.

Full path	OpenSplice/DurabilityService/Network/WaitForAttachment
Occurrences (min-max)	0 - 1
Child-elements	*Element ServiceName*
Required attributes	0
Optional attributes	*Attribute maxWaitCount*

3.4.2.6.1 Attribute *maxWaitCount*

This attribute specifies the number of times the Durability Service checks if the services specified in the *DurabilityService/Network/WaitForAttachment/ServiceName* elements are available before sending any data. An error is logged if one of the services still is unavailable afterwards. The service will continue after that, but this indicates a problem in the configuration and the service might not function correctly anymore.

Full path	OpenSplice/DurabilityService/Network/WaitForAttachment[@maxWaitCount]
Format	unsigned integer
Dimension	n/a
Default value	200
Valid values	1 - 1000
Required	false

3.4.2.6.2 Element *ServiceName*

This element specifies the name of the service(s) that the Durability Service waits for, before starting alignment activities for a specific topic-partition combination. In a multi-node environment the name of the Networking Service MUST be included here to assure a proper functioning of the Durability Service.

Full path	OpenSplice/DurabilityService/Network/ WaitForAttachment/ServiceName
Format	string
Dimension	name of an existing service
Default value	n/a
Valid values	any valid service name
Occurrences (min-max)	1 - *
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3 Element *Persistent*

Durable data is divided in transient and persistent data. Transient data must stay available for as long as at least one Durability Service is available in the system. For persistent data it is the same, but that type of data must also outlive the downtime of the system. The Durability Service stores the persistent data on permanent storage to realize this. This element can be used to fine-tune the behaviour of the Durability Service concerning the persistent properties of the data.

Note these elements are only available as part of the DDS persistence profile of OpenSplice.

Full path	OpenSplice/DurabilityService/Persistent
Occurrences (min-max)	0 - 1
Child-elements	*Element StoreDirectory* *Element StoreMode* *Element StoreSessionTime* *Element StoreSleepTime* *Element StoreOptimizeInterval* *Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.4.3.1 Element *StoreDirectory*

This element determines the location where the persistent data will be stored on disk. The value should point to a directory on disk. If this parameter is not configured, the Durability Service will not manage persistent data.

Full path	OpenSplice/DurabilityService/Persistent/StoreDirectory
Format	string
Dimension	path to directory
Default value	“” (empty string representing no persistent storage)
Valid values	depends on operating system
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3.2 Element *StoreMode*

This element specifies the plugin that is used to store the persistent data on disk. Currently only the XML plugin is supported, that will store persistent data in XML files.

Full path	OpenSplice/DurabilityService/Persistent/StoreMode
Format	enumeration
Dimension	n/a
Default value	XML
Valid values	XML
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3.3 Element *StoreSessionTime*

This element specifies the maximum session time (in seconds) for the persistency thread. After this period of time, it makes sure data is flushed to disk.

Full path	OpenSplice/DurabilityService/Persistent/StoreSessionTime
Format	float
Dimension	seconds
Default value	20.0
Valid values	5.0 - 60.0
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3.4 Element *StoreSleepTime*

This element specifies the period of time (in seconds) the persistency thread sleeps between two sessions. This allows influencing the CPU load of the persistency thread.

Full path	OpenSplice/DurabilityService/Persistent/StoreSleepTime
Format	float
Dimension	seconds
Default value	2.0
Valid values	0.5 - 10.0
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3.5 Element *StoreOptimizeInterval*

This element determines after how many write actions the persistent set for a specific partition-topic combination is optimized on disk.

Full path	OpenSplice/DurabilityService/Persistent/StoreOptimizeInterval
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 1000000000
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3.6 Element *Scheduling*

This element specifies the scheduling parameters used to control the thread that stores persistent data on permanent storage.

Full path	OpenSplice/DurabilityService/Persistent/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.4.3.6.1 Element *Class*

This element specifies the thread scheduling class that will be used by the persistent thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/DurabilityService/Persistent/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default

Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.3.6.2 Element *Priority*

This element specifies the thread priority that will be used by the persistent thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/DurabilityService/Persistent/ Scheduling/Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.4.3.6.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/DurabilityService/Persistent/ Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.4.4 Element *NameSpaces*

When a durability service wants to fulfill a particular role for some of the namespaces in a domain, it must have some way of deducing the desired behavior for those when encountered. For static, small-scale systems this can easily be solved by statically configurating this role-behavior for all relevant namespaces for each durability service in the domain.

In dynamic, large scale environments, the updating and maintaining of configurations for each durability service when new namespaces enter the domain can become quite cumbersome. Dynamic namespaces offer a solution for this problem.

Instead of specifying each namespace seperately, dynamic namespaces introduce the concept of namespace policies. A policy defines a generic role for the durability service, together with a namespace expression. This expression can contain wildcards, and is used to match against each namespace the durability service encounters in a domain. The first policy with a matching expression is then applied to the new namespace.

Specifying policies

Policies are specified in a fall-through manner, which means, that the first (top) policy to match a namespace is applied. Policies specify a range of options for namespaces, which tell the durability service how to handle the data. The following items can be configured:

* Durability
* Alignee
* Aligner

In the dynamic namespace configuration, the `NameSpace` element (a child of the NameSpaces element) only supports a `name` attribute, which is mandatory. This name will be used to match against policies.

An example of dynamic namespace configuration is shown on page 77, followed by a short note about *Backwards compatibility*.

`Full path`	OpenSplice/DurabilityService/NameSpaces
`Occurrences (min-max)`	1 - 1
`Child-elements`	*Element NameSpace* *Element Policy*
`Required attributes`	<none>
`Optional attributes`	<none>

3.4.4.1 Element *NameSpace*

The `NameSpace` element only supports a `name` attribute, which is mandatory. This name will be used to match against policies.

`Full path`	OpenSplice/DurabilityService/NameSpaces/ NameSpace
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	*Attribute name*
`Optional attributes`	<none>

3.4.4.1.0.1 Attribute *name*

This name is used to match against policies.

3.4.4.2 Element *Policy*

Policies are child elements of the NameSpaces element. The Policy element has a mandatory nameSpace attribute, in which a namespace expression is expected. This expression will be used to match with a namespace name.

`Full path`	OpenSplice/DurabilityService/NameSpaces/ Policy
`Occurrences (min-max)`	1 - *
`Child-elements`	<none>
`Required attributes`	*Attribute durability* *Attribute alignee* *Attribute aligner*
`Optional attributes`	<none>

3.4.4.2.1 Attribute *durability*

This element specifies how the durability service manages the data within the NameSpace. The original durability of the data (determined by the DataWriter that wrote it) can be ‘weakened’ (Persistent > Transient > Transient_local). This is useful to improve resource usage of the durability service in the situation where resource usage is more important then fault-tolerance. This parameter cannot be used to increase the original durability of samples. In case the value of this parameter is larger then the value a sample was published with, the sample will be handled as specified in the DataWriter durability QoS.

- **Persistent**: Maximum profile. A sample will be handled as specified in the durability Qos of the datawriter that wrote it.

- **Transient**: A sample will be stored transient at best.
- **Transient_Local**: A sample will be stored transient_local at best.
- **Durable**: Convenience value that is equal to persistent.

Full path	OpenSplice/DurabilityService/NameSpaces/Policy [@durability]
Format	enumeration
Dimension	n/a
Default value	Durable
Valid values	Persistent, Transient, Transient_Local, Durable
Required	true

3.4.4.2.2 Attribute *alignee*

This element determines how the durability service manages the data that matches the namespace. Scalability of durable data is an issue in large systems. Keeping all historical data on each node may not be feasible. Often nodes are interested in a small part of the total system data. They are driven by both performance (boot time, memory usage, network load, CPU load) and fault tolerance (the need for replicates).

- **Initial**: The durability service requests historical data at startup and caches it locally. Historical data will be available relatively fast for new local data readers and the system is more fault-tolerant. However, caching of historical data requires a relatively large amount of resources and a long boot time.
- **Lazy**: The Durability service caches historical data after local application interest arises for the first time and a remote Durability service aligns the first data reader. Historical data is available relatively slow for the first data reader, but for every new data reader it is relatively fast. The caching resources are only used when local interest in the data arises, so it only requires resources if there is actual local interest. However, this method provides no fault-tolerance for the domain, because the local Durability service is only partly a replica and is not able to provide historical data to remote Durability service and/or remote data readers.
- **On_Request**: The Durability service will not cache historical data, but will align each separate DataReader? on a request basis (in the situation where it calls wait_for_historical_data). Each new DataReader? that wants historical data therefore leads to a new alignment action. This is a good setting to limit the amount of resources used on the node, but will potentially lead to more network traffic. This method provides no fault-tolerance for the domain.

Full path	OpenSplice/DurabilityService/NameSpaces/Policy[@alignee]
Format	enumeration
Dimension	n/a
Default value	Initial
Valid values	Initial, Lazy, On_Request
Required	true

3.4.4.2.3 Attribute *aligner*

This mandatory attribute determines whether the durability service can act as aligner (provide historical data) for other durability services.

Full path	OpenSplice/DurabilityService/NameSpaces/Policy[@aligner]
Format	boolean
Default value	true
Valid values	true, false
Required	true

Example

Consider the following configuration example for the durability service:

```
<NameSpaces>
  <NameSpace name="Ns01">
    <Partition>A*</Partition>
  </NameSpace>
  <Policy nameSpace="Ns*" durability="Durable" alignee="Initial"
aligner="True"/>
  <Policy nameSpace="*" durability="Transient" alignee="Lazy"
aligner="False"/>
</NameSpaces>
```

From this configuration, the following behavior can be deduced:

- One namespace with name "Ns01" which holds all partitions with prefix "A" is created and assigned with the following policy: {durabilityKind=Durable, alignmentKind=Initial, aligner=True}
- All late-joining namespaces with prefix "Ns" will be assigned with the following policy: {durabilityKind=Durable, alignmentKind=Initial, aligner=True}
- All other late joining namespaces (without prefix "Ns") will be assigned with the following policy: {durabilityKind=Transient, alignmentKind=Lazy, aligner=False}

Backwards compatibility

Although the structure of the namespace configuration file has changed, the durability service still supports the old format, where the policies for a namespace are configured on the namespace element itself. An example of an old configuration:

```
<NameSpaces>
  <NameSpace durabilityKind="Durable"
alignmentKind="Initial_and_Aligner">
    <Partition>*</Partition>
  </NameSpace>
</NameSpaces>
```

Note that this configuration is deprecated and supported only by the durability service for backwards compatibility. New users should use the new configuration and existing users should migrate to this configuration whenever possible. The configurator will by default not accept the old format anymore. When for some reason migration to the new format is not possible or desired, a user can use the configurator from an older version, or edit the configuration file manually.

When converting from old to new format, all namespace elements must be provided with a name, and a matching policy for that name must be defined. For the above example of an old configuration, this could be the equivalent in the new format:

```
<NameSpaces>
  <NameSpace name="DefaultNameSpace">
    <Partition>*</Partition>
  </NameSpace>
  <Policy nameSpace="DefaultNameSpace" durability="Durable"
aligner="True" alignee="Initial"/>
</NameSpaces>
```

3.4.5 Element *Watchdog*

This element controls the characteristics of the Watchdog thread.

`Full path`	OpenSplice/DurabilityService/Watchdog
`Occurrences (min-max)`	0 - 1
`Child-elements`	*Element Scheduling*
`Required attributes`	<none>
`Optional attributes`	<none>

3.4.5.1 Element *Scheduling*

This element specifies the scheduling parameters used to control the watchdog thread.

Full path	OpenSplice/DurabilityService/Watchdog/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.4.5.1.1 Element *Class*

This element specifies the thread scheduling class that will be used by the watchdog thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/DurabilityService/Watchdog/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.5.1.2 Element *Priority*

This element specifies the thread priority that will be used by the watchdog thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/DurabilityService/Watchdog/Scheduling/Priority
Format	unsigned integer
Dimension	n/a

Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.4.5.1.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/DurabilityService/Persistent/ Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.4.6 Element *EntityNames*

This element specifies the names of the various entities used by the DurabilityService. The names specified here will be displayed in the OpenSplice DDS Tuner when viewing the DurabilityService.

Full path	OpenSplice/DurabilityService/EntityNames
Occurrences (min-max)	0 - 1
Child-elements	*Element Publisher* *Element Subscriber* *Element Partition*
Required attributes	<none>
Optional attributes	<none>

3.4.6.1 Element Publisher

This element specifies the name of the durability publisher.

Full path	OpenSplice/DurabilityService/EntityNames/Publisher
Format	string
Dimension	entity name
Default value	“durabilityPublisher”
Valid values	any string
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.6.2 Element Subscriber

This element specifies the name of the durability subscriber.

Full path	OpenSplice/DurabilityService/EntityNames/Subscriber
Format	string
Dimension	entity name
Default value	“durabilitySubscriber”
Valid values	any string
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.6.3 Element Partition

This element specifies the name of the durability partition.

Full path	OpenSplice/DurabilityService/EntityNames/Partition
Format	string
Dimension	partition name
Default value	“durabilityPartition”

Valid values	any valid partition name
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.7 Element Tracing

This element controls the amount and type of information that is written into the tracing log by the Durability Service. This is useful to track the Durability Service during application development. In the runtime system it should be turned off.

Full path	OpenSplice/DurabilityService/Tracing
Occurrences (min-max)	0 - 1
Child-elements	*Element OutputFile* *Element Timestamps* *Element Verbosity*
Required attributes	<none>
Optional attributes	<none>

3.4.7.1 Element *OutputFile*

This option specifies where the logging is printed to. Note that “stdout” is considered a legal value that represents “standard out” and “stderr” is a legal value representing “standard error”. The default value is an empty string, indicating that all tracing is disabled.

Full path	OpenSplice/DurabilityService/Tracing/OutputFile
Format	string
Dimension	file name
Default value	“” (empty string indicating no tracing)
Valid values	depends on operating system.
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.4.7.2 Element *Timestamps*

This element specifies whether the logging must contain timestamps.

Full path	OpenSplice/DurabilityService/Tracing/Timestamps
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute Absolute*

3.4.7.2.1 Attribute *Absolute*

This attribute specifies whether the timestamps are absolute or relative to the startup time of the service.

Full path	OpenSplice/DurabilityService/Tracing/ Timestamps[@absolute]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	false

3.4.7.3 Element *Verbosity*

This element specifies the verbosity level of the loggin information. The higher the level, the more (detailed) information will be logged.

Full path	OpenSplice/DurabilityService/Tracing/Verbosity
Format	enumeration
Dimension	n/a
Default value	INFO
Valid values	SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST
Occurrences (min-max)	0 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5 The Networking Service

When communication endpoints are located on different computing nodes, the data produced using the local DDS service must be communicated to the remote DDS service and the other way around. The Networking service provides a bridge between the local DDS service and a network interface.

Multiple Networking services can exist next to each other, each serving one physical network interface.

Please refer to section *1.6* on page 14 for notes about applications operating in multiple domains and interactions with the Networking Service.

The Networking service is responsible for forwarding data to the network and for receiving data from the network. It can be configured to distinguish multiple communication channels with different QoS policies assigned to be able to schedule sending and receival of specific messages to provide optimal performance for a specific application domain.

The networking configuration expects a root element named *OpenSplice/NetworkingService*. Within this root element, the networking daemon will look for several child-elements. Each of these is listed and explained.

Full path	OpenSplice/NetworkService
Occurrences (min-max)	0 - *
Child-elements	*Element General* *Element Partitioning* *Element Channels* *Element Discovery* *Element Tracing*
Required attributes	*Attribute name*
Optional attributes	<none>

3.5.1 Attribute *name*

This attribute identifies the configuration for the Networking Service. Multiple Networking Service configurations can be specified in one single resource. The actual applicable configuration is determined by the value of the *name* attribute, which must match the one specified under the *OpenSplice/Domain/Service[@name]* in the configuration of the Domain Service.

Full path	OpenSplice/NetworkService[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.5.2 Element *General*

This element contains general parameters that concern the networking service as a whole.

Full path	OpenSplice/NetworkService/General
Occurrences (min-max)	0 - *
Child-elements	*Element NetworkInterfaceAddress*
Required attributes	<none>
Optional attributes	<none>

3.5.2.1 Element *NetworkInterfaceAddress*

This element specifies which network interface card (NIC) should be used.

Full path	OpenSplice/NetworkService/General/ NetworkInterfaceAddress
Format	string
Dimension	n/a
Default value	"first available"
Valid values	"first available", any dotted decimal IPv4 address, or a symbolic name of a NIC
Occurrences (min-max)	0 - 1
Child-elements	<none>

Required attributes	<none>
Optional attributes	<none>
Remarks	The given interface should have the required capabilities, e.g. broadcasting

Every Networking service is bound to only one network interface card. The card can be uniquely identified by its corresponding IP address or by its symbolic name (*e.g.* `eth0`). If the value "`first available`" is entered here, OpenSplice will try to look up an interface that has the required capabilities.

NOTE: On Integrity, IP addresses *must* be given in the `NetworkInterfaceAddress` element for nodes which have more than one ethernet interface, as it is not possible to detect NIC broadcast/multicast capabilities automatically in this environment.

3.5.2.2 Element *Reconnection*

This element specifies the desired networking-behavior with respect to the validity of restoring lost connectivity with remote nodes. Here 'lost connectivity' means a prolonged inability to communicate with a known and still active remote node (typically because of network issues) that has resulted in such a node being declared 'dead' either by the topology-discovery or lost-reliability being detected by a reliable channel's reactivity-checking mechanism. If automatic reconnection is allowed, communication channels with the now-reachable-again node will be restored, even though reliable data might have been lost during the disconnection period.

Full path	OpenSplice/NetworkService/General/Reconnection
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	*Attribute allowed*
Optional attributes	<none>

3.5.2.2.1 Attribute *allowed*

This attribute specifies whether the network service must resume communication with another network service when it has already been seen before but has been disconnected for a while.

- *`false`* - Specifies that the network service must NOT resume communication.
- *`true`* - Specifies that the network service must resume communication.

Full path	OpenSplice/NetworkService/General/Reconnection[@allowed]
Format	boolean
Dimension	n/a
Default value	flase
Valid values	true, false
Required	true

3.5.3 Element *Partitioning*

The OpenSplice Networking service is capable of leveraging the network's multicast and routing capabilities. If some *a priori* knowledge about the participating nodes and their topic and partition interest is available, then the networking services in the system can be explicitly instructed to use specific unicast or multicast addresses for its networking traffic. This is done by means of so-called network partitions.

A network partition is defined by one or more unicast, multicast or broadcast IP addresses. Any networking service that is started will read the network partition settings and, if applicable, join the required multicast groups. For every sample distributed by the networking service, both its partition and topic name will be inspected. In combination with a set of network partition mapping rules, the service will determine which network partition the sample is written to. The mapping rules are configurable as well.

Using networking configuration, nodes can be disconnected from any networking partition. If a node is connected via a low speed interface, it is not capable of receiving high volume data. If the DCPS partitioning is designed carefully, high volume data is published into a specific partition, which in its turn is mapped onto a specific networking partition, which is itself only connected to those nodes that are capable of handling high volume data.

Full path	OpenSplice/NetworkService/Partitioning
Occurrences (min-max)	1 - 1
Child-elements	*Element GlobalPartition* *Element NetworkPartitions* *Element PartitionMappings*
Required attributes	<none>
Optional attributes	<none>

3.5.3.1 Element *GlobalPartition*

This element specifies the global or default networking partition. This global networking partition transports data that is either meant to be global, like discovery heartbeats, or that is not mapped onto any other networking partition.

`Full path`	OpenSplice/NetworkService/Partitioning/GlobalPartition
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	*Attribute Address*
`Optional attributes`	<none>

3.5.3.1.1 Attribute *Address*

The GlobalPartition address is a list of one or more unicast, multicast or broadcast addresses. If more than one address is specified, then the different addresses should separated by commas (`,`) semicolons (`;`) or spaces ( ). Samples for the global partition will be sent to all addresses that are specified in this list of addresses. To specify the default broadcast address, use the expression "`broadcast`". Addresses can be entered as dotted decimal notation or as the symbolic hostname, in which case OpenSplice will try to resolve the corresponding IP address.

`Full path`	OpenSplice/NetworkService/Partitioning/GlobalPartition[@Address]
`Format`	dotted decimal IPv4 address, symbolic host name or "broadcast"
`Dimension`	n/a
`Default value`	"broadcast"
`Valid values`	"broadcast", any dotted decimal IPv4 unicast or multicast address, or a resolvable symbolic hostname
`Required`	true
`Remarks`	The given interface should have the required capabilities, e.g. broadcasting or multicasting

3.5.3.2 Element *NetworkPartitions*

Networking configuration can contain a set of networking partitions, which are grouped under the NetworkPartitions element.

Full path	OpenSplice/NetworkService/Partitioning/ NetworkPartitions
Occurrences (min-max)	0 - 1
Child-elements	*Element NetworkPartition*
Required attributes	<none>
Optional attributes	<none>

3.5.3.2.1 Element *NetworkPartition*

Every NetworkPartition has a name, an address and a connected flag.

Full path	OpenSplice/NetworkService/Partitioning/ NetworkPartitions/NetworkPartition
Occurrences (min-max)	1 - *
Child-elements	<none>
Required attributes	*Attribute Address*
Optional attributes	*Attribute Name* *Attribute Connected* *Attribute Compression*

3.5.3.2.1.1 Attribute *Name*

A networking partition is uniquely identified by its name.

Full path	OpenSplice/NetworkService/Partitioning/NetworkPartitions /NetworkPartition[@Name]
Format	string
Dimension	n/a
Default value	string representation of the corresponding address
Valid values	any valid partition name
Required	false
Remarks	The name should be unique over all networking partitions.

3.5.3.2.1.2 Attribute *Address*

The address is a list of one or more unicast, multicast or broadcast addresses. If more than one address is specified, then the different addresses should separated by commas (**,**) semicolons (**;**) or spaces ( ). Samples for this partition will be sent to all addresses that are specified in this list of addresses. To specify the default

broadcast address, use the expression "`broadcast`". Addresses can be entered as dotted decimal notation or as the symbolic hostname, in which case OpenSplice will try to resolve the corresponding IP address.

`Full path`	OpenSplice/NetworkService/Partitioning/NetworkPartitions /NetworkPartition[@Address]
`Format`	dotted decimal IPv4 address, symbolic host name or "broadcast"
`Dimension`	n/a
`Default value`	"broadcast"
`Valid values`	"broadcast", any dotted decimal IPv4 unicast or multicast address or resolvable symbolic hostname
`Required`	true
`Remarks`	The given interface should have the required capabilities, e.g. broadcasting or multicasting.

3.5.3.2.1.3 Attribute *Connected*

A node can choose to be not connected to a networking partition by setting the Connected attribute.

`Full path`	OpenSplice/NetworkService/Partitioning/NetworkPartitions /NetworkPartition[@Connected]
`Format`	boolean
`Dimension`	n/a
`Default value`	true
`Valid values`	true, false
`Required`	false

If a node is connected to a networking partition, it will join the corresponding multicast group and it will receive data distributed over the partition. If it is not connected, data distributed over the partition will not reach the node but will be filtered by the networking interface or multicast enabled switches.

3.5.3.2.1.4 Attribute *Compression*

If the `Compression` attribute is set on a partition, outgoing data will be compressed before sending. Depending on the nature of the published data, this may improve performance, particularly when on a slow network.

`Full path`	OpenSplice/NetworkService/Partitioning/NetworkPartitions /NetworkPartition[@Connected]
`Format`	boolean
`Dimension`	n/a
`Default value`	false
`Valid values`	true, false
`Required`	false

3.5.3.3 Element *IgnoredPartitions*

This element is used to group the set of *IgnoredPartition* elements.

`Full path`	OpenSplice/NetworkService/Partitioning/ IgnoredPartitions
`Occurrences (min-max)`	0 - 1
`Child-elements`	*Element IgnoredPartition*
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.3.3.1 Element *IgnoredPartition*

This element can be used to create a "Local Partition" that is only available on the node on which it is specified, and therefore won't generate network-load. Any DCPS partition-topic combination specified in this element will not be distibuted by the Networking service.

`Full path`	OpenSplice/NetworkService/Partitioning/ IgnoredPartitions/IgnoredPartition
`Occurrences (min-max)`	1 - *
`Child-elements`	<none>
`Required attributes`	*Attribute DCPSPartitionTopic*
`Optional attributes`	<none>

3.5.3.3.1.1 Attribute *DCPSPartitionTopic*

The Networking service will match any DCPS messages to the DCPSPartitionTopic expression and determine if it matches. The PartitionExpression and TopicExpression are allowed to contain a '*' wildcard, meaning that anything matches. An exact match is considered better than a wildcard match. If a DCPS messages matches an expression it will not be send to the network.

`Full path`	OpenSplice/NetworkService/Partitioning/ IgnoredPartitions/IgnoredPartition [@DCPSPartitionTopic]
`Format`	PartitionExpression.TopicExpression
`Dimension`	n/a
`Default value`	*.*
`Valid values`	Expressions containing * wildcards
`Required`	true

3.5.3.4 Element *PartitionMappings*

This element is used to group a set of PartitionMapping elements.

`Full path`	OpenSplice/NetworkService/Partitioning/ PartitionMappings
`Occurrences (min-max)`	0 - 1
`Child-elements`	*Element PartitionMapping*
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.3.4.1 Element *PartitionMapping*

This element specifies a mapping between a network partition and a partition-topic combination.

`Full path`	OpenSplice/NetworkService/Partitioning/ PartitionMappings/PartitionMapping
`Occurrences (min-max)`	1 - *
`Child-elements`	<none>
`Required attributes`	*Attribute DCPSPartitionTopic* *Attribute NetworkPartition*
`Optional attributes`	<none>

In order to give networking partitions a meaning in the context of DCPS, mappings from DCPS partitions and topics onto networking partitions should be defined. Networking allows for a set of partition mappings to be defined.

3.5.3.4.1.1 Attribute *DCPSPartitionTopic*

The Networking Service will match any DCPS messages to the DCPSPartitionTopic expression and determine if it matches. The PartitionExpression and TopicExpression are allowed to contain a '*' wild card, meaning that anything matches. An exact match is considered better than a wild card match. For every DCPS message, the best matching partition is determined and the data is sent over the corresponding networking partition as specified by the matching *NetworkPartition* element.

Full path	OpenSplice/NetworkService/Partitioning/PartitionMappings /PartitionMapping[@DCPSPartitionTopic]
Format	PartitionExpression.TopicExpression
Dimension	n/a
Default value	*.*
Valid values	Expressions containing * wildcards
Required	true

3.5.3.4.1.2 Attribute *NetworkPartition*

The NetworkPartition attribute of a partition mapping defines that networking partition that data in a specific DCPS partition of a specific DCPS topic should be sent to.

Full path	OpenSplice/NetworkService/Partitioning/PartitionMappings /PartitionMapping[@NetworkPartition]
Format	string
Dimension	n/a
Default value	n/a
Valid values	Any name of a previously defined networking partition
Required	true

3.5.4 Element *Channels*

This element is used to group a set of Channels.

Full path	OpenSplice/NetworkService/Channels
Occurrences (min-max)	1 - 1
Child-elements	*Element Channel*
Required attributes	<none>
Optional attributes	<none>

The set of channels define the behaviour of the ‘network’ concerning aspects as priority, reliability and latency budget. By configuring a set of channels, the Networking Service is able to function as a ‘scheduler’ for the network bandwidth. It achieves this by using the application-defined DDS QoS policies of the data to select the most appropriate channel to send the data.

3.5.4.1 Element *Channel*

This element specifies all properties of an individual Channel.

Full path	OpenSplice/NetworkService/Channels/Channel
Occurrences (min-max)	1 - 42
Child-elements	*Element PortNr* *Element FragmentSize* *Element Resolution* *Element AdminQueueSize* *Element Sending* *Element Scheduling*
Required attributes	*Attribute name* *Attribute enabled* *Attribute reliable*
Optional attributes	*Attribute priority* *Attribute default*

The Networking Service will make sure messages with a higher priority precede messages with a lower priority and it uses the latency budget to assemble multiple messages into one UDP packet where possible, to optimise the bandwidth usage. Of course, its performance depends heavily on the compatibility of the configured channels with the used DDS QoS policies of the applications.

3.5.4.1.1 Attribute *name*

The name uniquely identifies the channel.

Full path	OpenSplice/NetworkService/Channels/Channel[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.5.4.1.2 Attribute *enabled*

This attribute toggles a channel on or off.

Full path	OpenSplice/NetworkService/Channels/Channel[@enabled]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	true

Toggling a channel between enabled and disabled is a quick alternative for commenting out the corresponding lines in the configuration file.

3.5.4.1.3 Attribute *reliable*

If this attribute is set to true, the channel sends all messages reliably. If not, data is sent only once (fire-and-forget).

Full path	OpenSplice/NetworkService/Channels/Channel[@reliable]
Format	boolean
Dimension	n/a
Default value	false
Valid values	true, false
Required	true
Remarks	This setting should be consistent over all nodes in the system

The specific channel a message is written into depends on the attached quality of service. Once a message has arrived in a channel, it will be transported with the quality of service attached to the channel. If the reliable attribute happens to be set to true, the message will be sent over the network using a reliability protocol.

3.5.4.1.4 Attribute *priority*

This attribute sets the transport priority of the channel.

Messages sent to the network have a transport_priority quality of service value. Selection of a networking channel is based on the priority requested by the message and the priority offered by the channel. The priority settings of the different channels divide the priority range into intervals. Within a channel, messages are sorted in order of priority.

`Full path`	OpenSplice/NetworkService/Channels/Channel[@priority]
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	0
`Valid values`	0 - maxInt
`Required`	false
`Remarks`	The specified priority value has no relation to the operating system threading priority.

3.5.4.1.5 Attribute *default*

This attribute indicates whether the channel is selected as the default channel in case no channel offers the quality of service requested by a message.

The networking channels should be configured corresponding to the quality of service settings that are expected to be requested by the applications. It might happen, however, that none of the available channels meets the requested quality of service for a specific message. In that case, the message will be written into the default channel.

`Full path`	OpenSplice/NetworkService/Channels/Channel[@default]
`Format`	boolean
`Dimension`	n/a
`Default value`	false
`Valid values`	true, false
`Required`	false
`Remarks`	Only one channel is allowed to have this attribute set to true

3.5.4.1.6 Element *PortNr*

This element specifies the port number used by the Channel. Messages for the channel are sent to the port number given. Each channel needs its own unique port number. Please note that 'reliable' channels use a second port, which is the specified *PortNr* + 1.

Full path	OpenSplice/NetworkService/Channels/Channel/PortNr
Format	unsigned integer
Dimension	n/a
Default value	3367
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.7 Element *FragmentSize*

The networking module will fragment large message into smaller fragments with size *FragmentSize*. These fragments are sent as datagrams to the UDP stack. OS-settings determine the maximum datagram size.

Full path	OpenSplice/NetworkService/Channels/Channel/FragmentSize
Format	unsigned integer
Dimension	bytes
Default value	1300
Valid values	200 - 65536 (if OS allows it)
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.8 Element *Resolution*

The resolution indicates the number of milliseconds that this channel sleeps between two consecutive resend or packing actions. Latency budget values are truncated to a multiple of *Resolution* milliseconds.

It is considered good practice to specify the Resolution consistently throughout the system.

`Full path`	OpenSplice/NetworkService/Channels/Channel/Resolution
`Format`	unsigned integer
`Dimension`	milliseconds
`Default value`	10
`Valid values`	1 - MaxInt
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.4.1.9 Element *AdminQueueSize*

For reliable channels the receiving side needs to keep the sending side informed about the received data and the received control messages.

This is done by means of an "AdminQueue". This setting determines the size of this queue, and it must be greater than the maximum number of reliable messages send or received during each "Resolution" milliseconds.

`Full path`	OpenSplice/NetworkService/Channels/Channel/AdminQueueSize
`Format`	unsigned integer
`Dimension`	messages
`Default value`	4000
`Valid values`	400 - maxInt
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.4.1.10 Element *Sending*

This element describes all properties for the transmitting side of the Channel.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending
Occurrences (min-max)	0 - 1
Child-elements	*Element QueueSize* *Element MaxBurstSize* *Element ThrottleLimit* *Element ThrottleThreshold* *Element MaxRetries* *Element RecoveryFactor* *Element DiffServField* *Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.1 Element *QueueSize*

This element specifies the number of messages the networking queue can contain.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/QueueSize
Format	unsigned integer
Dimension	messages
Default value	400
Valid values	1 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

Messages sent to the network are written into the networking queue. The networking service will read from this queue. If this queue is full, the writer writing into the queue is suspended and will retry until success. Note that a full networking queue is a symptom of an improperly designed system.

3.5.4.1.10.2 Element *MaxBurstSize*

Amount in bytes to be sent at maximum every "Resolution" milliseconds.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/MaxBurstSize
Format	unsigned integer
Dimension	bytes/(resolution interval)
Default value	200000
Valid values	1024 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.3 Element *ThrottleLimit*

Throttling will enable you to further limit (below MaxBurstSize) the amount of data that is sent every Resolution interval. This happens if one of the receiving nodes in the network indicates that it has trouble processing all incoming data. This value is the lower boundary of the range over which the throttling can adapt the limit. If this value is set to the same value (or higher) as MaxBurstSize throttling is disabled.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/ThrottleLimit
Format	unsigned integer
Dimension	bytes/(resolution interval)
Default value	10240
Valid values	128 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.4 Element *ThrottleThreshold*

This is the number of unprocessed network fragments that a node will store before it will inform the other nodes in the network that it has trouble processing the incoming data. Those other nodes can use this information to adjust their throttle values, effectively reducing the amount of incoming data in case of a temporary overflow, and increasing again when the node is able to catch up.

It is considered good practice to specify the ThrottleTreshold consistently throughout the system.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/ThrottleThreshold
Format	unsigned integer
Dimension	fragments
Default value	50
Valid values	2 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.5 Element *MaxRetries*

The number of retransmissions the service has to execute before considering the addressed node as not responding.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/MaxRetries
Format	unsigned integer
Dimension	n/a
Default value	100
Valid values	1 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.6 Element *RecoveryFactor*

A lost message is resent after Resolution * RecoveryFactor milliseconds.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/RecoveryFactor
Format	unsigned integer
Dimension	n/a
Default value	3
Valid values	2 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.7 Element *DiffServField*

This element describes the DiffServ setting the channel will apply to all its networking messages.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/DiffServField
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 255
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.8 Element *DontRoute*

The IP DONTROUTE socket option is set to the value specified.

Full Path	OpenSplice/NetworkService/Channels/Channel/Sending/DontRoute
Format	boolean
Dimension	n.a
Default value	true

Valid values	true, false
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.9 Element *TimeToLive*

For each UDP packet sent out, the IP Time To Live header value is set to the value specified.

Full Path	OpenSplice/NetworkService/Channels/Channel/Sending/TimeToLive
Format	unsigned integer
Dimension	n.a
Default value	0
Valid values	0-255
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.10 Element *Scheduling*

This element specifies the scheduling policies used to control the transmitter thread of the current Channel.

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/Scheduling
Occurrences (min-max)	0 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.10.1 Element *Class*

This element specifies the thread scheduling class that will be used by the transmitter thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/NetworkService/Channel/Channels/Sending/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.10.10.2 Element *Priority*

This element specifies the thread priority that will be used by the transmitter thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/NetworkService/Channel/Channels/Sending/Scheduling/Priority
Format	integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.5.4.1.10.10.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/NetworkService/Channel/Channels/Sending/Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.5.4.1.10.11 Element *CrcCheck*

This configuration element has been added in order to protect OpenSplice network packets from malicious attacks. CRCs (Cyclic Redundancy Checks) are specifically designed to protect against common types of errors on communication channels. When enabled, the integrity of delivered network packets from one DDS process to another is assured. There is a small performance cost to using this feature due to the addtional overhead of carrying out the CRC calculations.

When the sending side is enabled the network packet will contain a valid `crc` field.

Full Path	OpenSplice/NetworkService/Channel/Channels/Sending/CrcCheck
Format	boolean
Dimension	n/a
Default value	False
Valid values	True, False
Occurrences (min-max)	0 - n
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.11 Element *Receiving*

This element describes all properties for the receiving side of the Channel.

Full path	OpenSplice/NetworkService/Channels/Channel/Receiving
Occurrences (min-max)	0 - 1
Child-elements	*Element ReceiveBufferSize* *Element DefragBufferSize* *Element SMPOptimization* *Element MaxReliabBacklog* *Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.11.1 Element *ReceiveBufferSize*

The UDP receive buffer of the Channel socket is set to the value given. If many message are lost, the receive buffer size has to be increased.

Full path	OpenSplice/NetworkService/Channels/Channel/Receiving/ReceiveBufferSize
Format	unsigned integer
Dimension	bytes
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.11.2 Element *DefragBufferSize*

The maximum number of Fragment buffers that will be allocated for this channel. These buffers are used to store incoming fragments waiting to be processed, as well as fragments that are being processed.

Full path	OpenSplice/NetworkService/Channels/Channel/Receiving/DefragBufferSize
Format	unsigned integer
Dimension	fragments

Default value	5000 (For BestEffort Channels) 200000 (For Reliable Channels)
Valid values	500 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.11.3 Element *SMPOptimization*

This option will distribute the processing done for incoming fragements over multiple threads, which will lead to an improved throughput on SMP nodes.

Full path	OpenSplice/NetworkService/Channels/Channel/Receiving/SMPOptimization
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	*Attribute enabled*
Optional attributes	<none>

3.5.4.1.11.3.1 Attribute *enabled*

This attribute toggles the Optimization on or off.

Full path	OpenSplice/NetworkService/Channel/Channels/Receiving/SMPOptimization/[@enabled]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	true

3.5.4.1.11.4 Element *MaxReliabBacklog*

This is a lower limit to the DefragBufferSize that specifies the number of received fragments from a single remote node allocated for the purpose of order preservation because an earlier fragment from that remote node is missing. If this number is exceeded, then that particular remote node that didn't resend the missing fragent in time is considered dead for this channel.

`Full path`	OpenSplice/NetworkService/Channels/Channel/Receiving/MaxReliabBacklog
`Format`	unsigned integer
`Dimension`	fragments
`Default value`	1000
`Valid values`	100 - maxInt
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.4.1.11.5 Element *Scheduling*

This element specifies the scheduling policies used to control the receiver thread of the current Channel.

`Full path`	OpenSplice/NetworkService/Channels/Channel/Receiving/Scheduling
`Occurrences (min-max)`	1 - 1
`Child-elements`	*Element Class* *Element Priority*
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.4.1.11.5.1 Element *Class*

This element specifies the thread scheduling class that will be used by the receiver thread. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/NetworkService/Channels/Channel/Receiving/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	depends on operating system
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.4.1.11.5.2 Element Priority

This element specifies the thread priority that will be used by the receiver thread. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/NetworkService/Channel/Channels/Receiving/Scheduling/Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.5.4.1.11.5.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

`Full path`	OpenSplice/NetworkService/Channel/Channels/Receiving/Scheduling/Priority[@priority_kind]
`Format`	enum
`Dimension`	n/a
`Default value`	Relative
`Valid values`	Relative, Absolute
`Required`	false

3.5.4.1.11.6 Element *CrcCheck*

This configuration element has been added in order to protect OpenSplice network packets from malicious attacks. CRCs (Cyclic Redundancy Checks) are specifically designed to protect against common types of errors on communication channels. When enabled, the integrity of delivered network packets from one DDS process to another is assured. There is a small performance cost to using this feature due to the addtional overhead of carrying out the CRC calculations.

When the receiving side is enabled only network packets that contain a valid `crc` field are accepted.

`Full Path`	OpenSplice/NetworkService/Channel/Channels/Receiving/CrcCheck
`Format`	boolean
`Dimension`	n/a
`Default value`	False
`Valid values`	True, False
`Occurrences (min-max)`	0 - n
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.5 Element *Discovery*

This element is used to configure the various parameters of the Discovery Channel, which is used to discover all participating entities in the current Domain.

The purpose of the discovery process is to build-up and maintain a notion of all relevant active nodes within the domain. The relevance of discovered remote nodes can be defined statically (by definition of the so-called Global Partition) and/or can be dynamically expanded and maintained by the dynamic-discovery process driven by the node's Role (see *3.2.2* on page 27) and Scope (see *3.5.5.1* below).

Full path	OpenSplice/NetworkService/Discovery
Occurrences (min-max)	0 - 1
Child-elements	*Element ProbeList* *Element Active* *Element PortNr* *Element Sending* *Element Receiving*
Required attributes	<none>
Optional attributes	*Attribute Scope*

3.5.5.1 Attribute *Scope*

This attribute controls the dynamic discovery behaviour of this node within the current Domain. If it is not set, dynamic discovery will be disabled and the networking service will only communicate with nodes that can be reached through the predefined Global Partition. If the Scope attribute is specified, dynamic discovery is enabled and the networking service will be able to communicate with all nodes in the system that have a Role (see 3.2.2) that matches the Scope expression. The Scope expression can contain a comma separated list of wild-card role-expressions. If the role of any discovered node matches any of the wild-card expressions, the remote node is considered "a match" and will become part of the communication reach (i.e. the Global Partition) of the current domain.

Full path	OpenSplice/NetworkService/Discovery[@Scope]
Format	String
Dimension	n/a
Default value	
Valid values	any string
Required	false

3.5.5.2 Element *ProbeList*

This element contains the addresses of the nodes that will be contacted to retrieve an initial list of participating nodes in the current domain that match the specified Scope. Multiple ProbeList addresses can be entered by separating them by a colon

(,), semicolon (;) or space( ). The addresses can be entered as dotted decimal notation or as the symbolic `hostname`, in which case the middleware will try to resolve the corresponding IP address

`Full path`	OpenSplice/NetworkService/Discovery/ProbeList
`Format`	String
`Dimension`	n/a
`Default value`	
`Valid values`	any string
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.5.3 Element *Active*

This element can be used to enable or disable the Discovery Channel. In case the Discovery Channel is disabled, entities will only detect each others presence implicitly once messages are received for the first time.

`Full path`	OpenSplice/NetworkService/Discovery/Active
`Format`	boolean
`Dimension`	n/a
`Default value`	true
`Valid values`	true, false
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.5.4 Element *PortNr*

This element specifies the Port number used by the Discovery Channel.

`Full path`	OpenSplice/NetworkService/Discovery/PortNr
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	3369

Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.5.5 Element *Sending*

This element describes all properties for the transmitting side of the Discovery Channel.

Full path	OpenSplice/NetworkService/Discovery/Sending
Occurrences (min-max)	0 - 1
Child-elements	*Element Interval* *Element SafetyFactor* *Element SalvoSize* *Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.5.5.5.1 Element *Interval*

This element describes the interval at which remote nodes will expect heartbeats from this node.

Full path	OpenSplice/NetworkService/Discovery/Sending/Interval
Format	unsigned integer
Dimension	milliseconds
Default value	1000
Valid values	100 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.5.5.2 Element *SafetyFactor*

The SafetyFactor is used to set a margin (< 1) on the expected heartbeat interval. This avoids tight timing issues, since this node will send its heartbeats at a smaller interval than is expected by the remote nodes.

`Full path`	OpenSplice/NetworkService/Discovery/Sending/SafetyFactor
`Format`	float
`Dimension`	n/a
`Default value`	0.9
`Valid values`	0.2 - 1.0
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.5.5.3 Element *SalvoSize*

During starting and stopping, discovery messages are sent at higher frequency. This SalvoSize sets the number of messages to send during these phases.

`Full path`	OpenSplice/NetworkService/Discovery/Sending/SalvoSize
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	3
`Valid values`	1 - maxInt
`Occurrences (min-max)`	0 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	<none>

3.5.5.5.4 Element *Scheduling*

This element specifies the scheduling policies used to control the transmitter thread of the Discovery Channel.

Full path	OpenSplice/NetworkService/Discovery/Sending/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.5.5.5.4.1 Element *Class*

This element specifies the thread scheduling class that will be used by the transmitter thread of the Discovery Channel. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/NetworkService/Discovery/Sending/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.5.5.4.2 Element Priority

This element specifies the thread priority that will be used by the transmitter thread of the Discovery Channel. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

`Full path`	OpenSplice/NetworkService/Discovery/Sending/Scheduling/Priority
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	depends on operating system
`Valid values`	depends on operating system
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	*Attribute priority_kind*

3.5.5.5.4.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

`Full path`	OpenSplice/NetworkService/Discovery/Sending/Scheduling/Priority[@priority_kind]
`Format`	enum
`Dimension`	n/a
`Default value`	Relative
`Valid values`	Relative, Absolute
`Required`	false

3.5.5.6 Element *Receiving*

The Sending element describes all properties for the receiving side of the Discovery Channel.

Full path	OpenSplice/NetworkService/Discovery/Receiving
Occurrences (min-max)	0 - 1
Child-elements	*Element DeathDetectionCount* *Element ReceiveBufferSize* *Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.5.5.6.1 Element *DeathDetectionCount*

This element specifies how often a heartbeat from a remote node must miss its Interval before that remote node is considered dead.

Full path	OpenSplice/NetworkService/Discovery/Receiving/DeathDetectionCount
Format	unsigned integer
Dimension	n/a
Default value	6
Valid values	1 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.5.6.2 Element *ReceiveBufferSize*

The UDP receive buffer of the Discovery Channel socket is set to the value given. If many message are lost, the receive buffer size has to be increased.

Full path	OpenSplice/NetworkService/Discovery/Receiving/ReceiveBufferSize
Format	unsigned integer
Dimension	bytes
Default value	1000000
Valid values	depends on operating system
Occurrences (min-max)	0 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.5.6.3 Element *Scheduling*

This element specifies the scheduling policies used to control the receiver thread of the Discovery Channel.

Full path	OpenSplice/NetworkService/Discovery/Receiving/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.5.5.6.3.1 Element *Class*

This element specifies the thread scheduling class that will be used by the receiver thread of the Discovery Channel. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/NetworkService/Discovery/Receiving/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.5.6.3.2 Element *Priority*

This element specifies the thread priority that will be used by the receiver thread of the Discovery Channel. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

`Full path`	OpenSplice/NetworkService/Discovery/Receiving/Scheduling/Priority
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	depends on operating system
`Valid values`	depends on operating system
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	*Attribute priority_kind*

3.5.5.6.3.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

`Full path`	OpenSplice/NetworkService/Discovery/Receiving/Scheduling/Priority[@priority_kind]
`Format`	enum
`Dimension`	n/a
`Default value`	Relative
`Valid values`	Relative, Absolute
`Required`	false

3.5.6 Element *Tracing*

This element controls the amount and type of information that is written into the tracing log by the Networking Service. This is useful to track the Networking Service during application development. In the runtime system it should be turned off.

Full path	OpenSplice/NetworkService/Tracing
Occurrences (min-max)	0 - 1
Child-elements	*Element OutputFile* *Element Timestamps* *Element Categories*
Required attributes	<none>
Optional attributes	<none>

3.5.6.1 Element *OutputFile*

This option specifies where the logging is printed to. Note that “stdout” is considered a legal value that represents “standard out”. The default value is an empty string, indicating that the tracing log will be written to standard out.

Full path	OpenSplice/NetworkService/Tracing/OutputFile
Format	string
Dimension	file name
Default value	“” (empty string indicating stdout)
Valid values	depends on operating system.
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.2 Element *Timestamps*

This element specifies whether the logging must contain timestamps.

Full path	OpenSplice/NetworkService/Tracing/Timestamps
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute Absolute*

3.5.6.2.1 Attribute *Absolute*

This attribute specifies whether the timestamps are absolute or relative to the startup time of the service.

Full path	OpenSplice/NetworkService/Tracing/Timestamps[@absolute]
Format	boolean
Dimension	
Default value	true
Valid values	true, false
Required	false

3.5.6.3 Element *Categories*

This element contains the logging properties for various networking categories.

Full path	OpenSplice/NetworkService/Tracing/Categories
Occurrences (min-max)	0 - 1
Child-elements	*Element Default* *Element Configuration* *Element Construction* *Element Destruction* *Element Mainloop* *Element Groups* *Element Send* *Element Receive* *Element Test* *Element Discovery*
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.1 Element *Default*

This element specifies the tracing level used for categories that are not explicitly specified. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories/Default
Format	unsigned integer
Dimension	n/a

Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.2 Element *Configuration*

This element specifies the tracing level for the *Configuration* category. This includes the processing of all NetworkService parameters in the config file. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Configuration
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.3 Element *Construction*

This element specifies the tracing level for the *Construction* category. This includes the creation of all internal processing entities. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Construction
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.4 Element *Destruction*

This element specifies the tracing level for the *Destruction* category. This includes the destruction of all internal processing entities when the NetworkService terminates. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Destruction
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.5 Element *Mainloop*

This element specifies the tracing level for the *Mainloop* category. This includes information about each of the threads spawned by the NetworkService. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Mainloop
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.6 Element *Groups*

This element specifies the tracing level for the *Groups* category. This includes the management of local groups (partition-topic combinations). Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Groups
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.7 Element *Send*

This element specifies the tracing level for the *Send* category. This includes information about outgoing data. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Send
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.8 Element *Receive*

This element specifies the tracing level for the *Receive* category. This includes information about incoming data. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Receive
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.9 Element *Test*

This element specifies the tracing level for the *Test* category. This is a reserved category used for testing purposes. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Test
Format	unsigned integer
Dimension	
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.5.6.3.10 Element *Discovery*

This element specifies the tracing level for the *Discovery* category. This includes all activity related to the discovery channel. Level 0 indicates no tracing, level 6 indicates the most detailed level of tracing.

Full path	OpenSplice/NetworkService/Tracing/Categories /Discovery
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 6
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6 The Tuner Service

The TunerService configuration determines how the Tuner Service handles the incoming client connections. It expects a root element named *OpenSplice/TunerService*, in which several child-elements may be specified. Each of these are listed and explained.

Full path	OpenSplice/TunerService
Occurrences (min-max)	0 - *
Child-elements	*Element Client* *Element Server* *Element GarbageCollector* *Element LeaseManagement* *The DbmsConnect Service*
Required attributes	*Attribute name*
Optional attributes	<none>

3.6.1 Attribute *name*

This attribute identifies a configuration for the Tuner Service by name. Multiple Tuner Service configurations can be specified in one single resource file. The actual applicable configuration is determined by the value of the *name* attribute, which must match the one specified under the *OpenSplice/Domain/Service[@name]* in the configuration of the Domain Service.

Full path	OpenSplice/TunerService[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.6.2 Element *Client*

This element determines how the Tuner service handles the incoming client connections.

Full path	OpenSplice/TunerService/Client
Occurrences (min-max)	0 - 1
Child-elements	*Element LeasePeriod* *Element MaxClients* *Element MaxThreadsPerClient* *Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.6.2.1 Element *LeasePeriod*

This element determines the maximum amount of time in which a connected client must update its lease. This can be done implicitly by calling any function or explicitly by calling the update lease function. The Tuner tool will automatically update its lease when it is connected to the Tuner Service. This ensures that all resources are cleaned up automatically if the client fails to update its lease within this period.

Full path	OpenSplice/TunerService/Client/LeasePeriod
Format	float
Dimension	seconds

Default value	15.0
Valid values	10.0 - maxFloat
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.2.2 Element *MaxClients*

This element determines the maximum allowed number of clients that are allowed to be concurrently connected to the Tuner Service. Clients are identified by IP-address.

Full path	OpenSplice/TunerService/Client/MaxClients
Format	unsigned integer
Dimension	n/a
Default value	2
Valid values	1 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.2.3 Element *MaxThreadsPerClient*

This element specifies the maximum number of threads that the Tuner Service will create for one specific client. The number of threads determines the maximum number of concurrent requests for a client.

Full path	OpenSplice/TunerService/Client/ MaxThreadsPerClient
Format	unsigned integer
Dimension	n/a
Default value	2
Valid values	1 - maxInt
Occurrences (min-max)	0 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.2.4 Element *Scheduling*

This element specifies the scheduling policies used to control the threads that handle the client requests to the Tuner Service.

Full path	OpenSplice/TunerService/Client/Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.6.2.4.1 Element *Class*

This element specifies the thread scheduling class that will be used by the threads that handle client requests to the Tuner Service. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/TunerService/Client/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.2.4.2 Element *Priority*

This element specifies the thread priority that will be used by the threads that handle client requests to the Tuner Service. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

`Full path`	OpenSplice/TunerService/Client/Scheduling/ Priority
`Format`	unsigned integer
`Dimension`	n/a
`Default value`	depends on operating system
`Valid values`	depends on operating system
`Occurrences (min-max)`	1 - 1
`Child-elements`	<none>
`Required attributes`	<none>
`Optional attributes`	*Attribute priority_kind*

3.6.2.4.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

`Full path`	OpenSplice/TunerService/Client/Scheduling/ Priority[@priority_kind]
`Format`	enum
`Dimension`	n/a
`Default value`	Relative
`Valid values`	Relative, Absolute
`Required`	false

3.6.3 Element *Server*

This element determines the server side behaviour of the Tuner Service.

Full path	OpenSplice/TunerService/Server
Occurrences (min-max)	0 - 1
Child-elements	*Element Backlog* *Element PortNr* *Element Verbosity*
Required attributes	<none>
Optional attributes	<none>

3.6.3.1 Element *Backlog*

This element determines the maximum number of client requests that are allowed to be waiting when the maximum number of concurrent requests is reached.

Full path	OpenSplice/TunerService/Server/Backlog
Format	unsigned integer
Dimension	n/a
Default value	5
Valid values	0 - maxInt
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.3.2 Element *PortNr*

This element specifies the port number that the TunerService will use to listen for incoming requests. This port number must also be used by the Tuner tool to connect to this service.

Full path	OpenSplice/TunerService/Server/PortNr
Format	unsigned integer
Dimension	n/a
Default value	8000
Valid values	depends on operating system
Occurrences (min-max)	0 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.3.3 Element *Verbosity*

This element specifies the verbosity level of the logging of the service.

Full path	OpenSplice/TunerService/Server/Verbosity
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 5
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.4 Element *GarbageCollector*

This element specifies the behaviour of the garbage collection thread of the service.

Full path	OpenSplice/TunerService/GarbageCollector
Occurrences (min-max)	0 - 1
Child-elements	*Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.6.4.1 Element *Scheduling*

This element specifies the scheduling policies used to control the garbage collection thread of the Tuner Service.

Full path	OpenSplice/TunerService/GarbageCollection/ Scheduling
Occurrences (min-max)	1 - 1

Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.6.4.1.1 Element *Class*

This element specifies the thread scheduling class that will be used by the garbage collection thread of the Tuner Service. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/TunerService/GarbageCollection/Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.4.1.2 Element *Priority*

This element specifies the thread priority that will be used by the garbage collection thread of the Tuner Service. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/TunerService/GarbageCollection/Scheduling/ Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1

Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.6.4.1.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/TunerService/GarbageCollection/ Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.6.5 Element *LeaseManagement*

This element specifies the behaviour of the lease management thread of the Tuner Service.

Full path	OpenSplice/TunerService/LeaseManagement
Occurrences (min-max)	0 - 1
Child-elements	*Element Scheduling*
Required attributes	<none>
Optional attributes	<none>

3.6.5.1 Element *Scheduling*

This element specifies the scheduling policies used to control the lease management thread of the Tuner Service.

Full path	OpenSplice/TunerService/LeaseManagement/ Scheduling
Occurrences (min-max)	1 - 1
Child-elements	*Element Class* *Element Priority*
Required attributes	<none>
Optional attributes	<none>

3.6.5.1.1 Element *Class*

This element specifies the thread scheduling class that will be used by the lease management thread of the Tuner Service. The user may need the appropriate privileges from the underlying operating system to be able to assign some of the privileged scheduling classes.

Full path	OpenSplice/TunerService/LeaseManagement/ Scheduling/Class
Format	enumeration
Dimension	n/a
Default value	Default
Valid values	Timeshare, Realtime, Default
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.6.5.1.2 Element *Priority*

This element specifies the thread priority that will be used by the lease management thread of the Tuner Service. Only priorities that are supported by the underlying operating system can be assigned to this element. The user may need special privileges from the underlying operating system to be able to assign some of the privileged priorities.

Full path	OpenSplice/TunerService/LeaseManagement/ Scheduling/ Priority
Format	unsigned integer
Dimension	n/a
Default value	depends on operating system
Valid values	depends on operating system
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute priority_kind*

3.6.5.1.2.1 Attribute *priority_kind*

This attribute specifies whether the specified *Priority* is a relative or absolute priority.

Full path	OpenSplice/TunerService/LeaseManagement/Scheduling/Priority[@priority_kind]
Format	enum
Dimension	n/a
Default value	Relative
Valid values	Relative, Absolute
Required	false

3.7 The DbmsConnect Service

The DbmsConnect Service configuration is responsible for DDS to DBMS bridging and expects a root element named `OpenSplice/DbmsConnectService`. Within this root element, the DbmsConnect Service will look for several child-elements. Each of these is listed and explained.

Full path	OpenSplice/DbmsConnectService
Occurrences (min-max)	0 - *
Child-elements	*Element DdsToDbms* *Element DbmsToDds* *Element Tracing*
Required attributes	*Attribute name*
Optional attributes	<none>

3.7.1 Attribute *name*

This attribute identifies the configuration for the DBMS Service by name. Multiple DBMS Service configurations can be specified in one single resource file. The actual applicable configuration is determined by the value of the *name* attribute, which must match the one specified under the *OpenSplice/Domain/Service[@name]* in the configuration of the Domain Service.

Full path	OpenSplice/DBMSConnectService[@name]
Format	string
Dimension	n/a

Default value	n/a
Valid values	any string
Required	true

3.7.2 Element *DdsToDbms*

This element specifies the configuration properties concerning DDS to DBMS bridging.

Full path	OpenSplice/DBMSConnectService/DdsToDbms
Occurrences (min-max)	0 - 1
Child-elements	*Element NameSpace*
Required attributes	<none>
Optional attributes	*Attribute replication_mode*

3.7.2.1 Attribute *replication_mode*

This attribute specifies the default replication mode for all NameSpaces in the *DdsToDbms* element.

When replicating databases through DDS, the *NameSpace* elements in the *DbmsToDds* and *DdsToDbms* elements map a Table and Topic circularly. To prevent data-modifications from continuously cascading, modifications made by the DBMSConnect service itself should not trigger new updates in the DBMS nor in the DDS.

In replication mode, the DbmsConnect service ignores samples that were published by itself. (Currently that means that everything that is published on the same node as the DBMSConnect Service is considered to be of DBMSConnect origin and therefore ignored). That way, replication of changes that were copied from Dbms to DDS back into Dbms is avoided.

WARNING: This setting does not avoid replication of changes that were copied from DDS to Dbms back into DDS. For that purpose, the *replication_user* attribute of the *DbmsToDds* or *DbmsToDds/NameSpace* elements should be set appropriately as well!

Full path	OpenSplice/DbmsConnectService/ DdsToDbms[@replication_mode]
Format	boolean
Dimension	n/a
Default value	false

Valid values	true, false
Remarks	none
Required	no

3.7.2.2 Element *NameSpace*

This element specifies the responsibilities of the service concerning the bridging of particular data from DDS to DBMS. At least one *NameSpace* element has to be present in a *DdsToDbms* element.

Full path	OpenSplice/DBMSConnectService/DdsToDbms/NameSpace
Occurrences (min-max)	1 - *
Child-elements	*Element Mapping*
Required attributes	*Attribute dsn* *Attribute usr* *Attribute pwd*
Optional attributes	*Attribute name* *Attribute partition* *Attribute topic* *Attribute schema* *Attribute catalog* *Attribute replication_mode* *Attribute update_frequency* *Attribute odbc*

3.7.2.2.1 Attribute *dsn*

Represents the ODBC Data Source Name, that represents the DBMS where the service must bridge the DDS data to.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@dsn]
Format	string
Dimension	Data source name
Default value	n/a
Valid values	Any valid DSN
Required	true

3.7.2.2.2 Attribute *usr*

Represents the user name that is used when connecting to the DBMS.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@usr]
Format	string
Dimension	n/a
Default value	n/a
Valid values	Any valid username
Required	true

3.7.2.2.3 Attribute *pwd*

Represents the password that is used when connecting to the DBMS.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@pwd]
Format	string
Dimension	n/a
Default value	n/a
Valid values	Any valid password
Required	true

3.7.2.2.4 Attribute *name*

The name of the namespace. If not specified, the namespace will be named "(nameless)".

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@name]
Format	string
Dimension	n/a
Default value	(nameless)
Valid values	Any valid string
Required	false

3.7.2.2.5 Attribute *partition*

This attribute specifies an expression that represents one or more DDS partitions. It is allowed to use wildcards in the expression: a '*' represents any sequence of characters and a '?' represents a single character.

This expression is used to specify the partitions from which DDS samples must be 'bridged' to the DBMS domain.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@partition]
Format	string
Dimension	n/a
Default value	*
Valid values	Any valid DDS partition expression
Required	false

3.7.2.2.6 Attribute *topic*

This attribute specifies an expression that represents one or more DDS topics. It is allowed to use wildcards in the expression: a '*' represents any sequence of characters and a '?' represents a single character.

This expression is used to specify the topics from which DDS samples must be *bridged* to the DBMS domain. For every matching topic encountered in one or more of the specified partitions, it creates a separate table in the DBMS. The table name will match that of the topic, unless specified otherwise in the *table* attribute of a *Mapping* element.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@topic]
Format	string
Dimension	n/a
Default value	*
Valid values	Any valid DDS Topic expression
Required	false

3.7.2.2.7 Attribute *schema*

This attribute represents the schema that is used when communicating with the DBMS. The exact schema content may be dependent on the DBMS that is being used, so consult your DBMS documentation for more details on this subject.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@schema]
Format	string
Dimension	n/a

Default value	“” (empty string)
Valid values	Any valid string
Required	false

3.7.2.2.8 Attribute *catalog*

Represents the catalog that is used when communicating with the DBMS. The exact catalog content may be dependent on the DBMS that is being used, so consult your DBMS documentation for more details on this subject.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@catalog]
Format	string
Dimension	n/a
Default value	“” (empty string)
Valid values	Any valid string
Required	false

3.7.2.2.9 Attribute *replication_mode*

This attribute specifies the replication mode for the current *NameSpace* element. If not specified, the value will be inherited from the *replication_mode* of the parent *DdsToDbms* element, which if not explicitly specified defaults to `false`.

When replicating databases through DDS, the *NameSpace* elements in the *DbmsToDds* and *DdsToDbms* elements map a Table and Topic circularly. To prevent data-modifications from continuously cascading, modifications made by the DBMSConnect service itself should not trigger new updates in the DBMS.

In replication mode, the DbmsConnect service ignores samples that were published by itself. (Currently that means that everything that is published on the same node as the DBMSConnect Service is considered to be of DBMSConnect origin and therefore ignored). That way, replication of changes that were copied from Dbms to DDS back into Dbms is avoided.

WARNING: This setting does not avoid replication of changes that were copied from DDS to Dbms back into DDS. For that purpose, the *replication_user* attribute of the *DbmsToDds* or *DbmsToDds/NameSpace* elements should be set appropriately as well!

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@replication_mode]
Format	boolean
Dimension	
Default value	Inherited from parent *DdsToDbms* element
Valid values	true, false
Required	false

3.7.2.2.10 Attribute *update_frequency*

This attribute specifies the frequency (in Hz) at which the service will update the DBMS domain with DDS data. By default, it is 0.0 which means it is done event based (every time new DDS data arrives).

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@update_factor]
Format	float
Dimension	frequency (Hz)
Default value	0.0
Valid values	0.0 - maxFloat
Required	false

3.7.2.2.11 Attribute *odbc*

The service dynamically loads an ODBC library at runtime. This attribute specifies the name of the ODBC library to be loaded. Platform specific pre- and postfixes and extensions are automatically added.

If this attribute is not provided, the service will attempt to load the generic ODBC library. The resulting behaviour is dependent on the platform on which the DbmsConnect Service is running.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@odbc]
Format	string
Dimension	Library name

Default value	Platform dependent
Valid values	Any valid library name
Required	false

3.7.2.2.12 Element *Mapping*

This element specifies a modification to the way that the service handles a pre-configured set of data within the specified NameSpace. Its attributes are used to configure the responsibilities of the service concerning the bridging of data from DDS to DBMS.

3.7.2.2.12.1 Attribute topic

Full path	OpenSplice/DBMSConnectService/DdsToDbms/ NameSpace/Mapping
Occurrences (min-max)	0 - *
Child-elements	<none>
Required attributes	*Attribute topic*
Optional attributes	*Attribute table* *Attribute query* *Attribute filter*

This attribute specifies the name of the topic where the Mapping applies to.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/ NameSpace/Mapping[@topic]
Format	string
Dimension	Topic name
Default value	n/a
Valid values	Any valid DDS Topic name
Required	true

3.7.2.2.12.2 Attribute table

This attribute specifies an alternative name for the table that must be associated with the Topic. By default the table name is equal to the topic name.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace/Mapping[@table]
Format	string
Dimension	Table name
Default value	The name of the matching topic
Valid values	Any valid DBMS table name
Required	false

3.7.2.2.12.3 Attribute query

This attribute specifies an SQL query expression. Only DDS data that matches the query will be bridged to the DBMS domain. This is realized by means of a DCPS query condition. The default value is an empty string, representing all available samples of the selected topic.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace/Mapping[@query]
Format	string
Dimension	DDS query expression
Default value	“” (empty string representing all data of the specified topic)
Valid values	WHERE clause of an SQL expression
Required	false

3.7.2.2.12.4 Attribute filter

This attribute specifies an SQL content filter. Only DDS data that matches the filter will be bridged to the DBMS domain. This is realized by means of a DCPS ContentFilteredTopic. The default value is an empty string, representing all available samples of the selected topic.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace/Mapping[@filter]
Format	string
Dimension	DDS content filter
Default value	“” (empty string representing all data of the specified topic)
Valid values	WHERE clause of an SQL expression
Required	false

3.7.3 Element *DbmsToDds*

This element specifies the configuration properties concerning DDS to DBMS bridging.

Full path	OpenSplice/DBMSConnectService/DbmsToDds
Occurrences (min-max)	0 - 1
Child-elements	*Element NameSpace*
Required attributes	<none>
Optional attributes	*Attribute event_table_policy* *Attribute publish_initial_data* *Attribute replication_user* *Attribute trigger_policy*

3.7.3.1 Attribute *event_table_policy*

This attribute specifies the default setting of the event table policy for all *NameSpace* elements in the current *DbmsToDds* element.

An event table (sometimes referred to as ‘change table’ or ‘shadow table’) is a support-table that is slaved to an application-table, adding some status flags that are under the control of a trigger mechanism that responds to creation/modification/deletion events in the application-table.

The following policies are currently supported:

- **FULL**: (default) An ‘event table’ will always be created when the service connects, and will always be deleted when the service disconnects. In this mode, the service will replace the table if it already exists.
- **LAZY**: An ‘event table’ will only be created if it is not available when the service connects, and it will not be deleted when the service disconnects.
- **NONE**: An ‘event table’ will neither be created nor deleted by the service. For each specified *NameSpace*, the service will poll for the existence of a consistent table with a frequency specified in the coresponding *update_frequency* attribute. It will start using the table as soon as it is available. With this policy set, no initial data will be published regardless of the value of the applicable *publish_initial_data* attribute.

Full path	OpenSplice/DbmsConnectService/DbmsToDds[@event_table_policy]
Format	enum
Dimension	n/a

Default value	FULL
Valid values	FULL, LAZY, NONE
Required	no

3.7.3.2 Attribute *publish_initial_data*

This attribute specifies the default behaviour with respect to publishing initially available data in the DBMS to the DDS for all *NameSpace* elements in the current *DbmsToDds* element. If not specified, it defaults to `true`. The value of this attribute is ignored when the corresponding *event_table_policy* is set to NONE.

Full path	OpenSplice/DbmsConnectService/ DbmsToDds[@publish_initial_data]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	no

3.7.3.3 Attribute *replication_user*

This attribute specifies the default replication user for all *NameSpace* elements in the current *DdsToDbms* element.

When replicating databases through DDS, the *NameSpace* elements in the *DbmsToDds* and *DdsToDbms* elements map a Table and Topic circularly. To prevent data-modifications from continuously cascading, modifications made by the service itself should not trigger new updates in the DBMS nor in the DDS.

To distinguish between DBMS updates coming from an application and DBMS updates coming from DDS, an extra database user (the replication user) has to be introduced that differs from the application users. That way, replication of changes that were copied from DDS to Dbms back into DDS is avoided. The *replication_user* attribute specifies the user name of that replication user. An empty string (default value) indicates that there is no separate replication user.

WARNING: This setting does not avoid replication of changes that were copied from Dbms to DDS back into Dbms. For that purpose, the *replication_mode* attribute of the *DssToDbms* or *DssToDbms/NameSpace* elements should be set appropriately as well!

Full path	OpenSplice/DbmsConnectService/DbmsToDds[@replication_user]
Format	string
Dimension	n/a
Default value	“” (empty string indicating no replication user)
Valid values	Any valid SQL user name
Required	no

3.7.3.4 Attribute *trigger_policy*

This attribute specifies the default trigger policy for all *NameSpace* elements in the current *DbmsToDds* element.

Triggers are used to to update the event table in case of creation/modification/deletion events on the application-table.

The following policies are currently supported:

- **FULL**: (default) Triggers will always be created when the service connects, and will always be deleted when the service disconnects. In this mode, the service will replace the triggers if they already exists.
- **LAZY**: Triggers will only be created if they are not available when the service connects, and will not be deleted when the service disconnects.
- **NONE**: Triggers will neither be created nor deleted by the service. This allows you to build your own custom triggering mechanism.

Full path	OpenSplice/DbmsConnectService/DbmsToDds[@trigger_policy]
Format	enum
Dimension	n/a
Default value	FULL
Valid values	FULL, LAZY, NONE
Required	no

3.7.3.5 Element *NameSpace*

This element specifies the responsibilities of the service concerning the bridging of data from DBMS to DDS. At least one *NameSpace* element has to be present in a *DbmsToDds* element.

`Full path`	OpenSplice/DBMSConnectService/DbmsToDds/NameSpace
`Occurrences (min-max)`	1 - *
`Child-elements`	*Element Mapping*
`Required attributes`	*Attribute dsn* *Attribute usr* *Attribute pwd*
`Optional attributes`	*Attribute name* *Attribute partition* *Attribute table* *Attribute schema* *Attribute catalog* *Attribute force_key_equality* *Attribute event_table_policy* *Attribute publish_initial_data* *Attribute replication_user* *Attribute trigger_policy* *Attribute update_frequency* *Attribute odbc*

3.7.3.5.1 Attribute *dsn*

Represents the ODBC Data Source Name, that represents the DBMS where the service must bridge the DDS data from.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@dsn]
`Format`	string
`Dimension`	Data source name
`Default value`	n/a
`Valid values`	Any valid DSN
`Required`	true

3.7.3.5.2 Attribute *usr*

Represents the user name that is used when connecting to the DBMS.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@usr]
Format	string
Dimension	Username
Default value	n/a
Valid values	Any valid username
Required	true

3.7.3.5.3 Attribute *pwd*

Represents the password that is used when connecting to the DBMS.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@pwd]
Format	string
Dimension	Password
Default value	n/a
Valid values	Any valid password
Required	true

3.7.3.5.4 Attribute *name*

The name of the namespace. If not specified, the namespace will be named "(nameless)".

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@name]
Format	string
Dimension	n/a
Default value	(nameless)
Valid values	Any valid string
Required	false

3.7.3.5.5 Attribute *partition*

This attribute specifies an expression that represents one or more DDS partitions. It is allowed to use wildcards in the expression: a `'*'` represents any sequence of characters and a `'?'` represents one single character.

This expression is used to specify the DDS partition(s) where DBMS records will be written to as DDS samples by the service.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@partition]
Format	string
Dimension	n/a
Default value	*
Valid values	Any valid DDS partition expression
Required	false

3.7.3.5.6 Attribute *table*

This attribute specifies an expression that represents one or more DBMS tables. It is allowed to use wildcards in the expression: a `'*'` represents any sequence of characters and a `'?'` represents one single character.

This expression is used to specify the tables from which DBMS data must be 'bridged' to the DDS domain.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@table]
Format	string
Dimension	n/a
Default value	*
Valid values	Any Table expression
Required	false

3.7.3.5.7 Attribute *schema*

This attribute represents the schema that is used when communicating with the DBMS. The exact schema content may be dependent on the DBMS that is being used, so consult your DBMS documentation for more details on this subject.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@schema]
Format	string
Dimension	n/a

Default value	“” (empty string)
Valid values	Any valid string
Required	false

3.7.3.5.8 Attribute *catalog*

Represents the catalog that is used when communicating with the DBMS. The exact catalog content may be dependent on the DBMS that is being used, so consult your DBMS documentation for more details on this subject.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@catalog]
Format	string
Dimension	n/a
Default value	“” (empty string)
Valid values	Any valid string
Required	false

3.7.3.5.9 Attribute *force_key_equality*

This attribute specifies the default setting for *Mapping* elements in the current *NameSpace* element with regard to the enforcement of key equality between table and topic definitions. If true, key definitions from the table and topic must match, otherwise key definitions may differ.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@force_key_equality]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	no

3.7.3.5.10 Attribute *event_table_policy*

This attribute specifies the default setting of the event table policy for all *Mapping* elements in the current *NameSpace* element. If not specified, the value will be inherited from the *event_table_policy* of the parent *DbmsToDds* element, which if not explicitly specified defaults to FULL.

An event table (sometimes referred to as ‘change table’ or ‘shadow table’) is a support-table that is slaved to an application-table, adding some status flags that are under the control of a trigger mechanism that responds to creation/modification/deletion events in the application-table.

The following policies are currently supported:

- **FULL**: An ‘event table’ will always be created when the service connects, and will always be deleted when the service disconnects. In this mode, the service will replace the table if it already exists.
- **LAZY**: An ‘event table’ will only be created if it is not available when the service connects, and it will not be deleted when the service disconnects.
- **NONE**: An ‘event table’ will neither be created nor deleted by the service. For each specified *NameSpace*, the service will poll for the existence of a consistent table with a frequency specified in the coresponding *update_frequency* attribute. It will start using the table as soon as it is available. With this policy set, no initial data will be published regardless of the value of the applicable *publish_initial_data* attribute.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@event_table_policy]
`Format`	enum
`Dimension`	n/a
`Default value`	Inherited from parent *DbmsToDds* element
`Valid values`	FULL, LAZY, NONE
`Required`	no

3.7.3.5.11 Attribute *publish_initial_data*

This attribute specifies the default behaviour with respect to publishing initially available data in the DBMS to the DDS for all *Mapping* elements in the current *NameSpace* element. If not specified, the value will be inherited from the *publish_initial_data* of the parent *DbmsToDds* element, which defaults to `true`. The value of this attribute is ignored when the corresponding *event_table_policy* is set to NONE.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@publish_initial_data]
`Format`	boolean
`Dimension`	n/a

Default value	Inherited from parent *DbmsToDds* element
Valid values	true, false
Required	no

3.7.3.5.12 Attribute *replication_user*

This attribute specifies the default replication user for all *Mapping* elements in the current *NameSpace* element. If not specified, the value will be inherited from the *replication_user* of the parent *DbmsToDds* element, which by default has no separate replication user specified.

When replicating databases through DDS, the *NameSpace* elements in the *DbmsToDds* and *DdsToDbms* elements map a Table and Topic circularly. To prevent data-modifications from continuously cascading, modifications made by the service itself should not trigger new updates in the DBMS nor in the DDS.

To distinguish between DBMS updates coming from an application and DBMS updates coming from DDS, an extra database user (the replication user) has to be introduced that differs from the application users. That way, replication of changes that were copied from DDS to Dbms back into DDS is avoided. The *replication_user* attribute specifies the user name of that replication user. An empty string (default value) indicates that there is no separate replication user.

WARNING: This setting does not avoid replication of changes that were copied from Dbms to DDS back into Dbms. For that purpose, the *replication_mode* attribute of the *DssToDbms* or *DssToDbms/NameSpace* elements should be set appropriately as well!

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@replication_user]
Format	string
Dimension	n/a
Default value	Inherited from parent *DbmsToDds* element
Valid values	Any valid SQL user name
Required	no

3.7.3.5.13 Attribute *trigger_policy*

This attribute specifies the default trigger policy for all *Mapping* elements in the current *NameSpace* element. If not specified, the value will be inherited from the *trigger_policy* of the parent *DbmsToDds* element, which if not explicitly specified defaults to FULL.

Triggers are used to to update the event table in case of creation/modification/deletion events on the application-table.

The following policies are currently supported:

- **FULL**: Triggers will always be created when the service connects, and will always be deleted when the service disconnects. In this mode, the service will replace the triggers if they already exists.
- **LAZY**: Triggers will only be created if they are not available when the service connects, and will not be deleted when the service disconnects.
- **NONE**: Triggers will neither be created nor deleted by the service. This allows you to build your own custom triggering mechanism.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@trigger_policy]
`Format`	enum
`Dimension`	n/a
`Default value`	Inherited from parent *DbmsToDds* element
`Valid values`	FULL, LAZY, NONE
`Required`	no

3.7.3.5.14 Attribute *update_frequency*

This attribute specifies the frequency (in Hz) at which the service will update the DDS domain with DBMS data. The default value is 2.0Hz. Event-based updates are not supported. If 0.0Hz is specified, the default of 2.0Hz will be used.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@update_factor]
`Format`	float
`Dimension`	frequency (Hz)
`Default value`	2.0
`Valid values`	0.0 - maxFloat
`Required`	false

3.7.3.5.15 Attribute *odbc*

The service dynamically loads an ODBC library at runtime. This attribute specifies the name of the ODBC library to be loaded. Platform specific pre- and postfixes and extensions are automatically added.

If this attribute is not provided, the service will attempt to load the generic ODBC library. The resulting behaviour is dependent on the platform on which the DbmsConnect Service is running.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@odbc]
Format	string
Dimension	Library name
Default value	Platform dependent
Valid values	Any valid library name
Required	false

3.7.3.5.16 Element *Mapping*

This element specifies a modification to the way that the service handles a pre-configured set of data within the specified NameSpace. Its attributes are used to configure the responsibilities of the service concerning the bridging of data from DBMS to DDS.

Full path	OpenSplice/DBMSConnectService/DbmsToDds/NameSpace/Mapping
Occurrences (min-max)	0 - *
Child-elements	<none>
Required attributes	*Attribute table*
Optional attributes	*Attribute topic* *Attribute query* *Attribute force_key_equality* *Attribute event_table_policy* *Attribute publish_initial_data* *Attribute trigger_policy*

3.7.3.5.16.1 Attribute *table*

This attribute specifies the name of the table where the Mapping applies to.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@table]
Format	string
Dimension	Table name

Default value	n/a
Valid values	Any valid DBMS table name
Required	true

3.7.3.5.16.2 Attribute topic

This attribute specifies an alternative name for the topic that must be associated with the table. By default the topic name is equal to the table name.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@topic]
Format	string
Dimension	Topic name
Default value	The name of the matching table
Valid values	Any valid DDS Topic name
Required	false

3.7.3.5.16.3 Attribute query

Optional SQL query expression. Only DBMS data that matches the query will be bridged to the DDS domain. This is realized by means of a SQL query. The default value is an empty string, representing all available data in the selected table.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@query]
Format	string
Dimension	SQL expression
Default value	“” (empty string representing all data in the specified table)
Valid values	WHERE clause of an SQL expression
Required	false

3.7.3.5.16.4 Attribute *force_key_equality*

This attribute specifies the enforcement of key equality between table and topic definitions. If true, key definitions from the table and topic must match, otherwise key definitions may differ. If not specified, the value will be inherited from the *force_key_equality* of the parent *NameSpace* element, which if not explicitly specified defaults to true.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@force_key_equality]
`Format`	boolean
`Dimension`	n/a
`Default value`	Inherited from parent *NamseSpace* element
`Valid values`	true, false
`Required`	no

3.7.3.5.16.5 Attribute *event_table_policy*

This attribute specifies the event table policy in the current *Mapping* element. If not specified, the value will be inherited from the *event_table_policy* of the parent *NameSpace* element, which if not explicitly specified inherits from the *event_table_policy* of the parent *DbmsToDds* element, which defaults to FULL.

An event table (sometimes referred to as ‘change table’ or ‘shadow table’) is a support-table that is slaved to an application-table, adding some status flags that are under the control of a trigger mechanism that responds to creation/modification/deletion events in the application-table.

The following policies are currently supported:

- **FULL**: An ‘event table’ will always be created when the service connects, and will always be deleted when the service disconnects. In this mode, the service will replace the table if it already exists.
- **LAZY**: An ‘event table’ will only be created if it is not available when the service connects, and it will not be deleted when the service disconnects.

- **NONE**: An ‘event table’ will neither be created nor deleted by the service. For the specified table, the service will poll with a frequency specified in the coresponding *update_frequency* attribute of the parent *NameSpace*. It will start using the table as soon as it is available. With this policy set, no initial data will be published regardless of the value of the applicable *publish_initial_data* attribute.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@event_table_policy]
`Format`	enum
`Dimension`	n/a
`Default value`	Inherited from parent *NamseSpace* element
`Valid values`	FULL, LAZY, NONE
`Required`	no

3.7.3.5.16.6 Attribute *publish_initial_data*

This attribute specifies the behaviour with respect to publishing the initially available data specified in the current *Mapping* element from DBMS to DDS. If not specified, the value will be inherited from the *publish_initial_data* of the parent *NameSpace* element, which if not explicitly specified inherits from the *publish_initial_data* of the parent *DbmsToDds* element, which defaults to `true`. The value of this attribute is ignored when the corresponding *event_table_policy* is set to NONE.

`Full path`	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@publish_initial_data]
`Format`	boolean
`Dimension`	n/a
`Default value`	Inherited from parent *NamseSpace* element
`Valid values`	true, false
`Required`	no

3.7.3.5.16.7 Attribute trigger_policy

This attribute specifies the trigger policy for the current *Mapping* element. If not specified, the value will be inherited from the *trigger_policy* of the parent *NameSpace* element, which if not explicitly specified inherits from the *trigger_policy* of the parent *DbmsToDds* element, which defaults to FULL.

Triggers are used to to update the event table in case of creation/modification/deletion events on the application-table.

The following policies are currently supported:

- **FULL**: Triggers will always be created when the service connects, and will always be deleted when the service disconnects. In this mode, the service will replace the triggers if they already exists.
- **LAZY**: Triggers will only be created if they are not available when the service connects, and will not be deleted when the service disconnects.
- **NONE**: Triggers will neither be created nor deleted by the service. This allows you to build your own custom triggering mechanism.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@trigger_policy]
Format	enum
Dimension	n/a
Default value	Inherited from parent *NameSpace* element
Valid values	FULL, LAZY, NONE
Required	no

3.7.4 Element *Tracing*

This element controls all tracing aspects of the DbmsConnect Service.

Full path	OpenSplice/DbmsConnectService/Tracing
Occurrences (min-max)	0 - 1
Child-elements	*Element OutputFile* *Element Timestamps* *Element Verbosity*
Required attributes	<none>
Optional attributes	<none>

3.7.4.1 Element *OutputFile*

This element specifies where the tracing log is printed to. Note that “stdout” and “stderr” are considered legal values that represent “standard out” and “standard error” respectively. The default value is an empty string, indicating that all tracing is disabled.

Full path	OpenSplice/DbmsConnectService/Tracing/OutputFile
Format	string
Dimension	file name
Default value	“” (empty string indicating no tracing)

Valid values	depends on operating system.
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.7.4.2 Element *Timestamps*

This element specifies whether the logging must contain timestamps.

Full path	OpenSplice/DbmsConnectService/Tracing/Timestamps
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	*Attribute Absolute*

3.7.4.2.1 Attribute *Absolute*

This attribute specifies whether the timestamps are absolute or relative to the startup time of the service.

Full path	OpenSplice/DbmsConnectService/Tracing/Timestamps[@absolute]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	false

3.7.4.3 Element *Verbosity*

This element specifies the verbosity level of the logging.

Full path	OpenSplice/DbmsConnectService/Tracing/Verbosity
Format	enumeration
Dimension	n/a
Default value	INFO
Valid values	SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST
Occurrences (min-max)	0 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.8 The UserClock Service

The UserClock Service allows you to plugin a custom clock library, allowing OpenSplice to read the time from an external clock source. It expects a root element named *OpenSplice/UserClockService*. Within this root element, the userclock will look for several child-elements. Each of these is listed and explained.

Full path	OpenSplice/UserClockService
Occurrences (min-max)	0 - 1
Child-elements	*Element UserClockModule* *Element UserClockStart* *Element UserClockStop* *Element UserClockQuery*
Required attributes	*Attribute name*
Optional attributes	<none>

3.8.1 Attribute *name*

This attribute identifies the configuration for the UserClock Service. The value of the *name* attribute must match the one specified under the *OpenSplice/Domain/Service[@name]* in the configuration of the Domain Service.

Full path	OpenSplice/UserClockService[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.8.2 Element *UserClockModule*

This element specifies the User Clock Service library file. On UNIX like and Windows platforms this will be a shared library. On VxWorks this will be a reallocatable object file. On VxWorks this tag may be empty or discarded if the functions are pre-loaded on the target.

Full path	OpenSplice/UserClockService/UserClockModule
Format	string
Dimension	file name
Default value	n/a
Valid values	dependent on underlying operating system.
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	<none>
Optional attributes	<none>

3.8.3 Element *UserClockStart*

This element specifies if the user clock requires a start function to be called when the process first creates a participant.

Full path	OpenSplice/UserClockService/UserClockStart
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	*Attribute name*
Optional attributes	<none>

3.8.3.1 Attribute *name*

This attribute specifies the name of the start function. This start function should not have any parameters, and needs to return an int that represents 0 if there are no problems, and any other value if a problem is encountered.

Full path	OpenSplice/UserClockService/ UserClockStart[@name]
Format	string
Dimension	function name
Default value	clockStart
Valid values	name of any existing and accessable function
Required	true

3.8.4 Element *UserClockStop*

This element specifies if the user clock requires a stop function to be called when the process deletes the last participant.

Full path	OpenSplice/UserClockService/UserClockStop
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	*Attribute name*
Optional attributes	<none>

3.8.4.1 Attribute *name*

This attribute specifies the name of the stop function. This stop function should not have any parameters, and needs to return an int that represents 0 if there are no problems, and any other value if a problem is encountered.

Full path	OpenSplice/UserClockService/ UserClockStop[@name]
Format	string
Dimension	function name
Default value	clockStop
Valid values	name of any existing and accessable function
Required	true

3.8.5 Element *UserClockQuery*

This element specifies the clock query function.

Full path	OpenSplice/UserClockService/UserClockQuery
Occurrences (min-max)	1 - 1
Child-elements	<none>
Required attributes	*Attribute name*
Optional attributes	<none>

3.8.5.1 Attribute *name*

This attribute specifies the name of the function that gets the current time. This *clockGet* function should not have any parameters, and needs to return the current time as an *os_time* type.

The definition of the *os_time* type can be found in *os_time.h*:

```
typedef struct os_time {
    /** Seconds since 1-jan-1970 00:00 */
    os_timeSec tv_sec;
    /** Count of nanoseconds within the second */
    os_int32 tv_nsec;
    /** os_time can be used for a duration type with the following
        semantics for negative durations: tv_sec specifies the
        sign of the duration, tv_nsec is always possitive and added
        to the real value (thus real value is tv_sec+tv_nsec/10^9,
        for example { -1, 500000000 } is -0.5 seconds) */
} os_time;
```

Full path	OpenSplice/UserClockService/ UserClockQuery[@name]
Format	string
Dimension	function name
Default value	clockGet
Valid values	name of any existing and accessable function
Required	true

3.9 The DDSI Networking Service

The DDSI Networking configuration expects a root element named *OpenSplice/DDSIService*. Within this root element, the networking daemon will look for several child-elements. Each of these child elements is listed and explained in the following sections.

Full path	OpenSplice/DDSIService
Occurrences (min-max)	0 - *
Child-elements	3.9.2, *Element General*, on page 165 3.9.3, *Element Partitioning*, on page 166 3.9.4, *Element Channels*, on page 168 3.9.5, *Element Discovery*, on page 171 3.9.6, *Element Tracing*, on page 173
Required attributes	1
Optional attributes	0

3.9.1 Attribute *name*

This attribute identifies the configuration for the DDSI Networking Service. Multiple Networking Service configurations can be specified in one single resource. The actual applicable configuration is determined by the value of the *name* attribute, which must match the specified under the element *OpenSplice/Domain/Service[@name]* in the Domain Service configuration.

Full path	OpenSplice/DDSIService[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.9.2 Element *General*

The General element describes the networking service as a whole.

Full path	OpenSplice/DDSIService/General
Occurrences (min-max)	0 - 1
Child-elements	3.9.2.1, *Element NetworkInterfaceAddress*, on page 166
Required attributes	0
Optional attributes	0

3.9.2.1 Element *NetworkInterfaceAddress*

This element specifies which network interface card should be used.

Full path	OpenSplice/DDSIService/General/ NetworkInterfaceAddress
Format	string
Dimension	n/a
Default value	"first available"
Valid values	"first available" or any dotted decimal IPv4 address
Occurrences (min-max)	0 - 1
Child-elements	0
Required attributes	0
Optional attributes	0
Remarks	The given interface should have the required capabilities, e.g. broadcasting

Every DDSI Networking Service is bound to only one network interface card (NIC). The card can be uniquely identified by its corresponding IP address. If the value "first available" is entered here, the OpenSplice middleware will try to look up an interface that has the required capabilities.

3.9.3 Element *Partitioning*

This element defines the default partition being used for all data, except the data items being delivered using an dedicated network partition (support for dedicated Network Partitions will be available in later releases).

Full path	OpenSplice/DDSIService/Partitioning
Occurrences (min-max)	0 - 1

Child-elements	3.9.3.1, *Element GlobalPartition*, on page 167
Required attributes	0
Optional attributes	0

The OpenSplice DDSI Networking Service can be configured to use static routing via dedicated multicast capabilities. This requires the underlying operating system and network interfaces to be configured properly.

The multicast mechanism can be considered a simple implementation of the publish and subscribe technique. The DDSI Networking Service can be configured to make use of this by mapping DCPS partitions onto multicast addresses (networking partitions).

Every node is aware of all networking partitions. Using networking configuration, nodes can be disconnected from any networking partition. If a node is connected via a low speed interface, it is not capable of receiving high volume data. If the DCPS partitioning is designed carefully, high volume data is published into a specific partition, which is mapped onto a specific networking partition, which in turn is only connected to those nodes that are capable of handling high volume data.

3.9.3.1 Element GlobalPartition

This Element defines global or default networking partition.

Full path	OpenSplice/DDSIService/Partitioning/ GlobalPartition
Occurrences (min-max)	0 - 1
Child-elements	n/a
Required attributes	0
Optional attributes	1

3.9.3.1.1 Attribute *address*

This element identifies the global partition address.

Full path	OpenSplice/DDSIService/Partitioning/ GlobalPartition[@address]
Format	string
Dimension	n/a
Default value	239.255.0.1
Valid values	any dotted decimal IPv4 address
Required	false

3.9.4 Element *Channels*

The set of channels defines the behaviour of the service concerning priority. By configuring a set of channels, the DDSI Networking Service is able to function as a 'scheduler' for the network bandwidth. It achieves this by using the application-defined DDS QoS policies of the data to select the most appropriate channel to send the data.

Full path	OpenSplice/DDSIService/Channels
Occurrences (min-max)	0 - 1
Child-elements	3.9.4.1, *Element Channel*, on page 168
Required attributes	0
Optional attributes	0

3.9.4.1 Element *Channel*

The DDSI Networking Service will make sure messages with a higher priority precede messages with a lower priority.

Full path	OpenSplice/DDSIService/Channels/Channel
Occurrences (min-max)	0 - *
Child-elements	3.9.4.2, *Element FragmentSize*, on page 170 3.9.4.3, *Element GroupQueueSize*, on page 170 3.9.4.4, *Element PortNr*, on page 171
Required attributes	1
Optional attributes	3

3.9.4.1.1 Attribute *name*

This element uniquely identifies the channel.

Full path	OpenSplice/DDSIService/Channels/Channel[@name]
Format	string
Dimension	n/a
Default value	n/a
Valid values	any string
Required	true

3.9.4.1.2 Attribute *enabled*

This attribute toggles a channel on or off.

Full path	OpenSplice/DDSIService/Channels/Channel[@enabled]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	false

Toggling a channel between enabled and disabled is a quick alternative for commenting out the corresponding lines in the configuration file.

3.9.4.1.3 Attribute *priority*

This attribute sets the transport priority of the channel. Messages sent to the network have a *transport_priority* quality of service value. Selection of a networking channel is based on the priority requested by the message and the priority offered by the channel. The priority settings of the different channels divide the priority range into intervals. Within a channel, messages are sorted in order of priority.

Please note that this priority element does not have any relation to the operating system threading priority.

Full path	OpenSplice/DDSIService/Channels/Channel[@priority]
Format	unsigned integer
Dimension	n/a
Default value	0
Valid values	0 - 1000
Required	false

3.9.4.1.4 Attribute *default*

This attribute indicates whether the channel is used as the default channel in case no other channel offers the quality of service requested by a message.

The networking channels should be configured to match the quality of service settings that are expected to be requested by the applications. It might happen, however, that none of the available channels meets the requested quality of service for a specific message. In that case, the message will be written into the default channel.

Full path	OpenSplice/DDSIService/Channels/Channel[@default]
Format	boolean
Dimension	n/a
Default value	false
Valid values	true, false
Required	false
Remarks	Only one channel is allowed to have this attribute set to true

3.9.4.2 Element *FragmentSize*

This element defines the length of data fragments sent by this channel.

Full path	OpenSplice/DDSIService/Channels/Channel/FragmentSize
Format	unsigned integer
Dimension	n/a
Default value	1200
Valid values	100 - 10000
Occurrences (min-max)	0 - 1
Child-elements	n/a
Required attributes	0
Optional attributes	0

3.9.4.3 Element *GroupQueueSize*

This element defines the length of queue user by the network database reader.

Full path	OpenSplice/DDSIService/Channels/Channel/GroupQueueSize
Format	unsigned integer
Dimension	n/a
Default value	500
Valid values	100 - 10000
Occurrences (min-max)	0 - 1

Child-elements	n/a
Required attributes	0
Optional attributes	0

3.9.4.4 Element *PortNr*

Messages for the channel are sent to the given port number. Each channel needs its own unique port number.

Full path	OpenSplice/DDSIService/Channels/Channel/PortNr
Format	unsigned integer
Dimension	n/a
Default value	7412
Valid values	0 - 32000
Occurrences (min-max)	0 - 1
Child-elements	n/a
Required attributes	0
Optional attributes	0

3.9.5 Element *Discovery*

The Discovery Channel is used to discover all participating entities in the current domain. This element is used to configure the various parameters of the Discovery Channel.

Full path	OpenSplice/DDSIService/Discovery
Occurrences (min-max)	0- 1
Child-elements	3.9.5.2, *Element FragmentSize*, on page 172 3.9.5.3, *Element PortNr*, on page 172
Required attributes	0
Optional attributes	1

3.9.5.1 Attribute *enabled*

This attribute toggles Discovery on or off.

Full path	OpenSplice/DDSIService/Discovery[@enabled]
Format	boolean
Dimension	n/a
Default value	true
Valid values	true, false
Required	false

3.9.5.2 Element *FragmentSize*

This element defines the length of data fragments sent by this channel.

Full path	OpenSplice/DDSIService/Discovery/ FragmentSize
Format	unsigned integer
Dimension	n/a
Default value	1200
Valid values	100 - 10000
Occurrences (min-max)	0 - 1
Child-elements	n/a
Required attributes	0
Optional attributes	0

3.9.5.3 Element *PortNr*

This element specifies the port number used by the Discovery Channel.

Full path	OpenSplice/DDSIService/Discovery/PortNr
Format	unsigned integer
Dimension	n/a
Default value	7400
Valid values	0 - 32000
Occurrences (min-max)	0 - 1
Child-elements	n/a
Required attributes	0
Optional attributes	0

3.9.6 Element *Tracing*

This element controls the amount and type of information that is written into the tracing log by the DDSI service. This is useful to track the DDSI service during application development. In the runtime system it should be turned off.

Full path	OpenSplice/DDSIService/Tracing
Occurrences (min-max)	0 - 1
Child-elements	3.9.6.2, *Element OutputFile*, on page 173
Required attributes	0
Optional attributes	1

3.9.6.1 Attribute *enabled*

This attribute toggles tracing on or off.

Full path	OpenSplice/DDSIService/Tracing[@enabled]
Format	boolean
Dimension	n/a
Default value	false
Valid values	true, false
Required	false

3.9.6.2 Element *OutputFile*

This option specifies where the logging is printed to.

Full path	OpenSplice/DDSIService/Tracing/OutputFile
Format	string
Dimension	file name
Default value	ddsi-tracing-output.log
Valid values	depends on operating system. <stderr> and <stdout> are also supported.
Occurrences (min-max)	0 - 1
Child-elements	0
Required attributes	0
Optional attributes	0

3.9.7 Example Configuration

This section gives a DDSI Network Service example configuration. The example configuration defines a simple DDSI Networking Service.

Under the *OpenSplice/Domain* element (first part of the example configuration) the DDSI Service is defined, e.g., the name of the configuration is given. Under the *OpenSplice/DDSIService* element (second part of the example configuration) the properties of the DDSI Networking Service are configured, e.g., a default channel is defined under the *OpenSplice/DDSIService/Channels* element.

```
<OpenSplice>
    <Domain>
        <Name>OpenSplice ddsi configuration</Name>
        <Database>
            <Size>10000000</Size>
        </Database>
        <Service name="ddsi" enabled="true">
            <Command>ddsi</Command>
        </Service>
    </Domain>

    <DDSIService name="ddsi">
        <General>
            <NetworkInterfaceAddress>first available
            </NetworkInterfaceAddress>
        </General>
        <Channels>
            <Channel name="ch0" enabled="true" default="true"
priority="32">
                <FragmentSize>1200</FragmentSize>
                <PortNr>3340</PortNr>
                <GroupQueueSize>500</GroupQueueSize>
            </Channel>
        </Channels>
         <Discovery enabled="true">
            <FragmentSize>1200</FragmentSize>
            <PortNr>7400</PortNr>
        </Discovery>
        <Partitioning>
            <GlobalPartition Address="239.255.0.1" />
        </Partitioning>
        <Tracing enabled="false">
            <OutputFile>ddsi-tracing-output.log</OutputFile>
        </Tracing>
    </DDSIService>
</OpenSplice>
```

3.10 Example Reference Systems

The OpenSplice middleware can be deployed for different kinds of systems. This section identifies several different systems that will be used as reference systems throughout the rest of this manual. Each needs to be configured differently in order to fit its requirements. The intention of this section is to give the reader an impression of the possible differences in system requirements and the configuration aspects induced.

3.10.1 Zero Configuration System

The OpenSplice middleware comes with a default configuration file that is intended to give a satisfactory out-of-the-box experience. It suits the standard situation of a system containing a handful of nodes and where requirements on data distribution latencies, volumes and determinism are not too demanding.

Starting and running any systems that satisfy these conditions should not be a problem. Nodes can be started and shutdown without any extra configuration because the default discovery mechanism will keep track of the networking topology.

3.10.2 Single Node System

Systems that have to run only on a single node can be down scaled considerably by not starting the networking and durability daemons. The networking daemon is obviously not needed because its responsibility is forwarding data to and from the network, which is not present. The durability daemon is not needed because the OpenSplice libraries themselves are capable of handling durable data on a single node.

With a single node system, the OpenSplice middleware does not have too much influence on application behaviour. The application has full control over its own thread priorities and all OpenSplice activities will be executed under control of the application threads.

One exception on this is the listener thread. This thread is responsible for calling listener functions as described in the DDS specification.

3.10.3 Medium Size Static (Near) Real-time System

Many medium size systems have highly demanding requirements with respect to data distribution latencies, volumes and predictability. Such systems require configuration and tuning at many levels. The OpenSplice middleware will be an important player in the system and therefore is highly configurable in order to meet these requirements. Every section reflects on an aspect of the configuration.

3.10.3.1 High Volumes

The OpenSplice middleware architecture is designed for efficiently transporting many small messages. The networking daemon is capable of packing messages from different writing applications into one networking package. For this, the latency budget quality of service should be switched on. A latency budget allows the middleware to optimise on throughput. Messages will be collected and combined during an interval allowed by the latency budget. This concerns networking traffic only.

A channel that has to support high volumes should be configured to do so. By default, the resolution parameter is set to 50 ms. This means that latency budget values will be truncated to multiples of 50 ms, which is a suitable value. For efficient packing, the FragmentSize should be set to a large value, for example 8000. This means that data will be sent to the network in chunks of 8 kilobytes. A good value for MaxBurstSize depends on the speed of the attached network and on the networking load. If several writers start writing simultaneously at full speed during a longer period, receiving nodes will start losing packets. Therefore, the writers have to be slowed down to a suitable speed.

Note that message with a large latency budget might be overtaken by messages with a smaller latency budget, especially if they are transported via different networking channels.

3.10.3.2 Low Latencies

If messages are to be transported with requirements on their end to end times, a zero latency budget quality of service should be attached. This results in an immediate wake-up of the networking daemon at the moment that the message arrives in a networking queue. For optimal results with respect to end to end latencies, the thread priority of the corresponding networking channel should be higher than the thread priority of the writing application. with the current implementation, a context switch between the writing application and the networking channel is always required. With the correct priorities, the induced latency is minimized.

The value of the Resolution parameter has its consequences for using latency budgets. The networking daemon will ignore any latency budgets that have a value smaller than Resolution.

The effect of sending many messages with a zero latency budget is an increase of CPU load. The increasing number of context switches require extra processing. This quality of service should therefore be consciously used.

3.10.3.3 Responsiveness

Especially with respect to reliable transport over the network, responsiveness is an important aspect. Whenever a reliably sent message is lost on the network, the sending node has to initiate a resend. Since OpenSplice networking uses an acknowledgement protocol, it is the up to the sending side to decide when to resend a message. This behaviour can be tuned.

First of all, the Resolution parameter is important. This parameters gives the interval at which is checked if any messages have to be resent. The RecoveryFactor parameter indicates how many of these checks have to be executed before actually resending a message. If Resolution is scaled down, messages will be resent earlier. If Recovery factor is scaled down, message will be resent earlier as well.

3.10.3.4 Discovery

OpenSplice implements a discovery protocol for discovering other nodes in the system. As long as only one node is present, nothing has to be sent to the network. At the moment that at least two nodes are present, networking starts sending data to the network.