Document Title	Specification of DDS Service Discovery Protocol
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1111

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R24-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2024-11-27	R24-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and overview

This protocol specification details the OMG® DDS® data types, QoS policies, sequences and semantics of the AUTOSAR Protocol for Service Discovery over DDS.

AUTOSAR platforms employ, among others, the Service-Oriented Architecture (SOA) communications paradigm. In SOA systems, `Service Interface`s cohesively grouping Elements of various kinds are provided, required or both as `Service Instance`s by Applications.

In this context, OMG® DDS®, as enabled by the Classic Platform Dds Basic Software Module and the Adaptive Platform DDS Network Binding, can be used to advertise and discover `Service Instance`s.

1.1 Protocol purpose and objectives

This protocol defines how the OMG® DDS® middleware standard can be employed to advertise and discover `Service Instance`s.

1.2 Applicability of the protocol

This protocol applies to DDS `Service Instance` advertisement and discovery in:

- The AUTOSAR Classic Platform
- The AUTOSAR Adaptive Platform
- Non-AUTOSAR platforms targeting AUTOSAR interoperability

1.2.1 Constraints and assumptions

The following constraints and assumptions apply to the present document:

- The OMG® DDS® family of standards already define wire protocols, data type description formats, QoS policies and APIs
- Conversely, in this document "protocols" are described in terms of OMG® DDS® API interactions, QoS policies configuration and data type definitions involved in `Service Interface` advertisement and discovery processes

1.2.2 Limitations

Not applicable.

1.3 Dependencies

1.3.1 Dependencies to other protocol layers

The protocols described in this document rely, indirectly, upon the following OMG® DDS® protocol standards:

- DDS Wire Interoperability protocol (DDSI-RTPS) defined in [1]
- DDS-XTYPES Minimal Programming Interface and Network Interoperability Profiles defined in [2]

1.3.2 Dependencies to other standards and norms

The protocols described in this document target the following OMG® DDS® standards and profiles:

- DDS Minimum Profile defined in [3]
- DDS Wire Interoperability protocol (DDSI-RTPS) defined in [1]
- DDS-XTYPES Minimal Programming Interface and Network Interoperability Profiles defined in [2]

1.3.3 Dependencies to the Application Layer

Not applicable.

2 Use Cases

ID	*Name*	*Description*
UC_001	Provide a Service	An Application offers, and publishes samples of, an Instance of a Service Interface to other Applications in the network.
UC_002	Require a Servive	An Application attempts discovery of an Instance of a Service Interface, possibly offered by other Applications in the network.
UC_003	Consume a Service	An Application binds against a discovered Instance of a Service Interface, offered by other Applications in the network.
UC_004	Flexible communication paths	A developer wants to design his applications without knowing who will receive the data or who is sending the desired data. Dynamic Service Discovery enables dynamic creation of communication paths during run-time.

3 Related documentation

3.1 Input documents & related standards and norms

[1] DDS Interoperability Wire Protocol, Version 2.2
http://www.omg.org/spec/DDSI-RTPS/2.2

[2] Extensible and Dynamic Topic Types for DDS, Version 1.2
https://www.omg.org/spec/DDS-XTypes/1.2

[3] Data Distribution Service (DDS), Version 1.4
http://www.omg.org/spec/DDS/1.4

[4] ISO/IEC C++ 2003 Language DDS PSM, Version 1.0
https://www.omg.org/spec/DDS-PSM-Cxx/1.0

[5] Interface Definition Language (IDL), Version 4.2
https://www.omg.org/spec/IDL/4.2

3.2 Related specification

Not applicable.

4 Protocol Requirements

Implementation of this protocol requires an OMG® DDS® middleware implementation supporting:

- DDS Minimum Profile defined in [3]
- DDS Wire Interoperability protocol (DDSI-RTPS) defined in [1]
- DDS-XTYPES Minimal Programming Interface and Network Interoperability Profiles defined in [2]

4.1 Requirements Traceability

Requirement	Description	Satisfied by
[FO_RS_Dds_00001]	DDS Compliance	[FO_PRS_DDSSD_00101] [FO_PRS_DDSSD_00102] [FO_PRS_DDSSD_00103] [FO_PRS_DDSSD_00105] [FO_PRS_DDSSD_00106] [FO_PRS_DDSSD_00108] [FO_PRS_DDSSD_00109] [FO_PRS_DDSSD_00111] [FO_PRS_DDSSD_00201] [FO_PRS_DDSSD_00202] [FO_PRS_DDSSD_00203] [FO_PRS_DDSSD_00205] [FO_PRS_DDSSD_00206] [FO_PRS_DDSSD_00208]
[FO_RS_Dds_00005]	DDS Quality of Service	[FO_PRS_DDSSD_00101] [FO_PRS_DDSSD_00102] [FO_PRS_DDSSD_00103] [FO_PRS_DDSSD_00105] [FO_PRS_DDSSD_00106] [FO_PRS_DDSSD_00108] [FO_PRS_DDSSD_00109] [FO_PRS_DDSSD_00111] [FO_PRS_DDSSD_00202] [FO_PRS_DDSSD_00205]
[FO_RS_Dds_00007]	Type Definition	[FO_PRS_DDSSD_00202]
[FO_RS_Dds_00008]	Customization	[FO_PRS_DDSSD_00102] [FO_PRS_DDSSD_00103] [FO_PRS_DDSSD_00105] [FO_PRS_DDSSD_00106] [FO_PRS_DDSSD_00108] [FO_PRS_DDSSD_00109] [FO_PRS_DDSSD_00111] [FO_PRS_DDSSD_00201] [FO_PRS_DDSSD_00202] [FO_PRS_DDSSD_00203] [FO_PRS_DDSSD_00205] [FO_PRS_DDSSD_00206] [FO_PRS_DDSSD_00208]
[FO_RS_Dds_00015]	Publish	[FO_PRS_DDSSD_00201] [FO_PRS_DDSSD_00202]

▽

△

Requirement	Description	Satisfied by
[FO_RS_Dds_00016]	Subscribe	[FO_PRS_DDSSD_00201]

Table 4.1: Requirements Tracing

5 Definition of terms and acronyms

5.1 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
OMG	Object Management Group
QoS	Quality of Service
DDSI	Data Distribution Service Interoperability
RTPS	Real-Time Publish-Subscribe
XTYPES	eXtensible Types

Table 5.1: Acronyms and abbreviations used in the scope of this Document

5.2 Definition of terms

Terms:	Description:
Entity	The base class for all other DDS Entities.
DomainParticipant	Represents the participation of the application on a communication plane that isolates applications running on the same set of physical computers from each other.
Topic	Reprepsents the most basic description of the data to be published and subscribed.
Publisher	Provides the actual dissemination of publications.
DataWriter	Provides the application functionality to set the value of the data to be published under a given Topic.
Subscriber	Provides the actual reception of the data resulting from its subscriptions.
DataReader	Provides the application with (1) functionality to declare the data it wishes to receive (i.e., make a subscription) and (2) to access the data received by the attached Subscriber.
QoS Profile	Grouping of QoS Policy values applicable to one or more DDS Entities.

Table 5.2: Definition of terms in the scope of this Document

6 Protocol Specification

6.1 Introduction

In AUTOSAR, two distinct Service Discovery Protocols are configured by <DDSServiceInstanceDiscoveryType>, and specified in the following sections:

- The DomainParticipant QoS -based discovery protocol that leverages the USER_DATA QoS policy of DDS DomainParticipant Entities, assigning a purpose-specific format string to it as described in 6.7 below. This approach is fast and nimble, since no additional DDS Entities beyond Domain Participants need to be created to exercise discovery of Service Instances.
- The Topic -based discovery protocol that employs a purpose-specific Topic of a well-defined type to distribute Service Instance announcements in a publish-subscribe, instance-based fashion, as descriped in 6.8 below. This protocol, although more resource-demanding (DDS Entities down to a single DataWriter need to be created for advertising, same for a DataReader in discovery), enhances interoperability and enables advanced DDS features such as persistence, routing and durability.

In the scope of these protocols three distinct Resource Identification Mechanisms are configured by <DDSServiceInstanceResourceIdentifierType>, defining how Service Interfaces and their individual Instances (the "Resources") are uniquely instantiated and addressable with a particular DDS Domain:

- The Partition -based mechanism, where DDS Publisher and Subscriber Entity PARTITION QoS policy is leveraged to isolate each Service Instance and their consumers into a uniquely named DDS Partition. De-facto choice in:
  - DomainParticipant QoS -based discovery protocol.

  But also available in:
  - Topic -based discovery protocol.
- The Topic Prefix -based mechanism, where unique Service Instance Identifiers are included in all the DDS Topic names conforming the Service Interface. Available in:
  - Topic -based discovery protocol.
- The Instance -based mechanism, where in-band (i.e. included in AUTOSAR DDS Data Types) unique Service Instance Identifier Fields are used to uniquely identify different Service Instances. Available in:
  - Topic -based discovery protocol.

6.2 Message format

Message format is defined by the OMG® DDSI-RTPS standard ([1]).

6.3 Message types

Message types are defined by the OMG® DDSI-RTPS standard ([1]).

6.4 Services / Commands

Not applicable.

6.5 Sequences (lower layer)

Sequences are defined by the OMG® DDS® standard ([3]).

6.6 Error messages

Error messages are defined by the OMG® DDSI-RTPS standard ([1]).

6.7 Service Discovery via Domain Participant USER_DATA QoS policy

6.7.1 Service Instance Advertising

To advertise a Service Instance, the following operations have to be performed:

- [FO_PRS_DDSSD_00101] Assigning a DDS DomainParticipant Entity to a Service Instance.
- [FO_PRS_DDSSD_00102] Adding the Service Interface Identifier, Service Instance Identifier, and Service Interface Contract Versions to the DDS DomainParticipant's USER_DATA QoS policy.

[FO_PRS_DDSSD_00101] Assigning a DDS DomainParticipant Entity to a Service Instance

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005

⌈One or more Service Instances enter a particular DDS Domain, as configured by <DDSServiceInstanceDomainID>, through DDS DomainParticipant Entities. Different Service Instances in the same address space and DDS Domain may share the same DDS DomainParticipant Entity, as long as they use the same DDS QoS Profile, as configured by <DDSServiceInstanceQosProfile>.

Before creating a new DDS DomainParticipant Entity, existing DDS DomainParticipant Entities in the current process that match the configuration criteria specified above shall be looked up. If the search is successful, the DDS DomainParticipant Entity found shall be assigned to the Service Instance[1]; otherwise, a new DDS DomainParticipant Entity shall be created according to the desired configuration and assigned to the Service Instance.

Once a suitable DDS DomainParticipant Entity has either been created for, or assigned to, the Service Instance, one DDS Publisher Entity and one DDS Subscriber Entity to enclose all DDS DataWriter and DataReader Entities associated with the Service Instance shall be created. If the Partition-based Resource Identification Mechanism is employed (see this chapter's Introduction above), the PARTITION QoS of both the newly created DDS Publisher and Subscriber Entities shall contain the following partition name:

```
ara.com://services/<svcId>_<svcInId>
```

Where:

<svcId> is the value of <DDSServiceInterfaceID> for the Service Interface.

[1] These rules ensure the creation of only one DomainParticipant for a given Domain and QoS Profile.

<svcInId> is the stringified value of <DDSServiceInstanceID> for the Service Instance.

DDS Publisher and Subscriber Entities may be reused across differnet elements of Discovery and Service Instances; therefore, they shall not be removed until the enclosing DDS DomainParticipant Entity being prepared for release.⌋

[FO_PRS_DDSSD_00102] Adding Service Interface Identifier, Service Instance Identifier, and Service Interface Contract Versions to the DDS DomainParticipant's USER_DATA QoS Policy

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈Service advertisement shall configure the USER_DATA QoS Policy of the DDS DomainParticipant Entity associated with the Service Instance to propagate Service Interface Identifiers, Service Instance Identifiers, and Service Interface Contract Versions, using the native DDS discovery mechanisms defined in [1].

The USER_DATA QoS Policy inserts a user-defined value into the DDS DomainParticipant Entity's Participant Announcement messages. This information shall be used by proxies and native DDS applications to identify a DDS DomainParticipant as an "AUTOSAR DDS DomainParticipant" that hosts one or more Service Instances.

Service Interface Identifiers, Service Instance Identifiers, and Service Interface Contract Versions shall be encoded in the USER_DATA QoS Policy in string format according to the following pattern:

```
ara.com://services/<svcId>_<svcInId>-<svcMajVersion>.<svcMinVersion>
[&<svcId>_<svcInId>-<svcMajVersion>.<svcMinVersion>]*
```

Where:

<svcId> is the value of <DDSServiceInterfaceID> for the Service Interface.

<svcInId> is the stringified value of <DDSServiceInstanceID> for the Service Instance.

<svcMajVersion> is the stringified value of <DDSInterfaceMajorVersion> for the Service Interface.

<svcMinVersion> is the stringified value of <DDSInterfaceMinorVersion> for the Service Interface.

Because a DDS DomainParticipant Entity may host one or more Service Instances (as noted in [FO_PRS_DDSSD_00101] above), the syntax specified above allows appending one or more <svcId>_<svcInId>-<svcMajVersion>.<svcMinVersion> tuples to the USER_DATA QoS Policy value:

- If `USER_DATA` QoS policy value is empty, it shall be set to `"ara.com://services/<svcId>_<svcInId>-<svcMajVersion>.<svcMinVersion>"`.
- Otherwise, if `USER_DATA` QoS value is not empty, the `Service Interface`, Instance Identifiers and Contract Versions shall be appended to the current value preceded by an ampersand symbol (i.e., `"&<svcId>_<svcInId>-<svcMajVersion>.<svcMinVersion>"`).

⌋

[FO_PRS_DDSSD_00103] Stoping advertisement of a `Service Instance`

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈When instructed to stop advertising a `Service Instance`, the following operations shall be performed:

- Remove the appropriate `Service Interface`, `Service Instance` Identifiers and Service Contract Versions from the `USER_DATA` QoS Policy of the DDS `DomainParticipant` Entity hosting `Service Instance` being no longer advertised.
- Release the `Service Instance` -specific DDS `Publisher` and `Subscriber` Entities specifically set up for the `Service Instance` Partition.

⌋

6.7.2 Service Instance Discovery

When instructed to discover `Service Instances`, the following operations have to be performed:

- [FO_PRS_DDSSD_00105] Finding a DDS `DomainParticipant` Entity suitable for performing client-side operations.
- [FO_PRS_DDSSD_00106] Discovering remote `Service Instance`s in DDS `DomainParticipant` Entities.

[FO_PRS_DDSSD_00105] Finding a DDS `DomainParticipant` Entity suitable for performing client-side operations

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈A DDS `DomainParticipant` Entity capable of discovering and communicating with other DDS `DomainParticipant` Entities shall be assigned for `Service Instance` Discovery:

- If an existing DDS DomainParticipant Entity exists in the same address space matching configured DDS Domain ID (<DDSServiceInstanceDomainID>) and QoS profile (<DDSServiceInstanceQosProfile>), it may be reused.
- A new DDS DomainParticipant Entity shall be created otherwise.

Please refer to [FO_PRS_DDSSD_00101] for additional details on creating/assigning DDS DomainParticipant Entity.⌋

[FO_PRS_DDSSD_00106] Discovering remote Service Instances in DDS DomainParticipant Entities

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈DDS DomainParticipant Entities created or assigned in the context of Service Instance Discovery are responsible for discovering remote DomainParticipants assigned to Service Instances (see [FO_PRS_DDSSD_00105]).

To retrieve the list of discovered Service Instances, the list of remote DDS DomainParticipant Entities a local DDS DomainParticipant Entity has discovered so far shall be iterated. This shall be done by calling read() on the DDS DomainParticipant Entity's built-in DDS DataReader Entity for the DCPSParticipant Topic. DCPSParticipant is a standard DDS Topic defined in [1] that DDS DomainParticipant Entities use to inform other DDS DomainParticipant Entities of their presence in the network.

Among other things, DCPSParticipant Topics propagate the DDS DomainParticipant Entity's USER_DATA QoS policy value, providing all the necessary information to identify remote AUTOSAR DDS DomainParticipant Entities and the Service Instances they contain.

The content of the USER_DATA QoS policy value of each remote DDS DomainParticipant Entity shall be parsed according to the patterns described in [FO_PRS_DDSSD_00102], looking for Service Instance entries matching desired Service Interface Identifier, Service Instance Identifier and Service Interface Contract Version criteria.⌋

When instructed to start a continuous Service Instance search, the following operations have to be performed:

- [FO_PRS_DDSSD_00105] Finding a DDS DomainParticipant Entity suitable for performing client-side operations.
- [FO_PRS_DDSSD_00108] Defining a DDS Built-in DomainParticipant Topic Listener.
- [FO_PRS_DDSSD_00109] Binding a DDS BuiltinParticipantListener to a DDS DomainParticipant Entity.

[FO_PRS_DDSSD_00108] Defining a DDS BuiltinParticipantListener

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈An DDS Built-in `DomainParticipant Topic Listener` to handle notifications whenever a remote DDS `DomainParticipant` Entity is discovered shall be defined. This DDS Built-in `DomainParticipant Topic Listener` implementation shall derive from the standard `DataReaderListener` class [3], specifying that the data type of the samples to be handled is `ParticipantBuiltinTopicData`—the data type associated with the built-in DDS `DataReader` for samples of `DCPSParticipant` Topic [1].

The DDS Built-in `DomainParticipant Topic Listener` shall implement the following callbacks:

- `on_data_available()`, to notify when:
  - A new `Service Instance` is discovered in a DDS `DomainParticipant` Entity according to [FO_PRS_DDSSD_00106].
  - A known `Service Instance` ceases to be contained by a DDS `DomainParticipant` Entity according to [FO_PRS_DDSSD_00106].
- `on_liveliness_changed()`, to notify when:
  - A DDS `DomainParticipant` Entity assigned to a `Service Instance` ceases to exist (i.e., the instance state is either `NOT_ALIVE_DISPOSED` or `NOT_ALIVE_NO_WRITERS`).

⌋

[FO_PRS_DDSSD_00109] Binding a BuiltinParticipantListener to a DDS `DomainParticipant` Entity

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈To bind a DDS Built-in `DomainParticipant Topic Listener` to a DDS `DomainParticipant` Entity, a new DDS Built-in `DomainParticipant Topic Listener` object (see [FO_PRS_DDSSD_00108]) shall be created. Then the newly created DDS Built-in `DomainParticipant Topic Listener` shall be bound to the DDS `DomainParticipant` using `set_listener()` with `StatusMask = DATA_AVAILABLE_STATUS|DDS_DATA_AVAILABLE_STATUS`[2].

The DDS Built-in `DomainParticipant Topic Listener` shall be unbound when the enclosing DDS `DomainParticipant` Entity is destroyed accorting to [FO_PRS_DDSSD_00111].⌋

[2]Note that the syntax of `set_listener()` and `StatusMask` is described in terms of the DDS Platform-Independent Model specified in [3]. Different Platform-Specific Mappings, such as the DDS-CPP-PSM specified in [4], map these concepts into more language-friendly constructs.

When instructed to stop asynchronous `Service Instance` discovery, the following operations have to be performed:

- [FO_PRS_DDSSD_00111] Unbinding a DDS Built-in `DomainParticipant Topic Listener` from a DDS `DomainParticipant` Entity.

[FO_PRS_DDSSD_00111] Unbinding a BuiltinParticipantListener from a DDS `DomainParticipant` Entity

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈When instructed to unbind a DDS Built-in `DomainParticipant Topic Listener` from a DDS `DomainParticipant` Entity, `set_listener()` to disable the DDS Built-in `DomainParticipant Topic Listener` shall be called with `StatusMask = STATUS_MASK_NONE`, then the DDS Built-in `DomainParticipant Topic Listener` object shall be released.⌋

6.8 Service Discovery via Topic

6.8.1 Service Instance Advertising

To advertise a Service Instance, the following operations have to be performed:

- [FO_PRS_DDSSD_00201] Assigning a DDS DomainParticipant Entity to a Service Instance.
- [FO_PRS_DDSSD_00202] Publishing the Service Interface Identifier, Service Instance Identifier, Service Interface Contract Versions and Resource Identifier Type over the ara.com://services/discovery DDS Topic.

[FO_PRS_DDSSD_00201] Assigning a DDS DomainParticipant Entity to a Service Instance

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00008, FO_RS_Dds_00015, FO_RS_Dds_00016

⌈One or more Service Instances enter a particular DDS Domain, as configured by <DDSServiceInstanceDomainID>, through DDS DomainParticipant Entities. Different Service Instances in the same address space and DDS Domain may share the same DDS DomainParticipant Entity, as long as they use the same DDS QoS Profile, as configured by <DDSServiceInstanceQosProfile>.

Before creating a new DDS DomainParticipant Entity, existing DDS DomainParticipant Entities in the current process that match the configuration criteria specified above shall be looked up. If the search is successful, the DDS DomainParticipant Entity found shall be assigned to the Service Instance[3]; otherwise, a new DDS DomainParticipant Entity shall be created according to the desired configuration and assigned to the Service Instance.

Once a suitable DDS DomainParticipant Entity has either been created for, or assigned to, the Service Instance, one DDS Publisher Entity and one DDS Subscriber Entity to enclose all DDS DataWriter and DataReader Entities associated with the Service Instance shall be created. If the Partition-based Resource Identification Mechanism is employed (see this chapter's Introduction above), the PARTITION QoS of both the newly created DDS Publisher and Subscriber Entities shall contain the following partition name:

ara.com://services/<svcId>_<svcInId>

Where:

<svcId> is the value of <DDSServiceInterfaceID> for the Service Interface.

[3]These rules ensure the creation of only one DomainParticipant for a given Domain and QoS Profile.

<svcInId> is the stringified value of <DDSServiceInstanceID> for the Service Instance.

DDS Publisher and Subscriber Entities may be reused across different elements of Discovery and Service Instances; therefore, they shall not be removed until the enclosing DDS DomainParticipant Entity is being prepared for release.⌋

[FO_PRS_DDSSD_00202] Advertising Service Interface Identifier, Service Instance Identifier, and ServiceInterface Contract Versions and Resource Identifier Type over the ara.com://services/discovery DDS Topic

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00007, FO_RS_Dds_00008, FO_RS_Dds_00015

⌈Service advertisement shall configure DDS Topic, Publisher and DataWriter objects supporting the publication of announcement messages over the following topic:

ara.com://services/discovery

whose type is ServiceAnnouncementMessage and is defined as follows[4]:

```
module dds {
module ara {
module com {

enum ServiceInstanceResourceIdentifierType {
        SERVICE_INSTANCE_RESOURCE_PARTITION,
        SERVICE_INSTANCE_RESOURCE_TOPIC_PREFIX,
        SERVICE_INSTANCE_RESOURCE_INSTANCE_ID
};

struct ServiceVersion {
        uint32 major_version;
        uint32 minor_version;
};

struct ServiceAnnouncementMessage {
        @key string<256> interface_id;
        @key uint16 instance_id;
        ServiceVersion version;
        ServiceInstanceResourceIdentifierType identifier_type;
};

}; // module com
}; // module ara
}; // module dds
```

Where:

[4]DDS types are often defined in OMG IDL [5], which provides a standard language-independent format to represent data types and interfaces. Even though we use IDL throughout the specification to define data types, the use of IDL to is not mandated (i.e., a compliant implementation could choose to hand-craft these types, run code generation from an equivalent XML syntax, or run vendor-specific mechanisms to generate the actual data types).

interface_id is the value of <DDSServiceInterfaceID> for the Service Interface.

instance_id is the value of <DDSServiceInstanceID> for the Service Instance.

version.major_version is the value of <DDSInterfaceMajorVersion> for the Service Interface.

version.minor_version is the value of <DDSInterfaceMinorVersion> for the Service Interface.

identifier_type is the value of <DDSServiceInstanceResourceIdentifierType>.

In order to guarantee reception of ServiceAnnouncementMessage samples by all Service Interface consumers, including those joining after the Service Instance has been advertised, the following DDS DataWriter QoS policies shall be set for the ara.com://services/discovery topic:

- RELIABILITY set to RELIABLE
- HISTORY set to KEEP_LAST with DEPTH set to 1
- DURABILITY set to TRANSIENT_LOCAL

Once the ara.com://services/discovery topic DDS DataWriter is properly set up and ready to use, the offering Service Instance shall:

1. Instantiate a ServiceAnnouncementMessage sample, update it with the proper values uniquely identifying the Service Instance, and use it to register via register_instance() a unique instance (keyed by interface_id and instance_id)
2. Use the Instance Handle returned by the previous step to publish the sample via write()
3. Keep a copy the sample and the Instance Handle for use upon Service Instance tear down (see [FO_PRS_DDSSD_00203])

⌋

[FO_PRS_DDSSD_00203] Stopping advertisement of a Service Instance

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00008

⌈When instructed to stop advertising a Service Instance, the following operations shall be performed:

- Call dispose() using the sample and the Instance Handle kept during Service Instance announcement (see [FO_PRS_DDSSD_00202])

⌋

6.8.2 Service Instance Discovery

When instructed to find remote Service Instances, the following operations have to be performed:

- [FO_PRS_DDSSD_00205] Finding a DDS DomainParticipant Entity suitable for performing client-side operations.
- [FO_PRS_DDSSD_00206] Discovering remote Service Instances in DDS DomainParticipant Entities.

[FO_PRS_DDSSD_00205] Finding a DDS DomainParticipant Entity suitable for performing client-side operations

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00005, FO_RS_Dds_00008

⌈A DDS DomainParticipant Entity capable of discovering and communicating with other DDS DomainParticipant Entities shall be assigned for Service Instance Discovery:

- If an existing DDS DomainParticipant Entity exists in the same address space matching configured DDS Domain ID (<DDSServiceInstanceDomainID>) and QoS profile (<DDSServiceInstanceQosProfile>), it may be reused.
- A new DDS DomainParticipant Entity shall be created otherwise.

Please refer to [FO_PRS_DDSSD_00201] for additional details on creating/assigning DDS DomainParticipant Entity.⌋

[FO_PRS_DDSSD_00206] Discovering remote Service Instances through the ara.com://services/discovery topic

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00008

⌈DDS DomainParticipant, Subscriber and DataReader Entities created or assigned in the context of Service Instance Discoverty are responsible for discovering remote DDS DomainParticipants assigned to Service Instances (see [FO_PRS_DDSSD_00205]).

To retrieve a list of available Service Instances, inbound ServiceAnnouncementMessage samples from the ara.com://services/discovery topic shall be received and processed. This shall be done by calling read() on the DDS DataReader Entity matching the DDS Topic and QoS policies defined by [FO_PRS_DDSSD_00202].

The content of the ServiceAnnouncementMessage samples shall be parsed according to the patterns described in [FO_PRS_DDSSD_00202], looking for Service Instance entries matching desired Service Interface Identifier, Service Instance Identifier and Service Interface Contract Version criteria.⌋

When instructed to start a continuous `Service Instance` search, the following operations have to be performed:

- [FO_PRS_DDSSD_00205] Finding a DDS `DomainParticipant` Entity suitable for performing client-side operations.
- [FO_PRS_DDSSD_00206] Creating DDS `Topic` Entity and monitoring, via DDS asynchronous operation features such as Listeners and WaitSets, the `ara.com://services/discovery` topic.

[FO_PRS_DDSSD_00208] Stopping asyncrhonous discovery of e `Service Instances` in DDS `DomainParticipant` Entities

Status: DRAFT
Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00008

⌈When instructed to stop asynchronous `Service Instance` discovery, monitoring the arrival of `ServiceAnnouncementMessage` samples through the `ara.com://services/discovery` topic shall stop.⌋

7 Configuration parameters

This chapter lists all parameters the DDS Service Discovery protocol uses.

7.1 Service Discovery

DDS Protocol Parameter	Description	AP Config	CP Config
<DDSServiceInterfaceID>	ID of the DDS Service Interface	DdsServiceInterfaceDeployment.serviceInterfaceId	-
<DDSServiceInstanceID>	ID of the DDS Service Instance	DdsProvidedServiceInstance.serviceInstanceId and DdsRequiredServiceInstance.requiredServiceInstanceId	-
<DDSServiceInstanceDomainID>	ID of the DDS Domain where a DDS Service Instance operates	DdsProvidedServiceInstance.domainId and DdsRequiredServiceInstance.domainId	-
<DDSInterfaceMajorVersion>	Major Version of the DDS Service Interface	ServiceInterface.majorVersion	-
<DDSInterfaceMinorVersion>	Minor Version of the DDS Service Interface	ServiceInterface.minorVersion	-
<DDSServiceInstanceQosProfile>	QoS Profile for all Methods and Field Methods within a specific DDS Service Instance	DdsRequiredServiceInstance.qosProfile	-
<DDSServiceInstanceDiscoveryType>	Discovery mechanism for DDS Service Instance	DdsProvidedServiceInstance.discoveryType and DdsRequiredServiceInstance.discoveryType	-
<DDSServiceInstanceRessourceIdentifierType>	Resource Identification scheme for DDS Service Instance	DdsProvidedServiceInstance.resourceIdentifierType	-

Table 7.1: Mapping Table - DDS Protocol Parameters

8 Protocol usage and guidelines

This section is intentionally left empty.

A Appendix

This section is intentionally left empty.

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

Number	Heading
[FO_PRS_DDSSD_-00101]	Assigning a DDS `DomainParticipant` Entity to a `Service Instance`
[FO_PRS_DDSSD_-00102]	Adding `Service Interface` Identifier, `Service Instance` Identifier, and `Service Interface` Contract Versions to the DDS Domain Participant's USER_DATA QoS Policy
[FO_PRS_DDSSD_-00103]	Stoping advertisement of a `Service Instance`
[FO_PRS_DDSSD_-00105]	Finding a DDS `DomainParticipant` Entity suitable for performing client-side operations
[FO_PRS_DDSSD_-00106]	Discovering remote `Service Instance`s in DDS `DomainParticipant` Entities
[FO_PRS_DDSSD_-00108]	Defining a DDS BuiltinParticipantListener
[FO_PRS_DDSSD_-00109]	Binding a BuiltinParticipantListener to a DDS `DomainParticipant` Entity
[FO_PRS_DDSSD_-00111]	Unbinding a BuiltinParticipantListener from a DDS `DomainParticipant` Entity
[FO_PRS_DDSSD_-00201]	Assigning a DDS `DomainParticipant` Entity to a `Service Instance`
[FO_PRS_DDSSD_-00202]	Advertising `Service Interface` Identifier, `Service Instance` Identifier, and ServiceInterface Contract Versions and Resource Identifier Type over the `ara.com://services/discovery` DDS Topic
[FO_PRS_DDSSD_-00203]	Stopping advertisement of a `Service Instance`
[FO_PRS_DDSSD_-00205]	Finding a DDS `DomainParticipant` Entity suitable for performing client-side operations
[FO_PRS_DDSSD_-00206]	Discovering remote `Service Instance`s through the ara.com://services/discovery topic

▽

△

Number	Heading
[FO_PRS_DDSSD_-00208]	Stopping asyncrhonous discovery of e `Service Instance`s in DDS `DomainParticipant` Entities

Table B.1: Added Specification Items in R24-11

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none