

OpenSplice DDS

Version 3.4

Deployment Guide



OpenSplice DDS

DEPLOYMENT GUIDE



Part Number: OS-DG

Doc Issue 18, 27 May 2008

Copyright Notice

© 2008 PrismTech Limited. All rights reserved.

This document may be reproduced in whole but not in part.

The information contained in this document is subject to change without notice and is made available in good faith without liability on the part of PrismTech Limited or PrismTech Corporation.

All trademarks acknowledged.

A close-up, low-angle photograph of a computer keyboard, focusing on the central and right-hand keys. The keys are white with dark lettering. A white grid pattern is overlaid on the entire image, creating a sense of depth and perspective. The lighting is soft, highlighting the texture of the keys and the grid lines.

CONTENTS

Table of Contents

Preface

About the Deployment Guide	ix
Contacts	x

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	7
	1.3.2.4 OpenSplice Shared Memory Dump Tool	7
	1.3.2.5 OpenSplice Configurator (Beta)	8
	1.3.3 Stopping	8
	1.4 OpenSplice DDS Configuration	8
	1.4.1 Configuration Files	8
	1.4.2 Environment Variables	9
Chapter 2	Service Configuration	11
	2.1 Introduction	11
	2.2 The Domain Service	11
	2.2.1 Configuration	12
	2.2.1.1 Element name	12
	2.2.1.2 Database Parameters	12
	2.2.1.3 Element Size	12
	2.2.1.4 Lease Parameters	13
	2.2.1.5 Service Parameters	14
	2.3 The Durability Service	18
	2.3.1 Configuration	18
	2.3.1.1 Attribute name	18

2.3.1.2	Network Parameters	18
2.3.1.3	Persistent Parameters	25
2.3.1.4	NameSpaces and NameSpace Parameters	26
2.4	The Networking Service	28
2.4.1	Configuration	28
2.4.1.1	General Parameters	29
2.4.1.2	Partitioning Parameters	29
2.4.1.3	Channels and Channel Parameters	32
2.4.1.4	Discovery Parameters	39
2.5	The Tuner Service	43
2.5.1	Configuration	43
2.5.1.1	Client parameters	43
2.5.1.2	Server Parameters	44
2.6	The DBMS Service	45
2.6.1	DdsToDbms Parameters	45
2.6.1.1	NameSpace Parameters	45
2.6.2	DbmsToDds Parameters	49
2.6.2.1	NameSpace Parameters	49
2.7	Example Reference Systems	53
2.7.1	Zero Configuration System	53
2.7.2	Single Node System	54
2.7.3	Medium Size Static (Near) Real-time System	54
2.7.3.1	High Volumes	54
2.7.3.2	Low Latencies	55
2.7.3.3	Responsiveness	55
2.7.3.4	Discovery	55
2.8	Troubleshooting	56
2.8.1	Basic Problems	56
2.8.1.1	Configuration	56
2.8.1.2	Incorrect Format	56
2.8.1.3	Required Value Not Found	56
2.8.2	Shared Memory	56
2.8.2.1	Requested Shared Memory Too Large	56
2.8.2.2	Shared Memory Segment Already Exists	56
2.8.3	Service Liveliness	56
2.8.3.1	Service Will Not Start	56
2.8.3.2	Service Died	56
2.8.4	Networking	56
2.8.4.1	Socket Still In Use	56
2.8.4.2	Routing Table Not Correctly Configured	56
2.8.4.3	Fragment-size Too Large	56
2.8.4.4	Message Rejected	56

2.8.4.5	Receive Buffer Too Small	56
2.9	Complete List of Error Messages	56
2.10	Known problems.	56
<i>Appendix A</i>	Quick Reference	59
	Conventions	59
	Domain Service	61
	Durability Service	65
	Networking Service	70
	Tuner Service.	77
	DbmsConnect Service	79
	Index	87

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

Appendix A, *Quick Reference*, lists all configuration parameters.

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.



Item contains helpful hint or special information.



Information applies to Windows (e.g. NT, 2000, XP) only.



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



C language specific



C++ language specific



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 x, 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.

Corporate Headquarters

PrismTech Corporation
6 Lincoln Knoll Lane
Suite 100
Burlington, MA
01803
USA

Tel: +1 781 270 1177
Fax: +1 781 238 1700

Web: <http://www.prismtech.com>
General Enquiries: info@prismtech.com

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

The background of the slide is a close-up, low-angle photograph of a computer keyboard. The keys are white and slightly worn. A white grid pattern is overlaid on the entire image, creating a sense of depth and perspective. The text is centered in the upper half of the image.

DEPLOYING OPENSPLICE DDS

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 its 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 visa versa

The OpenSplice DDS middleware can be easily configured, on the fly, using its pluggable architecture: the services which are needed can be specified as well as their optimum configuration for the applicable application domain, including networking parameters, and durability levels for example). *Figure 1* shows an overview of the pluggable service architecture of OpenSplice 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 is described below.

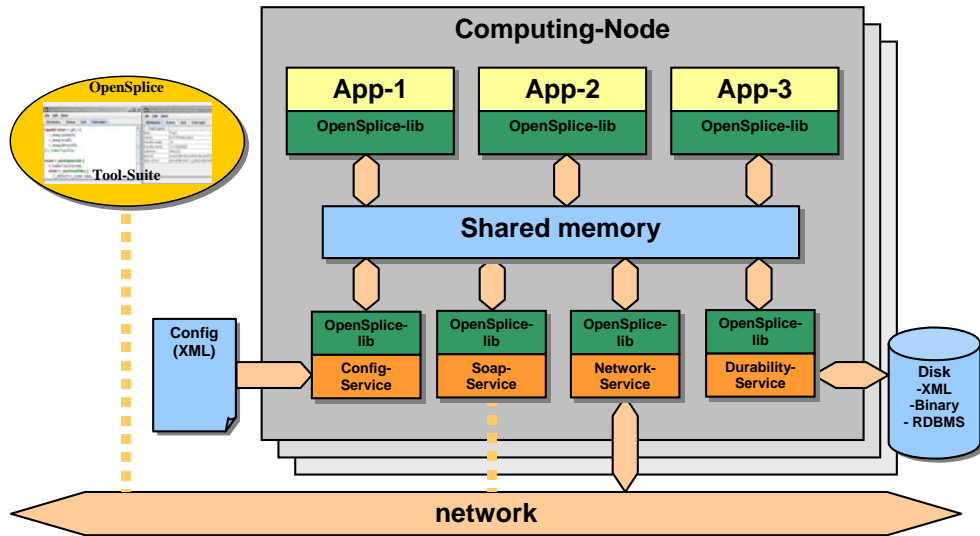


Figure 1 The OpenSplice Single Node Architecture

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. 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 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 are determined by the configuration of the Domain Service. The available configuration parameters with their purpose are discussed in Section 2.2, *The Domain Service*, on page 11.

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 the one 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 are determined by the configuration of the Durability Service. The usage and the available configuration parameters and their purpose are discussed in Section 2.3, *The Durability Service*, on page 18.

1.2.3 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 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 assigned to be able to schedule sending and receiving of specific messages to provide optimal performance for a specific application domain.

The exact fulfilment of these responsibilities are determined by the configuration of the Networking Service. The usage and the available configuration parameters and their purpose are discussed in Section 2.4, *The Networking Service*, on page 28.

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, from any *reachable* location, monitor and control OpenSplice services as well as the applications that use OpenSplice for the distribution of their data.

The exact fulfilment of these responsibilities are determined by the configuration of the Tuner Service. The usage and the available configuration parameters and their purpose are discussed in Section 2.5, *The Tuner Service*, on page 43.

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 *visa 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 are determined by the configuration of the DBMS Service. The DBMS Service's use plus its configuration parameters and purpose are described in Section 2.6, *The DBMS Service*, on page 45

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 this is done by opening a command prompt and executing the `release.bat` file which is located in the root directory of the OpenSplice installation.:

```
% release.bat
```

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.

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 to these files.

The location where the information and error messages are stored can be overridden by setting the *SPLICE_INFOFILE* and *SPLICE_ERRORFILE* variables, respectively, in the environment. They can be set to a location on disk (by specifying a path), to standard out (by specifying *<stdout>*) and to standard error (by specifying *<stderr>*).

1.3.2.2 OpenSplice Tuner

The intention of OpenSplice Tuner, *ospltun*, 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.

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.

1.4 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 supported (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 in 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.

2 Service Configuration

2.1 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 a 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 (for instance 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 described 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.

2.2 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. This section describes the configuration parameters for this service.

2.2.1 Configuration

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

2.2.1.1 Element *name*

This element specifies the name of the instantiated DDS domain.

Full path	OpenSplice/Domain/Name
Format	string
Dimension	Not Available
Default value	“The default Domain”
Valid values	any string
Occurrences (min-max)	1 - 1

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

2.2.1.2 Database Parameters

The Database element contains information about the nodal administration (shared memory) to be used. Currently, it has only one child element.

2.2.1.3 Element *Size*

This element specifies the size of the shared memory database.

Full path	OpenSplice/Domain/Database/Size
Format	unsigned long
Dimension	bytes
Default value	10485670
Valid values	depends on operating system
Occurrences (min-max)	1 - 1

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.

2.2.1.4 Lease Parameters

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

2.2.1.4.1 Element *ExpiryTime*

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

Full path	OpenSplice/Domain/Lease/ExpiryTime
Format	float
Dimension	seconds
Default value	0.2 - ∞
Valid values	none
Occurrences (min-max)	0 - 1

Every OpenSplice DDS service, including the Domain Service itself, has to announce its liveness 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. For increasing it is the other way around.

2.2.1.4.2 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 liveness ‘on time’. The *update_factor* attribute introduces some elasticity in this mechanism by making the services assert their liveness more often than required by the *ExpiryTime*. Services will report their liveness every *ExpiryTime* multiplied by this *update_factor*.

Full path	OpenSplice/Domain/Lease/ExpiryTime[@update_factor]
Format	float
Dimension	Not Available
Default value	0.1
Valid values	0.05 - 0.9
Remarks	none
Required	no

2.2.1.5 Service Parameters

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

2.2.1.5.1 Attribute *enabled*

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

Full path	OpenSplice/Domain/Service[@enabled]
Format	boolean
Dimension	Not Available
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.

2.2.1.5.2 Attribute *name*

This attribute gives the name by which the configuration identifies it.

Full path	OpenSplice/Domain/Service[@name]
Format	string
Dimension	Not Available
Default value	none
Valid values	any string
Required	yes

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

2.2.1.5.3 Element *Command*

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

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

OpenSplice comes with a set of pluggable services. The Command element specifies the name or path to the actual service executable. The Domain Service will search in the current path for the corresponding executable and will start it as a new process.

2.2.1.5.4 Element *Configuration*

This element allows overriding the URI to the configuration resource that is passed on to the service by the Domain Service when it starts it.

Full path	OpenSplice/Domain /Service/Configuration
Format	string
Dimension	URI
Default value	none
Valid values	Any URI to a file resource.
Occurrences (min-max)	0 - 1

The Domain Service passes on the URI where it is started with itself (by the *ospl* tool) to the service when it starts it, by default. This is valid when the configuration of the service is to be found in the same resource as the configuration of the Domain Service (This is the convenient situation in most cases). If the configuration is to be found in another resource the URI that identifies the resource must be specified using this element.

2.2.1.5.5 Element *Class*

This element specifies the type of OS scheduling class will be used by the Domain Service to create the service process.

Full path	OpenSplice/Domain /Service/Scheduling/Class
Format	enumeration
Dimension	Not Available

Default value	Default
Valid values	Timeshare, Realtime, Default
Remarks	The user needs the appropriate privileges to be able to use this option.
Occurrences (min-max)	0 - 1

Services can only be started within the scheduling classes that are supported by the underlying operating system.

2.2.1.5.6 Element *Priority*

This element specifies the process priority that will be used by the Domain Service to start the service.

Full path	OpenSplice/Domain /Service/Scheduling/Priority
Format	natural
Dimension	Not Available
Default value	depends on operating system
Valid values	depends on operating system
Remarks	The user needs the appropriate privileges to be able to use this option.
Occurrences (min-max)	0 - 1

Services can only be started with priorities that are supported by the underlying operating system.

2.2.1.5.7 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	Not Available
Default value	Relative
Valid values	Relative, Absolute
Required	false

2.2.1.5.8 Element *Locking*

Sets the locking policy of the service process, indicating whether to lock pages in physical memory or not.

Full path	OpenSplice/Domain /Service/Locking
Format	boolean
Dimension	
Default value	depends on operating system
Valid values	true, false
Remarks	The user needs the appropriate privileges to be able to use this option.
Occurrences (min-max)	0 - 1

With the 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 started service.

2.2.1.5.9 Element *FailureAction*

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

Full path	OpenSplice/Domain /Service/FailureAction
Format	enumeration
Dimension	
Default value	skip
Valid values	skip, kill, restart, systemhalt
Remarks	none
Occurrences (min-max)	0 - 1

Each service reports its liveness regularly using the shared administration. If the service fails to do so, the Domain Service will assume the service to have 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).

2.3 The Durability Service

2.3.1 Configuration

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.

2.3.1.1 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	Not Available
Default value	-
Valid values	any string
Required	true

2.3.1.2 Network Parameters

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* parameters provide 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.

2.3.1.2.1 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.

Full path	OpenSplice/DurabilityService/Network[@latency_budget]
Format	float
Dimension	Not Available
Default value	0.0
Valid values	Any float
Required	false

latency_budget specifies the maximum acceptable delay 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.

2.3.1.2.2 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.

Full path	OpenSplice/DurabilityService/Network[@transport_priority]
Format	integer
Dimension	Not Available
Default value	0
Valid values	Any integer
Required	false

latency_budget 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.

2.3.1.2.3 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.

2.3.1.2.4 Attribute *latency_budget*

This attribute controls the latency budget QoS setting that is used by the Durability Service for sending its heartbeats.

Full path	OpenSplice/DurabilityService/Network/Heartbeat[@latency_budget]
Format	float
Dimension	Not Available
Default value	0.0
Valid values	Any float
Required	false

It overrides the value of the *DurabilityService/Network[@latency_budget]* for the sending of heartbeats only.

2.3.1.2.5 Attribute *transport_priority*

This attribute controls the transport priority QoS setting that is used by the Durability Service for sending its heartbeats.

Full path	OpenSplice/DurabilityService/Network/Heartbeat[@transport_priority]
Format	integer
Dimension	Not Available
Default value	0
Valid values	Any integer
Required	false

It overrides the value of the *DurabilityService/Network[@transport_priority]* for the sending of heartbeats only.

2.3.1.2.6 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.

Full path	OpenSplice/DurabilityService/Network/Heartbeat/ExpiryTime
Format	float
Dimension	seconds
Default value	5.0
Valid values	1.0 - 20.0
Remarks	none
Occurrences (min-max)	0 - 1

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

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 than required by the *ExpiryTime*. The Durability Service will report its liveness every *ExpiryTime* multiplied by this *update_factor*.

Full path	OpenSplice/DurabilityService/Network/Heartbeat/ExpiryTime[@update_factor]
Format	float
Dimension	
Default value	0.5
Valid values	0.1 - 0.9
Remarks	none
Required	false

2.3.1.2.7 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
Remarks	none
Occurrences (min-max)	0 - 1

2.3.1.2.8 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.

2.3.1.2.9 Attribute *latency_budget*

This attribute controls the latency budget QoS setting that is used by the Durability Service for the alignment of data.

Full path	OpenSplice/DurabilityService/Network/Alignment[@latency_budget]
Format	float
Dimension	Not Available
Default value	0.0
Valid values	Any float
Required	false

It overrules the value of the *DurabilityService/Network[@latency_budget]* for the alignment of data only.

2.3.1.2.10 Attribute *transport_priority*

This attribute controls the transport priority QoS setting that is used by the Durability Service for the alignment of data.

Full path	OpenSplice/DurabilityService/Network/Alignment[@transport_priority]
Format	integer
Dimension	Not Available
Default value	0
Valid values	Any integer
Required	false

It overrides the value of the *DurabilityService/Network[@transport_priority]* for the alignment of data only.

2.3.1.2.11 Element *RequestCombinePeriod*

When the Durability Service detects an inconsistency with another Durability Service, it requests that service to align him. 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 value of this element determines the maximum amount of time the Durability Service is allowed to wait with alignment after an alignment request has been received.

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

It has been split up in a setting that is used during the startup period (Initial) of the Durability Service and one for the period after that (Operational). 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 at more then two computing nodes.

Full path	OpenSplice/DurabilityService/Network/Alignment/RequestCombinePeriod/Operational
Format	float
Dimension	seconds
Default value	0.5
Valid values	0.01 - 5.0
Occurrences (min-max)	0 - 1

2.3.1.2.12 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 Network service is ready to send the data. This element allows specifying what services must be available and how long the Durability Service must wait for them to become available before sending any data.

2.3.1.2.13 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 afterwards, but this indicates a problem in the configuration and the service might not function correctly anymore.

Full path	OpenSplice/DurabilityService/Network/WaitForAttachment[@maxWaitCount]
Format	natural
Dimension	
Default value	200
Valid values	1 - 1000
Remarks	none
Required	false

2.3.1.2.14 Element *ServiceName*

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/Service Name
Format	string
Dimension	
Default value	none
Valid values	any string
Remarks	none
Occurrences (min-max)	0 - *

2.3.1.3 Persistent Parameters

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 parameters are only available as part of the DDS persistence profile of OpenSplice.

2.3.1.3.1 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	

Valid values	depends on operating system
Remarks	none
Occurrences (min-max)	0 - 1

2.3.1.3.2 Element *StoreMode*

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

Full path	OpenSplice/DurabilityService/Persistent/StoreMode
Format	enumeration
Dimension	
Default value	XML
Valid values	XML
Remarks	none
Occurrences (min-max)	0 - 1

2.3.1.4 NameSpaces and Namespace Parameters

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, on one hand driven by application interest, on the other hand driven by fault-tolerance (the need for replicates). This setting controls which historical data is managed by this Durability Service (both transient and persistent).

A namespace describes a dependency between data in two or more partitions by means of a partition expression. The dependency specifies that the data within one of the partitions has no right to exist separately from the data in the other partition(s). Namespaces determine which data must be managed by the Durability Service. Data that does not match any of the namespaces, is ignored by the Durability Service.

Currently the namespaces must be equally configured, except for their alignmentKind settings, which can be configured as desired per node. In the future it will also be possible to configure other namespace parameters per node.

2.3.1.4.1 Attribute *durabilityKind*

This element determines the durability kind of the data that matches the namespace.

Persistent - Only persistent data matches the namespace.

Transient - Only transient data matches the namespace.

Transient_Local - Only transient local data matches the namespace.

Durable - All durable data (persistent, transient and transient local) matches the namespace.

Full path	OpenSplice/DurabilityService/NameSpaces/NameSpace[@durabilityKind]
Format	enumeration
Dimension	
Default value	Durable
Valid values	Durable, Persistent, Transient, Transient_Local
Remarks	none
Required	false

2.3.1.4.2 Attribute *alignmentKind*

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). The durability service provides the following mechanisms to request and provide historical data:

Initial_And_Aligner - 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. The Durability Service will also provide historical data to other Durability Services on a request basis.

Initial - The same as *Initial_And_Aligner*, except the Durability Service will not provide other Durability Services with historical data. This is useful when applications on the current node need all the processing and network bandwidth and dynamic interference is not allowed.

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.

Full path	OpenSplice/DurabilityService/Namespace/namespace[@alignmentKind]
Format	enumeration
Dimension	
Default value	Initial_And_Aligner
Valid values	Initial_And_Aligner, Initial, Lazy
Remarks	none
Required	false

2.3.1.4.3 Element *Partition*

This element specifies a partition expression that matches the namespace. A namespace consists of a set of partition expressions. Together they determine the partitions that belong to the namespace. Make sure the different namespaces do not have an overlap in partitions. The default configuration has one namespace containing all partitions. A partition may contain the wildcards '*' to match any number of characters and '?' to match one single character.

Full path	OpenSplice/DurabilityService/Namespace/Partition
Format	string
Dimension	
Default value	none
Valid values	any string
Remarks	none
Occurrences (min-max)	0 - *

2.4 The Networking Service

2.4.1 Configuration

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

2.4.1.1 General Parameters

General parameters concern the networking service as a whole.

2.4.1.1.1 Element *NetworkInterfaceAddress*

This element specifies which network interface card should be used.

Full path	OpenSplice/NetworkService/General/NetworkInterfaceAddress
Format	string
Dimension	
Default value	"first available"
Valid values	"first available" or any dotted decimal IPv4 address
Remarks	The given interface should have the required capabilities, e.g. broadcasting

Every 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.

2.4.1.2 Partitioning Parameters

The OpenSplice Networking Service can be configured to use multicast capabilities. This requires the underlying operating system and network to be configured properly.

The multicast mechanism can be considered a simple implementation of the publish and subscribe technique. The 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 on its turn is mapped onto a specific networking partition, which on its turn is only connected to those nodes that are capable of handling high volume data.

2.4.1.2.1 Element *GlobalPartition*

This element specifies the global or default networking partition.

Attribute Address

This attribute identifies the broadcast or multicast address to be used for global communication elements.

The 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[@Address]
Format	dotted decimal IPv4 address
Dimension	
Default value	"broadcast"
Valid values	"broadcast" or any dotted decimal IPv4 class D address
Remarks	The given interface should have the required capabilities, e.g. broadcasting or multicasting

2.4.1.2.2 Element *NetworkPartitions* and *NetworkPartition*

Networking configuration can contain a set of networking partitions, which are grouped under the *NetworkPartitions* element. Every *NetworkPartition* has a name, an address and a connected flag.

Attribute Name

A networking partition is uniquely identified by its name.

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

Attribute Address

Each networking partition is mapped onto either the broadcast address or a multicast address.

Full path	OpenSplice/NetworkService/Partitioning/NetworkPartitions/NetworkPartition[@Address]
Format	dotted decimal IPv4 address
Dimension	
Default value	"broadcast"
Valid values	"broadcast" or any dotted decimal IPv4 class D address
Remarks	The time to live parameter currently has a fixed value of 16.

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	
Default value	true
Valid values	true, false
Remarks	

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.

2.4.1.2.3 Element *PartitionMappings* and *PartitionMapping*

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.

Attribute PartitionTopic

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, which is given by the following attribute.

Full path	OpenSplice/NetworkService/Partitioning/PartitionMappings/PartitionMapping[@DCPSPartitionTopic]
Format	PartitionExpression.TopicExpression
Dimension	
Default value	none
Valid values	Expressions containing * wildcards
Remarks	

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	
Default value	none
Valid values	Any name of a previously defined networking partition
Remarks	

2.4.1.3 Channels and Channel Parameters

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.

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.

2.4.1.3.1 Attribute *name*

This element uniquely identifies the channel.

Full path	OpenSplice/NetworkService/Channels/Channel[@name]
Format	string
Dimension	
Default value	none
Valid values	any string
Remarks	This is a required attribute

2.4.1.3.2 Attribute *enabled*

This attribute toggles a channel on or off.

Full path	OpenSplice/NetworkService/Channels/Channel[@enabled]
Format	boolean
Dimension	
Default value	false
Valid values	true, false
Remarks	none

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

2.4.1.3.3 Attribute *reliable*

Set this attribute to true for channel that communicates reliably.

Full path	OpenSplice/NetworkService/Channels/Channel[@reliable]
Format	boolean
Dimension	
Default value	false
Valid values	true, false
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.

2.4.1.3.4 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.

Please note that this Priority element does not have any relation to the operating system threading priority.

Full path	OpenSplice/NetworkService/Channels/Channel[@priority]
Format	natural
Dimension	
Default value	0
Valid values	0 - maxInt
Remarks	none

2.4.1.3.5 Attribute *default*

This attribute indicates whether the channel is used as the default channel at the moment that 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	
Default value	false
Valid values	true, false
Remarks	Only one channel is allowed to have this attribute set to true

2.4.1.3.6 Element *PortNr*

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 *PortNr* + 1.

Full path	OpenSplice/NetworkService/Channels/Channel/PortNr
Format	natural
Dimension	
Default value	3367
Valid values	depends on operating system
Remarks	none

2.4.1.3.7 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	natural
Dimension	bytes
Default value	1300
Valid values	200 - 65536 (if OS allows it)
Remarks	none

2.4.1.3.8 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.

Full path	OpenSplice/NetworkService/Channels/Channel/Resolution
Format	natural
Dimension	milliseconds
Default value	50
Valid values	10 - MaxInt
Remarks	none

Full path	OpenSplice/NetworkService/Channels/Channel/Multicast[@enabled]
Format	boolean
Dimension	
Default value	false
Valid values	true, false
Remarks	none

Full path	OpenSplice/NetworkService/Channels/Channel/Multicast/Address
Format	dotted decimal IPv4 address
Dimension	
Default value	230.0.0.1
Valid values	IPv4 class D address
Remarks	none

Full path	OpenSplice/NetworkService/Channels/Channel/Multicast/TimeToLive
Format	natural
Dimension	hops
Default value	32
Valid values	1 - 256
Remarks	none

2.4.1.3.9 Sending Parameters

Element QueueSize

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/QueueSize
Format	natural
Dimension	messages
Default value	4000
Valid values	1 - maxInt
Remarks	none

Element MaxBurstSize

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/MaxBurstSize
Format	natural
Dimension	bytes/(resolution interval)
Default value	10000
Valid values	
Remarks	none

Element MaxRetries

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/MaxRetries
Format	natural
Dimension	
Default value	10
Valid values	1 - maxInt
Remarks	none

Element RecoveryFactor

Full path	OpenSplice/NetworkService/Channels/Channel/RecoveryFactor
Format	natural
Dimension	
Default value	5
Valid values	1 - maxInt
Remarks	none

Element DiffServField

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/DiffServField
Format	natural
Dimension	

Default value	0
Valid values	0 - 255
Remarks	none

2.4.1.3.10 Receiving Parameters

Element ReceiveBufferSize

Full path	OpenSplice/NetworkService/Channels/Channel/Receiving/ReceiveBufferSize
Format	natural
Dimension	bytes
Default value	65500
Valid values	depends on operating system
Remarks	none

2.4.1.3.11 Scheduling Parameters

Element Priority

Full paths	OpenSplice/NetworkService/Channels/Channel/Sending/Scheduling/Priority OpenSplice/NetworkService/Channels/Channel/Receiving/Scheduling/Priority
Format	natural
Dimension	
Default value	depends on operating system
Valid values	depends on operating system
Remarks	none

Element Class

Full path	OpenSplice/NetworkService/Channels/Channel/Sending/Scheduling/Class OpenSplice/NetworkService/Channels/Channel/Receiving/Scheduling/Class
Format	enumeration
Dimension	

Default value	depends on operating system
Valid values	default, timeshare, realtime
Remarks	none

2.4.1.4 Discovery Parameters

Element Active

Full path	OpenSplice/NetworkService/Discovery/Active
Format	boolean
Dimension	
Default value	true
Valid values	true, false
Remarks	none

2.4.1.4.1 Element PortNr

Full path	OpenSplice/NetworkService/Discovery/PortNr
Format	natural
Dimension	
Default value	3367
Valid values	depends on operating system
Remarks	none

Full path	OpenSplice/NetworkService/Discovery/Multicast[@enabled]
Format	boolean
Dimension	
Default value	false
Valid values	true, false
Remarks	none

Full path	OpenSplice/NetworkService/Discovery/Multicast/Address
Format	dotted decimal IPv4 address
Dimension	
Default value	230.0.0.1
Valid values	IPv4 class D address
Remarks	none

Full path	OpenSplice/NetworkService/Discovery/Multicast/TimeToLive
Format	natural
Dimension	hops
Default value	32
Valid values	1 - 256
Remarks	none

2.4.1.4.2 Sending Parameters

Element Interval

Full path	OpenSplice/NetworkService/Discovery/Sending/Interval
Format	natural
Dimension	milliseconds
Default value	1000
Valid values	100 - maxInt
Remarks	none

Element SafetyFactor

Full path	OpenSplice/NetworkService/Discovery/Sending/SafetyFactor
Format	float
Dimension	

Default value	0.9
Valid values	0.2 - 1.0
Remarks	none

Element SalvoSize

Full path	OpenSplice/NetworkService/Discovery/Sending/SalvoSize
Format	natural
Dimension	
Default value	3
Valid values	1 - maxInt
Remarks	none

Element DiffServField

Full path	OpenSplice/NetworkService/Discovery/Sending/DiffServField
Format	natural
Dimension	
Default value	0
Valid values	0 - 255
Remarks	none

2.4.1.4.3 Receiving Parameters

Element DeathDetectionCount

Full path	OpenSplice/NetworkService/Discovery/Receiving/DeathDetectionCount
Format	natural
Dimension	bytes
Default value	65500
Valid values	depends on operating system
Remarks	none

Element ReceiveBufferSize

Full path	OpenSplice/NetworkService/Discovery/Receiving/ReceiveBufferSize
Format	natural
Dimension	bytes
Default value	65500
Valid values	depends on operating system
Remarks	none

2.4.1.4.4 Scheduling Parameters*Element Priority*

Full paths	OpenSplice/NetworkService/Discovery/Sending/Scheduling/PriorityOpenSplice/NetworkService/Discovery/Receiving/Scheduling/Priority
Format	natural
Dimension	
Default value	depends on operating system
Valid values	depends on operating system
Remarks	none

Element Schedule

Full path	OpenSplice/NetworkService/Discovery/Sending/Scheduling/ScheduleOpenSplice/NetworkService/Discovery/Receiving/Scheduling/Priority
Format	enumeration
Dimension	
Default value	depends on operating system
Valid values	default, timeshare, realtime
Remarks	none

2.5 The Tuner Service

2.5.1 Configuration

2.5.1.1 Client parameters

These parameters determine how the Tuner Service handles the incoming client connections.

2.5.1.1.1 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	10.0
Valid values	10.0 - maxFloat
Remarks	none
Occurrences (min-max)	0 - 1

2.5.1.1.2 Element *MaxClients*

This element determines 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	natural
Dimension	
Default value	2
Valid values	1 - maxInt
Remarks	none
Occurrences (min-max)	0 - 1

2.5.1.1.3 Element *MaxThreadsPerClient*

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	natural
Dimension	
Default value	2
Valid values	1 - maxInt
Remarks	none
Occurrences (min-max)	0 - 1

2.5.1.2 Server Parameters

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

2.5.1.2.1 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	natural
Dimension	
Default value	5
Valid values	0 - maxInt
Remarks	none
Occurrences (min-max)	0 - 1

2.5.1.2.2 Element *PortNr*

This element determines 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	natural
Dimension	

Default value	8000
Valid values	depends on operating system
Remarks	none

2.5.1.2.3 Element *Verbosity*

Determines the verbosity level of the logging of the service.

Full path	OpenSplice/TunerService/Server/Verbosity
Format	natural
Dimension	
Default value	0
Valid values	0 - maxInt
Remarks	none

2.6 The DBMS Service

2.6.1 *DdsToDbms* Parameters

This element holds the configuration of the service concerning DDS to DBMS bridging.

2.6.1.1 *NameSpace* Parameters

This element is used to configure the responsibilities of the service concerning the bridging of data from DDS to DBMS.

2.6.1.1.1 Attribute *Partition*

This attribute is an expression, with wildcards, that represents one or more DDS partitions. 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	

Default value	*
Valid values	Any valid DDS partition expression
Remarks	None

2.6.1.1.2 Attribute *Topic*

This is an expression, with wildcards, that represents one or more DDS topics. 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.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@topic]
Format	String
Dimension	
Default value	*
Valid values	Any Topic expression
Remarks	None

2.6.1.1.3 Attribute *update_frequency*

Optional attribute that determines the frequency in 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_frequency]
Format	float
Dimension	seconds
Default value	0.0
Valid values	0.0 - maxFloat
Remarks	None

2.6.1.1.4 Attribute *odbc*

The service dynamically loads an ODBC library at runtime. This attribute determines 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.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@odbc]
Format	String
Dimension	Library name
Default value	Platform dependent
Valid values	Any library name
Remarks	None

2.6.1.1.5 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	""
Valid values	Any valid DSN
Remarks	None

2.6.1.1.6 Attribute *usr*

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

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@usr]
Format	String
Dimension	Username
Default value	""
Valid values	Any valid username
Remarks	None

2.6.1.1.7 Attribute *pwd*

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

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace[@pwd]
Format	String
Dimension	Password
Default value	""
Valid values	Any valid password
Remarks	None

2.6.1.1.8 Element *Mapping*

A Mapping allows modification of the way that the service handles a specific topic that matches the NameSpace.

Attribute topic

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	
Valid values	Any valid DDS Topic name
Remarks	None

Attribute table

Optional name of the table that must be associated with the Topic. This means the service will map the specified topic to the specified table and will also store data samples in this table as well. 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	
Valid values	Any valid DBMS table name
Remarks	None

Attribute query

Optional DDS 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.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace/Mapping[@query]
Format	String
Dimension	DDS query expression
Default value	
Valid values	WHERE clause of an SQL expression
Remarks	None

Attribute filter

Optional DDS content filter. Only DDS data that matches the query will be bridged to the DBMS domain. This is realized by means of a DCPS content filterd topic.

Full path	OpenSplice/DbmsConnectService/DdsToDbms/NameSpace/Mapping[@filter]
Format	String
Dimension	DDS content filter
Default value	
Valid values	DDS content filtered expression
Remarks	None

2.6.2 DbmsToDds Parameters

Element that holds the configuration of the service concerning DBMS to DDS bridging.

2.6.2.1 NameSpace Parameters

Element that is used to configure the responsibilities of the service concerning the bridging of data from DBMS to DDS.

2.6.2.1.1 Attribute *partition*

An expression, with wildcards, that represents one or more DDS partitions. A * represents any sequence of characters and a "?" represents one single character.

This expression is used to specify the 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	
Default value	*
Valid values	Any valid DDS partition expression
Remarks	None

2.6.2.1.2 Attribute *table*

An expression, with wildcards, that represents one or more DBMS tables. A * represents any sequence of characters and a "?" represents one single character.

This expression is used to specify the tables from which records must be 'bridged' to the DDS domain.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@table]
Format	String
Dimension	
Default value	*
Valid values	Any Table expression
Remarks	None

2.6.2.1.3 Attribute *update_frequency*

Optional attribute that determines the frequency in which the service will update the DDS domain with DBMS data.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@update_frequency]
Format	float
Dimension	seconds

Default value	0.5
Valid values	0.5 - maxFloat
Remarks	None

2.6.2.1.4 Attribute *odbc*

The service dynamically loads an ODBC library at runtime. This attribute determines 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.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace[@odbc]
Format	String
Dimension	Library name
Default value	Platform dependent
Valid values	Any library name
Remarks	None

2.6.2.1.5 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/DbmsToDds/NameSpace[@dsn]
Format	String
Dimension	Data source name
Default value	""
Valid values	Any valid DSN
Remarks	None

2.6.2.1.6 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	""
Valid values	Any valid username
Remarks	None

2.6.2.1.7 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	
Valid values	Any valid password
Remarks	None

2.6.2.1.8 Element *Mapping*

A Mapping allows modification of the way that the service handles a specific table that matches the NameSpace.

Attribute table

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@table]
Format	String
Dimension	Table name
Default value	
Valid values	Any valid DBMS table name
Remarks	None

Attribute topic

The name of the table where the Mapping applies to.

Optional name of the Topic that must be associated with the table. This means the service will map the specified table to the specified topic and will also publish data samples using this Topic as well. By default the table name is equal to the topic name.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@topic]
Format	String
Dimension	Topic name
Default value	
Valid values	Any valid DDS Topic name
Remarks	None

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.

Full path	OpenSplice/DbmsConnectService/DbmsToDds/NameSpace/Mapping[@query]
Format	String
Dimension	SQL expression
Default value	
Valid values	WHERE clause of an SQL expression
Remarks	None

2.7 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.

2.7.1 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 be no problem. Nodes can be started and shutdown without any extra configuration because the default discovery mechanism will keep track of the networking topology.

2.7.2 Single Node System

Systems that have to run on a single node only 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.

2.7.3 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.

2.7.3.1 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 epitomise 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 correctly 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.

2.7.3.2 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.

2.7.3.3 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 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 parameter 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.

2.7.3.4 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.

2.8 Troubleshooting

Configuring the OpenSplice middleware sometimes requires intimate knowledge of the exact meaning of the parameters. Wrong configuration might lead to un-expected problems. During the years, specific problems have appeared to occur frequently. This section gives an overview of well-known problems and their solutions.

2.8.1 Basic Problems

2.8.1.1 Configuration

2.8.1.2 Incorrect Format

2.8.1.3 Required Value Not Found

2.8.2 Shared Memory

2.8.2.1 Requested Shared Memory Too Large

2.8.2.2 Shared Memory Segment Already Exists

2.8.3 Service Liveliness

2.8.3.1 Service Will Not Start

2.8.3.2 Service Died

2.8.4 Networking

2.8.4.1 Socket Still In Use

2.8.4.2 Routing Table Not Correctly Configured

2.8.4.3 Fragment-size Too Large

2.8.4.4 Message Rejected

2.8.4.5 Receive Buffer Too Small

2.9 Complete List of Error Messages

2.10 Known problems

A close-up, low-angle photograph of a computer keyboard, focusing on the central and right-hand keys. The keys are white with dark lettering. A white grid pattern is overlaid on the entire image, creating a sense of depth and perspective. The lighting is soft, highlighting the texture of the keys and the grid lines.

APPENDICES

Appendix



Quick Reference

This appendix lists all configuration parameters and variables mentioned in this guide.

Conventions


The conventions used in Appendix A are:

- Any XML code fragments shown here contain valid structures and are shown as they would appear in the configuration file. However, the actual contents sometimes are symbolic (see below).
 - The symbolic phrase *no default* indicates that a default is not available and is therefore not applicable.
 - The symbolic phrase *OS-specific* indicates that every operating systems has its own default. The defaults are given in a separate section.
- A configuration table gives a description of every setting for every service. Each column in the tables use the meanings defined in Table 1, *Definitions*.

Table 1 Definitions

Term	Meaning
Name	The name of the setting is equal to the XML tag as used in the instantiation file. A name containing the character '#' indicates an attribute of the element preceding the '#' character. The name after the '#' character is the name of the attribute.
Path	The path represents the location of the setting in the XML tree in XPath notation. It is given relative to the splice root node <i>//OpenSplice</i> . The notation [<i>@attribute</i>] means that the setting is specified as an attribute of a tag.

Table 1 Definitions (Continued)

Term	Meaning
Default	The default value is the value used when the parameter is not set or not correctly set. A parameter is not correctly set if its format does not match the expected format. In this situation, a warning is issued mentioning the offending setting and the default value used. When the cell contains 'Not Available', no default value is specified, implying that the parameter is not optional. The default value 'OS' indicates that the default value is platform specific.
Format	<p>The format indicates which format the setting value should correspond to. Possible formats are</p> <p><i>Natural</i> - non-negative integer values <i>Float</i> - real (floating point) values <i>Boolean</i> - true or false values, case insensitive <i>String</i> - any string value.</p> <p> Special characters in strings must be used with care, since the <![CDATA[...]]> syntax is currently not supported.</p>
Valid values	If possible, the valid range of values is printed. For some values, this setting depends on the operating system or its con-figuration. In that case, the validity is unspecified.
Corresponding (C)	Any service that specifies this column requires some parameters to be consistently defined over all nodes. For example, all nodes need to agree on which UDP port to use.
Meaning	The last column briefly explains the meaning of the setting.

Domain Service

The following XML definition shows the default values of all configuration parameters for the spliced daemon. The default value is used for a parameter if it is not completely or not correctly set in the configuration.

```
<OpenSplice>
  <Domain>
    <Name>c_base_A02</Name>
    <Lease>
      <ExpiryTime update_factor="0.5">5.0</ExpiryTime>
    </Lease>
    <Database>
      <Size>10485670</Size>
    </Database>
  </Domain>
</OpenSplice>
```

By default, no services are specified. If a service is specified, the following defaults hold for the different tags.

```
<OpenSplice>
  <Domain>
    <Service name="(no default)">
      <Command>(no default)</Command>
      <Configuration>(no default)</Configuration>
      <Scheduling>
        <Priority>(OS-specific)</Priority>
        <Class>(OS-specific)</Class>
      </Scheduling>
      <Locking>(OS-specific)</Locking>
      <FailureAction>skip</FailureAction>
    </Service>
  </Domain>
</OpenSplice>
```

Table 2 gives a brief description of all Domain Service configuration parameters. The values for **Path** are relative to the root node `//OpenSplice/Domain`.

Table 2 Domain Service Configuration Parameters

Name	Path	Default	Format	Valid Values	Meaning
[@Name]		"The default Domain"	String	Currently only the default is supported.	Specifies the name of the domain that the Domain Service must create.
Size	Database	1048567	Natural	The maximum possible value depends on the user privileges.	Specifies the size of the splice database. On most platforms you need 'root' privileges to set large sizes.
ExpiryTime	Lease	20.0	Float	0.2 - ∞	The maximum interval allowed between two liveliness assertions from any service. If this interval has exceeded, the service will be declared dead.
ExpiryTime[@update_factor]	Lease	0.5	Float	0.1 - 1.0	A factor specifying the actual period between two liveliness assertions. This introduces a margin that allows heartbeats to get lost without immediate consequences.
ServiceTerminatePeriod		10.0	Float	1.0 - 60.0	Determines the maximum amount of time in seconds that the Domain Service will wait for the other services to terminate elegantly before stopping them by force when it is instructed to stop itself.
The values for Path are relative to the root node <code>//OpenSplice/Domain</code> .					

Table 2 Domain Service Configuration Parameters (Continued)

Name	Path	Default	Format	Valid Values	Meaning
Command	Service	N.A	String	Name or path to executable of the service	When no path is given the splice service will search for the executable using the PATH environment variable. The splice service will always check the file for read and execute permissions.
Configuration	Service	N.A	String	Currently only file:// URI's	The configuration URI used by the service to read its configuration parameters
Class	Service/Scheduling	OS	String	timeshare, realtime	Sets the scheduling policy for the service process. The user needs the appropriate privileges to be able to use this option.
Locking	Service	OS	Boolean	true, false	Sets the locking policy of the service process, indicating whether to lock pages in physical memory or not. The user needs the appropriate privileges to be able to use this option.
The values for <i>Path</i> are relative to the root node <i>//OpenSplice/Domain</i> .					

Table 2 Domain Service Configuration Parameters (Continued)

Name	Path	Default	Format	Valid Values	Meaning
Priority	Service/Scheduling	OS	Natural	OS dependent	Sets the process priority of the service. The user needs the appropriate privileges to be able to use this option.
Priority[@priority_kind]	Service/Scheduling/ Priority	OS	String	relative, absolute	Specifies whether the specified priority is absolute or relative.
FailureAction	Service	"skip"	"String"	"skip", "kill", "restart", "systemhalt"	Tells the splice service what action to take when the service dies. "skip": no action, "kill": send SIGKILL to service, "re-start": start service again, "systemhalt": splice service terminates as all other running services.
The values for <i>Path</i> are relative to the root node <i>//OpenSplice/Domain</i> .					

Durability Service

The following XML definition shows all possible durability settings and their defaults.

```
<OpenSplice>
  <!-- Domain Part -->
  <Domain>
    <Service name="durability">
      <Command>durability</Command>
      <Configuration></Configuration>
    </Service>
  </Domain>

  <!-- Durability Service Part -->
  <DurabilityService name="durability">
    <Network latency_budget="0.0" transport_priority="0">
      <Heartbeat latency_budget="0.0" transport_priority="0">
        <ExpiryTime update_factor="10.0">20.0</ExpiryTime>
      </Heartbeat>
      <InitialDiscoveryPeriod>1.0</InitialDiscoveryPeriod>
      <Alignment latency_budget="0.0" transport_priority="0">
        <RequestCombinePeriod>
          <Initial>3.0</Initial>
          <Operational>0.1</Operational>
        </RequestCombinePeriod>
      </Alignment>
      <WaitForAttachment maxWaitCount="10">
        <ServiceName>networking</ServiceName>
      </WaitForAttachment>
    </Network>
    <Persistent>
      <StoreDirectory>/tmp/store</StoreDirectory>
      <StoreMode>XML</StoreMode>
      <StoreSleepTime>0.5</StoreSleepTime>
    </Persistent>
    <NameSpaces>
      <NameSpace durabilityKind="DURABLE"
        alignmentKind="INITIAL_AND_ALIGNER">
        <Partition>*</Partition>
      </NameSpace>
    </NameSpaces>
  </DurabilityService>
</OpenSplice>
```

Table 3, *Durability Parameters*, gives an overview of all parameters. The values given for the **Path** are relative to the root node `//OpenSplice/DurabilityService`.

Table 3 Durability Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
Network[@latency_budget]		0.0	Float	0.0 - max float		Determines the latency budget that is used by the Durability Service for protocol messages that are sent over the network meant to communicate with remote Durability Services.
Network[@transport_priority]		0	Int	0 - max int		Determines the transport priority that is used by the Durability Service for protocol messages that are sent over the network meant to communicate with remote Durability Services.
ExpiryTime	Network/Heartbeat	5.0	Float	1.0 - 20.0	X	The maximum interval allowed between two heartbeats from a fellow durability daemon. If this interval has exceeded, the fellow daemon will be declared dead.
ExpiryTime[@update_factor]	Network/Heartbeat	0.5	Float	0.1 - 1.0	X	A factor specifying the actual period between two heartbeats sent. This introduces a margin that allows heartbeats to get lost without immediate consequences.
Heartbeat[@latency_budget]	Network	0.0	Float	0.0 - max float		Determines the latency budget that is used by the Durability Service for heartbeat messages that are sent over the network meant to communicate with remote Durability Services. This value overrules the setting defined under Network[@latency_budget]
The values given for the Path are relative to the root node <code>//OpenSplice/DurabilityService</code> .						

Table 3 Durability Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
Heartbeat[@transport_priority]	Network	0	Int	0 - max int		Determines the transport priority that is used by the Durability Service for heartbeat messages that are sent over the network meant to communicate with remote Durability Services. This value overrules the setting defined under Network[@transport_priority]
InitialDiscoveryPeriod	Network	3.0	Float	0.1 - 10		The period during which the service will wait for fellow services to reply to each of the initial askStatus requests.
Alignment[@latency_budget]	Network	0.0	Float	0.0 - max float		Determines the latency budget that is used by the Durability Service for alignment messages that are sent over the network meant to communicate with remote Durability Services. This value overrules the setting defined under Network[@latency_budget]
Alignment[@transport_priority]	Network	0	Int	0 - max int		Determines the transport priority that is used by the Durability Service for alignment messages that are sent over the network meant to communicate with remote Durability Services. This value overrules the setting defined under Network[@transport_priority]
Initial	Network/Alignment/RequestCombinePeriod	0.5	Float	0.01 - 5		The minimum time (in seconds) between the time of a request for aligning and the start of the actual aligning during the startup phase of the durability service.
The values given for the Path are relative to the root node <code>//OpenSplice/DurabilityService</code> .						

Table 3 Durability Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
Operational	Network/Alignment/RequestCombinePeriod	0.5	Float	0.01 - 5		The minimum time (in seconds) between the time of a request for aligning and the start of the actual aligning during the operational phase of the durability service.
WaitForAttachment[@maxWaitCount]	Network	200	Natural	1 - 1000		Durability will have to wait for other services to initialize. This parameter
ServiceName	Network/WaitForAttachment	Not Available	String	Any string not containing XML mark-up characters		The time (in seconds) the service will wait for "GroupAttachServices" to attach to a group.
StoreDirectory	Persistent	" "	String	Any platform dependent path to the store location.		This option specifies the location of the persistent store. If this option is not set, the durability service will not manage persistent data.
StoreMode	Persistent	"XML"	String	"XML"		Controls the format in which the data is stored and read. This means the setting must be consistent between runs. Currently, XML is the single supported format. Only the first 3 characters of the string are significant to identify the wanted value.
QueueSize	Persistent	1001	Natural	100 - 10000		Maximum number of samples in the persistent queue. The available resources determine the maximum value of this setting.
The values given for the <i>Path</i> are relative to the root node //OpenSplice/DurabilityService.						

Table 3 Durability Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
NameSpace[@durabilityKind]	NameSpaces	Durable	Enum	Durable, Persistent, Transient, Transient_Local		This attribute selects the kind of durable data that is aligned within this namespace.
NameSpace[@alignmentKind]	NameSpaces	Initial_And_Aligner	Enum	Initial_And_Aligner, Initial, Lazy		This attribute specifies the role of this service within this namespace, with respect to alignment functionality.
Partition	NameSpaces/NameSpace	Not Available	String	Any string not containing XML mark-up characters.		For this namespace, only data for the specified partitions will be aligned. The expression may contain the wildcards '*' and '?'. Multiple elements are allowed.
The values given for the Path are relative to the root node <i>//OpenSplice/DurabilityService</i> .						

Networking Service

The following XML definition shows all possible networking settings and their defaults.

```
<OpenSplice>
  <!-- Domain Part -->
  <Domain>
    <Service name="networking">
      <Command>networking</Command>
      <Configuration></Configuration>
    </Service>
  </Domain>

  <!-- Networking Daemon Part -->
  <NetworkService name="networking">
    <Partitioning>
      <GlobalPartition Address="broadcast"/>
      <NetworkPartitions>
        <NetworkPartition Name="Example" Address="230.230.1.1"
          Connect="true"/>
      </NetworkPartitions>
      <PartitionMappings>
        <PartitionMapping NetworkPartition="Example"
          DCPSPartitionTopic="ExamplePartition.ExampleTopic"/>
      </PartitionMappings>
    </Partitioning>
    <General>
      <NetworkInterfaceAddress>
        first available
      </NetworkInterfaceAddress>
    </General>
    <Channels>
      <!-- Currently, at most 10 channels allowed -->
      <Channel name="(no default)" reliable="false"
        default="false" enabled="true" priority="2">
        <PortNr>3367</PortNr>
        <FragmentSize>1300</FragmentSize>
        <Resolution>50</Resolution>
        <Sending>
          <QueueSize>4000</QueueSize>
          <MaxBurstSize>10</MaxBurstSize>
          <MaxRetries>10</MaxRetries>
          <RecoveryFactor>5</RecoveryFactor>
          <DiffServField>0</DiffServField>
          <Scheduling>
            <Priority>(OS-specific)</Priority>
            <Schedule>default</Schedule>
          </Scheduling>
        </Sending>
        <Receiving>
          <ReceiveBufferSize>65500</ReceiveBufferSize>
          <Scheduling>
            <Priority>(OS-specific)</Priority>
            <Schedule>default</Schedule>
          </Scheduling>
        </Receiving>
      </Channel>
    </Channels>
  </NetworkService>
</OpenSplice>
```

```
<Discovery>
  <!-- a special channel that contains all channel
        elements and a bit more: -->
  <Active>true</Active>
  <Sending>
    <Interval>1000</Interval>
    <SafetyFactor>0.9</SafetyFactor>
    <SalvoSize>3</SalvoSize>
  </Sending>
  <Receiving>
    <DeathDetectionCount>6</DeathDetectionCount>
  </Receiving>
</Discovery>
</NetworkService>
</OpenSplice>
```

Table 4, *Networking Parameters*, provides an overview of all parameters. The values given for the **Path** is relative to the root node `//OpenSplice/NetworkService`.

Table 4 Networking Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
NetworkInterfaceAddress	General	"first available"	String	"first available" or any dotted decimal IPv4 address		String representation of the networking interface to use. The interface given should be broadcast enabled. If the default is not used, the setting should represent the primary IPv4 address of the interface.
GlobalPartition[@Address]	Partitioning	"broadcast"	Class D IP	"broadcast" or class D IP	X	String representation of the default networking address that should be used for global data communication
NetworkPartition[@Name]	Partitioning/Network Partitions		String	unique name	X	Unique name identifying the networking partition
NetworkPartition[@Address]	Partitioning/Network Partitions	"broadcast"	Class D IP	"broadcast" or class D IP	X	String representation of the networking address for the given networking partition
NetworkPartition[@Connected]	Partitioning/Network Partitions	true	Boolean	false/true		Connects or disconnects a node from the given networking partitions
PartitionMapping[@DCSPartitionTopic]	Partitioning/Partition Mappings	Not Available	String.String	Valid DCPS names and * wild card	X	Identifies an expression for matching DCPS partition/topic combinations to networking partitions
PartitionMapping[@NetworkPartition]	Partitioning/Partition Mappings	Not Available	String	Previously defined network partition name	X	Points to the networking partition that has to be used for this mapping.
The values given for the Path is relative to the root node <code>//OpenSplice/NetworkService</code> .						

Table 4 Networking Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
Channel[@name]	Channels	Not Available	Not Available	Not Available		Every channel has to have a unique name
Channel[@reliable]	Channels	false	Boolean	false/true		This parameter indicates whether the channel should send messages reliably or not.
Channel[@default]	Channels	false	Boolean	false/true		If a quality of service has been requested that does not match any of the channels, the default channel is selected. Only one channel can be the default channel.
Channel[@enabled]	Channels	true	Boolean	false/true		The parameter can be used to switch on or off a specific channel.
Channel[@priority]	Channels	0	Int	0 - max int		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.
PortNr	Channels/Channel	3367	Natural	OS specific	X	UDP messages for the best effort channel are sent to the port number given. Best effort is currently the only channel available.
The values given for the Path is relative to the root node //OpenSplice/NetworkService.						

Table 4 Networking Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
FragmentSize	Channels/Channel	1300	Natural	200 - OS specific, usually 65536		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
Resolution	Channels/Channel	50	Natural	20 - MaxInt	X	The Resolution parameters indicates the minimal latency and reliability resolution in milliseconds. All latency budget values and actions are executed on a multiple of Resolution milliseconds. Resending of reliable data is done on a multiple of Resolution milliseconds.
QueueSize	Channels/Channel/ Sending	4000	Natural	1 - MaxInt		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 best effort messages are dropped and reliable messages are rejected. In the last case, the writer writing into the queue is suspended and will retry until success. This setting determines the number of messages the networking queue can contain. Note that a full networking queue is a symptom of an improperly designed system.
MaxBurstSize	Channels/Channel/ Sending	10	Natural	1 - MaxInt		During Resolution milliseconds, at most MaxBurstSize bytes of data are sent to the network. Resends of reliable data is not counted.
The values given for the <i>Path</i> is relative to the root node <i>//OpenSplice/NetworkService</i> .						

Table 4 Networking Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
MaxRetries	Channels/Channel/ Sending	10	Natural	1 - MaxInt		Reliable data that can not be sent for MaxRetries times is considered lost.
RecoveryFactor	Channel/Channel/ Sending	5	Natural	1 - MaxInt		Reliable data is resent at the moment that it has not been acknowledged during RecoveryFactor ticks of Resolution milliseconds.
DiffServField	Channels/Channel/ Sending	0	Natural	0 - 255		
Priority	Channels/Channel/ Sending/Scheduling	Not Available	Natural	OS specific		The sending thread of the Channel is created with a priority given by this parameter
Class	Channels/Channel/ Sending/Scheduling	“default”	String	"default", "timeshare", "realtime"		The sending thread of the Channel is created with a scheduling class given by this parameter
ReceiveBufferSize	Channels/Channel/ Receiving	65500	Natural	OS specific		The UDP receive buffer of the best effort channel socket is set to the value given. If many message are lost, the receive buffer size has to be increased. Best effort is currently the only channel available.
Priority	Channels/Channel/ Receiving/Scheduling	Not Available	Natural	OS specific		The receiving thread of the Channel is created with a priority given by this parameter
Class	Channels/Channel/Re ceiving/Scheduling	“default”	String	"default", "timeshare", "realtime"		The receiving thread of the Channel is created with a scheduling class given by this parameter
The values given for the Path is relative to the root node <i>//OpenSplice/NetworkService</i> .						

Table 4 Networking Parameters (Continued)

Name	Path	Default	Format	Valid Values	C	Meaning
Active	Discovery	true	Boolean	false/true		By default, the discovery protocol for determining the starting and stopping of nodes is switched on. When switched off, data is always sent to the network
Interval	Discovery/Sending	1000	Natural	100 - MaxInt		The interval in milliseconds for sending heartbeats
SafetyFactor	Discovery/Sending	0.9	Float	0.2 - 1.0		Heartbeats are sent every SafetyFactor * Interval milliseconds.
SalvoSize	Discovery/Sending	3	Natural	1 - MaxInt		State changes are quickly sent to the network SalvoSize times to decrease the reaction time
DeathDetectionCount	Discovery/Receiving	6	Natural	1 - MaxInt		A node is declared dead after DeathDetectionCount heartbeats have been lost.
The values given for the <i>Path</i> is relative to the root node <i>//OpenSplice/NetworkService</i> .						

Tuner Service

This chapter provides the specification of the configuration for the Tuner Service. The purpose of the service is to provide remote access to the node for OpenSplice tooling. First the XML representation of all settings is given. This representation contains all default values as used by the Tuner Service and after that the meaning and format of each setting is explained in a table.

The following XML definition shows the most used settings and their defaults.

```
<OpenSplice>
  <!-- Domain Part -->
  <Domain>
    <Service name="cmsoap">
      <Command>cmsoap</Command>
      <Configuration></Configuration>
    </Service>
  </Domain>
  <TunerService name="cmsoap">
    <Client>
      <LeasePeriod>5.0</LeasePeriod>
      <MaxClients>2</MaxClients>
      <MaxThreadsPerClient>2</MaxThreadsPerClient>
    </Client>
    <Server>
      <Backlog>5</Backlog>
      <Port>8000</Port>
      <Verbosity>0</Verbosity>
    </Server>
  </TunerService>
</OpenSplice>
```

Table 5, *Tuner Service Parameters* provides an overview of all parameters. The values given for the **Path** is relative to the root node `//OpenSplice/TunerService`.

Table 5 Tuner Service Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
LeasePeriod	Client	10.0	Float	10.0 - ∞		The lease period for clients that are connected to the service. Clients that are connected to this service have to update their lease every ClientLeasePeriod (seconds.nanoseconds). Calling any function of the service or explicitly calling the update lease routine of the service can achieve this. If the client fails to update its lease within this period, all its resources are cleaned up.
MaxClients	Client	2	Natural	1 - ∞		The maximum number of concurrently connected clients.
MaxThreadsPerClient	Client	2	Natural	1 - ∞		The maximum number of concurrently running threads for one specific client.
Backlog	Server	5	Natural	0 - ∞		The maximum number of requests that may wait to be handled when the maximum number of clients or the maximum number of threads is reached.
PortNr	Server	8000	Natural	1 - 65535		The port number where the service attaches to and listens for requests.
Verbosity	Server	0	Natural	0 - ∞		The verbosity level of the service. This is for debugging purposes only and therefore should always be 0.
The values given for the Path is relative to the root node <code>//OpenSplice/TunerService</code> .						

DbmsConnect Service

This section provides the DbmsConnect Service's configuration specification. The service provides facilities for exchanging data between OpenSplice and one or more DBMS'. The XML definition of all settings is provided below. The XML definition contains all of the default values that are used by the Dbmsconnect Service. Table 6, *DbmsConnect Service Parameters*, on page 80, provides descriptions for each setting.

The following XML shows the most used settings and their defaults.

Table 6, *DbmsConnect Service Parameters*, provides an overview of all parameters . Note that the values shown for **Path** are the path relative from the root node *//OpenSplice/DbmsConnectService*.

Table 6 DbmsConnect Service Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
DdsToDbms		N/A	N/A	N/A		<i>Element that holds the configuration of the service concerning DDS to DBMS bridging.</i>
NameSpace	DdsToDbms	N/A	N/A	N/A		Element that is used to configure the responsibilities of the service concerning the bridging of data from DDS to DBMS.
NameSpace@partition	DdsToDbms	"*"	String	Any valid DDS partition expression		An expression, with wildcards, that represents one or more DDS partitions. 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.
NameSpace@topic	DdsToDbms	"*"	String	Any valid DDS topic expression		An expression, with wildcards, that represents one or more DDS topic names. 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.
The values shown for Path are the path relative from the root node <i>//OpenSplice/DbmsConnectService</i> .						

Table 6 DbmsConnect Service Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
Namespace@update_frequency	DdsToDbms	0.0	Float	Any valid time in seconds		Optional attribute that determines the frequency in 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)
Namespace@dsn	DdsToDbms	""	String	Any valid ODBC data source name		Represents the ODBC Data Source Name, that represents the DBMS where the service must bridge the DDS data to.
Namespace@usr	DdsToDbms	""	String	Any valid ODBC user name		Represents the user name that is used when connecting to the DBMS.
Namespace@pwd	DdsToDbms	""	String	Any valid ODBC password		Represents the password that is used when connecting to the DBMS.
Namespace@odbc	DdsToDbms	Platform dependent	String	A valid library name		The service dynamically loads an ODBC library at runtime. This attribute determines 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.
Mapping@topic	DdsToDbms/ Namespace	N/A	String	Any valid DDS Topic name		The name of the topic where the Mapping applies to.
The values shown for Path are the path relative from the root node <i>//OpenSplice/DbmsConnectService</i> .						

Table 6 DbmsConnect Service Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
Mapping@table	DdsToDbms/ NameSpace	N/A	String	Any valid DBMS table name		Optional name of the table that must be associated with the topic. This means the service will create a table with the specified name for the specified topic and will also store data samples in this table as well. By default the table name is equal to the topic name.
Mapping@query	DdsToDbms/ NameSpace	N/A	String	Any valid DDS query.		Optional DDS 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.
Mapping@filter	DdsToDbms/ NameSpace	N/A	String	Any valid DDS content filter.		Optional DDS content filter. Only DDS data that matches the query will be bridged to the DBMS domain. This is realized by means of a DCPS content filterd topic.
DbmsToDds		N/A	N/A	N/A		Element that holds the configuration of the service concerning DBMS to DDS bridging.
NameSpace	DbmsToDds	N/A	N/A	N/A		Element that is used to configure the responsibilities of the service concerning the bridging of data from DBMS to DDS.
NameSpace@partition	DbmsToDds	"*"	String	Any valid DDS partition expression		An expression, with wildcards, that represents one or more DDS partitions. A "*" represents any sequence of characters and a "?" represents a single character. This expression is used to specify the partitions from which DBMS records must be 'bridged' to the DDS domain.
The values shown for Path are the path relative from the root node <i>//OpenSplice/DbmsConnectService</i> .						

Table 6 DbmsConnect Service Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
NameSpace@table	DbmsToDds	“*”	String	Any table expression		An expression, with wildcards, that represents one or more DBMS tables. A "*" represents any sequence of characters and a "?" represents a single character. This expression is used to specify the DBMS table(s) from which records must be 'bridged' to the DDS domain.
NameSpace@update_frequency	DbmsToDds	0.5	Float	Any valid time in seconds		Optional attribute that determines the frequency in which the service will update the DDS domain with DBMS data.
NameSpace@dsn	DbmsToDds	“”	String	Any valid ODBC data source name		Represents the ODBC Data Source Name, that represents the DBMS where the service must retrieve the data from.
NameSpace@usr	DbmsToDds	“”	String	Any valid ODBC user name		Represents the user name that is used when connecting to the DBMS.
NameSpace@pwd	DbmsToDds	“”	String	Any valid ODBC password		Represents the password that is used when connecting to the DBMS.
NameSpace@odbc	DbmsToDds	Platform dependent	String	A valid library name		The service dynamically loads an ODBC library at runtime. This attribute determines 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 values shown for Path are the path relative from the root node //OpenSplice/DbmsConnectService.						

Table 6 DbmsConnect Service Parameters

Name	Path	Default	Format	Valid Values	C	Meaning
Mapping@table	DbmsToDds/ NameSpace	N/A	String	Any valid DDS Topic name		The name of the table where the Mapping applies to.
Mapping@topic	DbmsToDds/ NameSpace	N/A	String	Any valid DBMS table name		Optional name of the topic that must be associated with the table. This means the service will create a topic with the specified name for the table and will also write DDS data samples using this topic as well. By default the table name is equal to the topic name.
Mapping@query	DbmsToDds/ NameSpace	N/A	String	Any valid DDS 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 an SQL query.
The values shown for Path are the path relative from the root node <i>//OpenSplice/DbmsConnectService</i> .						

A close-up, low-angle photograph of a computer keyboard, focusing on the central and lower-right keys. The keys are white with dark lettering. A white grid pattern is overlaid on the entire image, creating a sense of depth and perspective. The word "INDEX" is printed in a dark blue, serif font in the upper right quadrant.

INDEX

Index

A

Attribute update_factor. 13

B

Basic Problems. 56

C

CMSOAP Daemon. 43

Complete List of Error Messages. 56

Conventions 59

D

Database Parameters. 12

Discovery 55

Durability Daemon. 18

E

Element ExpiryTime 13

Element name. 12

Element size. 12

Example Reference Systems 53

F

Fragment size 56

Frequently occurring application requirements . 56

H

High Volumes. 54

K

Known problems 56

L

Lease Parameters 13

Low Latencies 55

M

Message Rejected. 56

N

Networking 56 Networking Daemon 28, 70

R

Receive Buffer Too Small. 56 Responsiveness 55
Requested Shared Memory Too Large 56 Routing Table 56
Required Value Not Found 56

S

Service Died 56 Shared Memory Segment Already Exists 56
Service Liveliness 56 Single Node System 54
Service Will Not Start 56 Socket Still In Use. 56
Shared Memory 56 Spliced Daemon 11, 61

T

Troubleshooting 56

Z

Zero Configuration System 53