OpenSplice DDS

Version 5.x

Migration Guide

PRISMTECH

OpenSplice DDS

MIGRATION GUIDE

Part Number: OS-MG5

Doc Issue 01, 26 Mar 10

Copyright Notice

Preface

About this Migration Guide

This *Migration Guide* is intended to help users migrate their existing code from OpenSplice DDS version ***4.x*** to OpenSplice DDS version ***5.x***.

This guide is **only** intended for customers who are currently using *OpenSplice* ***V4.x*** and who want to migrate to *OpenSplice* ***5.x***. Migration from older versions up to V4.x is covered by separate Migration Guides.

This guide ***does not*** cover all of the differences which exist between the OpenSplice DDS V5.x and previous versions, only those which are needed for compatibility.

i

The C language binding is provided with a special *legacy mode* which enables pre-version 5.x code to be used without modification.

Conventions

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

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

i

Item contains helpful hint or special information.

WIN

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

UNIX

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

C

C language specific

C++

C++ language specific

Java

Java language specific

Hypertext links are shown as *blue italic underlined.*

On-Line (PDF) versions of this document: Items shown as cross references, *e.g. Contacts* on page iv, 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*** indicate new terms, or emphasise an item.

Arial Bold indicate Graphical User Interface (GUI) elements and commands, for example, **File > Save** from a menu.

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

Contacts

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

USA Corporate Headquarters

PrismTech Corporation
400 TradeCenter
Suite 5900
Woburn, MA
01801
USA

Tel: +1 781 569 5819

European Head Office

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

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

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

Migration to Version 5.x

CHAPTER

1 Incompatibilities and Solutions

This guide is intended to help users of OpenSplice DDS Version 4 with the migration of their existing code-base to the new version of OpenSplice DDS (V5.x).

Not all changes in the product are described here, as this document is focused on those changes that cause incompatibilities between existing code and the new version of OpenSplice.

A complete list of all changes in the product can be found in the release notes.

1.1 Structural Changes

1.1.1 ANY_STATUS

The `ANY_STATUS` is now deprecated and replaced by `STATUS_MASK_ANY_V1_2` due to the introduction of an additional status, but the first one will remain working throughout the V5 series.

1.1.2 STATUS_MASK_NONE

In OpenSplice V5 a new status mask (`STATUS_MASK_NONE`) has been introduced. This status mask addition is a precursor to the addition of this new status mask to the DDS 1.3 specification. The new special constant `STATUS_MASK_NONE` can be used to indicate that an entity listener or an entity `StatusCondition` should not respond to any of its available statuses. Listeners take precedence over `StatusConditions` when handling status changes. In such cases the DDS will propagate the statuses to the entities factory. If there is no factory that consumes the status then the status is disregarded. The example on the next page (using the SAC api) demonstrates its use.

```
DDS_DomainParticipantFactory dpf;
DDS_DomainParticipant dp;
DDS_DomainId_t domain = NULL;

dpf = DDS_DomainParticipantFactory_get_instance();
/* Create a domain participant with a NIL listener and
 * STATUS_MASK_NONE, which will thus forward any status changes
 * to the appropiate StatusCondition
 */
dp = DDS_DomainParticipantFactory_create_participant (
    dpf,
    domain,
    DDS_PARTICIPANT_QOS_DEFAULT,
    NULL,/* NIL listener */
    DDS_STATUS_MASK_NONE);
```

In OpenSplice V4 example code the status mask `STATUS_ANY` (now `STATUS_MASK_ANY_V1_2`) was sometimes used when creating an entity with a `NIL` listener, this is still a valid usecase but has a very different result compared to using the `STATUS_MASK_NONE`. An entity with the combination of a `NIL` listener and `STATUS_ANY` results in all status change events being consumed by that entity and not propagated to its factory or an appropiate `StatusCondition`. An example (using the SAC api) demonstrates this usage:

```
DDS_DomainParticipantFactory dpf;
DDS_DomainParticipant dp;
DDS_DomainId_t domain = NULL;

dpf = DDS_DomainParticipantFactory_get_instance();
/* Create a domain participant with a NIL listener and
 * DDS_STATUS_MASK_ANY_V1_2, which will thus consume all status
 * changes events and not propagate them.
 */
dp = DDS_DomainParticipantFactory_create_participant (
    dpf,
    domain,
    DDS_PARTICIPANT_QOS_DEFAULT,
    NULL,/* NIL listener */
    DDS_STATUS_MASK_ANY_V1_2);
```

1.1.3 DataReaderQos

The `DataReaderQos` has been extended with three additional policies for all language bindings:

- `SubscriptionKeyQosPolicy`
- `ReaderLifespanQosPolicy`
- `ShareQosPolicy`

Furthermore, the already existing `ReaderDatalifecycleQosPolicy` has been extended with an additional attribute.

If the default values for the newly-introduced policies are used, the behaviour will be exactly the same as before. In case applications already use the `get_default_datareader_qos()` method to obtain it and just override the values that need to be different from the default, no changes need to be made to migrate. In case the application allocates the `QoS` itself and sets all policies, the default values as specified below and in the following sections should be set before providing it as an argument to any DDS method call.

V4:

```
struct DataReaderQos {
    DurabilityQosPolicy durability;
    DeadlineQosPolicy deadline;
    LatencyBudgetQosPolicy latency_budget;
    LivelinessQosPolicy liveliness;
    ReliabilityQosPolicy reliability;
    DestinationOrderQosPolicy destination_order;
    HistoryQosPolicy history;
    ResourceLimitsQosPolicy resource_limits;
    UserDataQosPolicy user_data;
    OwnershipQosPolicy ownership;
    TimeBasedFilterQosPolicy time_based_filter;
    ReaderDataLifecycleQosPolicy reader_data_lifecycle;
};
```

V5:

```
struct DataReaderQos {
    DurabilityQosPolicy durability;
    DeadlineQosPolicy deadline;
    LatencyBudgetQosPolicy latency_budget;
    LivelinessQosPolicy liveliness;
    ReliabilityQosPolicy reliability;
    DestinationOrderQosPolicy destination_order;
    HistoryQosPolicy history;
    ResourceLimitsQosPolicy resource_limits;
    UserDataQosPolicy user_data;
    OwnershipQosPolicy ownership;
    TimeBasedFilterQosPolicy time_based_filter;
    ReaderDataLifecycleQosPolicy reader_data_lifecycle;
    SubscriptionKeyQosPolicy subscription_keys;
    ReaderLifespanQosPolicy reader_lifespan;
    ShareQosPolicy share;
};
```

1.1.3.1 SubscriptionKeyQosPolicy

The `SubscriptionKeyQosPolicy` is a newly-introduced `QoS` policy. By default the `use_key_list` is set to `false` and the `key_list` is an empty sequence.

V4:

(*Not available*)

V5:

```
struct SubscriptionKeyQosPolicy {
    boolean use_key_list;
    StringSeq key_list;
};
```

1.1.3.2 ReaderLifespanQosPolicy

The `ReaderLifespanQosPolicy` is a newly-introduced `QoS` policy. By default the `use_lifespan` is set to `false` and the `duration` is set to `DURATION_INFINITE`.

V4:

(*Not available*)

V5:

```
struct ReaderLifespanQosPolicy {
    boolean use_lifespan;
    Duration_t duration;
};
```

1.1.3.3 ShareQosPolicy

The `ShareQosPolicy` is a newly-introduced `QoS` policy. By default the `enable` is set to `false` and the `name` is an empty string.

V4:

(*Not available*)

V5:

```
struct ShareQosPolicy {
    string name;
    boolean enable;
};
```

1.1.3.4 ReaderDataLifecycleQosPolicy

The already existing `ReaderDataLifecycleQosPolicy` has been extended with one additional attribute. By default the `enable_invalid_samples` is set to `false`.

V4:

```
struct ReaderDataLifecycleQosPolicy {
    Duration_t autopurge_nowriter_samples_delay;
    Duration_t autopurge_disposed_samples_delay;
};
```

V5:

```
struct ReaderDataLifecycleQosPolicy {
    Duration_t autopurge_nowriter_samples_delay;
    Duration_t autopurge_disposed_samples_delay;
    boolean enable_invalid_samples;
};
```

1.1.4 SubscriberQos

The `SubscriberQos` has been extended with one additional policy for all language bindings:

`ShareQosPolicy`

If the default values for the newly-introduced policy is used, the behaviour will be exactly the same as before. In case applications already use the `get_default_subscriber_qos()` method to obtain it and just override the values that need to be different from the default, no changes need to be made to migrate. In case the application allocates the `QoS` itself and sets all policies, the default values as specified below should be set before providing it as an argument to any DDS method call.

V4:

```
struct SubscriberQos {
    PresentationQosPolicy presentation;
    PartitionQosPolicy partition;
    GroupDataQosPolicy group_data;
    EntityFactoryQosPolicy entity_factory;
};
```

V5:

```
struct SubscriberQos {
    PresentationQosPolicy presentation;
    PartitionQosPolicy partition;
    GroupDataQosPolicy group_data;
    EntityFactoryQosPolicy entity_factory;
    ShareQosPolicy share;
};
```

1.1.4.1 ShareQosPolicy

The `ShareQosPolicy` is a newly-introduced `QoS` policy. By default the `enable` is set to `false` and the `name` is an empty string.

V4:

(*Not available*)

V5:

```
struct ShareQosPolicy {
    string name;
    boolean enable;
};
```

1.1.5 DataWriterQos

The `WriterDataLifecycleQosPolicy` of the `DataWriterQos` has been extended with two additional attributes.

If the default values for the newly-introduced attributes are used, the behaviour will be exactly the same as before. In case applications already use the `get_default_datawriter_qos()` method to obtain it and just override the values that need to be different from the default, no changes need to be made to migrate. In case the application allocates the `QoS` itself and sets all policies, the default values as specified below should be set before providing it as an argument to any DDS method call.

1.1.5.1 WriterDataLifecycleQosPolicy

The already existing `WriterDataLifecycleQosPolicy` has been extended with two additional attributes.

By default both the `autopurge_suspended_samples_delay` and the `autounregister_instance_delay` are set to `DURATION_INFINITE`.

V4:

```
struct WriterDataLifecycleQosPolicy {
    boolean autodispose_unregistered_instances;
};
```

V5:

```
struct WriterDataLifecycleQosPolicy {
    boolean autodispose_unregistered_instances;
    Duration_t autopurge_suspended_samples_delay;
    Duration_t autounregister_instance_delay;
};
```

1.2 Other changes

1.2.1 OSPL_HOME

In the V5.x series the `OSPL_HOME` environment variable is no longer set on the Windows platform. This allows multiple OpenSplice DDS installations to reside on the same machine.