European Component Oriented Architecture (ECOA) Collaboration Programme: Architecture Specification Part 10: Ada Language Binding

BAE Ref No: IAWG-ECOA-TR-003
Dassault Ref No: DGT 144476-C

Issue: 3

Prepared by
BAE Systems (Operations) Limited and Dassault Aviation

This specification is developed by BAE SYSTEMS, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés . AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE SYSTEMS, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés . AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Note: *This specification represents the output of a research programme and contains mature high-level concepts, though low-level mechanisms and interfaces remain under development and are subject to change. This standard of documentation is recommended as appropriate for limited lab-based evaluation only. Product development based on this standard of documentation is not recommended.*

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Contents

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Tables

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

0 Introduction

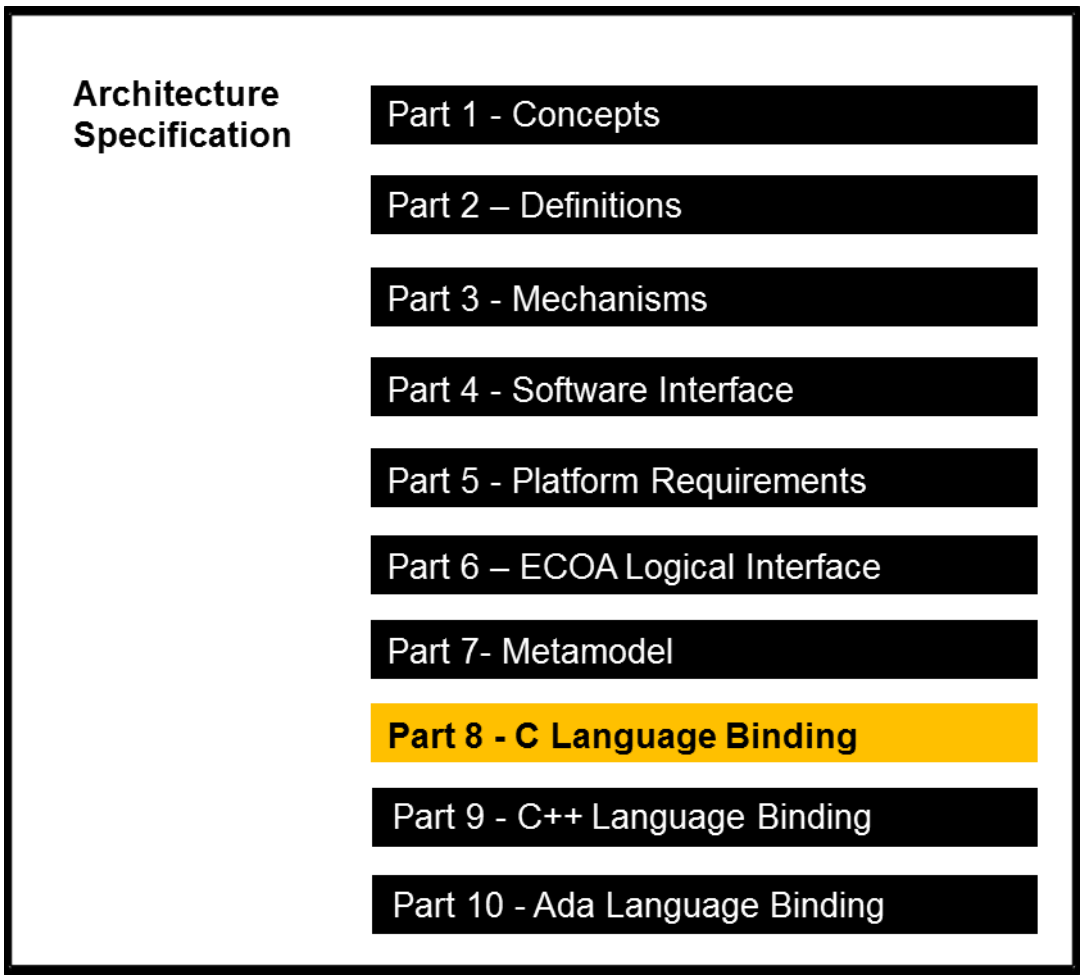

Figure 1 ECOA Documentation

This Architecture Specification provides the definitive specification for creating ECOA-based systems. It describes the standardised programming interfaces and data-model that allow a developer to construct an ECOA-based system. The details of the other documents comprising the rest of this Architecture Specification can be found in Section 3.

This document is Part 10 of the Architecture Specification, and describes the Ada 95 (reference ISO/IEC 8652:1995(E) with COR.1:2000) language binding for the module and container APIs that facilitate communication between the module instances and their container in an ECOA system.

The document is structured as follows:

- Section 6 describes the Module to Language Mapping;
- Section 7 describes the method of passing parameters;
- Section 8 describes the Module Context;
- Section 9 describes the pre-defined types that are provided and the types that can be derived from them;
- Section 10 describes the Module Interface;
- Section 11 describes the Container Interface;

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

- Section 12 describes the Fault Handler Interface;
- Section 13 provides a reference Ada specification for the ECOA package, usable in any Ada binding implementation;

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

1 Scope

This purpose of this Architecture Specification is to establish a uniform method for design, development and integration of software systems using a component oriented approach.

2 Warning

This specification represents the output of a research programme and contains mature high-level concepts, though low-level mechanisms and interfaces remain under development and are subject to change. This standard of documentation is recommended as appropriate for limited lab-based evaluation only. Product development based on this standard of documentation is not recommended.

3 Normative References

Ref	Description
Architecture Specification Part 1	
	IAWG-ECOA-TR-001 / DGT 144474
	Issue 3
	Architecture Specification Part 1 – Concepts
Architecture Specification Part 2	
	IAWG-ECOA-TR-012 / DGT 144487
	Issue 3
	Architecture Specification Part 2 – Definitions
Architecture Specification Part 3	
	IAWG-ECOA-TR-007 / DGT 144482
	Issue 3
	Architecture Specification Part 3 – Mechanisms
Architecture Specification Part 4	
	IAWG-ECOA-TR-010 / DGT 144485
	Issue 3
	Architecture Specification Part 4 – Software Interface
Architecture Specification Part 5	
	IAWG-ECOA-TR-008 / DGT 144483
	Issue 3
	Architecture Specification Part 5 – Platform Requirements
Architecture Specification Part 6	
	IAWG-ECOA-TR-006 / DGT 144481
	Issue 3
	Architecture Specification Part 6 – ECOA Logical Interface
Architecture Specification Part 7	
	IAWG-ECOA-TR-011 / DGT 144486
	Issue 3

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Architecture Specification Part 7 – Metamodel

Architecture Specification Part 8

IAWG-ECOA-TR-004 / DGT 144477

Issue 3

Architecture Specification Part 8 – C Language Binding

Architecture Specification Part 9

IAWG-ECOA-TR-005 / DGT 144478

Issue 3

Architecture Specification Part 9 – C++ Language Binding

Architecture Specification Part 10

IAWG-ECOA-TR-003 / DGT 144476

Issue 3

Architecture Specification Part 10 – Ada language Binding

ISO/IEC 8652:1995(E) with COR.1:2000

Ada95 Reference Manual

Issue 1

ISO/IEC 9899:1999(E) Programming Languages – C

ISO/IEC 14882:2003(E) Programming Languages C++

4 Definitions

For the purpose of this standard, the definitions given in Architecture Specification Part 2.

5 Abbreviations

API	Application Programming Interface
ECOA	European Component Oriented Architecture
UK	United Kingdom
UTC	Coordinated Universal Time
XML	eXtensible Markup Language

6 Module to Language Mapping

This section gives an overview of the Module and Container APIs, in terms of filename and the overall structure of the files.

The Ada 95 language allows tagged types (which allow object-oriented behaviour), however the Ada bindings will not use tagged types. This corresponds to traditional use within the avionics industry in the UK. Therefore the mapping is similar to C, apart from support for proper namespacing using Packages. The filename mapping is specified in Table 1.

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

The Module Interface will be composed of a set of procedures corresponding to each entry-point of the Module Implementation. The declaration of these procedures will be accessible in a package spec file called `#module_impl_name#.ads`.

The Container Interface will be composed of a set of procedures corresponding to the required operations. The declaration of these procedures will be accessible in a package spec file called `#module_impl_name#_Container.ads`.

A dedicated structure named Context_Type, and called Module Context structure in the rest of the document will be generated by the ECOA toolchain in the Module Container specification (`#module_impl_name#_Container.ads`) and shall be extended by the Module implementer to contain all the user variables of the Module. This structure will be allocated by the container before Module Instance start-up and passed to the Module Instance in each activation entry-point (i.e. received events, received request-response and asynchronous request-response sent call-back).

Table 1 Filename Mapping for Ada 95

Filename	Use
#module_impl_name#.ads	Package **#module_impl_name#** specifies the module interface.
#module_impl_name#.adb	Package body **#module_impl_name#** implements the module interface.
#module_impl_name#_Container.{ads|adb}	Package **#module_impl_name#**_Container specifies and implements the container Interface (functions provided by the container and callable by the module). It also specifies the standard module context information.
#module_impl_name#_User_Context.ads	Extensions to Module Context.

Templates for the files in Table 1 are provided below:

6.1 Module Interface Template

```
-------------------------------------------------------------------
-- @file "#module_impl_name#.ads"
-- This is the Module Interface package spec. for Module #module_impl_name#
-- This file is generated by the ECOA tools and shall not be modified.
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;
-- Include container
with #module_impl_name#_Container#;

package #module_impl_name# is

   -- Event operation handlers specifications

   #list of event operations specifications#

   -- Request-Response operation handlers specifications

   #list_of_request_response_operations_specifications#

   -- Lifecycle operation handlers specifications
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
   #list_of_lifecycle_operations_specifications#

end #module_impl_name#;
```

```
-------------------------------------------------------------------
-- @file "#module impl name#.adb"
-- This is the Module Interface package for Module #module_impl_name#
-- This file can be considered a template with the operation stubs
-- autogenerated by the ECOA toolset and filled in by the module
-- developer.
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;
-- Include container
with #module_impl_name#_Container#;
-- Additional children or other packages implementing the module
with #additional_with_clauses#;

package body #module_impl_name# is


   -- Event operation handlers

   #list_of_event_operations#

   -- Request-Response operation handlers

   #list_of_request_response_operations#

   -- Lifecycle operation handlers

   #list_of_lifecycle_operations#

end module_impl_name#;
```

6.2 Container Interface Template

```
-------------------------------------------------------------------
-- @file "#module_impl_name#_Container.ads"
-- This is the Module Container package for Module #module impl name#
-- This file is generated by the ECOA tools and shall not be modified.
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;
-- Include module user context
with #module_impl_name#_User_Context;

package #module_impl_name#_Container is

   -- Module Implementation Context data type is specified here. This enables a
   -- module instance to hold its own private data in a non-OO fashion.

   type Context_Type is record
        -- Standard container context information
        Operation_Timestamp : ECOA.Timestamp_Type;

        -- A hook to implementation dependant private data
        Platform_Hook : System.Address;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
      -- Information that is private to a module implementation
      User_Context        : #module_impl_name#_User_Context.User_Context_Type;
   end record;

   -- Event operation call specifications
   #event_operation_call_specifications#

   -- Request-response call specifications
   #request_response_call_specifications#

   -- Versioned data call specifications
   #versioned_data_call_specifications#

   -- Functional parameters call specifications
   #propertys_call_specifications#

   -- Logging services API call specifications
   #logging_services_call_specifications#

   -- Time Services API call specifications
   #time_services_call_specifications#


end #module_impl_name#_Container;
```

6.3 User Module Context Template

```
-------------------------------------------------------------------
-- @file "#module_impl_name#_User_Context.ads"
-- This is the module implementation private user context data type
-- that is included in the module context.
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;

package #module_impl_name#_User_Context is

   type User_Context_Type is record
       -- Declare the User Module Context "local" data here.

   end record;

end module_impl_name#_User_Context;
```

7 Parameters

In the Ada programming language, the manner in which parameters are passed is specified as '**in**', '**out**' or '**in out**'. '**in'** Parameters are only passed into a procedure; '**out**' parameters are only passed out from a procedure; and **'in out**' parameters are passed in, modified and passed out from a procedure. The compiler then makes an appropriate choice as to whether to pass-by-value or pass-by-reference.

Table 2 Parameter Typing

	Input parameter	*Output parameter*	*Input and Output parameter*
Simple type	**in**	**out**	**in out**
Complex type	**in**	**out**	**in out**

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

NOTE: within the API bindings, parameters are passed as '**in**' if the behaviour of the specific API warrants it, overriding the standard conventions defined above

8 Module Context

In the Ada language, the Module Context is a structure which holds both the user local data (called "User Module Context") and Infrastructure-level technical data (which is implementation dependant). The structure is defined in the Container Interface.

The following shows the Ada syntax for the Module Context:

```
-------------------------------------------------------------------
-- @file "#module_impl_name#_Container.ads"
-- This is the Module Container package for Module #module_impl_name#
-- This file is generated by the ECOA tools and shall not be modified.
-------------------------------------------------------------------
with System;

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;
-- Include module user context
with #module_impl_name#_User_Context;

package #module_impl_name#_Container is

   -- Module Implementation Context data type is specified here. This enables a
   -- module instance to hold its own private data in a non-OO fashion.

   type Context_Type is record
-- Standard container context information
Operation_Timestamp : ECOA.Timestamp_Type;

-- A hook to implementation dependant private data
Platform_Hook : System.Address;

-- Information that is private to a module implementation
      User Context : #module impl name# User Context.User Context Type;
   end record;

-- ...

end #module_impl_name#_Container;
```

8.1 User Module Context

The Ada syntax for the user context is shown below (including an example data item; `My_Counter`):

```
-------------------------------------------------------------------
-- @file "#module_impl_name#_User_Context.ads"
-- This is the module implementation private user context data type
-- that is included in the module context.
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;

package #module_impl_name#_User_Context is

   type User_Context_Type is record
       -- Example user context
      My_Counter : ECOA.Unsigned_8_Type;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
   end record;

end module_impl_name#_User_Context;
```

The following example illustrates the usage of the Module context in the entry-point corresponding to an event-received:

```
-------------------------------------------------------------------
-- @file "#module_impl_name#.adb"
-- Generic operation implementation example
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;
-- Include container
with #module impl name# Container#;
-- Additional children or other packages implementing the module
with #additional_with_clauses#;


package body #module_impl_name# is


   procedure #operation_name#_Received
     (Context : in out #module_impl_name#_Container.Context_Type;
      #parameters#)
   is

   begin

   -- To be implemented by the module.

   -- Increments a local user defined counter.
   Context.User_Context.My_Counter := Context.User_Context.My_Counter + 1;

   end #operation_name#_Received;


end module_impl_name#;
```

NB: currently, the user extensions to Module Context need to be known by the container in order to allocate the required memory area. This means that the component supplier is requested to provide the associated header file. If the supplier does not want to divulge the original contents of the header file, then:

- It may be replaced by an array with a size equivalent to the original data; or
- Memory management may be dealt with internally to the code, using memory allocation functions[1].

To extend the Module Context structure, the module implementer shall define the User Module Context structure, named `#module_impl_name#_User_Context`, in a package spec file called `#module_impl_name#_User_Context.ads`. All the private data of the Module Implementation shall be added as members of this record, and will be accessible within the "User_Context" field of the Module Context.

[1] The current ECOA architecture specification does not specify any memory allocation function. So, this case may lead to non portable code.

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

The Module Context structure will be passed by the Container to the Module as the first parameter for each operation that will activate the Module instance (i.e. received events, received request-response and asynchronous request-response sent call-back). This structure shall be passed by the Module to all container interface API functions it can call.

The Module Context will also be used by the Container to automatically timestamp operations on the emitter/requester side using an ECOA-provided attribute called `operation_timestamp`. The Container also provides a utility function to retrieve this from the Module Instance Context. The way this structure is populated by the ECOA infrastructure is detailed in reference ISO/IEC 8652:1995(E) with COR.1:2000.

9 Types

This section describes the convention for creating namespaces, and how the ECOA pre-defined types and derived types are represented in Ada.

9.1 Filenames and Namespace

The type definitons are contained within one or more namespaces: all types for specific namespace `#namespace1#_#namespace2#_[…]-#namespacen#` shall be placed in a file called `#namespace1#_#namespace2#_[…]-#namespacen#.ads`.

The syntax that follows shall be used to declare variable `#variable_name#` of data type `#data_type_name#:`

```
--
-- @file #namespace1# -#namespace2# -[…]-#namespacen#.ads
-- This is data-type declaration file
-- This file is generated by the ECOA tools and shall not be modified
--

package #namespace1#.#namespace2#.[…].#namespacen# is

#variable_name# : #data_type_name#;
-- Other definitions

end #namespace1#.#namespace2#.[…].#namespacen#;
```

9.2 Predefined Types

Predefined types in Ada 95, shown in Table 3, shall be located in the “ECOA” namespace and hence in `ECOA.ads`.

Table 3 Ada 95 ECOA Types

ECOA Predefined Type	Ada 95 Type
ECOA:boolean8	ECOA.Boolean_8_Type
ECOA:int8	ECOA.Signed_8_Type
ECOA:char8	ECOA.Character_8_Type
ECOA:byte	ECOA.Byte_Type
ECOA:int16	ECOA.Signed_16_Type

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

ECOA:int32	ECOA.Signed_32_Type
ECOA:int64	ECOA.Signed_64_Type
ECOA:uint8	ECOA.Unsigned_8_Type
ECOA:uint16	ECOA.Unsigned_16_Type
ECOA:uint32	ECOA.Unsigned_32_Type
ECOA:uint64	ECOA.Unsigned_64_Type
ECOA:float32	ECOA.Float_32_Type
ECOA:double64	ECOA.Float_64_Type

Ada provides the 'First and 'Last attributes, so there is no requirement to refer to explicit constants for the maximum and minimum values of the type range.

All predefined types shall be specified with a representation clause to ensure they occupy the correct number of bits, and have the correct alignment.

The data types described in the following sections are also defined in the ECOA namespace.

9.2.1 ECOA:return_status

In Ada `ECOA:return_status` translates to `ECOA.Return_Status_Type`, with the enumerated values shown below:

```
package ECOA is

-- ...

type Return_Status_Type is new Unsigned_32_Type;
    Return_Status_Type_OK                          : constant Return_Status_Type := 0;
    Return_Status_Type_INVALID_HANDLE              : constant Return_Status_Type := 1;
    Return_Status_Type_DATA_NOT_INITIALIZED        : constant Return_Status_Type := 2;
    Return_Status_Type_NO_DATA                     : constant Return_Status_Type := 3;
    Return_Status_Type_INVALID_IDENTIFIER          : constant Return_Status_Type := 4;
    Return_Status_Type_NO_RESPONSE                 : constant Return_Status_Type := 5;
    Return_Status_Type_OPERATION_ALREADY_PENDING   : constant Return_Status_Type := 6;
    Return_Status_Type_INVALID_SERVICE_ID          : constant Return_Status_Type := 7;
    Return_Status_Type_CLOCK_UNSYNCHRONIZED        : constant Return_Status_Type := 8;
    Return_Status_Type_INVALID_TRANSITION          : constant Return_Status_Type := 9;
    Return_Status_Type_RESOURCE_NOT_AVAILABLE      : constant Return_Status_Type := 10;
    Return_Status_Type_OPERATION_NOT_AVAILABLE     : constant Return_Status_Type := 11;
    Return_Status_Type_PENDING_STATE_TRANSITION    : constant Return_Status_Type := 12;

-- ...

end ECOA;
```

9.2.2 ECOA:hr_time

In Ada, Seconds and Nanosecond types are defined as follows:

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
package ECOA is

-- ...

  type Seconds_Type is mod 2 ** 32;
  for Seconds_Type'Size use 32;
  for Seconds_Type'Alignment use 4;

  type Nanoseconds_Type is range 0 .. 10 ** 9 - 1;
  for Nanoseconds_Type'Size use 32;
  for Nanoseconds_Type'Alignment use 4;

-- ...

end ECOA;
```

The binding for time is:

```
package ECOA is

-- ...

type HR_Time_Type is
   record
      Seconds     : Seconds_Type;
      Nanoseconds : Nanoseconds Type;
   end record;
for HR_Time_Type'size use 64;
for HR_Time_Type'Alignment use 4;

-- ...

end ECOA;
```

9.2.3 ECOA:global_time

Global time is defined as:

```
package ECOA is

-- ...

type Global_Time_Type is
   record
      Seconds     : Seconds_Type;
      Nanoseconds : Nanoseconds Type;
   end record;
  for Global_Time_Type'size use 64;
  for Global_Time_Type'Alignment use 4;

-- ...

end ECOA;
```

9.2.4 ECOA:duration

Duration is defined as:

```
package ECOA is

-- ...
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
type Duration_Type is
   record
      Seconds     : Seconds_Type;
      Nanoseconds : Nanoseconds_Type;
   end record;
  for Duration_Type'size use 64;
  for Duration_Type'Alignment use 4;

-- ...

end ECOA;
```

9.2.5 ECOA:timestamp

The syntax for defining a timestamp, for use by operations etc., is:

```
package ECOA is

-- ...

type Timestamp_Type is
   record
      Seconds     : Seconds_Type;
      Nanoseconds : Nanoseconds Type;
   end record;
  for Timestamp_Type'size use 64;
  for Timestamp_Type'Alignment use 4;

-- ...

end ECOA;
```

9.2.6 ECOA:log

The syntax for a log is:

```
package ECOA is

-- ...

type Log_Elements_Index_Type is range 0..255;
type Log_Elements_Type is array (Log_Elements_Index_Type) of ECOA.Character_8_Type;

type Log Type is
   record
      Current_Size : Log_Elements_Index_Type;
      Data         : Log_Elements_Type;
   end record;
for Log_Elements_Type'size use 2048;
for Log_Elements_Type'Alignment use 4;

-- ...

end ECOA;
```

9.2.7 ECOA:module_states_type

In Ada `ECOA:module_states_type` translates to `ECOA.Module_States_Type`, with the enumerated values shown below:

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
package ECOA is

-- ...

type Module_States_Type is new Unsigned_32_Type;
Module_States_Type_IDLE    : constant Module_States_Type := 0;
Module_States_Type_READY   : constant Module_States_Type := 1;
Module_States_Type_RUNNING : constant Module_States_Type := 2;

-- ...

end ECOA;
```

9.2.8 ECOA:module_error_type

In Ada `ECOA:module_error_type` translates to `ECOA.Module_Error_Type`, with the enumerated values shown below:

```
package ECOA is

-- ...

type Module_Error_Type is new Unsigned_32_Type;
Error_Type_ERROR       : constant Module_Error_Type := 0;
Error_Type_FATAL_ERROR : constant Module_Error_Type := 1;

-- ...

end ECOA;
```

9.2.9 ECOA:error_id

In C the syntax for an `ECOA:error_id` is:

```
package ECOA is

-- ...

type Error Id is new Unsigned 32 Type;

 -- ...

end ECOA;
```

9.2.10 ECOA:asset_id

In C the syntax for an `ECOA:asset_id` is:

```
package ECOA is

-- ...

type Asset Id is new Unsigned 32 Type;

-- ...

end ECOA;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

9.2.11 ECOA:asset_type

In C `ECOA:asset_type` translates to `ECOA.Asset_Type`, with the enumerated values shown below:

```
package ECOA is

-- ...

type Asset_Type is new Unsigned_32_Type;
Asset_Type_COMPONENT         : constant Asset_Type := 0;
Asset_Type_PROTECTION_DOMAIN : constant Asset_Type := 1;
Asset_Type_NODE              : constant Asset_Type := 2;
Asset_Type_PLATFORM          : constant Asset_Type := 3;
Asset_Type_SERVICE           : constant Asset_Type := 4;
Asset_Type_DEPLOYMENT        : constant Asset_Type := 5;

-- ...

end ECOA;
```

9.2.12 ECOA:error_type

In C `ECOA:error_type` translates to `ECOA.Error_Type`, with the enumerated values shown below:

```
package ECOA is

-- ...

type Error_Type is new Unsigned_32_Type;;
Error Type RESOURCE NOT AVAILABLE : constant Error Type := 0;
Error Type UNAVAILABLE            : constant Error Type := 1;
Error_Type_MEMORY_VIOLATION       : constant Error_Type := 2;
Error_Type_NUMERICAL_ERROR        : constant Error_Type := 3;
Error_Type_ILLEGAL_INSTRUCTION    : constant Error_Type := 4;
Error_Type_STACK_OVERFLOW         : constant Error_Type := 5;
Error Type DEADLINE VIOLATION     : constant Error Type := 6;
Error Type OVERFLOW               : constant Error Type := 7;
Error_Type_UNDERFLOW              : constant Error_Type := 8;
Error_Type_ILLEGAL_INPUT_ARGS     : constant Error_Type := 9;
Error_Type_ILLEGAL_INPUT_ARGS     : constant Error_Type := 10;
Error_Type_ERROR                  : constant Error_Type := 11;
Error Type FATAL ERROR            : constant Error Type := 12;
Error_Type_HARDWARE_FAULT         : constant Error_Type := 13;
Error_Type_POWER_FAIL             : constant Error_Type := 14;
Error_Type_COMMUNICATION_ERROR    : constant Error_Type := 15;
Error_Type_INVALID_CONFIG         : constant Error_Type := 16;
Error Type INITIALISATION PROBLEM : constant Error Type := 17;
Error_Type_CLOCK_UNSYNCHRONIZED   : constant Error_Type := 18;
Error_Type_UNKNOWN_OPERATION : constant Error_Type := 19;
Error_Type_OPERATION_OVERRATED    : constant Error_Type := 20;
Error_Type_OPERATION_UNDERRATED   : constant Error_Type := 21;

-- ...

end ECOA;
```

9.2.13 ECOA:recovery_action_type

In C `ECOA:recovery_action_type` translates to `ECOA.Recovery_Action_Type`, with the enumerated values shown below:

```
package ECOA is
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
-- ...

type Recovery_Action_Type is new Unsigned_32_Type;
Recovery_Action_Type_SHUTDOWN_COMPONENT : constant Recovery_Action_Type := 0;
Recovery_Action_Type_COLD_RESTART       : constant Recovery_Action_Type := 1;
Recovery_Action_Type_WARM_RESTART       : constant Recovery_Action_Type := 2;
Recovery_Action_Type_CHANGE_DEPLOYMENT  : constant Recovery_Action_Type := 3;

-- ...

end ECOA;
```

9.3 Derived Types

This Section describes the derived types that can be constructed from the ECOA pre-defined types.

9.3.1 Simple Types

The Ada syntax for a Simple Type called "`#simple_type_name#`" with an optional restricted range, which is derived from a Predefined Type is:

```
type #simple_type_name# is new #predef_type_name# range #min# .. #max#;
```

9.3.2 Constants

The syntax for declaring a constant called "`#constant_name#`" of type `#type_name#` in Ada is as follows:

```
#constant_name# : constant #type_name# := #constant_value#;
```

Where `#constant_value#` is either an integer or a floating-point value, compatible with the type.

9.3.3 Enumerations

For an enumerated type named `#enum_type_name#`, a set of constants named from `#enum_value_name_1#` to `#enum_value_name_n#` are defined with a set of optional values named `#enum_value_value_1#` to `#enum_value_value_n#`. The syntax is defined below.

The order of fields in the type shall follow the order of fields in the XML definition.

```
type #enum_type_name# is new #base_type_name#;
#enum_type_name#_#enum_value_name_1# : constant #enum_type_name# := #enum_value_value_1#;
#enum type name# #enum value name 2# : constant #enum type name# := #enum value value 2#;
[…]
#enum_type_name#_#enum_value_name_n# : constant #enum_type_name# := #enum_value_value_n#;
```

Where:

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

- #enum_value_name_X# is the name of a label
- #enum_value_value_X# is the optional value of the label. If not set, this value is computed from the previous label value, by adding 1 (or set to 0 if it is the first label of the enumeration).

9.3.4 Records

The Ada syntax for a record type named #record_type_name# with a set of fields named #field_name1# to #field_namen# of given types #data_type_1# to #data_type_n# is given below.

The order of fields in the Ada record shall follow the order of fields in the XML definition.

```
type #record_type_name# is
   record
      #field name1# : #data type 1#;
      #field_name2# : #data_type_2#;
      […]
      #field_namen# : #data_type_n#;
   end record;
```

9.3.5 Variant Records

The syntax for a variant record named #variant_record_type_name# containing:

- a set of fields (named #field_name1# to #field_namen#) of given types #data_type_1# to #data_type_n#
- optional fields (named #optional_field_name1# to #optional_field_namen#) of type (#optional_type_name1# to #optional_type_namen#) with selector #selector_name# of type #selector_type_name#

is given below.

The order of fields in the Ada record shall follow the order of fields in the XML definition.

```
-- #selector type name# can be of any simple predefined type, or an enumeration

type #variant_record_type_name# (#selector_name# : #selector_type_name#) is
   record
      #field_name1# : #data_type_1#;
      #field name2# : #data type 2#;
      […]
      #field_namen# : #data_type_n#;
      case #selector_name# is
         when #selector_value_constant1# =>
            #optional_field_name1# : #optional_type_name1#;
         when #selector value constant2# =>
            #optional_field_name1# : #optional_type_name1#;
         […]
         when #selector_value_constantn# =>
           #optional_field_namen# : #optional_type_namen#;
      end case;
   end record;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

9.3.6 Fixed Arrays

The Ada syntax for a fixed array named `#array_type_name#` of `#max_number#` elements with index range `0 to #max_number#-1`, and with elements of type `#data_type_name#` is given below. The index to an array must be specified as a distinct type.

```
type #array_type_name#_Index is range 0..#max_number#-1;
type #array_type_name# is array (#array_type_name#_Index) of #data_type_name#;
```

9.3.7 Variable Arrays

The Ada syntax for a variable array (named `#var_array_type_name#`) of `#max_number#` elements with index range `0 to #max_number#`, and with elements of type `#data_type_name#` and a current size of `Current_Size` is given below.

```
type #var_array_type_name#_Index is range 0..#max_number#-1;
type #var array type name# Data is array (#var array type name# Index) of #data type name#;

type #var_array_type_name# is
   record
      Current_Size : #var_array_type_name#_Index;
      Data         : #var_array_type_name#_Data;
   end record;
```

10 Module Interface

10.1 Operations

This section contains details of the operations that comprise the module API i.e. the operations that can invoked by the container on a module.

10.1.1 Request-response

10.1.1.1 Request Received

The following is the Ada syntax for invoking a request received by a module instance, where `#module_impl_name#` is the name of the module implementation providing the service and `#operation_name#` is the operation name. The same syntax is applicable for both synchronous and asynchronous request-response operations.

```
package #module_impl_name# is

-- ...

procedure #operation_name#_Request_Received
  (Context : in out #module_impl_name#_Container.Context_Type;
   ID      : in     ECOA.Unsigned_32_Type;
   #parameters_in#);

-- ...

end #module_impl_name#;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

10.1.1.2 Response received

The following is the Ada syntax for an operation used by the container to send a response to an asynchronous request response operation to the module instance that originally issued the request, where `#module_impl_name#` is the name of the module implementation providing the service and `#operation_name#` is the operation name. (The reply to a synchronous request response is provided by the return of the response).

```
package #module_impl_name# is

-- ...

procedure #operation_name#_Response_Received
   (Context : in out #module_impl_name#_Container.Context_Type;
    ID      : in     ECOA.Unsigned_32_Type;
    Status  : in     ECOA.Return_Status_Type;
    #parameters out#);

-- ...

end #module_impl_name#;
```

NOTE: the "`#parameters_out#` are the **'out**' parameters of the original procedure and are passed as '**in**' parameters, so they are not modified by the container.

10.1.2 Versioned Data

10.1.2.1 Updated

The following is the Ada syntax that is used by the container to inform a module instance that reads an item of versioned data that new data has been written.

```
package #module_impl_name# is

-- ...

procedure #operation_name#_Updated
   (Context     : in out #module_impl_name#_Container.Context_Type;
    Status      : in     ECOA.Return_Status_Type;
    Data_Handle : in     #module_impl_name#_Container.#operation_name#_Handle_Type);

-- ...

end #module_impl_name#;
```

10.1.3 Events

10.1.3.1 Received

The following is the Ada syntax for an event received by a module instance.

```
package #module_impl_name# is

-- ...

procedure #operation_name#_Received
   (Context : in out #module_impl_name#_Container.Context_Type;
    #parameters#);
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
-- ...

end #module_impl_name#;
```

10.2 Module Lifecycle

This section describes the procedures that are used to perform the required module lifecycle activities.

10.2.1 Generic Module API

The following operations are applicable to supervision, non-supervision, trigger and dynamic-trigger module instances.

10.2.1.1 Initialize_Received

The Ada syntax for a procedure to initialise a module instance is:

```
package #module_impl_name# is

-- ...

procedure INITIALIZE_Received
   (Context : in out #module impl name# Container.Context Type);

-- ...

end #module_impl_name#;
```

10.2.1.2 Start_Received

The Ada syntax for a procedure to start a module instance is:

```
package #module_impl_name# is

-- ...

procedure START Received
  (Context : in out #module_impl_name#_Container.Context_Type);

-- ...

end #module_impl_name#;
```

10.2.1.3 Stop_Received

The Ada syntax for a procedure to stop a module instance is:

```
package #module_impl_name# is

-- ...

procedure STOP_Received
  (Context : in out #module_impl_name#_Container.Context_Type);

-- ...
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
end #module_impl_name#;
```

10.2.1.4 Shutdown_Received

The Ada syntax for a procedure to shutdown a module instance is:

```
package #module_impl_name# is

-- ...

procedure SHUTDOWN_Received
  (Context : in out #module_impl_name#_Container.Context_Type);

-- ...

end #module_impl_name#;
```

10.2.1.5 Reinitialize_Received

The Ada syntax for a procedure to reinitialise a module instance is:

```
package #module_impl_name# is

-- ...

procedure REINITIALIZE Received
   (Context : in out #module_impl_name#_Container.Context_Type);

-- ...

end #module_impl_name#;
```

10.2.2 Supervision Module API

The Ada syntax for an operation that is used by the container to notify the supervision module that a module/trigger/dynamic trigger has changed state is:

```
package #supervision_module_impl_name#

procedure Lifecycle_Notification_#module_instance_name#
   (Context        : in out #module impl name# Container.Context Type;
    Previous_State : in     ECOA.Module_States_Type;
    New_State      : in     ECOA.Module_States_Type);

end #supervision_module_impl_name#;
```

Note: the supervision module API will contain a Lifecycle Notification procedure for every module/trigger/dynamic trigger in the Component i.e. the above API will be duplicated for every `#module_instance_name#` module/trigger/dynamic trigger in the Component.

`ECOA.Module_States_Type` is an enumerated type that contains all of the possible lifecycle states of the module instance: see section 9.2.7.

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

10.3 Service Availability

This section contains details of the operations which allow the container to notify the supervision module of a client component about changes to the availability of required services.

10.3.1 Service Availability Changed

The following is the Ada syntax for an operation used by the container to invoke a service availability changed operation to a supervision module instance. The operation will only be available if the component has one or more required services. The `Reference_ID_Type` is an enumeration type defined in the Container Interface (Section 11.4.4).

```
package #supervision_module_impl_name# is

-- ...

procedure Service_Availability_Changed
  (Context   : in out #supervision module impl name# Container.Context Type;
   Instance  : in     #supervision_module_impl_name#_Container.Reference_ID_Type;
   Available : in     ECOA.Boolean_8_Type);

-- ...

end #module_impl_name#;
```

10.3.2 Service Provider Changed

The following is the Ada syntax for an operation used by the container to invoke a service provider changed operation to a supervision module instance. The operation will only be available if the component has one or more required services. The `Reference_ID_Type` is an enumeration type defined in the Container Interface (Section 11.4.4).

```
package #supervision module impl name# is

-- ...

procedure Service_Provider_Changed
  (Context  : in out #supervision_module_impl_name#_Container.Context_Type;
   Instance : in     #supervision module impl name# Container.Reference ID Type);

-- ...

end #module_impl_name#;
```

10.4 Error_notification binding at application level

The Ada syntax for the container to report an error to the supervision module instance is:

```
package #supervision_module_impl_name# is

-- ...

procedure Error_Notification__#module_instance_name#
  (Context      : in out #supervision_module_impl_name#_Container.Context_Type;
   Module_error : in     ECOA.Module_Error_Type);

-- ...
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
end #module_impl_name#;
```

11 Container Interface

This section contains details of the operations that comprise the container API i.e. the operations that can be called by a module.

11.1 Operations

11.1.1 Request Response

11.1.1.1 Response Send

The Ada syntax, applicable to both synchronous and asynchronous request response operations, for sending a reply is:

```
package #module_impl_name#_Container is

-- ...

procedure #operation_name#_Response_Send
  (Context    : in out Context_Type;
   ID         : in     ECOA.Unsigned_32_Type;
   #parameters_out#;
   Status     :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

NOTE: the "`#parameters_out#` in the above code snippet are the out parameters of the original request, not of this operation: they are passed as '**in**' values, as they should not be modified by the container. The `ID` parameter is that which was passed in during the invocation of the request received operation.

11.1.1.2 Synchronous Request

The Ada syntax for a module instance to perform a synchronous request response operation is:

```
package #module_impl_name#_Container is

-- ...

procedure #operation_name#_Request_Sync
  (Context    : in out Context_Type;
   #parameters_in#;
   #parameters_out#;
   Status     :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

11.1.1.3 Asynchronous Request

The Ada syntax for a module instance to perform an asynchronous request response operation is:

```
package #module_impl_name#_Container is

-- ...

procedure #operation_name#_Request_Async
  (Context    : in out Context_Type;
   ID         :    out ECOA.Unsigned_32_Type;
   #parameters_in#);

-- ...

end #module_impl_name#_Container;
```

11.1.2 Versioned Data

This section contains the Ada syntax for versioned data operations, which allow a module instance to:

- Get (request) Read Access
- Release Read Access
- Get (request) Write Access
- Cancel Write Access (without writing new data)
- Publish (write) new data (automatically releases write access)

11.1.2.1 Get Read Access

```
package #module impl name# Container is

#operation_name#_Handle_Platform_Hook_Size : constant := 32;
type #operation_name#_Handle_Platform_Hook_Type is array
                           (0..#operation_name#_Handle_Platform_Hook_Size-1) of ECOA.Byte_Type;

--
-- The following is the data handle structure associated to the data operation
-- called #operation_name# of data-type #type_name#
--

type #operation_name#_Data_Access_Type is access all #type_name#;

type #operation_name#_Handle_Type is
  record
    Data_Access     : #operation_name#_Data_Access_Type;
    Timestamp       : ECOA.Timestamp_Type;
    Platform Hook   : #operation name# Handle Platform Hook Type;
  end record;

-- ...

procedure #operation_name#_Get_Read_Access
  (Context                : in out Context Type;
   Data_Handle            :    out #operation_name#_Handle_Type;
   Status                 :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

11.1.2.2 Release Read Access

```
package #module impl name# Container is

-- ...

procedure #operation_name#_Release_Read_Access
  (Context                : in out Context_Type;
   Data Handle            : in     #operation name# Handle Type;
   Status                 :    out ECOA.Return Status Type);

-- ...

end #module_impl_name#_Container;
```

11.1.2.3 Get Write Access

```
package #module_impl_name#_Container is

#operation_name#_Handle_Platform_Hook_Size : constant := 32;
type #operation_name#_Handle_Platform_Hook_Type is array
                          (0..#operation_name#_Handle_Platform_Hook_Size-1) of ECOA.Byte_Type;

type #operation_name#_Data_Access_Type is access all #type_name#

type #operation_name#_Handle_Type is
  record
    Data_Access     : #operation_name#_Data_Access_Type;
    Timestamp       : ECOA.Timestamp_Type;
    Platform_Hook   : #operation_name#_Handle_Platform_Hook_Type;
  end record;

-- ...

procedure #operation_name#_Get_Write_Access
  (Context                : in out Context Type;
   Data_Handle            :    out #operation_name#_Handle_Type;
   Status                 :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

11.1.2.4 Cancel Write Access

```
package #module_impl_name#_Container is

-- ...

procedure #operation_name#_Cancel_Write_Access
  (Context                : in out Context Type;
   Data_Handle            : in     #operation_name#_Handle_Type;
   Status                 :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

11.1.2.5 Publish Write Access

```
package #module impl name# Container is

-- ...

procedure #operation_name#_Publish_Write_Access
  (Context                : in out Context_Type;
   Data Handle            : in     #operation name# Handle Type;
   Status                 :    out ECOA.Return Status Type);

-- ...

end #module_impl_name#_Container;
```

11.1.3 Events

11.1.3.1 Send

The Ada syntax for a module instance to perform an event send operation is:

```
package #module impl name# Container is

-- ...

procedure #operation_name#_Send
  (Context      : in out Context_Type;
   #parameters#);

-- ...

end #module_impl_name#_Container;
```

11.2 Properties

This section describes the syntax for the Get_Value operation to request the module properties.

11.2.1 Get Value

The syntax for Get_Value is shown below where:

- `#property_name#` is the name of the property used in the component definition.
- `#property_type_name#` is the name of the data-type of the property.

```
package #module_impl_name#_Container is

-- ...

procedure Get_#property_name#_Value
  (Context : in out Context_Type;
   Value   :    out #property_type_name#);

-- ...

end #module_impl_name#_Container;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

11.3 Module Lifecycle

This section describes the container operations that are used to perform the required module lifecycle activities.

11.3.1 Non-Supervision Container API

Container operations are only available to supervision modules to allow them to manage the module lifecycle of non-supervision modules.

11.3.2 Supervision Container API

The Ada Syntax for the procedures that are called by the supervision to request the container to command a module/trigger/dynamic trigger instance to change (lifecycle) state is:

```
package #module impl name# Container is

procedure Get_Lifecycle_State_#module_instance_name#
  (Context       : in out Context_Type;
   Current_State :    out ECOA.Module_States_Type);

procedure Stop #module instance name#
  (Context        : in out Context_Type;
   Status         :    out ECOA.Return_Status_Type);

procedure Start_#module_instance_name#
  (Context        : in out Context_Type;
   Status         :    out ECOA.Return Status Type);

procedure Initialize_#module_instance_name#
  (Context        : in out Context_Type;
   Status         :    out ECOA.Return_Status_Type);

procedure Shutdown_#module_instance_name#
  (Context        : in out Context_Type;
   Status         :    out ECOA.Return_Status_Type);

end #module_impl_name#_Container;
```

An instance of each of the above operations is created for each module/trigger/dynamic trigger instance in the component, where `#module_instance_name#` above represents the name of the module/trigger/dynamic trigger instance.

11.4 Service Availability

This section contains details of the operations which allow supervision modules to set the availability of provided services or get the availability of required services.

11.4.1 Set Service Availability (Server Side)

The following is the Ada syntax for invoking the set service availability operation by a supervision module instance. The operation will only be available if the component has one or more provided services. The service instance is identified by the enumeration type `service_id` defined in the Container Interface (Section 11.4.3).

```
package #supervision_module_impl_name#_Container is

-- ...
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
procedure Set Service Availability
  (Context   : in out Context_Type;
   Instance  : in     Service_ID_Type;
   Available : in     ECOA.Boolean_8_Type;
   Status    :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

11.4.2 Get Service Availability (Client Side)

The following is the Ada syntax for invoking the get service availability operation by a supervision module instance. The operation will only be available if the component has one or more required services. The service instance is identified by the enumeration type `reference_id` defined in the Container Interface (Section 11.4.4).

```
package #supervision module impl name# Container is

-- ...

procedure Get_Service_Availability
  (Context   : in out Context_Type;
   Instance  : in     Reference ID Type;
   Available :    out ECOA.Boolean_8_Type;
   Status    :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

11.4.3 Service ID Enumeration

In Ada `service_id` translates to `Service_ID_Type`.

This enumeration has a value for each element *<service/>* defined in the file `.componentType`, whose name is given by its attribute *name* and the numeric value is the position (starting by 0).

The `service_id` enumeration is only available if the component provides one or more services.

```
package #supervision_module_impl_name#_Container is

-- ...

type Service_ID_Type is new ECOA.Unsigned_32_Type;
Service_ID_Type_#service_instance_name# : constant Service_ID_Type := 0;

-- ...

end #supervision_module_impl_name#_Container;
```

11.4.4 Reference ID Enumeration

In Ada `reference_id` translates to `Reference_ID_Type`.

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

This enumeration has a value for each element <*reference*/> defined in the file `.componentType`, whose name is given by its attribute *name* and the numeric value is the position (starting by 0).

The `reference_id` enumeration is only available if the component requires one or more services.

```
package #supervision_module_impl_name#_Container is

-- ...

type Reference_ID_Type is new ECOA.Unsigned_32_Type;
Reference_ID_Type_#reference_instance_name# : constant Reference_ID_Type := 0;

-- ...

end #supervision_module_impl_name#_Container;
```

11.5 Logging and Fault Management

This section describes the Ada syntax for the logging and fault management procedures provided by the container. There are six procedures:

- Trace: a detailed runtime trace to assist with debugging
- Debug: debug information
- Info: to log runtime events that are of interest e.g. changes of module state
- Warning: to report and log warnings
- Raise_Error: to report an error from which the application may be able to recover
- Raise_Fatal_Error: to raise a severe error from which the application cannot recover.

11.5.1 Log_Trace Binding

```
package #module_impl_name#_Container is

-- ...

procedure Log_Trace
  (Context : in out Context_Type;
   Log     : in     ECOA.Log_Type);

-- ...

end #module_impl_name#_Container;
```

11.5.2 Log_Debug Binding

```
package #module_impl_name#_Container is

-- ...

procedure Log_Debug
  (Context : in out Context_Type;
   Log     : in     ECOA.Log_Type);

-- ...
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
end #module_impl_name#_Container;
```

11.5.3 Log_Info Binding

```
package #module impl name# Container is

-- ...

procedure Log_Info
  (Context  : in out Context_Type;
   Log      : in     ECOA.Log Type);

-- ...

end #module_impl_name#_Container;
```

11.5.4 Log_Warning Binding

```
package #module_impl_name#_Container is

-- ...

procedure Log_Warning
  (Context  : in out Context_Type;
   Log      : in     ECOA.Log_Type);

-- ...

end #module_impl_name#_Container;
```

11.5.5 Raise_Error Binding

```
package #module_impl_name#_Container is

-- ...

procedure Raise_Error
  (Context  : in out Context_Type;
   Log      : in     ECOA.Log_Type);

-- ...

end #module_impl_name#_Container;
```

11.5.6 Raise_Fatal_Error Binding

```
package #module_impl_name#_Container is

-- ...

procedure Raise Fatal Error
  (Context  : in out Context Type;
   Log      : in     ECOA.Log_Type);

-- ...

end #module_impl_name#_Container;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

11.6 Time Services

This section contains the Ada syntax for the time services provided to module instances by the container.

11.6.1 Get_Relative_Local_Time

```
package #module_impl_name#_Container is

-- ...

procedure Get Relative Local Time
  (Context             : in out Context_Type;
   Relative_Local_Time :    out ECOA.HR_Time_Type;
   Status              :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

11.6.2 Get_UTC_Time

```
package #module_impl_name#_Container is

-- ...

procedure Get UTC Time
  (Context  : in out Context_Type;
   UTC_Time :    out ECOA.Global_Time_Type;
   Status   :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

11.6.3 Get_Absolute_System_Time

```
package #module_impl_name#_Container is

-- ...

procedure
   Get Absolute System Time
   (Context               : in out Context_Type;
    Absolute_System_Time :    out ECOA.Global_Time_Type;
    Status               :    out ECOA.Return_Status_Type);

-- ...

end #module_impl_name#_Container;
```

11.6.4 Get_Relative_Local_Time_Resolution

```
package #module_impl_name#_Container is

-- ...

procedure
   Get Relative Local Time Resolution
   (Context                         : in out Context_Type;
    Relative_Local_Time_Resolution :    out ECOA.Duration);
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
-- ...

end #module_impl_name#_Container;
```

11.6.5 Get_UTC_Time_Resolution

```
package #module_impl_name#_Container is

-- ...

procedure
   Get_UTC_Time_Resolution
   (Context            : in out Context_Type;
    UTC_Time_Resolution :    out ECOA.Duration);

-- ...

end #module_impl_name#_Container;
```

11.6.6 Get_Absolute_System_Time_Resolution

```
package #module_impl_name#_Container is

-- ...

procedure
   Get_Absolute_System_Time_Resolution
   (Context                         : in out Context_Type;
    Absolute System Time Resolution :    out ECOA.Duration);

-- ...

end #module_impl_name#_Container;
```

12 Fault Handler Interface

12.1 Error_notification binding at Fault Handler level

The Ada syntax for the container to report an error to a Fault Handler is:

```
package #fault_handler_impl_name# is

-- ...

procedure Error Notification
  (Context        : in out #fault_handler_impl_name#_Container.Context_Type;
   Error_id   : in     ECOA.Error_Id;
   Timestamp      : in     ECOA.Timestamp_Type;
   Asset_id       : in     ECOA.Asset_Id;
   Asset_type     : in     ECOA.Asset_Type;
   Error type : in     ECOA.Error Type);

-- ...

end #fault_handler_impl_name#;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

12.2 Recovery_Action Binding

This section contains the Ada syntax for the recovery action service provided to Fault Handlers by the container.

```
package #fault_handler_impl_name#_Container is

-- ...

procedure
  Recovery_Action
  (Context         : in out Context_Type;
   Recovery_action : in     ECOA.Recovery_Action_Type;
   Asset_id        : in     ECOA.Asset_Id;
   Asset type      : in     ECOA.Asset Type;
   Status          :    out ECOA.Return Status Type);

-- ...

end #fault_handler_impl_name#_Container;
```

13 Reference Ada Specification

```
package ECOA is

  type Boolean_8_Type is new Boolean;
  for Boolean_8_Type'Size use 8;
  type Character_8_Type is new Character;
  for Character_8_Type'Size use 8;
  type Signed_8_Type is range -127 .. 127;
  for Signed 8 Type'Size use 8;
  type Signed_16_Type is range -32768 .. 32768;
  for Signed_16_Type'Size use 16;
  type Signed_32_Type is range -2147483647 .. 2147483647;
  for Signed_32_Type'Size use 32;
  type Signed_64_Type is range -9223372036854775807 .. 9223372036854775807;
  for Signed 64 Type'Size use 64;
  type Unsigned_8_Type is mod 2 ** 8;
  for Unsigned_8_Type'Size use 8;
  type Unsigned_16_Type is mod 2 ** 16;
  for Unsigned_16_Type'Size use 16;
  type Unsigned_32_Type is mod 2 ** 32;
  for Unsigned_32_Type'Size use 32;
  type Unsigned_64_Type is mod 2 ** 64;
  for Unsigned_64_Type'Size use 64;
  type Float_32_Type is digits 6;
  for Float_32_Type'Size use 32;
  type Float_64_Type is digits 15;
  for Float_64_Type'Size use 64;
  type Byte_Type is mod 2 ** 8;
  for Byte_Type'Size use 8;

  type Return Status Type is new Unsigned 32 Type;
    Return_Status_Type_OK                     : constant Return_Status_Type := 0;
    Return_Status_Type_INVALID_HANDLE         : constant Return_Status_Type := 1;
    Return_Status_Type_DATA_NOT_INITIALIZED   : constant Return_Status_Type := 2;
    Return_Status_Type_NO_DATA                : constant Return_Status_Type := 3;
    Return Status Type INVALID IDENTIFIER     : constant Return Status Type := 4;
    Return Status Type NO RESPONSE            : constant Return Status Type := 5;
    Return_Status_Type_OPERATION_ALREADY_PENDING : constant Return_Status_Type := 6;
    Return_Status_Type_INVALID_SERVICE_ID     : constant Return_Status_Type := 7;
    Return_Status_Type_CLOCK_UNSYNCHRONIZED   : constant Return_Status_Type := 8;
    Return_Status_Type_INVALID_TRANSITION     : constant Return_Status_Type := 9;
    Return Status Type RESOURCE NOT AVAILABLE : constant Return Status Type := 10;
    Return Status Type OPERATION NOT AVAILABLE : constant Return Status Type := 11;
    Return_Status_Type_PENDING_STATE_TRANSITION : constant Return_Status_Type := 12;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
type Seconds_Type is mod 2 ** 32;
for Seconds_Type'Size use 32;
for Seconds_Type'Alignment use 4;

type Nanoseconds_Type is range 0 .. 999999999;
for Nanoseconds_Type'Size use 32;
for Nanoseconds_Type'Alignment use 4;

type HR_Time_Type is record
  Seconds     : Seconds_Type := 0;
  Nanoseconds : Nanoseconds_Type := 0;
end record;
for HR_Time_Type'size use 64;
for HR_Time_Type'Alignment use 4;

type Global_Time_Type is record
  Seconds     : Seconds_Type := 0;
  Nanoseconds : Nanoseconds_Type := 0;
end record;
for Global_Time_Type'size use 64;
for Global_Time_Type'Alignment use 4;

type Timestamp_Type is record
  Seconds     : Seconds_Type := 0;
  Nanoseconds : Nanoseconds_Type := 0;
end record;
for Timestamp_Type'size use 64;
for Timestamp_Type'Alignment use 4;

type Duration_Type is record
  Seconds     : Seconds_Type := 0;
  Nanoseconds : Nanoseconds_Type := 0;
end record;
for Duration_Type'size use 64;
for Duration_Type'Alignment use 4;

type Log_Elements_Index_Type is range 0 .. 255;
for Log_Elements_Index_Type'size use 32;
for Log_Elements_Index_Type'Alignment use 4;

type Log_Elements_Type is array (Log_Elements_Index_Type) of ECOA.Character_8_Type;
for Log_Elements_Type'size use 2048;
for Log_Elements_Type'Alignment use 4;

type Log_Type is record
  Current_Size : Log_Elements_Index_Type  := Log_Elements_Index_Type'First;
  Data         : Log_Elements_Type        := (others => ECOA.Character_8_Type'First);
end record;
for Log_Type'size use 2080;
for Log_Type'Alignment use 4;

type Module_States_Type is new Unsigned_32_Type;
  Module_States_Type_IDLE    : constant Module_States_Type := 0;
  Module_States_Type_READY   : constant Module_States_Type := 1;
  Module_States_Type_RUNNING : constant Module_States_Type := 2;

type Module_Error_Type is new Unsigned_32_Type;
  Error_Type_ERROR       : constant Module_Error_Type := 0;
  Error_Type_FATAL_ERROR : constant Module_Error_Type := 1;

type Error_Id is new Unsigned_32_Type;

type Asset_Id is new Unsigned_32_Type;

type Asset_Type is new Unsigned_32_Type;
  Asset_Type_COMPONENT         : constant Asset_Type := 0;
  Asset_Type_PROTECTION_DOMAIN : constant Asset_Type := 1;
  Asset_Type_NODE              : constant Asset_Type := 2;
  Asset_Type_PLATFORM          : constant Asset_Type := 3;
  Asset_Type_SERVICE           : constant Asset_Type := 4;
  Asset_Type_DEPLOYMENT        : constant Asset_Type := 5;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
  type Error Type is new Unsigned 32 Type;;
    Error_Type_RESOURCE_NOT_AVAILABLE : constant Error_Type := 0;
    Error_Type_UNAVAILABLE            : constant Error_Type := 1;
    Error_Type_MEMORY_VIOLATION       : constant Error_Type := 2;
    Error_Type_NUMERICAL_ERROR        : constant Error_Type := 3;
    Error_Type_ILLEGAL_INSTRUCTION    : constant Error_Type := 4;
    Error_Type_STACK_OVERFLOW         : constant Error_Type := 5;
    Error_Type_DEADLINE_VIOLATION     : constant Error_Type := 6;
    Error_Type_OVERFLOW               : constant Error_Type := 7;
    Error_Type_UNDERFLOW              : constant Error_Type := 8;
    Error_Type_ILLEGAL_INPUT_ARGS     : constant Error_Type := 9;
    Error_Type_ILLEGAL_INPUT_ARGS     : constant Error_Type := 10;
    Error_Type_ERROR                  : constant Error_Type := 11;
    Error_Type_FATAL_ERROR            : constant Error_Type := 12;
    Error_Type_HARDWARE_FAULT         : constant Error_Type := 13;
    Error_Type_POWER_FAIL             : constant Error_Type := 14;
    Error_Type_COMMUNICATION_ERROR    : constant Error_Type := 15;
    Error_Type_INVALID_CONFIG         : constant Error_Type := 16;
    Error_Type_INITIALISATION_PROBLEM : constant Error_Type := 17;
    Error_Type_CLOCK_UNSYNCHRONIZED   : constant Error_Type := 18;
    Error_Type_UNKNOWN_OPERATION : constant Error_Type := 19;
    Error Type OPERATION OVERRATED    : constant Error Type := 20;
    Error Type OPERATION UNDERRATED   : constant Error Type := 21;

  type Recovery_Action_Type is new Unsigned_32_Type;
    Recovery_Action_Type_SHUTDOWN_COMPONENT : constant Recovery_Action_Type := 0;
    Recovery_Action_Type_COLD_RESTART       : constant Recovery_Action_Type := 1;
    Recovery Action Type WARM RESTART       : constant Recovery Action Type := 2;
    Recovery_Action_Type_CHANGE_DEPLOYMENT  : constant Recovery_Action_Type := 3;

end ECOA;
```

This specification is developed by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, AgustaWestland Limited, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Selex ES Ltd. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.