European Component Oriented Architecture (ECOA®) Collaboration Programme: Architecture Specification Part 11: High Integrity Ada Language Binding

BAE Ref No: IAWG-ECOA-TR-031
Dassault Ref No: DGT 154934-B

Issue: 6

Prepared by
BAE Systems (Operations) Limited and Dassault Aviation

Note: *This specification represents the output of a research programme. Compliance with this specification shall not in itself relieve any person from any legal obligations imposed upon them. Product development should rely on the DefStan or BNAE publications of the ECOA standard.*

Contents

Figures

Tables

0 Introduction

This Architecture Specification provides the 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. It uses terms defined in the Definitions (Architecture Specification Part 2). The details of the other documents comprising the rest of this Architecture Specification can be found in Section 3.

This document describes a SPARK 95 compatible High Integrity Ada 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 basic 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;
- Section 12 describes the Container Types;
- Section 13 describes the External Interface;
- Section 14 describes the Default Values;
- Section 15 describes Trigger Instances;
- Section 16 describes Dynamic Trigger Instances;
- Section 17 provides a reference High Integrity Ada specification for the ECOA® package, usable in any High Integrity Ada binding implementation;

1 Scope

This Architecture Specification specifies 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. Compliance with this specification shall not in itself relieve any person from any legal obligations imposed upon them. Product development should rely on the DefStan or BNAE publications of the ECOA standard.

3 Normative References

Architecture Specification Part 1	IAWG-ECOA-TR-001 / DGT 144474 Issue 6 Architecture Specification Part 1 – Concepts
Architecture Specification Part 2	IAWG-ECOA-TR-012 / DGT 144487 Issue 6 Architecture Specification Part 2 – Definitions
Architecture Specification Part 3	IAWG-ECOA-TR-007 / DGT 144482 Issue 6 Architecture Specification Part 3 – Mechanisms
Architecture Specification Part 4	IAWG-ECOA-TR-010 / DGT 144485 Issue 6 Architecture Specification Part 4 – Software Interface
Architecture Specification Part 5	IAWG-ECOA-TR-008 / DGT 144483 Issue 6 Architecture Specification Part 5 – High Level Platform Requirements
Architecture Specification Part 6	IAWG-ECOA-TR-006 / DGT 144481 Issue 6 Architecture Specification Part 6 – ECOA® Logical Interface
Architecture Specification Part 7	IAWG-ECOA-TR-011 / DGT 144486 Issue 6 Architecture Specification Part 7 – Metamodel
Architecture Specification Part 8	IAWG-ECOA-TR-004 / DGT 144477 Issue 6 Architecture Specification Part 8 – C Language Binding
Architecture Specification Part 9	IAWG-ECOA-TR-005 / DGT 144478 Issue 6 Architecture Specification Part 9 – C++ Language Binding
Architecture Specification Part 10	IAWG-ECOA-TR-003 / DGT 144476 Issue 6 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++
SPARK_LRM	The SPADE Ada Kernel (including RavenSPARK) Issue 7.3

4 Definitions

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

5 Abbreviations

API	Application Programming Interface
ECOA	European Component Oriented Architecture. ECOA® is a registered trademark.
PINFO	Persistent Information
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 filenames whose mapping is specified in Table 1.

The Module Interface will be composed of a set of procedures corresponding to the Provided Operations of the Module Implementation. The declaration of these procedures will be accessible in a package specification 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 specification file called `#module_impl_name#_Container.ads`.

The Container Types will be composed of the types which the Module Implementation needs in order to declare, use and store various handles. The declaration of these types will be accessible in a package specification file called `#module_impl_name#_Container_Types.ads`.

A dedicated structure named `Context_Type`, and referred to as the Module Context structure in the rest of the document, may be defined in the Module Container specification (`#module_impl_name#_Container.ads`) if either user context or warm start context are used. It shall contain the optional user defined context variables of the Module Instance and the optional warm start context variable. This structure will be allocated by the Container if required before Module Instance start-up and passed to the Module Instance in each activation entry-point.

Since it is desirable to simplify the interactions between the Module and Container when using this binding, it differs from the other bindings in the following areas:

- In the case where the User and Warm Start Context are not required, the Module Context parameter will be removed from the Module Interface. This is to ensure that it remains SPARK 95 compliant, as the Module Context record (see section 6.4) would be null in this case.
- The inclusion of the Context parameter in the standard ECOA Container operation calls is not desirable in high integrity systems as it may lead to corrupted context data being passed into the Container. For this reason the Context parameter has been removed from all Container Operations. The Container must therefore determine the calling Module Instance by other means.

Table 1 Filename Mapping for High Integrity Ada

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#_Container.adb	Package body **#module_impl_name#**_Container implements the Container interface.

Filename	Use
#module_impl_name#_Container_Types.ads	Package **#module_impl_name#**_Container_Types specifies Container Types declaration (Container-level data types usable by the Module).
#module_impl_name#_User_Context.ads	Optional user extensions to the Module Context.

Figure 1 shows the files within the system, their relationships, and whether they are automatically generated or user generated.

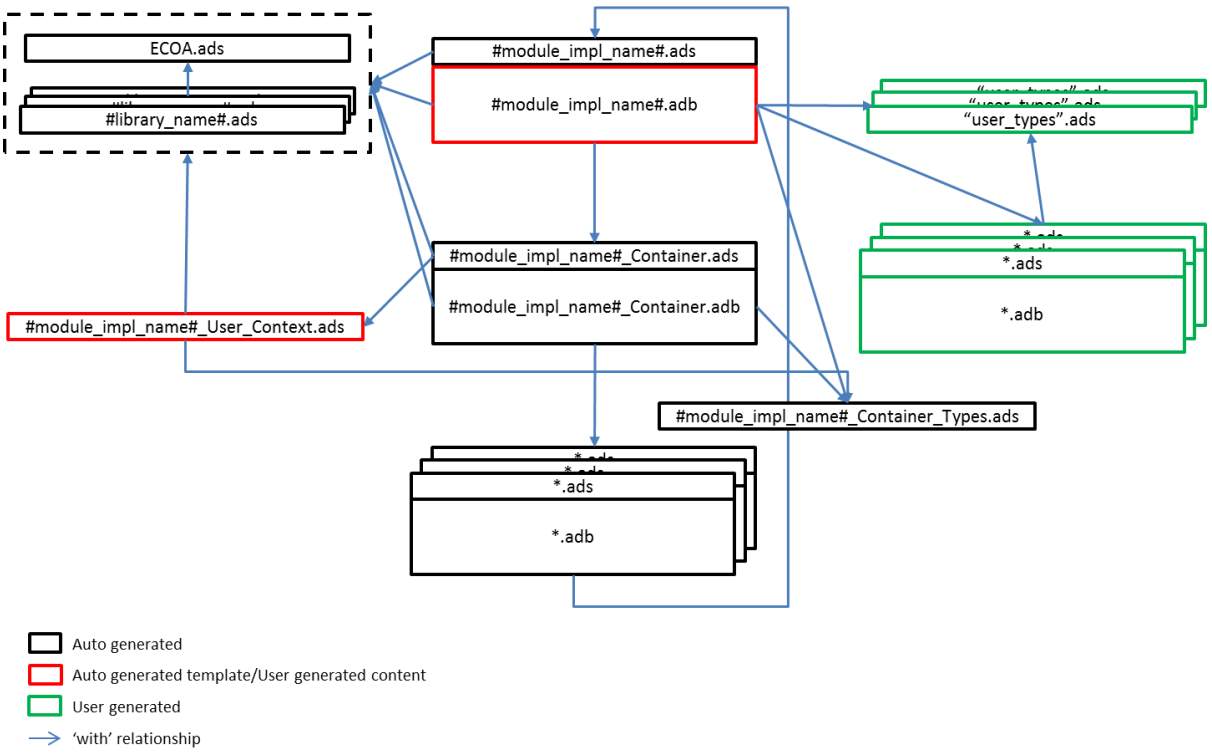

Figure 1 High Integrity Ada Files Organization

All auto generated files (shown in black outline) do not require modification by the user, and are there to provide the functionality of the Container and the interfaces to or from it. In addition, the files for ECOA base types and any user defined libraries containing types are automatically generated.

The auto generated template/user generated content files (shown in red outline) may be auto generated as empty or 'null' templates, which require populating by the user. Alternatively, these files may be manually generated by the user if desired; however they must comply with the ECOA template for these files, as they interface to the auto generated ECOA files.

The user generated files (shown in green outline) are written by the developer, if needed, to implement the Module functionality (and any local types). These files may be utility files used by many Module implementations, or be specific to each Module implementation.

These user defined files must be written using the same rules as the Module implementations, as they are an extension of them. For example the code must be written to be 'thread safe' (Architecture Specification Part 1, subsection 7.6).

6.1 Module Interface Template

```
-------------------------------------------------------------------
-- @file #module_impl_name#.ads
-- Module Interface package specification for Module #module_impl_name#
-- Generated automatically from specification; do not modify here
-------------------------------------------------------------------

-- 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#

   -- Versioned Data Notifying operation handlers specifications
   #list_of_versioned_data_notifying_operations_specifications#

   -- Lifecycle operation handlers specifications
   #list_of_lifecycle_operations_specifications#

   -- Error notification handler specification if this module is a
   -- Fault Handler
   #error_notification_operation_specification#

end #module_impl_name#;
```

```
-------------------------------------------------------------------
-- @file #module_impl_name#.adb
-- Module Interface package for Module #module_impl_name#
-- Generated automatically from specification; do not modify here
-- 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#

   -- Versioned Data Notifying operation handlers
   #list_of_versioned_data_notifying_operations#

   -- Lifecycle operation handlers
   #list_of_lifecycle_operations#

   -- Error notification handler specification if this module is a
   -- Fault Handler
   #error_notification_operation_specification#

end #module_impl_name#;
```

6.2 Container Interface Template

```
-------------------------------------------------------------------
-- @file #module_impl_name#_Container.ads
-- Container Interface package specification for Module #module_impl_name#
-- Generated automatically from specification; do not modify here
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;
-- Additionally Created Types
with #additionally_created_types#;
-- Include Container Types
with #module_impl_name#_Container_Types;
-- Optional User Context: the "#module_impl_name#_User_Context.ads" header
```

```
-- inclusion is optional (depends if user and/or warm start context are being
-- used
with #module_impl_name#_User_Context;

package #module_impl_name#_Container is

   -- Optional Module Implementation Context data type is specified here. This
   -- enables a module instance to hold its own private data in a non-OO
   -- fashion. If there is no User Context or Warm Start Context then the
   -- Context_Type will not be defined or used.
   type Context_Type is record
      -- Optional information that is private to a module implementation
      User_Context : #module_impl_name#_User_Context.User_Context_Type;

      -- Optional Warm Start information that is private to a module
      -- implementation
      Warm_Start_Context  :
        #module_impl_name#_User_Context.Warm_Start_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
   #properties_call_specifications#

   -- Logging services API call specifications
   #logging_services_call_specifications#

   -- Recovery action service API call specification if this is a
   -- Fault Handler module
   #recovery_action_call_specification#

   -- Time Services API call specifications
   #time_services_call_specifications#

   -- Persistent Information management operations
```

```
   #PINFO_read_call_specifications#
   #PINFO_seek_call_specifications#

   -- Optional Context management operation
   #save_warm_start_context_operation#

end #module_impl_name#_Container;
```

6.3 Container Types Template

```
-------------------------------------------------------------------
-- @file #module_impl_name#_Container_Types.ads
-- Container Types package specification for Module #module_impl_name#
-- Generated automatically from specification; do not modify here
-------------------------------------------------------------------

-- Standard ECOA Types
with ECOA;

package #module_impl_name#_Container_Types is

   -- The following describes the data types generated with regard to APIs:
   -- For any Versioned Data Read Access: data_handle
   #versioned_data_read_handles#
   -- For any Versioned Data Write Access: data_handle
   #versioned_data_write_handles#

end #module_impl_name#_Container_Types;
```

6.4 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;
-- Include Container Types
with #module_impl_name#_Container_Types;
-- 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.
       #user_context_data_declarations#
   end record;

   type Warm_Start_Context_Type is record
       -- Declare the Module Warm Start Context "local" data here.
       #warm_start_context_data_declarations#
   end record;

end module_impl_name#_User_Context;
```

Data declared within the Module User Context and the Module Warm Start Context can be of any type.

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 makes an appropriate choice as to whether to pass-by-value or pass-by-reference as determined by the parameter type.

8 Module Context

In the High Integrity Ada language binding, the Module Context is a structure which holds user local data (called “User Module Context” and “Warm Start Context”). User context and warm start context features may be optionally selected in Module Type declarations using metamodel attributes. The presence or absence of declarations of corresponding fields in Module code must be in accordance with the selections made in the Module Type declaration. The structure is defined in the Container Interface.

The following shows the syntax for the Module Context:

```
-------------------------------------------------------------------
-- @file #module_impl_name#_Container.ads
-- Container package specification for Module #module_impl_name#
-- Generated automatically from specification; do not modify here
-------------------------------------------------------------------
with System;

-- Standard ECOA Types
with ECOA;
-- Include Container Types
with #module_impl_name#_Container_Types;
-- Additionally Created Types
with #additionally_created_types#;
-- Optional User Context: the "#module_impl_name#_User_Context.ads" header
-- inclusion is optional (depends if user and/or warm start context are being
-- used
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
   -- When the optional user context is used, the type
   -- #module_impl_name#_User_Context.User_Context_Type shall be defined by the
   -- user in the #module_impl_name#_User_Context.ads file to carry the module
   -- implementation private data.
   -- #module_impl_name#_user_context user shall be declared as follows:
      User_Context : #module_impl_name#_User_Context.User_Context_Type;

   -- When the optional warm start context is used, the type
   -- #module_impl_name#_User Context.Warm_Start_Context_Type shall be defined
   -- by the user in the #module_impl_name#_User_Context.ads file to carry the
   -- module implementation warm start private data.
```

```
      Warm_Start_Context  :
        #module_impl_name#_User_Context.Warm_Start_Context_Type;

   end record;

end #module_impl_name#_Container;
```

8.1 User Module Context

The syntax for the optional user context is shown below (including an example data item; `My_Counter`) and the Module Warm Start Context (including an example data item `My_Data` and validity flag `Warm_Start_Valid`. The Module User Context header file is needed only if the user context and/or warm start context are used:

```
-----------------------------------------------------------------
-- @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;
-- Include Container Types
with #module_impl_name#_Container_Types;
-- 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;
   end record;

   type Warm_Start_Context_Type is record
      -- Example warm start context
      Warm_Start_Valid : ECOA.Boolean_8_Type; -- example validity flag
      My_Data : ECOA.Unsigned_32_Type;
   end record;

end #module_impl_name#_User_Context;
```

EXAMPLE The following 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#;
```

The optional 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.

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 specification 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.

The optional Module Context structure will be passed by the Container to the Module as the first parameter for each operation (i.e. received events, received request-response and asynchronous request-response sent call-back) if either the User Context or Warm Start Context are used. The Module Context defines the instance of the Module being invoked by the operation.

9 Types

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

9.1 Filenames and Namespace

The type definitions are contained within one or more namespaces: all types for specific namespace defined in `#namespace1#[__#namespacen#].types.xml` shall be placed in a file called `#namespace1#[-#namespacen#].ads`.

Below is an example of a simple type being defined within a nested namespace.

```
--
-- @file #namespace1#[-#namespacen#].ads
-- Data-type declaration file
-- Generated automatically from specification; do not modify here
--

package #namespace1#[.#namespacen#] is

   subtype #simple_type_name# is #basic_type_name# range
     #basic_type_name#(#min#) .. #basic_type_name#(#max#);

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

9.2 Basic Types

Basic types, shown in Table 2, shall be located in the “ECOA” namespace and hence in `ECOA.ads`. Note that this is the same filename (and namespace) used by the ECOA Ada Binding (Architecture Specification Part 10), and as such will cause issues if the two language bindings are mixed within the same protection domain. It is recommended that only one of the Ada bindings are used for any Components within a Protection Domain as this will avoid any namespace clashes. Also note that any tooling that is used to generate the ECOA.ads file (for both bindings) will need to ensure that it manages the separation of the files within the development system.

Table 2 High Integrity Ada Basic Types

ECOA Basic Type	High Integrity Ada 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
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 define explicit constants for the maximum and minimum values of the type range.

All basic types shall be specified with a representation clause to ensure the type declared will fit into that many bits, and have the correct alignment.

```
package ECOA is

   …

   subtype Boolean_8_Type is Boolean;

   type Signed_8_Type is range -2 ** 7 + 1 .. 2 ** 7 - 1;
   for Signed_8_Type'Size use 8;
   for Signed_8_Type'Alignment use 1;

   type Unsigned_8_Type is mod 2 ** 8;
   for Unsigned_8_Type'Size use 8;
   for Unsigned_8_Type'Alignment use 1;

   subtype Character_8_Type is Character range Character'Val(0) ..
     Character'Val(127);

   type Byte_8_Type is mod 2 ** 8;
   for Byte_8_Type'Size use 8;
```

```
   for Byte_8_Type'Alignment use 1;

   type Signed_16_Type is range -2 ** 15 + 1 .. 2 ** 15 - 1;
   for Signed_16_Type'Size use 16;
   for Signed_16_Type'Alignment use 2;

   type Unsigned_16_Type is mod 2 ** 16;
   for Unsigned_16_Type'Size use 16;
   for Unsigned_16_Type'Alignment use 2;

   type Signed_32_Type is range -2 ** 31 + 1  .. 2 ** 31 - 1;
   for Signed_32_Type'Size use 32;
   for Signed_32_Type'Alignment use 4;

   type Unsigned_32_Type is mod 2 ** 32;
   for Unsigned_32_Type'Size use 32;
   for Unsigned_32_Type'Alignment use 4;

   type Signed_64_Type is range -2 ** 63 + 1  .. 2 ** 63 - 1;
   for Signed_64_Type'Size use 64;
   for Signed_64_Type'Alignment use 8;

   type Unsigned_64_Type is mod 2 ** 64;
   for Unsigned_64_Type'Size use 64;
   for Unsigned_64_Type'Alignment use 8;

   type Float_32_Type is digits 6 range -3.402823466e+38 .. 3.402823466e+38;
   for Float_32_Type'Size use 32;
   for Float_32_Type'Alignment use 4;

   type Float_64_Type is digits 15 range -1.7976931348623157e+308 ..
      1.7976931348623157e+308;
   for Float_64_Type'Size use 64;
   for Float_64_Type'Alignment use 8;

   …

end ECOA;
```

`System_Address_Type` is derived from `System.Address` as a private type to allow memory addresses to be referenced.

NOTE Any code that uses this type to reference a memory location (usually overlaid on a variable) will have to be 'hidden' in order to be SPARK compliant.

```
with Ada.Unchecked_Conversion;
```

```
with System;

package ECOA is

   …

   type System_Address_Type is private;

   Default_System_Address : constant System_Address_Type;

   …

   private
      --# hide ECOA;
      type System_Address_Type is new System.Address;

      function To_System_Address is new
        Ada.Unchecked_Conversion(System.Address, System_Address_Type);

      Default_System_Address : constant System_Address_Type :=
        To_System_Address(System.Null_Address);

end ECOA;
```

9.3 Derived Types

9.3.1 Simple Types

The syntax for a Simple Type called `#simple_type_name#` with an optional restricted range, which is a subtype of a Basic Type is:

```
subtype #simple_type_name# is #basic_type_name# range
  #predef_type_name#(#min#) .. #basic_type_name#(#max#);
```

9.3.2 Constants

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

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

Where `#constant_value#` is either an integer or a real 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
   (#enum_type_name#_#enum_value_name_1#,
    #enum_type_name#_#enum_value_name_2#,
    --…
    #enum_type_name#_#enum_value_name_n#);
for #enum_type_name# use
   (#enum_type_name#_#enum_value_name_1# => #enum_value_value_1#,
    #enum_type_name#_#enum_value_name_2# => #enum_value_value_2#,
    --…
    #enum_type_name#_#enum_value_name_n# => #enum_value_value_n#);
for #enum_type_name#'Size use #base_type_name#'Size;
for #enum_type_name#'Alignment use #base_type_name#'Alignment;
```

Where:

- `#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 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 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 Variant records are not supported in SPARK 95 Ada, and so will not be directly supported as a type in this binding.

However it is possible to implement a normal record which contains a field that may be used as a selector, and contain a superset of fields used for any variant. This does not have the advantage of efficiencies in storage, or the normal protection of Ada with respect to accessing a field when using the wrong selector,

and is implemented entirely within the applications software. The Container will treat it as a normal record type, and therefore will manipulate/transfer the data of the full record every time.

NOTE This approach is not recommended for use in high integrity applications, but it may be used with care and correct design rationale.

9.3.6 Fixed Arrays

The 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 syntax for a variable array (named `#var_array_type_name#`) of `#max_number#` elements with index range `0 to #max_number#-1`, and with elements of type `#data_type_name#` and a current size of `Current_Size` is given below.

```
type #var_array_type_name#_Size is range 0..#max_number#;
for #var_array_type_name#_Size'Size use ECOA.Unsigned_32_Type'Size;
subtype #var_array_type_name#_Index is #var_array_type_name#_Size 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#_Size;
      Data         : #var_array_type_name#_Data;
   end record;
```

9.4 Predefined Types

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

9.4.1 ECOA:return_status

9.4.1.1 ECOA:Return_Status_Type

The following definition of `Return_Status_Type` is not used directly by this binding, however it is provided to allow interoperability between Components using the other bindings. A user may define an interface between Components that contains a parameter of type `ECOA:return_status` – this definition allows a Component written using this binding to interpret it correctly.

```
package ECOA is
   …
```

```
   type Return_Status_Type is
      (Return_Status_Type_OK,
       Return_Status_Type_INVALID_HANDLE,
       Return_Status_Type_DATA_NOT_INITIALIZED,
       Return_Status_Type_NO_DATA,
       Return_Status_Type_INVALID_IDENTIFIER,
       Return_Status_Type_NO_RESPONSE,
       Return_Status_Type_OPERATION_ALREADY_PENDING,
       Return_Status_Type_CLOCK_UNSYNCHRONIZED,
       Return_Status_Type_RESOURCE_NOT_AVAILABLE,
       Return_Status_Type_OPERATION_NOT_AVAILABLE,
       Return_Status_Type_INVALID_PARAMETER);
   for Return_Status_Type use
      (Return_Status_Type_OK                          => 0,
       Return_Status_Type_INVALID_HANDLE              => 1,
       Return_Status_Type_DATA_NOT_INITIALIZED        => 2,
       Return_Status_Type_NO_DATA                     => 3,
       Return_Status_Type_INVALID_IDENTIFIER          => 4,
       Return_Status_Type_NO_RESPONSE                 => 5,
       Return_Status_Type_OPERATION_ALREADY_PENDING   => 6,
       Return_Status_Type_CLOCK_UNSYNCHRONIZED        => 7,
       Return_Status_Type_RESOURCE_NOT_AVAILABLE      => 8,
       Return_Status_Type_OPERATION_NOT_AVAILABLE     => 9,
       Return_Status_Type_INVALID_PARAMETER           => 10);
   for Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

A unique return status type shall be defined for each operation which returns a status. The return status values remain consistent with those defined in the ECOA Specification. The return status types are defined below:

9.4.1.2 ECOA:Response_Received_Return_Status_Type

```
package ECOA is
   …
   type Response_Received_Return_Status_Type is
      (Response_Received_Return_Status_Type_OK,
       Response_Received_Return_Status_Type_NO_RESPONSE);
   for Response_Received_Return_Status_Type use
      (Response_Received_Return_Status_Type_OK          => 0,
       Response_Received_Return_Status_Type_NO_RESPONSE => 5);
```

```
   for Response_Received_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Response_Received_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.3 ECOA:Request_Sync_Return_Status_Type

```
package ECOA is
   …
   type Request_Sync_Return_Status_Type is
      (Request_Sync_Return_Status_Type_OK,
       Request_Sync_Return_Status_Type_NO_RESPONSE);
   for Request_Sync_Return_Status_Type use
      (Request_Sync_Return_Status_Type_OK          => 0,
       Request_Sync_Return_Status_Type_NO_RESPONSE => 5);
   for Request_Sync_Return_Status_Type'Size use Unsigned_32_Type'Size;
   for Request_Sync_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.4 ECOA:Request_Async_Return_Status_Type

```
package ECOA is
   …
   type Request_Async_Return_Status_Type is
      (Request_Async_Return_Status_Type_OK,
       Request_Async_Return_Status_Type_RESOURCE_NOT_AVAILABLE);
   for Request_Async_Return_Status_Type use
      (Request_Async_Return_Status_Type_OK                    => 0,
       Request_Async_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 8);
   for Request_Async_Return_Status_Type'Size use Unsigned_32_Type'Size;
   for Request_Async_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.5 ECOA:Response_Send_Return_Status_Type

```
package ECOA is
   …
   type Response_Send_Return_Status_Type is
```

```
      (Response_Send_Return_Status_Type_OK,
       Response_Send_Return_Status_Type_INVALID_IDENTIFIER);
   for Response_Send_Return_Status_Type use
      (Response_Send_Return_Status_Type_OK                => 0,
       Response_Send_Return_Status_Type_INVALID_IDENTIFIER => 4);
   for Response_Send_Return_Status_Type'Size use Unsigned_32_Type'Size;
   for Response_Send_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.6 ECOA:Get_Read_Access_Return_Status_Type

```
package ECOA is
   …
   type Get_Read_Access_Return_Status_Type is
      (Get_Read_Access_Return_Status_Type_OK,
       Get_Read_Access_Return_Status_Type_INVALID_HANDLE,
       Get_Read_Access_Return_Status_Type_NO_DATA,
       Get_Read_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE);
   for Get_Read_Access_Return_Status_Type use
      (Get_Read_Access_Return_Status_Type_OK                  => 0,
       Get_Read_Access_Return_Status_Type_INVALID_HANDLE      => 1,
       Get_Read_Access_Return_Status_Type_NO_DATA             => 3,
       Get_Read_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 8);
   for Get_Read_Access_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Get_Read_Access_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;   …
end ECOA;
```

9.4.1.7 ECOA:Release_Read_Access_Return_Status_Type

```
package ECOA is
   …
   type Release_Read_Access_Return_Status_Type is
      (Release_Read_Access_Return_Status_Type_OK,
       Release_Read_Access_Return_Status_Type_INVALID_HANDLE);
   for Release_Read_Access_Return_Status_Type use
      (Release_Read_Access_Return_Status_Type_OK             => 0,
       Release_Read_Access_Return_Status_Type_INVALID_HANDLE => 1);
   for Release_Read_Access_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Release_Read_Access_Return_Status_Type'Alignment use
```

```
      Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.8 ECOA:Get_Write_Access_Return_Status_Type

```
package ECOA is
   …
   type Get_Write_Access_Return_Status_Type is
      (Get_Write_Access_Return_Status_Type_OK,
       Get_Write_Access_Return_Status_Type_INVALID_HANDLE,
       Get_Write_Access_Return_Status_Type_DATA_NOT_INITIALIZED,
       Get_Write_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE);
   for Get_Write_Access_Return_Status_Type use
      (Get_Write_Access_Return_Status_Type_OK                    => 0,
       Get_Write_Access_Return_Status_Type_INVALID_HANDLE        => 1,
       Get_Write_Access_Return_Status_Type_DATA_NOT_INITIALIZED  => 2,
       Get_Write_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 8);
   for Get_Write_Access_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Get_Write_Access_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.9 ECOA:Cancel_Write_Access_Return_Status_Type

```
package ECOA is
   …
   type Cancel_Write_Access_Return_Status_Type is
      (Cancel_Write_Access_Return_Status_Type_OK,
       Cancel_Write_Access_Return_Status_Type_INVALID_HANDLE);
   for Cancel_Write_Access_Return_Status_Type use
      (Cancel_Write_Access_Return_Status_Type_OK             => 0,
       Cancel_Write_Access_Return_Status_Type_INVALID_HANDLE => 1);
   for Cancel_Write_Access_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Cancel_Write_Access_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.10 ECOA:Publish_Write_Access_Return_Status_Type

```
package ECOA is
   …
   type Publish_Write_Access_Return_Status_Type is
      (Publish_Write_Access_Return_Status_Type_OK,
       Publish_Write_Access_Return_Status_Type_INVALID_HANDLE);
   for Publish_Write_Access_Return_Status_Type use
      (Publish_Write_Access_Return_Status_Type_OK             => 0,
       Publish_Write_Access_Return_Status_Type_INVALID_HANDLE => 1);
   for Publish_Write_Access_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Publish_Write_Access_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.11 ECOA:Get_UTC_Time_Return_Status_Type

```
package ECOA is
   …
   type Get_UTC_Time_Return_Status_Type is
      (Get_UTC_Time_Return_Status_Type_OK,
       Get_UTC_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED,
       Get_UTC_Time_Return_Status_Type_OPERATION_NOT_AVAILABLE);
   for Get_UTC_Time_Return_Status_Type use
      (Get_UTC_Time_Return_Status_Type_OK                    => 0,
       Get_UTC_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED  => 7,
       Get_UTC_Time_Return_Status_Type_OPERATION_NOT_AVAILABLE => 9);
   for Get_UTC_Time_Return_Status_Type'Size use Unsigned_32_Type'Size;
   for Get_UTC_Time_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.12 ECOA:Get_Absolute_System_Time_Return_Status_Type

```
package ECOA is
   …
   type Get_Absolute_System_Time_Return_Status_Type is
      (Get_Absolute_System_Time_Return_Status_Type_OK,
      Get_Absolute_System_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED,
      Get_Absolute_System_Time_Return_Status_Type_OPERATION_NOT_AVAILABLE);
   for Get_Absolute_System_Time_Return_Status_Type use
```

```
      (Get_Absolute_System_Time_Return_Status_Type_OK                    => 0,
       Get_Absolute_System_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED => 7,
       Get_Absolute_System_Time_Return_Status_Type_OPERATION_NOT_AVAILABLE =>
         9);
   for Get_Absolute_System_Time_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Get_Absolute_System_Time_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.13 ECOA:Recovery_Action_Return_Status_Type

```
package ECOA is
   …
   type Recovery_Action_Return_Status_Type is
      (Recovery_Action_Return_Status_Type_OK,
       Recovery_Action_Return_Status_Type_INVALID_IDENTIFIER,
       Recovery_Action_Return_Status_Type_OPERATION_ALREADY_PENDING,
       Recovery_Action_Return_Status_Type_OPERATION_NOT_AVAILABLE);
   for Recovery_Action_Return_Status_Type use
      (Recovery_Action_Return_Status_Type_OK                        => 0,
       Recovery_Action_Return_Status_Type_INVALID_IDENTIFIER        => 4,
       Recovery_Action_Return_Status_Type_OPERATION_ALREADY_PENDING => 6,
       Recovery_Action_Return_Status_Type_OPERATION_NOT_AVAILABLE   => 9);
   for Recovery_Action_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Recovery_Action_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.14 ECOA:Read_Return_Status_Type

```
package ECOA is
   …
   type Read_Return_Status_Type is
      (Read_Return_Status_Type_OK,
       Read_Return_Status_Type_RESOURCE_NOT_AVAILABLE,
       Read_Return_Status_Type_INVALID_PARAMETER);
   for Read_Return_Status_Type use
      (Read_Return_Status_Type_OK                    => 0,
       Read_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 8,
       Read_Return_Status_Type_INVALID_PARAMETER      => 10);
```

```
   for Read_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Read_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.1.15 ECOA:Seek_Return_Status_Type

```
package ECOA is
   …
   type Seek_Return_Status_Type is
      (Seek_Return_Status_Type_OK,
       Seek_Return_Status_Type_RESOURCE_NOT_AVAILABLE,
       Seek_Return_Status_Type_INVALID_PARAMETER);
   for Seek_Return_Status_Type use
      (Seek_Return_Status_Type_OK                  => 0,
       Seek_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 8,
       Seek_Return_Status_Type_INVALID_PARAMETER   => 10);
   for Seek_Return_Status_Type'Size use
     Unsigned_32_Type'Size;
   for Seek_Return_Status_Type'Alignment use
     Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.2 ECOA:hr_time

The binding for `hr_time` makes use of `ECOA:Seconds` and `ECOA:Nanoseconds` types (section 9.4.14), and is defined as:

```
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;
```

9.4.3 ECOA:global_time

The binding for `global_time` makes use of `ECOA:Seconds` and `ECOA:Nanoseconds` types (section 9.4.14), and 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;
```

9.4.4 ECOA:duration

The binding for `duration` makes use of `ECOA:Seconds` and `ECOA:Nanoseconds` types (section 9.4.14), and is defined as:

```
package ECOA is
   …
   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;
```

9.4.5 ECOA:log

The syntax for a `log` is:

```
package ECOA is
   …
   type Log_Elements_Size_Type is range 0..256;
   for Log_Elements_Size_Type'Size use 32;
   for Log_Elements_Size_Type'Alignment use 4;
   subtype Log_Elements_Index_Type is Log_Elements_Size_Type range 0..255;

   type Log_Elements_Type is array (Log_Elements_Index_Type) of
     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_Size_Type;
         Data         : Log_Elements_Type;
      end record;
   for Log_Type'Size use 2080;
   for Log_Type'Alignment use 4;
   …
end ECOA;
```

9.4.6 ECOA:error_id

`ECOA:error_id` translates to `ECOA.Error_ID_Type` shown below:

```
package ECOA is
   …
   subtype Error_ID_Type is Unsigned_32_Type
     range Unsigned_32_Type'Range;
   …
end ECOA;
```

9.4.7 ECOA:error_code

`ECOA:error_code` translates to `ECOA.Error_Code_Type` shown below:

```
package ECOA is
   …
   subtype Error_Code_Type is Unsigned_32_Type
     range Unsigned_32_Type'Range;
   …
end ECOA;
```

9.4.8 ECOA:asset_id

`ECOA:asset_id` translates to `ECOA.Asset_ID_Type` shown below:

```
package ECOA is
   …
   subtype Asset_ID_Type is Unsigned_32_Type
     range Unsigned_32_Type'Range;
   …
end ECOA;
```

The `ECOA:asset_id` definitions will be generated as constants declared in a file named ECOA_Assets.ads using the following syntax:

```
-- File ECOA_Assets.ads
with ECOA;

package ECOA_Assets is


   CMP_#component_instance_name1# : constant ECOA.Asset_Id_Type := #CMP_ID1#;
   CMP_#component_instance_name2# : constant ECOA.Asset_Id_Type := #CMP_ID2#;
   CMP_#component_instance_nameN# : constant ECOA.Asset_Id_Type := #CMP_IDN#;

   PD_#protection_domain_name1# : constant ECOA.Asset_Id_Type := #PD_ID1#;
   PD_#protection_domain_name2# : constant ECOA.Asset_Id_Type := #PD_ID2#;
   PD_#protection_domain_nameN# : constant ECOA.Asset_Id_Type := #PD_IDN#;

   NOD_#computing_node_name1# : constant ECOA.Asset_Id_Type := #NOD_ID1#;
   NOD_#computing_node_name2# : constant ECOA.Asset_Id_Type := #NOD_ID2#;
   NOD_#computing_node_nameN# : constant ECOA.Asset_Id_Type := #NOD_IDN#;

   PF_#computing_platform_name1# : constant ECOA.Asset_Id_Type := #PF_ID1#;
   PF_#computing_platform_name2# : constant ECOA.Asset_Id_Type := #PF_ID2#;
   PF_#computing_platform_nameN# : constant ECOA.Asset_Id_Type := #PF_IDN#;

   SOP_#service_operation_name1# : constant ECOA.Asset_Id_Type := #ELI_UID#;
   SOP_#service_operation_name2# : constant ECOA.Asset_Id_Type := #ELI_UID#;
   SOP_#service_operation_nameN# : constant ECOA.Asset_Id_Type := #ELI_UID#;

   DEP_#deployment_name1# : constant ECOA.Asset_Id_Type := #DEP_ID1#;
   DEP_#deployment_name2# : constant ECOA.Asset_Id_Type := #DEP_ID2#;
   DEP_#deployment_nameN# : constant ECOA.Asset_Id_Type := #DEP_IDN#;

end ECOA_Assets;
```

9.4.9 ECOA:asset_type

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

```
package ECOA is
   …
   type Asset_Type is
      (Asset_Type_COMPONENT,
       Asset_Type_PROTECTION_DOMAIN,
       Asset_Type_NODE,
```

```
        Asset_Type_PLATFORM,
        Asset_Type_SERVICE,
        Asset_Type_DEPLOYMENT);
    for Asset_Type use
       (Asset_Type_COMPONENT         => 0,
        Asset_Type_PROTECTION_DOMAIN => 1,
        Asset_Type_NODE              => 2,
        Asset_Type_PLATFORM          => 3,
        Asset_Type_SERVICE           => 4,
        Asset_Type_DEPLOYMENT        => 5);
    for Asset_Type'Size use Unsigned_32_Type'Size;
    for Asset_Type'Alignment use Unsigned_32_Type'Alignment;
    …
end ECOA;
```

9.4.10 ECOA:error_type

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

```
package ECOA is
    …
    type Error_Type is
       (Error_Type_RESOURCE_NOT_AVAILABLE,
        Error_Type_UNAVAILABLE,
        Error_Type_MEMORY_VIOLATION,
        Error_Type_NUMERICAL_ERROR,
        Error_Type_ILLEGAL_INSTRUCTION,
        Error_Type_STACK_OVERFLOW,
        Error_Type_DEADLINE_VIOLATION,
        Error_Type_OVERFLOW,
        Error_Type_UNDERFLOW,
        Error_Type_ILLEGAL_INPUT_ARGS,
        Error_Type_ILLEGAL_OUTPUT_ARGS,
        Error_Type_ERROR,
        Error_Type_FATAL_ERROR,
        Error_Type_HARDWARE_FAULT,
        Error_Type_POWER_FAIL,
        Error_Type_COMMUNICATION_ERROR,
        Error_Type_INVALID_CONFIG,
        Error_Type_INITIALISATION_PROBLEM,
        Error_Type_CLOCK_UNSYNCHRONIZED,
        Error_Type_UNKNOWN_OPERATION,
        Error_Type_OPERATION_OVERRATED,
        Error_Type_OPERATION_UNDERRATED);
    for Error_Type use
```

```
      (Error_Type_RESOURCE_NOT_AVAILABLE => 0,
       Error_Type_UNAVAILABLE            => 1,
       Error_Type_MEMORY_VIOLATION       => 2,
       Error_Type_NUMERICAL_ERROR        => 3,
       Error_Type_ILLEGAL_INSTRUCTION    => 4,
       Error_Type_STACK_OVERFLOW         => 5,
       Error_Type_DEADLINE_VIOLATION     => 6,
       Error_Type_OVERFLOW               => 7,
       Error_Type_UNDERFLOW              => 8,
       Error_Type_ILLEGAL_INPUT_ARGS     => 9,
       Error_Type_ILLEGAL_OUTPUT_ARGS    => 10,
       Error_Type_ERROR                  => 11,
       Error_Type_FATAL_ERROR            => 12,
       Error_Type_HARDWARE_FAULT         => 13,
       Error_Type_POWER_FAIL             => 14,
       Error_Type_COMMUNICATION_ERROR    => 15,
       Error_Type_INVALID_CONFIG         => 16,
       Error_Type_INITIALISATION_PROBLEM => 17,
       Error_Type_CLOCK_UNSYNCHRONIZED   => 18,
       Error_Type_UNKNOWN_OPERATION      => 19,
       Error_Type_OPERATION_OVERRATED    => 20,
       Error_Type_OPERATION_UNDERRATED   => 21);
   for Error_Type'Size use Unsigned_32_Type'Size;
   for Error_Type'Alignment use Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.11 ECOA:recovery_action_type

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

```
package ECOA is
   …
   type Recovery_Action_Type is
      (Recovery_Action_Type_SHUTDOWN,
       Recovery_Action_Type_COLD_RESTART,
       Recovery_Action_Type_WARM_RESTART,
       Recovery_Action_Type_CHANGE_DEPLOYMENT);
   for Asset_Type use
      (Recovery_Action_Type_SHUTDOWN          => 0,
       Recovery_Action_Type_COLD_RESTART      => 1,
       Recovery_Action_Type_WARM_RESTART      => 2,
       Recovery_Action_Type_CHANGE_DEPLOYMENT => 3);
   for Recovery_Action_Type'Size use Unsigned_32_Type'Size;
```

```
   for Recovery_Action_Type'Alignment use Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.12 ECOA:pinfo_filename

The syntax for `ECOA:pinfo_filename` is:

```
package ECOA is
   …
   type PINFO_Filename_Elements_Size_Type is range 0..256;
   for PINFO_Filename_Elements_Size_Type'Size use 32;
   for PINFO_Filename_Elements_Size_Type'Alignment use 4;
   subtype PINFO_Filename_Elements_Index_Type is
PINFO_Filename_Elements_Size_Type range 0..255;

   type PINFO_Filename_Elements_Type is
     array (PINFO_Filename_Elements_Index_Type) of Character_8_Type;
   for PINFO_Filename_Elements_Type'Size use 2048;
   for PINFO_Filename_Elements_Type'Alignment use 4;

   type PINFO_Filename_Type is
      record
         Current_Size : PINFO_Filename_Elements_Size_Type;
         Data         : PINFO_Filename_Elements_Type;
      end record;
   for PINFO_Filename_Type'Size use 2080;
   for PINFO_Filename_Type'Alignment use 4;
   …
end ECOA;
```

9.4.13 ECOA:seek_whence_type

The syntax for `ECOA:seek_whence_type` is:

```
package ECOA is
   …
   type Seek_Whence_Type is
      (Seek_Whence_Type_SEEK_SET,
       Seek_Whence_Type_SEEK_CUR,
       Seek_Whence_Type_SEEK_END);
   for Seek_Whence_Type use
      (Seek_Whence_Type_SEEK_SET => 0,
       Seek_Whence_Type_SEEK_CUR => 1,
       Seek_Whence_Type_SEEK_END => 2);
```

```
   for Seek_Whence_Type'Size use Unsigned_32_Type'Size;
   for Seek_Whence_Type'Alignment use Unsigned_32_Type'Alignment;
   …
end ECOA;
```

9.4.14 ECOA:seconds and ECOA:nanoseconds

`ECOA:Seconds_Type` and `ECOA:Nanoseconds_Type`[1] are defined as follows:

```
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;
```

9.4.15 ECOA:request_response_id_type

The `ECOA:Request_Response_ID_Type` is defined as follows:

```
package ECOA is
   …
   type Request_Response_ID_Type is mod 2 ** 32;
   for Request_Response_ID_Type'Size use 32;
   for Request_Response_ID_Type'Alignment use 4;
   …
end ECOA;
```

9.4.16 ECOA:pinfo_size_type

The `ECOA:PINFO_Size_Type` is defined as follows:

```
package ECOA is
   …
```

[1] With the difference of C and C++ bindings, the High Integrity Ada binding defines new types suitable for time management by limiting the possible values of the considered temporal units.

```
   type PINFO_Size_Type is mod 2 ** 32;
   for PINFO_Size_Type'Size use 32;
   for PINFO_Size_Type'Alignment use 4;
   …
end ECOA;
```

9.4.17 ECOA:pinfo_offset_type

The `ECOA:PINFO_Offset_Type` is defined as follows:

```
package ECOA is
   …
   type PINFO_Offset_Type is range -2 ** 31 + 1  .. 2 ** 31 - 1;
   for PINFO_Offset_Type'Size use 32;
   for PINFO_Offset_Type'Alignment use 4;
   …
end ECOA;
```

9.4.18 ECOA:pinfo_position_type

The `ECOA:PINFO_Position_Type` is defined as follows:

```
package ECOA is
   …
   type PINFO_Position_Type is mod 2 ** 32;
   for PINFO_Position_Type'Size use 32;
   for PINFO_Position_Type'Alignment use 4;
   …
end ECOA;
```

10 Module Interface

10.1 Operations

This section contains details of the operations that comprise the Module API i.e. the operations that can be invoked by the Container on a Module.

10.1.1 Request-Response

10.1.1.1 Request Received

The following is the 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.Request_Response_ID_Type;
      #request_parameters#);

end #module_impl_name#;
```

Parameters:

Context	This provides access to the context associated to the particular Module Instance.
ID	The ID is assigned by the Container and used to associate the response to the initiating Request.
#request_parameters#	Correspond to the ordered list of input parameters specified for the Request Received operation.

10.1.1.2 Response Received

The following is the 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.Request_Response_ID_Type;
      Status  : in     ECOA.Response_Received_Return_Status_Type;
      #response_parameters#);

end #module_impl_name#;
```

Parameters:

Context	This provides access to the context associated to the particular Module Instance.

ID	The ID is used by the Module to associate the response with the initiating Request.
Status	The status of the Request-Response operation. ECOA.Response_Received_Return_Status_Type_OK shall be returned if the Request-Response operation was successful. ECOA.Response_Received_Return_Status_Type_NO_RESPONSE shall be returned if no response was received. The parameters will be set to the default for the type.
#response_parameters#	Correspond to the ordered list of output parameters specified for the original Request operation. *Note: that they are passed as 'in' parameters to the Module.*

10.1.2 Versioned Data Updated

The following is the 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.

```
-- Include Container Types
with #module_impl_name#_Container_Types;

package #module_impl_name# is

   procedure #operation_name#_Updated
     [(Context      : in out #module_impl_name#_Container.Context_Type)];

end #module_impl_name#;
```

Parameters:

Context	This provides access to the context associated to the particular Module Instance.

10.1.3 Event Received

The following is the 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;]
      #event_parameters#)];

end #module_impl_name#;
```

Parameters:

Context	This provides access to the context associated to the particular Module Instance.
#event_parameters#	Correspond to the ordered list of input parameters specified for the Event Received operation.

10.2 Module Lifecycle

The following operations are applicable to application, trigger and dynamic-trigger Module instances.

10.2.1 Initialize_Received

The 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#;
```

Parameters:

Context This provides access to the context associated to the particular Module Instance.

10.2.2 Start_Received

The 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#;
```

Parameters:

Context This provides access to the context associated to the particular Module Instance.

10.2.3 Stop_Received

The 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)];

end #module_impl_name#;
```

Parameters:

Context This provides access to the context associated to the particular Module Instance.

10.2.4 Shutdown_Received

The 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#;
```

Parameters:

Context This provides access to the context associated to the particular Module Instance.

10.3 Error_notification binding at Fault Handler level

The 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_Type;
      Timestamp   : in     ECOA.Global_Time_Type;
      Asset_Id    : in     ECOA.Asset_Id_Type;
      Asset_Type  : in     ECOA.Asset_Type;
      Error_Type  : in     ECOA.Error_Type;
      Error_Code  : in     ECOA.Error_Code_Type);

end #fault_handler_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.

The inclusion of the Context parameter in the standard ECOA Container operation calls is not desirable in high integrity systems as it may lead to corrupted context data being passed into the Container. For this reason the Context parameter has been removed from all Container Operations. The Container must therefore determine the calling Module Instance by other means.

All operations described in this section will be contained within a package named `#module_impl_name#_Container`.

11.1 Operations

11.1.1 Request Response

11.1.1.1 Response Send

The 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
     (ID          : in     ECOA.Request_Response_ID_Type;
      #response_parameters#;
      Status      :    out ECOA.Response_Send_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

ID	This will contain an ID which should match that which was passed into the Module during the invocation of the Request Received operation.
#response_parameters#	Correspond to the ordered list of output parameters specified for the original Request Send operation. Note: that they are passed as 'in' parameters, so are not modified by the Container.
Status	The status of the response operation. `ECOA.Response_Send_Return_Status_Type_OK` shall be returned if the response operation was successful. `ECOA.Response_Send_Return_Status_Type_INVALID_IDENTIFIER` shall be returned if the Module has provided an invalid (unknown) ID.

11.1.1.2 Synchronous Request

The syntax for a Module Instance to perform a synchronous request response operation is:

```
package #module_impl_name#_Container is

   procedure #operation_name#_Request_Sync
     (#request_parameters#;
      #response_parameters#;
      Status :    out ECOA.Request_Sync_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

#request_parameters#	Correspond to the ordered list of input parameters specified for the Request Send operation.

#response_parameters#	Correspond to the ordered list of output parameters specified for the Request Send operation.
Status	The status of the Request-Response operation. `ECOA.Request_Sync_Return_Status_Type_OK` shall be returned if the Request-Response operation was successful. `ECOA.Request_Sync_Return_Status_Type_NO_RESPONSE` shall be returned if no response was received. The out parameters will be set to the default for the type.

11.1.1.3 Asynchronous Request

The syntax for a Module Instance to perform an asynchronous request response operation is:

```
package #module_impl_name#_Container is

   procedure #operation_name#_Request_Async
     (ID      :    out ECOA.Request_Response_ID_Type;
      #request_parameters#;
      Status  :    out ECOA.Request_Async_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

ID	The ID is assigned by the Container and used by the Module to associate the response with the initiating Request.
#request_parameters#	Correspond to the ordered list of input parameters specified for the Request Send operation.
Status	The status of the Request-Response operation. `ECOA.Request_Async_Return_Status_Type_OK` shall be returned if the Request-Response operation was successful. `ECOA.Request_Async_Return_Status_Type_RESOURCE_NOT_AVAILABLE` shall be returned if maxConcurrentRequests has been exceeded.

11.1.2 Versioned Data

This section contains the 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)

NOTE The definition of versioned data handles involved in all `#operation_name#` is done in the Container types file, as specified in Section 12.1.

NOTE The High Integrity Ada binding does not allow disabling access control on versioned data links. Consequently, only the behaviour with access control is specified for versioned data APIs in this section.

11.1.2.1 Get Read Access

```
-- Include Container Types
with #module_impl_name#_Container_Types;

package #module_impl_name#_Container is

   procedure #operation_name#_Get_Read_Access
     (Data_Handle :    out
        #module_impl_name#_Container_Types.#operation_name#_Handle_Type;
      Status      :    out ECOA.Get_Read_Access_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Data Handle	The Data Handle contains the version of the data, and a Stamp.
Status	The status of the get read access operation. `ECOA.Get_Read_Access_Return_Status_Type_OK` shall be returned if the get read access operation was successful. The Data Handle will be updated with a Stamp and a version of the data. `ECOA.Get_Read_Access_Return_Status_Type_NO_DATA` shall be returned if no data has been previously written to the repository. The data element of the handle will be set to the default for the type. `ECOA.Get_Read_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE` shall be returned if the maximum number of concurrent read accesses has been reached or if the Container is unable to provide a copy of the data. The data element of the handle will be set to the default for the type.

11.1.2.2 Release Read Access

```
-- Include Container Types
with #module_impl_name#_Container_Types;

package #module_impl_name#_Container is

   procedure #operation_name#_Release_Read_Access
     (Data_Handle : in
         #module_impl_name#_Container_Types.#operation_name#_Handle_Type;
      Status      :    out ECOA.Release_Read_Access_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Data Handle	The Data Handle contains the version of the data, and a Stamp.

Status — The status of the release read access operation.
ECOA.Release_Read_Access_Return_Status_Type_OK shall be returned if the release read access operation was successful.
ECOA.Release_Read_Access_Return_Status_Type_INVALID_HANDLE shall be returned if a corresponding get read access operation has not been successfully completed.

11.1.2.3 Get Write Access

```
-- Include Container Types
with #module_impl_name#_Container_Types;

package #module_impl_name#_Container is

   procedure #operation_name#_Get_Write_Access
     (Data_Handle :    out
        #module_impl_name#_Container_Types.#operation_name#_Handle_Type;
      Status      :    out ECOA.Get_Write_Access_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Data Handle — The Data Handle contains the version of the data, and a Stamp.

Status — The status of the get write access operation.
ECOA.Get_Write_Access_Return_Status_Type_OK shall be returned if the get write access operation was successful. The Data Handle will be updated with a Stamp and a version of the data.
ECOA.Get_Write_Access_Return_Status_Type_DATA_NOT_INITIALIZED shall be returned if no data has been previously written to the repository. The data element of the handle will be set to the default for the type.
ECOA.Get_Write_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE shall be returned if the maximum number of concurrent write accesses has been reached or if the Container is unable to provide a write-access copy of the data. The Data Handle will be set to the default for the type.

11.1.2.4 Cancel Write Access

```
-- Include Container Types
with #module_impl_name#_Container_Types;

package #module_impl_name#_Container is

   procedure #operation_name#_Cancel_Write_Access
     (Data_Handle : in
        #module_impl_name#_Container_Types.#operation_name#_Handle_Type;
      Status      :    out ECOA.Cancel_Write_Access_Return_Status_Type);
```

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

Parameters:

Data Handle	The Data Handle contains the version of the data, and a Stamp.
Status	The status of the cancel write access operation. `ECOA.Cancel_Write_Access_Return_Status_Type_OK` shall be returned if the cancel write access operation was successful. `ECOA.Cancel_Write_Access_Return_Status_Type_INVALID_HANDLE` shall be returned if a corresponding get write access operation has not been successfully completed.

11.1.2.5 Publish Write Access

```
-- Include Container Types
with #module_impl_name#_Container_Types;

package #module_impl_name#_Container is

   procedure #operation_name#_Publish_Write_Access
     (Data_Handle : in
        #module_impl_name#_Container_Types.#operation_name#_Handle_Type;
      Status      :    out ECOA.Publish_Write_Access_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Data Handle	The Data Handle contains the version of the data, and a Stamp.
Status	The status of the publish write access operation. `ECOA.Publish_Write_Access_Return_Status_Type_OK` shall be returned if the publish write access operation was successful. The repository will have been updated with the data and Stamp. `ECOA.Publish_Write_Access_Return_Status_Type_INVALID_HANDLE` shall be returned if a corresponding get write access operation has not been successfully completed. `ECOA.Publish_Write_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE` shall be returned if the Container was unable to write the Versioned Data to the repository. A subsequent cancel write access or publish write access must be performed by the Module in order to release the write access in this scenario.

11.1.3 Events

11.1.3.1 Send

The syntax for a Module Instance to perform an event send operation is:

```
package #module_impl_name#_Container is

```

```
   procedure #operation_name#_Send
     (#event_parameters#);

end #module_impl_name#_Container;
```

Parameters:

#event_parameters#	Correspond to the ordered list of input parameters specified for the event Send operation.

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
     (Value : out #property_type_name#);

end #module_impl_name#_Container;
```

Parameters:

Value	Contains the value of the requested property.

11.2.2 Expressing Property Values

Not applicable to this binding.

11.2.3 Example of Defining and Using Properties

Not applicable to this binding.

11.3 Logging and Fault Management

This section describes the 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.

11.3.1 Log_Trace

```
package #module_impl_name#_Container is

   procedure Log_Trace
     (Log      : in      ECOA.Log_Type);

end #module_impl_name#_Container;
```

Parameters:

Log	This is a block of data containing the message to be logged.

11.3.2 Log_Debug

```
package #module_impl_name#_Container is

   procedure Log_Debug
     (Log      : in      ECOA.Log_Type);

end #module_impl_name#_Container;
```

Parameters:

Log	This is a block of data containing the message to be logged.

11.3.3 Log_Info

```
package #module_impl_name#_Container is

   procedure Log_Info
     (Log      : in      ECOA.Log_Type);

end #module_impl_name#_Container;
```

Parameters:

Log	This is a block of data containing the message to be logged.

11.3.4 Log_Warning

```
package #module_impl_name#_Container is

   procedure Log_Warning
     (Log      : in      ECOA.Log_Type);

```

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

Parameters:

Log	This is a block of data containing the message to be logged.

11.3.5 Raise_Error

```
package #module_impl_name#_Container is

   procedure Raise_Error
     (Log        : in     ECOA.Log_Type;
      Error_Code : in     ECOA.Error_Code_Type);

end #module_impl_name#_Container;
```

Parameters:

Log	This is a block of data containing the message to be logged.

11.3.6 Raise_Fatal_Error

```
package #module_impl_name#_Container is

   procedure Raise_Fatal_Error
     (Log        : in     ECOA.Log_Type;
      Error_Code : in     ECOA.Error_Code_Type);

end #module_impl_name#_Container;
```

Parameters:

Log	This is a block of data containing the message to be logged.

11.4 Time Services

11.4.1 Get_Relative_Local_Time

```
package #module_impl_name#_Container is

   procedure Get_Relative_Local_Time
     (Relative_Local_Time :    out ECOA.HR_Time_Type);

end #module_impl_name#_Container;
```

Parameters:

Relative_Local_Time	This is the relative local time returned by the operation.

11.4.2 Get_UTC_Time

```
package #module_impl_name#_Container is

   procedure Get_UTC_Time
     (UTC_Time :    out ECOA.Global_Time_Type;
      Status   :    out ECOA.Get_UTC_Time_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

UTC_Time	This is the UTC time returned by the operation.
Status	The status of the get UTC time operation. `ECOA.Get_UTC_Time_Return_Status_Type_OK` shall be returned if the time operation has been successful. `ECOA.Get_UTC_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED` shall be returned if the clock has not been synchronized. A valid time value shall still be returned. `ECOA.Get_UTC_Time_Return_Status_Type_OPERATION_NOT_AVAILABLE` shall be returned if UTC time is not available. A zero time value shall be returned.

11.4.3 Get_Absolute_System_Time

```
package #module_impl_name#_Container is

   procedure Get_Absolute_System_Time
     (Absolute_System_Time :    out ECOA.Global_Time_Type;
      Status               :    out
        ECOA.Get_Absolute_System_Time_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Absolute_System_Time	This is the absolute system time returned by the operation.
Status	The status of the get absolute system time operation. `ECOA.Get_Absolute_System_Time_Return_Status_Type_OK` if the time operation has been successful. `ECOA.Get_Absolute_System_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED` shall be returned if the clock has not been synchronized. A valid time value shall still be returned. `ECOA.Get_Absolute_System_Time_Return_Status_Type_OPERATION_NOT_AVAILABLE` shall be returned if absolute system time is not available. A zero time value shall be returned.

11.4.4 Get_Relative_Local_Time_Resolution

```
package #module_impl_name#_Container is

   procedure Get_Relative_Local_Time_Resolution
     (Relative_Local_Time_Resolution :    out ECOA.Duration);

end #module_impl_name#_Container;
```

Parameters:

Relative_Local_Time_Resolution This is the relative local time resolution returned by the operation.

11.4.5 Get_UTC_Time_Resolution

```
package #module_impl_name#_Container is

   procedure Get_UTC_Time_Resolution
     (UTC_Time_Resolution :    out ECOA.Duration);

end #module_impl_name#_Container;
```

Parameters:

UTC_Time_Resolution This is the UTC time resolution returned by the operation.

11.4.6 Get_Absolute_System_Time_Resolution

```
package #module_impl_name#_Container is

   procedure Get_Absolute_System_Time_Resolution
     (Absolute_System_Time_Resolution :    out ECOA.Duration);

end #module_impl_name#_Container;
```

Parameters:

Absolute_System_Time_Resolution This is the absolute system time resolution returned by the operation.

11.5 Persistent Information management (PINFO)

11.5.1 PINFO Read

The syntax for a Module Instance to read persistent data (PINFO) is:

```
package #module_impl_name#_Container is

   procedure Read_#PINFOname#
     (Memory_Address  : in     ECOA.System_Address_Type;
```

```
      In_Size          : in     ECOA.PINFO_Size_Type;
      Out_Size         :    out ECOA.PINFO_Size_Type;
      Status           :    out ECOA.Read_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Memory_Address	This is the memory address where the read bytes will be stored.
In_Size	This is the requested number of bytes to read.
Out_Size	This is the number of bytes actually returned by the read.
Status	The status of the PINFO Read operation ECOA.Read_Return_Status_Type_OK if the read operation has been successful. ECOA.Read_Return_Status_Type_RESOURCE_NOT_AVAILABLE shall be returned if an infrastructure error occurred. ECOA.Read_Return_Status_Type_INVALID_PARAMETER shall be returned if the memory address is null or not accessible.PINFO write

11.5.2 PINFO Seek

The syntax for a Module Instance to seek within persistent data (PINFO) is:

```
package #module_impl_name#_Container is

   procedure Seek_#PINFOname#
     (Offset       : in     ECOA.PINFO_Offset_Type;
      Whence       : in     ECOA.Seek_Whence_Type;
      New_Position :    out ECOA.PINFO_Position_Type;
      Status       :    out ECOA.Seek_Return_Status_Type);

end #module_impl_name#_Container;
```

Parameters:

Offset	This is how much to offset the PINFO Index.
Whence	This is the reference from where the Offset is applied
New_Position	This is the resultant position of the PINFO index.
Status	The status of the PINFO Read operation ECOA.Write_Return_Status_Type_OK if the read operation has been successful. ECOA.Write_Return_Status_Type_RESOURCE_NOT_AVAILABLE shall be returned if an infrastructure error occurred. ECOA.Write_Return_Status_Type_INVALID_PARAMETER shall be returned

if the Offset and Whence values, when combined with the current PINFO Index results in an invalid PINFO Index.

11.5.3 Example of Defining Private PINFO

Not applicable to this binding.

11.5.4 Example of Defining Public PINFO

Not applicable to this binding.

11.6 Recovery Action

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

```
package #fault_handler_impl_name#_Container is

   procedure Recovery_Action
     (Recovery_Action : in     ECOA.Recovery_Action_Type;
      Asset_Id        : in     ECOA.Asset_Id_Type;
      Asset_Type      : in     ECOA.Asset_Type;
      Status          :    out ECOA.Recovery_Action_Return_Status_Type);

end #fault_handler_impl_name#_Container;
```

Parameters:

`Recovery_Action`	This is the recovery action to apply.
`Asset_Id`	This is ID of the Asset that the recovery action applies to.
`Asset_Type`	This is the type of Asset that the recovery action applies to.

11.7 Save Warm Start Context

The syntax for a Module Instance to save its warm start (non-volatile) context is:

```
package #module_impl_name#_Container is

   procedure Save_Warm_Start_Context;

end #module_impl_name#_Container;
```

12 Container Types

This section contains details of the data types that comprise the Container API i.e. the data types that can be used by a module.

12.1 Versioned Data Handles

This section contains the syntax in order to define data handles for versioned data operations defined in the Container Interface.

The definition of the Data element within the Versioned Data Handle is defined as a 'pointer' or Access type within the standard ECOA language bindings. Since SPARK 95 does not allow Access types, this has been changed to be defined as a real data element. This change has an effect on which part of the system allocates the memory for use by the Versioned Data handles. In ECOA it is the responsibility of the Container to allocate the required memory to a Versioned Data Handle, however with the High Integrity Binding it is now the responsibility of the application to allocate the memory that is populated by the Container operations defined in section 11.1.2. It remains the responsibility of the Container to allocate the memory used by the Versioned Data Updated Module operation defined in section 10.1.2.

NOTE The inclusion of the 'platform hook' field in the standard ECOA Data Handle is not desirable in high integrity systems as it may lead to corruption of platform data. For this reason the 'platform hook' field has been removed from the Data Handle structure. Any additional information required by the platform to manage the Versioned Data will have to be implemented in an alternative way.

In addition ECOA defines the Data field before the Stamp field, however for consistency with the Module Context type it is desirable to define the Stamp field first.

```
package #module_impl_name#_Container_Types is

type #operation_name#_Handle_Type is
  record
    Stamp    : ECOA.Unsigned_32_Type;
    Data     : #type_name#;
  end record;

end #module_impl_name#_Container_Types;
```

13 External Interface

This section contains the syntax for the ECOA external interface provided to non-ECOA software by the Container.

NOTE The choice of the language for generating external APIs is made separately from the choice of the language for generating ECOA modules APIs. The choice of supported languages is made depending on needs that are to be taken into account in platform procurement requirements.

```
-- @file #component_implementation_name#_External_Interface.ads
-- External Interface specification for Component
-- Implementation #component_implementation_name#
-- Generated automatically from specification; do not modify here

package #component_implementation_name#_External_Interface is

   procedure #external_operation_name#(#event_parameters#);

end #component_implementation_name#_External_Interface;
```

14 Default Values

Not applicable to this binding.

15 Trigger Instances

Not applicable to this binding.

16 Dynamic Trigger Instances

Not applicable to this binding.

17 Reference High Integrity Ada Specification

```
package ECOA is

   subtype Boolean_8_Type is Boolean;

   subtype Character_8_Type is Character
     range Character'Val(0) .. Character'Val(127);

   type Signed_8_Type is range -2 ** 7 + 1 .. 2 ** 7 - 1;
   for Signed_8_Type'Size use 8;
   for Signed_8_Type'Alignment use 1;

   type Signed_16_Type is range -2 ** 15 + 1 .. 2 ** 15 - 1;
   for Signed_16_Type'Size use 16;
   for Signed_16_Type'Alignment use 2;

   type Signed_32_Type is range -2 ** 31 + 1 .. 2 ** 31 - 1;
   for Signed_32_Type'Size use 32;
   for Signed_32_Type'Alignment use 4;

   type Signed_64_Type is range -2 ** 63 + 1 .. 2 ** 63 - 1;
   for Signed_64_Type'Size use 64;
   for Signed_64_Type'Alignment use 8;

   type Unsigned_8_Type is mod 2 ** 8;
   for Unsigned_8_Type'Size use 8;
   for Unsigned_8_Type'Alignment use 1;

   type Unsigned_16_Type is mod 2 ** 16;
   for Unsigned_16_Type'Size use 16;
   for Unsigned_16_Type'Alignment use 2;

   type Unsigned_32_Type is mod 2 ** 32;
```

```
   for Unsigned_32_Type'Size use 32;
   for Unsigned_32_Type'Alignment use 4;

   type Unsigned_64_Type is mod 2 ** 64;
   for Unsigned_64_Type'Size use 64;
   for Unsigned_64_Type'Alignment use 8;

   type Float_32_Type is digits 6 range -3.402823466e+38 .. 3.402823466e+38;
   for Float_32_Type'Size use 32;
   for Float_32_Type'Alignment use 4;

   type Float_64_Type is digits 15 range -1.7976931348623157e+308 ..
     1.7976931348623157e+308;
   for Float_64_Type'Size use 64;
   for Float_64_Type'Alignment use 8;

   type Byte_8_Type is mod 2 ** 8;
   for Byte_8_Type'Size use 8;
   for Byte_8_Type'Alignment use 1;

   type Return_Status_Type is
      (Return_Status_Type_OK,
       Return_Status_Type_INVALID_HANDLE,
       Return_Status_Type_DATA_NOT_INITIALIZED,
       Return_Status_Type_NO_DATA,
       Return_Status_Type_INVALID_IDENTIFIER,
       Return_Status_Type_NO_RESPONSE,
       Return_Status_Type_OPERATION_ALREADY_PENDING,
       Return_Status_Type_CLOCK_UNSYNCHRONIZED,
       Return_Status_Type_RESOURCE_NOT_AVAILABLE,
       Return_Status_Type_OPERATION_NOT_AVAILABLE,
       Return_Status_Type_INVALID_PARAMETER);
   for Return_Status_Type use
      (Return_Status_Type_OK                          => 0,
       Return_Status_Type_INVALID_HANDLE              => 1,
       Return_Status_Type_DATA_NOT_INITIALIZED        => 2,
       Return_Status_Type_NO_DATA                     => 3,
       Return_Status_Type_INVALID_IDENTIFIER          => 4,
       Return_Status_Type_NO_RESPONSE                 => 5,
       Return_Status_Type_OPERATION_ALREADY_PENDING => 6,
       Return_Status_Type_CLOCK_UNSYNCHRONIZED        => 7,
       Return_Status_Type_RESOURCE_NOT_AVAILABLE      => 8,
       Return_Status_Type_OPERATION_NOT_AVAILABLE     => 9,
       Return_Status_Type_INVALID_PARAMETER           => 10);
```

```
for Return_Status_Type'Size use
  Unsigned_32_Type'Size;
for Return_Status_Type'Alignment use
  Unsigned_32_Type'Alignment;

type Response_Received_Return_Status_Type is
  (Response_Received_Return_Status_Type_OK,
   Response_Received_Return_Status_Type_NO_RESPONSE);
for Response_Received_Return_Status_Type use
  (Response_Received_Return_Status_Type_OK          => 0,
   Response_Received_Return_Status_Type_NO_RESPONSE => 5);
for Response_Received_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Response_Received_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Request_Sync_Return_Status_Type is
  (Request_Sync_Return_Status_Type_OK,
   Request_Sync_Return_Status_Type_NO_RESPONSE);
for Request_Sync_Return_Status_Type use
  (Request_Sync_Return_Status_Type_OK          => 0,
   Request_Sync_Return_Status_Type_NO_RESPONSE => 5);
for Request_Sync_Return_Status_Type'Size use Unsigned_32_Type'Size;
for Request_Sync_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Request_Async_Return_Status_Type is
   (Request_Async_Return_Status_Type_OK,
    Request_Async_Return_Status_Type_RESOURCE_NOT_AVAILABLE);
for Request_Async_Return_Status_Type use
   (Request_Async_Return_Status_Type_OK                    => 0,
    Request_Async_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 8);
for Request_Async_Return_Status_Type'Size use Unsigned_32_Type'Size;
for Request_Async_Return_Status_Type'Alignment use
  Unsigned_32_Type'Alignment;

type Response_Send_Return_Status_Type is
  (Response_Send_Return_Status_Type_OK,
   Response_Send_Return_Status_Type_INVALID_IDENTIFIER);
for Response_Send_Return_Status_Type use
  (Response_Send_Return_Status_Type_OK                => 0,
   Response_Send_Return_Status_Type_INVALID_IDENTIFIER => 4);
for Response_Send_Return_Status_Type'Size use Unsigned_32_Type'Size;
for Response_Send_Return_Status_Type'Alignment
```

```
  use Unsigned_32_Type'Alignment;

type Get_Read_Access_Return_Status_Type is
  (Get_Read_Access_Return_Status_Type_OK,
   Get_Read_Access_Return_Status_Type_INVALID_HANDLE,
   Get_Read_Access_Return_Status_Type_NO_DATA,
   Get_Read_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE);
for Get_Read_Access_Return_Status_Type use
  (Get_Read_Access_Return_Status_Type_OK                    => 0,
   Get_Read_Access_Return_Status_Type_INVALID_HANDLE        => 1,
   Get_Read_Access_Return_Status_Type_NO_DATA               => 3,
   Get_Read_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 10);
for Get_Read_Access_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Get_Read_Access_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Release_Read_Access_Return_Status_Type is
  (Release_Read_Access_Return_Status_Type_OK,
   Release_Read_Access_Return_Status_Type_INVALID_HANDLE);
for Release_Read_Access_Return_Status_Type use
  (Release_Read_Access_Return_Status_Type_OK             => 0,
   Release_Read_Access_Return_Status_Type_INVALID_HANDLE => 1);
for Release_Read_Access_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Release_Read_Access_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Get_Write_Access_Return_Status_Type is
  (Get_Write_Access_Return_Status_Type_OK,
   Get_Write_Access_Return_Status_Type_INVALID_HANDLE,
   Get_Write_Access_Return_Status_Type_DATA_NOT_INITIALIZED,
   Get_Write_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE);
for Get_Write_Access_Return_Status_Type use
  (Get_Write_Access_Return_Status_Type_OK                    => 0,
   Get_Write_Access_Return_Status_Type_INVALID_HANDLE        => 1,
   Get_Write_Access_Return_Status_Type_DATA_NOT_INITIALIZED   => 2,
   Get_Write_Access_Return_Status_Type_RESOURCE_NOT_AVAILABLE => 10);
for Get_Write_Access_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Get_Write_Access_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Cancel_Write_Access_Return_Status_Type is
```

```
   (Cancel_Write_Access_Return_Status_Type_OK,
    Cancel_Write_Access_Return_Status_Type_INVALID_HANDLE);
for Cancel_Write_Access_Return_Status_Type use
   (Cancel_Write_Access_Return_Status_Type_OK             => 0,
    Cancel_Write_Access_Return_Status_Type_INVALID_HANDLE => 1);
for Cancel_Write_Access_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Cancel_Write_Access_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Publish_Write_Access_Return_Status_Type is
   (Publish_Write_Access_Return_Status_Type_OK,
    Publish_Write_Access_Return_Status_Type_INVALID_HANDLE);
for Publish_Write_Access_Return_Status_Type use
   (Publish_Write_Access_Return_Status_Type_OK             => 0,
    Publish_Write_Access_Return_Status_Type_INVALID_HANDLE => 1);
for Publish_Write_Access_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Publish_Write_Access_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Get_UTC_Time_Return_Status_Type is
   (Get_UTC_Time_Return_Status_Type_OK,
    Get_UTC_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED);
for Get_UTC_Time_Return_Status_Type use
   (Get_UTC_Time_Return_Status_Type_OK                  => 0,
    Get_UTC_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED => 8);
for Get_UTC_Time_Return_Status_Type'Size use Unsigned_32_Type'Size;
for Get_UTC_Time_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;

type Get_Absolute_System_Time_Return_Status_Type is
   (Get_Absolute_System_Time_Return_Status_Type_OK,
    Get_Absolute_System_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED);
for Get_Absolute_System_Time_Return_Status_Type use
   (Get_Absolute_System_Time_Return_Status_Type_OK
      => 0,
    Get_Absolute_System_Time_Return_Status_Type_CLOCK_UNSYNCHRONIZED
      => 8);
for Get_Absolute_System_Time_Return_Status_Type'Size
  use Unsigned_32_Type'Size;
for Get_Absolute_System_Time_Return_Status_Type'Alignment
  use Unsigned_32_Type'Alignment;
```

```
type Recovery_Action_Return_Status_Type is
   (Recovery_Action_Return_Status_Type_OK,
    Recovery_Action_Return_Status_Type_INVALID_IDENTIFIER,
    Recovery_Action_Return_Status_Type_OPERATION_ALREADY_PENDING,
    Recovery_Action_Return_Status_Type_OPERATION_NOT_AVAILABLE);
for Recovery_Action_Return_Status_Type use
   (Recovery_Action_Return_Status_Type_OK
     => 0,
    Recovery_Action_Return_Status_Type_INVALID_IDENTIFIER
     => 4,
    Recovery_Action_Return_Status_Type_OPERATION_ALREADY_PENDING
     => 6,
    Recovery_Action_Return_Status_Type_OPERATION_NOT_AVAILABLE
     => 11);
for Recovery_Action_Return_Status_Type'Size use
  Unsigned_32_Type'Size;
for Recovery_Action_Return_Status_Type'Alignment use
  Unsigned_32_Type'Alignment;

type Read_Return_Status_Type is
   (Read_Return_Status_Type_OK,
    Read_Return_Status_Type_RESOURCE_NOT_AVAILABLE,
    Read_Return_Status_Type_INVALID_PARAMETER);
for Read_Return_Status_Type use
   (Read_Return_Status_Type_OK
     => 0,
    Read_Return_Status_Type_RESOURCE_NOT_AVAILABLE
     => 10,
    Read_Return_Status_Type_INVALID_PARAMETER
     => 13);
for Read_Return_Status_Type'Size use
  Unsigned_32_Type'Size;
for Read_Return_Status_Type'Alignment use
  Unsigned_32_Type'Alignment;

type Seek_Return_Status_Type is
   (Seek_Return_Status_Type_OK,
    Seek_Return_Status_Type_RESOURCE_NOT_AVAILABLE,
    Seek_Return_Status_Type_INVALID_PARAMETER);
for Seek_Return_Status_Type use
   (Seek_Return_Status_Type_OK
     => 0,
    Seek_Return_Status_Type_RESOURCE_NOT_AVAILABLE
     => 10,
```

```
      Seek_Return_Status_Type_INVALID_PARAMETER
        => 13);
  for Seek_Return_Status_Type'Size use Unsigned_32_Type'Size;
  for Seek_Return_Status_Type'Alignment use
    Unsigned_32_Type'Alignment;

  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;

  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;

  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;

  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;

  type Log_Elements_Size_Type is range 0..256;
  for Log_Elements_Size_Type'Size use 32;
  for Log_Elements_Size_Type'Alignment use 4;
  subtype Log_Elements_Index_Type is Log_Elements_Size_Type range 0..255;

  type Log_Elements_Type is array (Log_Elements_Index_Type) of
    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_Size_Type;
      Data         : Log_Elements_Type;
   end record;
for Log_Type'Size use 2080;
for Log_Type'Alignment use 4;

subtype Error_ID_Type is Unsigned_32_Type
  range Unsigned_32_Type'First .. Unsigned_32_Type'Last;

subtype Asset_ID_Type is Unsigned_32_Type
  range Unsigned_32_Type'First .. Unsigned_32_Type'Last;

type Asset_Type is
  (Asset_Type_COMPONENT,
   Asset_Type_PROTECTION_DOMAIN,
   Asset_Type_NODE,
   Asset_Type_PLATFORM,
   Asset_Type_SERVICE,
   Asset_Type_DEPLOYMENT);
for Asset_Type use
  (Asset_Type_COMPONENT          => 0,
   Asset_Type_PROTECTION_DOMAIN => 1,
   Asset_Type_NODE               => 2,
   Asset_Type_PLATFORM           => 3,
   Asset_Type_SERVICE            => 4,
   Asset_Type_DEPLOYMENT         => 5);
for Asset_Type'Size use 32;
for Asset_Type'Alignment use 4;

type Error_Type is
  (Error_Type_RESOURCE_NOT_AVAILABLE,
   Error_Type_UNAVAILABLE,
   Error_Type_MEMORY_VIOLATION,
   Error_Type_NUMERICAL_ERROR,
   Error_Type_ILLEGAL_INSTRUCTION,
   Error_Type_STACK_OVERFLOW,
   Error_Type_DEADLINE_VIOLATION,
   Error_Type_OVERFLOW,
   Error_Type_UNDERFLOW,
```

```
      Error_Type_ILLEGAL_INPUT_ARGS,
      Error_Type_ILLEGAL_OUTPUT_ARGS,
      Error_Type_ERROR,
      Error_Type_FATAL_ERROR,
      Error_Type_HARDWARE_FAULT,
      Error_Type_POWER_FAIL,
      Error_Type_COMMUNICATION_ERROR,
      Error_Type_INVALID_CONFIG,
      Error_Type_INITIALISATION_PROBLEM,
      Error_Type_CLOCK_UNSYNCHRONIZED,
      Error_Type_UNKNOWN_OPERATION,
      Error_Type_OPERATION_OVERRATED,
      Error_Type_OPERATION_UNDERRATED);
   for Error_Type use
     (Error_Type_RESOURCE_NOT_AVAILABLE =>  0,
      Error_Type_UNAVAILABLE            =>  1,
      Error_Type_MEMORY_VIOLATION       =>  2,
      Error_Type_NUMERICAL_ERROR        =>  3,
      Error_Type_ILLEGAL_INSTRUCTION    =>  4,
      Error_Type_STACK_OVERFLOW         =>  5,
      Error_Type_DEADLINE_VIOLATION     =>  6,
      Error_Type_OVERFLOW               =>  7,
      Error_Type_UNDERFLOW              =>  8,
      Error_Type_ILLEGAL_INPUT_ARGS     =>  9,
      Error_Type_ILLEGAL_OUTPUT_ARGS    => 10,
      Error_Type_ERROR                  => 11,
      Error_Type_FATAL_ERROR            => 12,
      Error_Type_HARDWARE_FAULT         => 13,
      Error_Type_POWER_FAIL             => 14,
      Error_Type_COMMUNICATION_ERROR    => 15,
      Error_Type_INVALID_CONFIG         => 16,
      Error_Type_INITIALISATION_PROBLEM => 17,
      Error_Type_CLOCK_UNSYNCHRONIZED   => 18,
      Error_Type_UNKNOWN_OPERATION      => 19,
      Error_Type_OPERATION_OVERRATED    => 20,
      Error_Type_OPERATION_UNDERRATED   => 21);
   for Error_Type'Size use 32;
   for Error_Type'Alignment use 4;

   type Recovery_Action_Type is
     (Recovery_Action_Type_SHUTDOWN,
      Recovery_Action_Type_COLD_RESTART,
      Recovery_Action_Type_WARM_RESTART,
      Recovery_Action_Type_CHANGE_DEPLOYMENT);
```

```
for Recovery_Action_Type use
  (Recovery_Action_Type_SHUTDOWN          => 0,
   Recovery_Action_Type_COLD_RESTART      => 1,
   Recovery_Action_Type_WARM_RESTART      => 2,
   Recovery_Action_Type_CHANGE_DEPLOYMENT => 3);
for Recovery_Action_Type'Size use 32;
for Recovery_Action_Type'Alignment use 4;

type PINFO_Filename_Elements_Size_Type is range 0..256;
for PINFO_Filename_Elements_Size_Type'Size use 32;
for PINFO_Filename_Elements_Size_Type'Alignment use 4;
subtype PINFO_Filename_Elements_Index_Type is
  PINFO_Filename_Elements_Size_Type range 0..255;

type PINFO_Filename_Elements_Type is
  array (PINFO_Filename_Elements_Index_Type) of Character_8_Type;
for PINFO_Filename_Elements_Type'Size use 2048;
for PINFO_Filename_Elements_Type'Alignment use 4;

type PINFO_Filename_Type is
   record
      Current_Size : PINFO_Filename_Elements_Size_Type;
      Data         : PINFO_Filename_Elements_Type;
   end record;
for PINFO_Filename_Type'Size use 2080;
for PINFO_Filename_Type'Alignment use 4;

type Seek_Whence_Type is
   (Seek_Whence_Type_SEEK_SET,
    Seek_Whence_Type_SEEK_CUR,
    Seek_Whence_Type_SEEK_END);
for Seek_Whence_Type use
   (Seek_Whence_Type_SEEK_SET => 0,
    Seek_Whence_Type_SEEK_CUR => 1,
    Seek_Whence_Type_SEEK_END => 2);
for Seek_Whence_Type'Size use Unsigned_32_Type'Size;
for Seek_Whence_Type'Alignment use Unsigned_32_Type'Alignment;

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

type PINFO_Size_Type is mod 2 ** 32;
for PINFO_Size_Type'Size use 32;
```

```
   for PINFO_Size_Type'Alignment use 4;

   type PINFO_Offset_Type is range -2 ** 31 + 1  .. 2 ** 31 - 1;
   for PINFO_Offset_Type'Size use 32;
   for PINFO_Offset_Type'Alignment use 4;

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

   type System_Address_Type is private;

   Default_System_Address : constant System_Address_Type;

   private
      --# hide ECOA;
      type System_Address_Type is new System.Address;

      function To_System_Address is new
        Ada.Unchecked_Conversion(System.Address, System_Address_Type);

      Default_System_Address : constant System_Address_Type :=
        To_System_Address(System.Null_Address);

end ECOA;
```