European Component Oriented Architecture (ECOA®) Collaboration Programme: Preliminary version of the ECOA Architecture Specification Part 7: Metamodel

Dassault Ref No: DGT 2041086-A
Thales DMS Ref No: 69398925-035 --

Issue: 7

Prepared by
Dassault Aviation and Thales DMS

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Note: *This specification is preliminary and is subject to further adjustments. Consequently, users are advised to exercise caution when relying on the information herein. No warranties are provided regarding the completeness or accuracy of the information in this preliminary version. The final version of the document will be released to reflect further improvements.*

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

This Page Intentionally Left Blank

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

This Page Intentionally Left Blank

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Contents

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. The details of the other documents comprising the rest of this Architecture Specification can be found in Section 3.

This document is Part 7 of the Architecture Specification, and contains the Metamodel and XML schema definitions for an ECOA® system.

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 shall rely on the BNAE publications of the ECOA standard.

3 Normative References

Architecture Specification Part 1	Dassault Ref No: DGT 2041078-A Thales DMS Ref No: 69398915-035 -- Issue 7 Architecture Specification Part 1 – Concepts
Architecture Specification Part 2	Dassault Ref No: DGT 2041081-A Thales DMS Ref No: 69398916-035 -- Issue 7 Architecture Specification Part 2 – Definitions
Architecture Specification Part 3	Dassault Ref No: DGT 2041082-A Thales DMS Ref No: 69398917-035 -- Issue 7 Architecture Specification Part 3 – Mechanisms
Architecture Specification Part 4	Dassault Ref No: DGT 2041083-A Thales DMS Ref No: 69398918-035 -- Issue 7 Architecture Specification Part 4 – Software Interface
Architecture Specification Part 5	Dassault Ref No: DGT 2041084-A Thales DMS Ref No: 69398919-035 -- Issue 7 Architecture Specification Part 5 – High Level Platform Requirements
Architecture Specification Part 6	Dassault Ref No: DGT 2041491-A Thales DMS Ref No: 69398920-035 -- Issue 7 Architecture Specification Part 6 – Options
Architecture Specification Part 7	Dassault Ref No: DGT 2041086-A Thales DMS Ref No: 69398925-035 -- Issue 7 Architecture Specification Part 7 – Metamodel

4 Definitions

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

5 Abbreviations

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ECOA	European Component Oriented Architecture. ECOA® is a registered trademark.
FIFO	First In, First Out
gRPC	Google Remote Procedure Calls
ID	Identifier
IT	Information Technology
OS	Operating System
PINFO	Persistent Information
QoS	Quality of Service
UML	Unified Modelling Language
URL	Universal Resource Locator
XML	eXtensible Markup Language
XSD	XML Schema Definition

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6 ECOA Meta Models

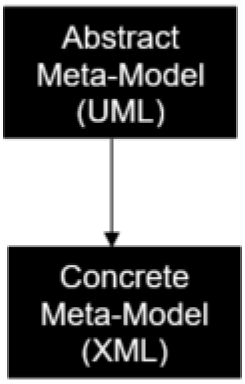


Figure 1 ECOA Meta Models

The structure of an ECOA software architecture is specified as an abstract metamodel which describes the ECOA software architecture's data elements and their relationships. UML entity-relationship diagrams are used to present the model information which can be found in Section 6.1.

Ultimately, the requirements to exchange components, and automatically instantiate technical software architecture from them, require a precisely-specified and machine-readable version of the model. This is known as the concrete metamodel and the implementation is based on XML language specified in several XSD files. Section 6.2 defines the concrete metamodel.

It is envisaged that the ECOA implementers will ultimately develop tool support that enables ECOA information to be captured in high-level design tools that support, for example, UML. However, the XML-based concrete metamodel will remain the standard for exchange of information (e.g. between component suppliers and system integrators).

6.1 Abstract Metamodel

6.1.1 Overview

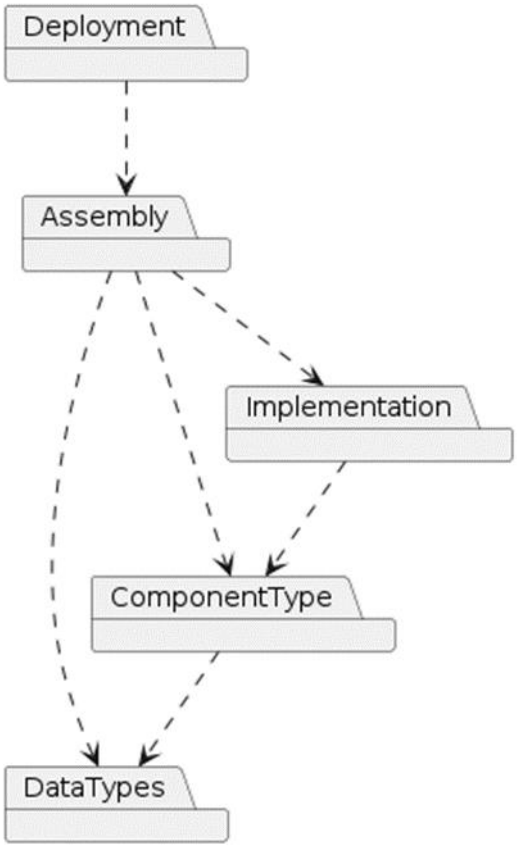


Figure 2 Overview of the ECOA Metamodel

The following sections detail the ECOA abstract metamodel. They aim to provide a definition of all the concepts and objects that need to be formalized to describe an ECOA Application. This abstract metamodel is designed using a set of self-sufficient views of a UML model; each view of the metamodel is describing one given concept. An overview of this metamodel is shown in Figure 2, which describes overarching dependencies between main ECOA concepts.

The DataTypes part allows to formalize definitions of common data structures found in every programming language: integer and floating point numbers of different sizes, arrays, records (or structs), enumerations, etc.

A Component Type (section 6.1.2) captures the interface of a component and is formalized in terms of the operations that exchange information with other components.

A (component) Implementation captures the realization of a Component Type in terms of software (source code or binary form). A Component Type may have different Implementations.

An Assembly model (section 6.1.4) describes the logical structure of an ECOA Application, in a hardware-independent fashion. Essentially, it instanciates some Components and describes links between them.

In addition, a Component is either "concrete", i.e. directly implemented in a given programming language, or "composite", i.e. made of an assembly of other Components.

An ECOA Application is fully formalized by a Deployment model (section 6.1.6) which specifies how Components are deployed on a Computing Node: it details how components are mapped onto tasks and executables.

An ECOA Application is always deployed on a single Computing Node. ECOA Standard extensions may be defined in order to formalize the deployment of several Applications on several Computing Nodes ("multi-applications assembly").

6.1.2 Component Types

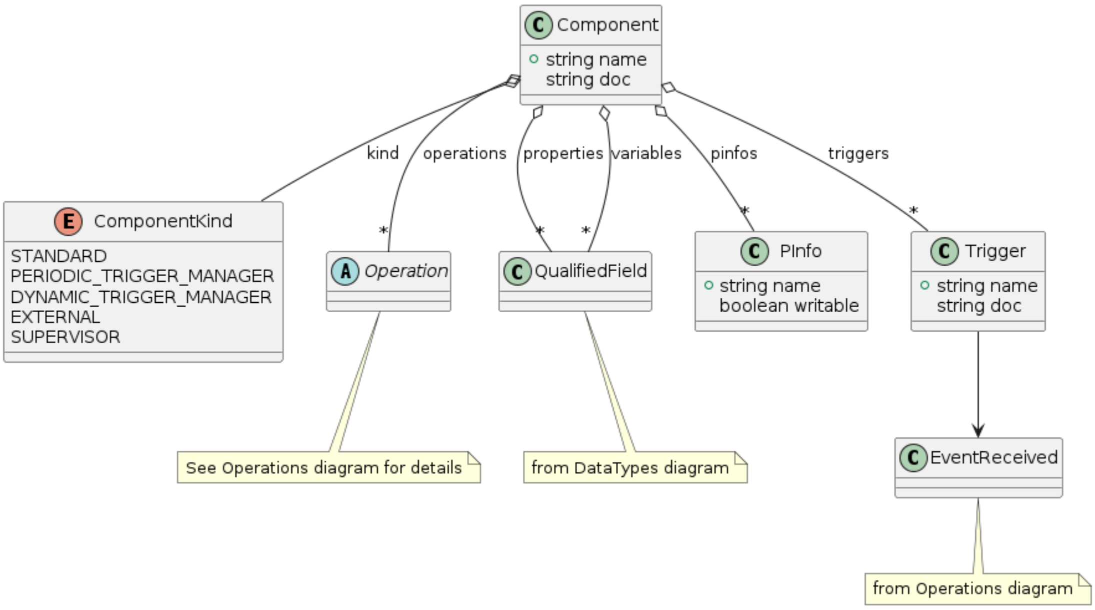


Figure 3 Component Type metamodel

Note: In this figure and the following figures, a round green symbol next to an attribute name means that this attribute is mandatory. All other attributes are optional, often because they have a default value.

A **Component Type** is a set of **Operations, Properties**, **Pinfo, Triggers** and **Variables** definitions. A Component Type defines the interface of a Component.

A **Trigger** definition is a way for the component to generate events at a specified time, that are sent to itself. The time can be dynamically set at runtime through a specific container API. The definition of a trigger in a component will generate a specific trigger API in the container (to start the trigger) and a specific API in the component to manage the action associated to the trigger (what will happen when the trigger is expired).

Note: Trigger defintions are meaningful for concrete implementations only. They have no effect on composite implementations.

PINFO, persistent information, is a way of accessing, by default in read only mode, persistent information as if it were stored locally in a file through read and seek APIs. A Component Type may contain an optional

pinfo items for declaring PINFO accessible to instances of that Component Type. Each pinfo has a unique name in the Component Type scope.

A pinfo item may also be declared 'writable'. In this case, the API will include additional functions allowing the component to modify the persistent information. This feature is part of [OPTION PINFO_WRITE] and is not supported by all platforms.

An **Operation** is either a **Data Operation**, an **Event Operation** or a **RequestResponse Operation**.

Operation names shall be unique in a Component Type.

A Data Operation may be **written** or **read** by the component.

An Event Operation may be **sent** or **received** by the component.

A RequestResponse Operation may be **required** or **provided** by the Component.

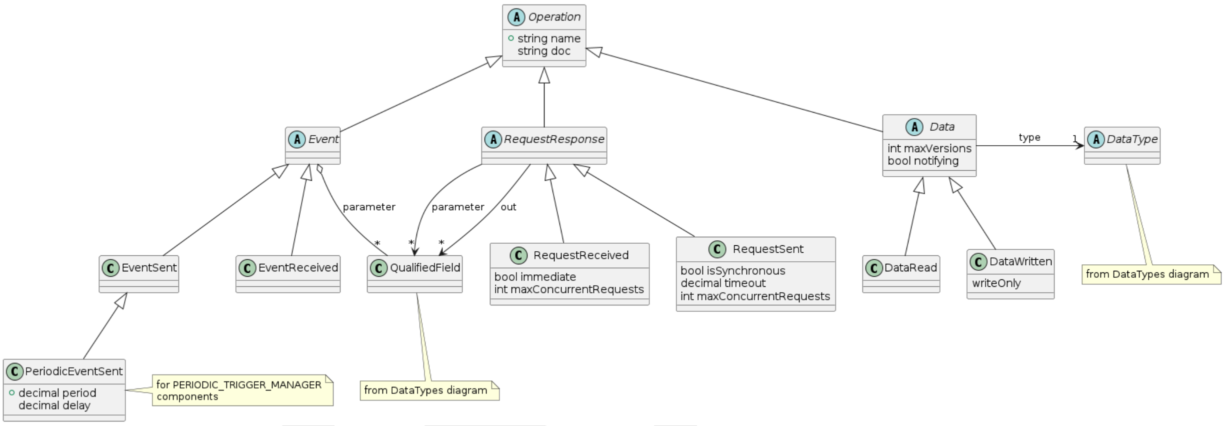


Figure 4 - Component Type Operations

Each **DataRead** *(pronounce "data red")* operation that indicates the component is a reader of a versioned data has the following attributes:

- **maxVersions**: the maximum number of versions that the component may access simultaneously in read mode (default is 1),
- **notifying**: if ‘true’, an entry point is generated by the platform tooling and is called by the container each time it is aware of a data update (default is false).

Each **DataWritten** operation that indicates the component is the writer of a given versioned data relies on the following attributes:

- **maxVersions**: the maximum of versions that the component may access simultaneously in read-write mode (default is 1).

- **writeOnly**: the possibility for a writer of a versioned data to have a “write only” access to the data, thereby avoiding the platform to initialize the local copy of the data with the current value. When this flag is set, the writer component will not have access to the current value of the shared data (default is false).

Note that, except when writeOnly is true, a 'write' operation includes an implicit 'read' operation. Thus the attribute 'notifying' is also applicable to 'write' operations. This attribute is only meaningful if writeOnly=false.

Each **RequestReceived** operation which is the entry-point called on the receiving of a request has the following attribute:

- **immediate**: if 'true', the response is immediate, i.e. sent when the execution of the entry point of the receiver component ends its execution. There is no "send response" container API to call (default is false).
- **maxConcurrentRequests**: the maximum number of R/R IDs that the component may retain for that entry-point before sending the associated replies (default value is 8). If this number is reached, additional R/R cannot be retained and are discarded by the Container.

Each **RequestSent** operation which indicates the component may send a request towards another component relies on the following attributes

- **isSynchronous**: if true, the R/R is synchronous: the call blocks the calling component until the receiving of the response or the expiration of the timeout (default is false).
- **timeout**: the maximum time spent waiting for a reply
When isSynchronous is true, it is the maximum time during which the component is blocked waiting for a reply. If the timeout is set to 0 or negative, the R/R is an indefinite blocking call.
When isSynchronous is false, it is the maximum time before the Container sends a NO_RESPONSE to the component. If the timeout is set to 0 or negative, the Container will only send a response to the component if there is a reply (default value is 0).

The operations' names will appear in the component's container API. Each operation name shall be unique for a given Component Type.

A Component Instance corresponds to an instance of a Component Implementation, which itself is of a defined Component Type. A Component Instance has its own internal state. The container activates the Component Instance as long as there are incoming activating operations for it.

The notion of Component Instance provides the ability to instantiate, a number of times, the same software code in multiple execution contexts and with different configurations (via Properties) inside an ECOA Application.

6.1.2.1 Specific kinds of Component Type

A component type may be a standard one that may define all the classical features described in the previous chapter. The STANDARD component type defines component that are completely HW agnostic, easily reusable and only dependant on their Component Type definition (contract).

But a software architecture is never so ideal, and specific kind of component types are available to address that.

A **PERIODIC_TRIGGER_MANAGER** component type is a specific type of component that will not require any line of manual code and that is dedicated to sending periodic events. These events are declared as classical sent events, without parameters, but with a **period** and optionally a **delay**. In the assembly model, those events are then managed as usual and may be linked with any event received operation that has no parameter. The code for this component is fully generated and not customisable. PERIODIC_TRIGGER_MANAGER components are part of the ECOA Core Architecture Specification.

A **DYNAMIC_TRIGGER_MANAGER** component type is a specific type of component that will not require any line of manual code and that is dedicated to sending events with a delay. The component defines a received event named 'in' with a parameter 'expiryTime' and other parameters Pi, and a sent event named 'out' with the parameters Pi. It also defines a 'reset' received event. In the assembly model, those events are then managed as usual. The code for this component is fully generated and not customisable. DYNAMIC_TRIGGER_MANAGER components are part of ECOA option Architeture Specification [OPTION DYNAMIC TRIGGER].

An **EXTERNAL** component type is a specific type of component that will manage external interfaces that are not defined in ECOA models (connection to a specific data bus, management of a specific HW, etc.). This kind of components usually need to have a dedicated task to wait, for example, for messages coming from the non-ECOA world. This kind of components is the only one that will not respect the mono-threading rule and that will have a dedicated additional thread, called the “external” thread. The priority of the external thread is "immediately inferior to" the priority of the normal thread of the component (defined in the deployment model). EXTERNAL components are part of the ECOA Core Architecture Specification.

A **SUPERVISOR** component type is the only type of component that will have the knowledge of the current Application's assembly and deployment. This component can access the list of component instances of the Application, access to the state of all component instances, and control it (initialize, start, stop, shutdown cf. Component state diagram). It can also start and stop the Executables of the Application. This feature is not supported by all platforms, it is part of [OPTION SUPERVISION].

6.1.3 Component Implementation

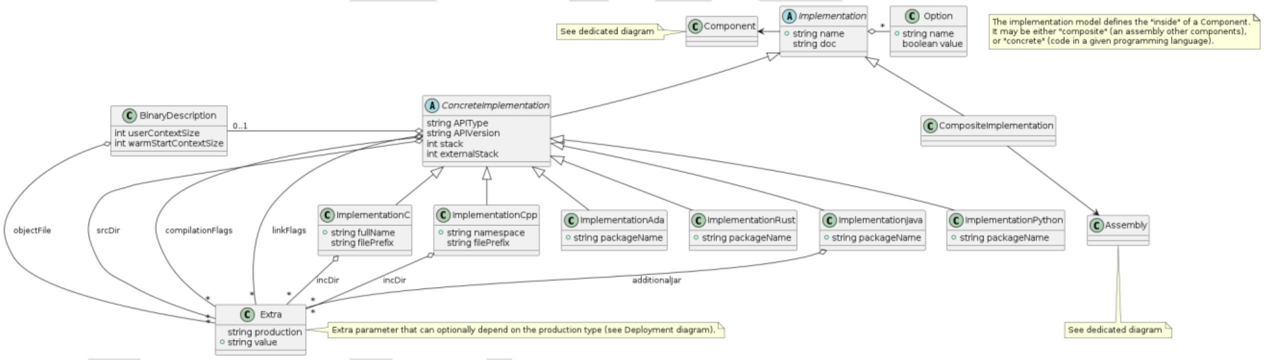


Figure 5 Component Implementation metamodel

A software realization of a Component Type is described by a **Component Implementation**.

There are two kinds of implementations for a component: concrete implementations and composite implementations.

A concrete ComponentImplementation (**ConcreteImplementation**) corresponds to a piece of software implementing in a certain programming language a given Component Type. It must be executable in a single thread (no parallelism, no internal synchronisation), except for the case of EXTERNAL components.

A composite ComponentImplementation (**CompositeImplementation**) corresponds to an assembly of other components (see Assembly model below). The notion of composite is recursive, i.e. some of those other components may themselves be composite.

The **stack** attribute defines the maximum stack size, in bytes, used by the implementation, considering all entry points of the component contract. This value will be used to calculate the thread stack. The default value is 32768 bytes.

The **externalStack** is similar, but for the additional thread of an EXTERNAL component.- The default value is the same as for the stack attribute.

The **APIType** is an optional attribute that identifies the name of the language binding used by this implementation. , The default value is "standard", corresponding to the language binding defined by the ECOA Standard.

The **APIVersion** is an optional attribute that identifies a possible version of the API, used by this implementation. If this attribute is not specified, no version check will be performed by tools.

If a **BinaryDescription** element is present, then the component is assumed to be provided in compiled form and not in source form. In this case, the BinaryDescription element defines a list of object files or libraries (typically with extension ".o" or ".a") that make up the compiled code of the component implementation. These files can be specific to a given production type. This allows, for example, to provide a component implementation with compiled code for both Intel and ARM architectures. The filenames of object files are relative to the implementation directory.

The BinaryDescription also defines sizes for the data used by a component: the user context size, and the warm start context size (default is 0). When the source code is provided, the execution platform can compute theses sizes itself (with a sizeof() in C, for example). But when the source code is not provided, these sizes must be provided in addition to the binary code. They must be sufficient large to contain the corresponding data structures.

The **Option** elements allow to declare that the component implementation requires some features of the software execution platform. The list of possible options is extensible. Every option has a boolean value.

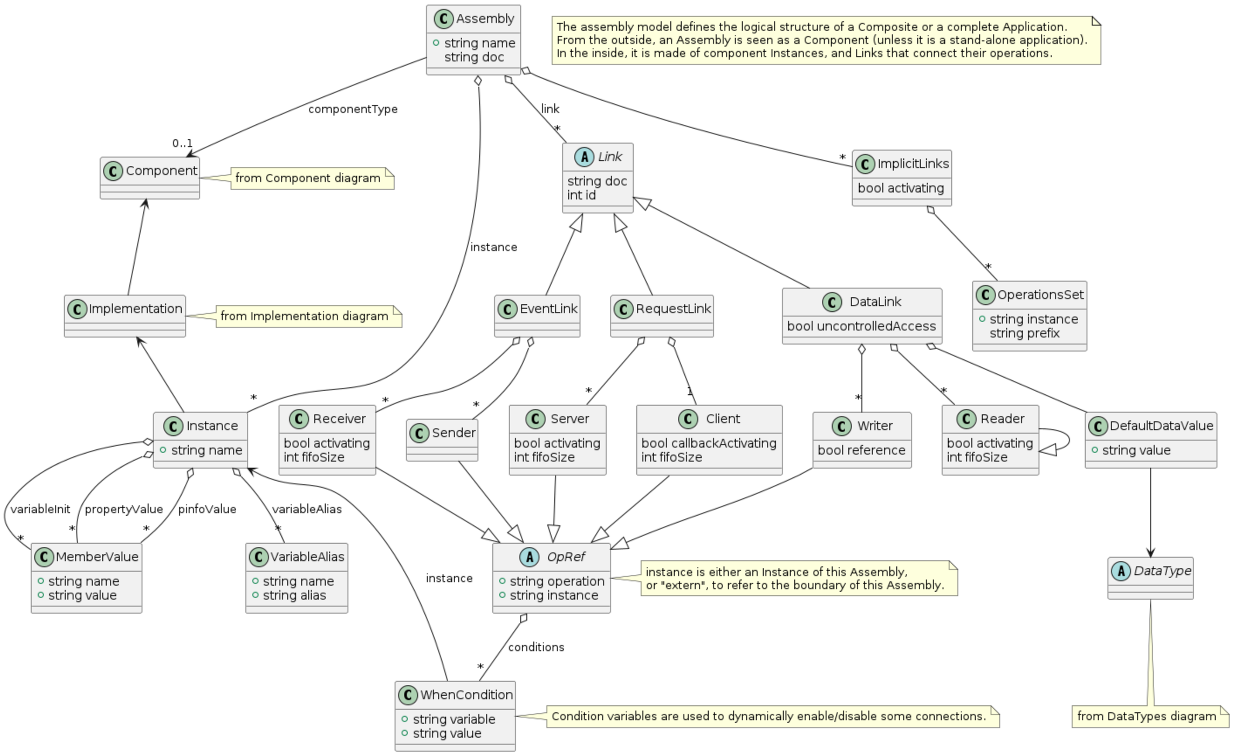
Among all the possible options, some have a name and a semantic that is defined in the ECOA Architecture Specification.

The options that are defined in the ECOA Architecture Specification are the following:

Name	**Description of the option**
needsLocalTime	The component's code uses the "Relative Local Time" clock. If false, the corresponding API may not be avaiblable.
needsSystemTime	The component's code uses the "Absolute System Time" clock. If false, the corresponding API may not be avaiblable.
needsUTCTime	The component's code uses the "UTC Time" clock. If false, the corresponding API may not be avaiblable.
needsTimeResolution	The component's code uses the container " clock. If false, the corresponding API may not be avaiblable.

hasReset	If true, an entry-point exists for the RESET transition of the component (and thus "RESET__received" needs to be implemented by this implementation).
isFaultHandler	To indicate if the component is a Fault Handler (i.e. has a fault handling API). This feature is not supported by all platforms [OPTION FAULT HANDLER].
autostartExternalThread	Applicable only for "EXTERNAL" components. If true, the external thread is started automatically, before component init. If false, the external thread must be started by the component code using the container API. This feature is not supported by all platforms [OPTION AUTO START EXTERNAL TASK].
hasWarmStartContext	To indicate if the component has a "warm start context". This feature is only supported by platforms that implement [OPTION WARM_CONTEXT].

6.1.4 Assembly



The assembly model defines the logical structure of a Composite or a complete Application. From the outside, an Assembly is seen as a Component (unless it is a stand-alone application). In the inside, it is made of component Instances, and Links that connect their operations.

from Component diagram

from Implementation diagram

instance is either an Instance of this Assembly, or "extern", to refer to the boundary of this Assembly.

Condition variables are used to dynamically enable/disable some connections.

from DataTypes diagram

Figure 6 Assembly metamodel

An **Assembly** describes the logical structure of an ECOA application, or the structure of a Composite Component. It is independent of any physical deployment on hardware platforms. Different Assemblies may be used in order to describe different ways to instanciate and link ECOA components.

An Assembly may reference a **ComponentType**, which defines its interface. If this reference is absent, it means that the Assembly has an empty interface (i.e. containing no operations and no properties). This is the case for the assembly of a "closed" ECOA application, e.g. when an application does not communicate with the outside world in terms of ECOA interfaces.

An Assembly is made of **Components instances** and **Operation links**.

6.1.4.1 Component Instances

A Component Instance corresponds to the instantiation of a **Component Implementation** (which can be concrete or composite). It has a set of instantiation parameters known as Properties defined in the Component type definition. It also has a set PINFO values, variable initial values, and variable aliases.

Inside a composite, the value of a property or a PINFO can be set to the value of a corresponding property or a PINFO of the encompassing composite.

For example:

- Component A is a composite instance that contains instances B and C
- Component A has a property X of type int32, which is set to value 32.
- Component B has a property Y of type int32, which is set to value '$X', which is evaluated to 32 in this context.

Note: In the context of an assembly, the simple term "**instance**" designates a Component Instance.

6.1.4.2 Operation Links

The **Operation Links** describe the interactions/synchronisations within the same Assembly. Interactions between Component Instances are specified using **DataLink**, **EventLink** or **RequestLink** (aka RequestResponseLink), depending on the kind of component operations that are linked. These three kinds of links are oriented and have different possible multiplicities:

- A DataLink may have n writers and p readers (a unique data-writer is recommended but not mandatory),
- A RequestLink has exactly one client, and may have several servers for failover cases (if several clients are wanted, create one RequestLink per client, connected to the same server),
- An EventLink can have n possible senders and p possible receivers of the event.

A component instance operation is referred to by an **instance** attribute, and an **operation** attribute.

The **instance** attribute must be either the name of an instance defined locally in the assembly, or the special name "extern", which references the boundaries of the assembly (see next section for details).

The **operation** attribute must be the name of an operation defined by the ComponentType associated to the instance referenced by attribute **instance**.

Other attributes can be associated to an operation reference, in order to define:

- if the operation is activating or not (**activating** or **callbackActivating**, default is true),
- the maximum number of waiting operation calls for this operation (**fifoSize**, default is 8) – waiting means the operation calls have not been taken out of the infrastructure.

Attributes 'activating' and 'fifoSize' are also applicable to read and written data. This is only meaningful when the correponding component's operation is notifying, and concerns the behaviour of notification events associated to the data.

A given operation of a component can be part of different OperationLinks, with the following exceptions:.

- A RequestSent operation shall be part of only one RequestLink,
- A DataRead operation shall be part of only one DataLink.
- A DataWritten operation with attribute 'writeOnly' set to false shall be part of only one DataLink

Each DataLink is associated to a Data that it represents and that is shared within the Application. Each DataLink has an attribute, **uncontrolledAccess**, which indicates if concurrent access to the Data is controlled by the ECOA Infrastructure or not (default is false, which corresponds to a controlled access).

When a DataLink is tagged as controlled, it may connect several writers, as well as several readers. The ECOA Infrastructure ensures that any writer or reader can access a local copy of the Data, this copy being the most recent value accessible on the platform among those produced by writers declared in the DataLink.

When a DataLink is tagged as not controlled, it may also connect several writers, as well as several readers. The local data repository is accessed directly by all readers and writers declared in the Datalink, thereby concurrent access between Component Instances is not managed by the Infrastructure, and must be managed at application level under the responsibility of the Application Supplier.

Operation Links can have associated QoS attributes, which may be used, for example, to described the dynamic behaviour of an information exchange (average rate, throughput, min/max repetition interval, etc.). These QoS attributes are expressed as metadata, and are not defined in the ECOA Architecture Specification.

6.1.4.3 Operations signature compatibilty

All the operations referenced in an Operation Link must share the same signature.

The signature of an operation includes:

- its nature (event, data, or request-response)
- the number of its parameters (always 1 for data operations, 0 or more for events and request-reponses)
- the structure of the types of all its parameters

Note that the following elements are not included in the signature of an operation:

- the name of the operation,
- the name of data types,
- the name of fields, in a record ou variant record data type,
- the definition of "simple" types, which are considered equal to the base type,
- the name of enum values, in an enum data type.

Thus it is possible to link together operations that have different names, and different data types.

Predefined data types are only compatible with themselves.

Examples of compatible types:

- record{a:int8, b:int16} is compatible with record{c:int8, d:int16}
- simple{int32, min=0, max=100} is compatible with int32
- enum{a=0, b=2} is compatible with enum{c=0, d=2}

Examples of incompatible types:

- record{a:int8, b:int16} is not compatible with record{a:int8, b:int16, c:int8 }
- enum{a=0, b=1} is not compatible with enum{a=0, b=2}
- int8 is not compatible with uint8
- int8 is not compatible with int16
- int8 is not compatible with boolean8

6.1.4.4 Conditional links

Conditional links are Operation Links containing one (or more) WhenCondition element.

A WhenCondition element references an instance, defined locally in the assembly, which must be a SUPERVISOR component.

It also contains the name of a variable, which must be defined by this SUPERVISOR component, and a value. The value of a variable is modifiable at run-time, at any moment, by the component that defines it.

The meaning of the WhenCondition element is that the operation is active only when the variable's value is strictly equal to the specified value. An inactive operation behaves as if it were not connected:

- in case of a event operation, it is not received or sent,
- in case of a data operation, it is not read (no value will be returned in case of a read access) or written (the write access will succeed but the value will not be modified),
- in case of a request-response operation, the request will fail from the client's perspective, the server will not receive any request nor be able to send any response.

6.1.4.5 Instance "extern" in Operation Links

The Operation links may define "promotion" links between internal and external operations. Promotion links are operation links that use the “extern” keyword to designate the component instance that owns the assembly. Note that an extern operation, seen from the "inside" of a composite component, has a direction opposite to the direction of the related operation defined at component type level, which defines an interface seen from the "outside" of the component.

The special name "extern" references the boundaries of the assembly. It must be used to connect an internal component of the assembly to the "outside" of the assembly.

Note: The name "extern" as an instance name is reserved, which means that no Component Instance defined in the assembly can have the name "extern".

References to operations of the special instance "extern" are called external operations.

The "outside" of the assembly may be the outside of the application (in case of the toplevel assembly of an application), or other components (in case of a composite instanciated in an application or in another composite).

Note that all interactions with the outside of the assembly must be defined explicitely, just like this is the case for internal interactions between components (e.g. to connect operation A of a component with external operation A, an Operation Link must be defined).

A single Operation Link may involve multiple internal operations and external operations.

However, the connection of an external operation with another external operation of the opposite direction ("pass-through") is not allowed.

External operations have their direction reversed, compared to their original definition in the Component Type model, because these operations are "seen from the inside". For example, when defining the internal assembly of a Composite that *sends* the event A (with a EventSent metamodel element), the event A of instance "extern" may be used as a *Receiver*, not a *Sender,* in a EventLink.

The following figure illustrates the internal assembly of a Composite containing two Components.

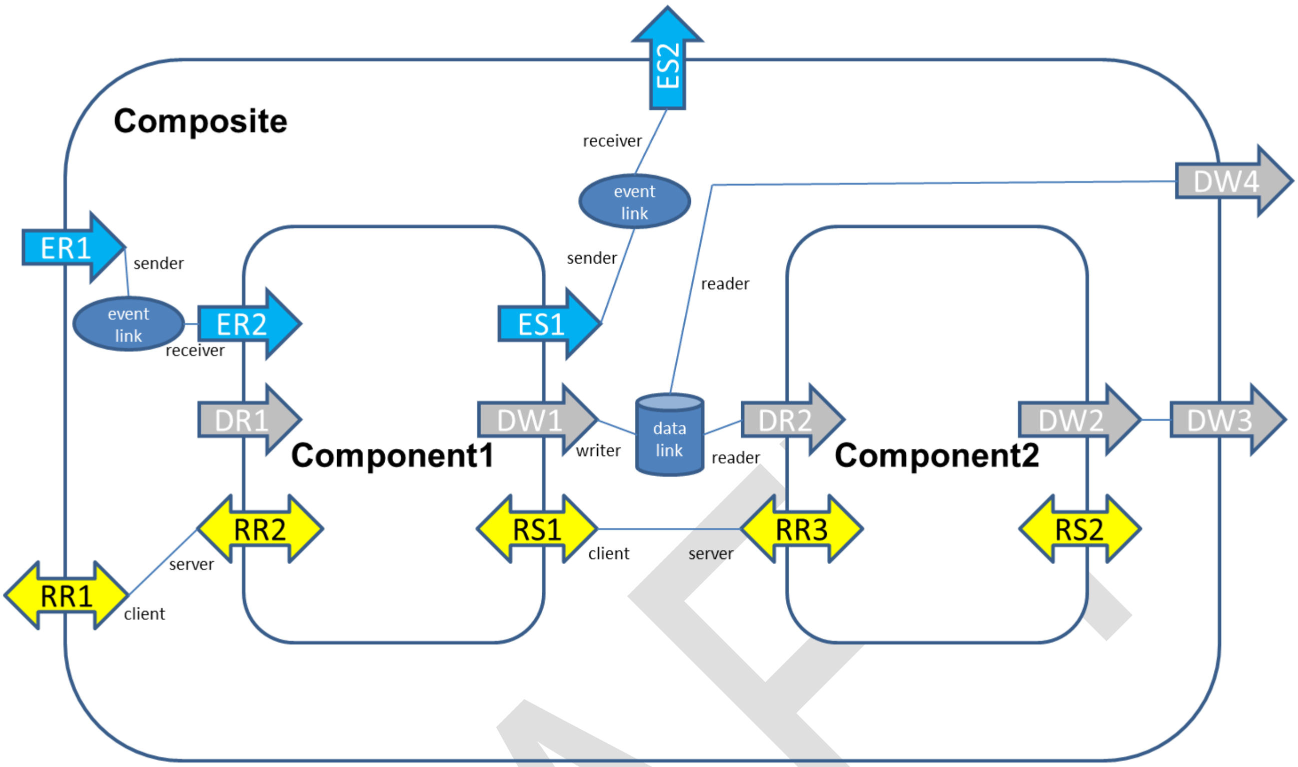


Figure 7 Internal assembly of a composite

In this example, ER1 is an EventReceived in the definition of Composite, but it is used as a sender in the internal assembly. It is connected internally to ER2, on Component1 (delegation link), which means that all events received by Composite.ER1 will be routed to Component2.ER2.

Similarly, a RequestLink between RR1 and RR2 delegates the handling of Composite.RR1 to Component1.RR2.

The RequestLink between RS1 and RR3 is a purely internal Operation Link.

The DataLink between DW1, DR2 and DW4 illustrates the case of an Operation Link that is both internal and external: the data values written by DW1 are read by DR2, but they are also made available to the outside of the composite through DW4.

6.1.4.6 Implicit Operation Links

6.1.4.6.1 Overview

In addition to normal operation links, some "implicit" operation links can be defined.

Implicit links are automatic generators of operation links. Each Implicit Link is equivalent to a set of EventLinks, DataLinks or RequestLinks.

The semantics of Implicit Operation Links is strictly limited to the MetaModel, so this mechanism is only described here and not in other documents.

6.1.4.6.2 Rationale

Implicit links are a way to simplify the assembly when connecting operations that have the same name, or related names.

Implicit links may be used to ease the mapping to ECOA of other architectural models, where operations are not considered individually, but grouped in higher-level interfacing concepts (called here "interfaces"). The following hypothesis are made:

- each interface corresponds to several ECOA operations
- the full name of an operation is obtained by concatenation: <interface_name><operation_name>

6.1.4.6.3 Description

Each element ImplicitLinks is defined by a set of references to operations (from different component instances). Each OperationSet element encompasses a set of operations that belong to the instance whose name matches the attribute "instance" (including the special instance "extern"). Moreover, the name of each operation must start with the value of the attribute "prefix" (default value of prefix is the empty string).

The attribute "instance" on elements OperationSet may contain (any number of times) the character "*", which is interpreted as a joker matching any sequence of characters. This allows to refer to several instances at once.

All these operations must have the same nature and have compatible parameter types.

The ImplicitLinks element connects all these operations together, provided that:

1. the name of operations, minus the prefix, is the same,
2. the operation does not appear in any normal OperationLink.

The rule that an operation involved in a normal Operation Link is never involved in an Implicit Link allows to define general rules (Implicit Links), as well as exceptions to these rules (normal Links).

Implicit Operation Links are cumulative : if an operation appears in several operation links, it may be connected multiple times.

Implicit Links must always be defined after all normal Operation Links.

The order in which Implicit Operation Links are defined is not significant.

An ImplicitLinks has an attribute 'activating', which will be used to create all derived operation links.

Other attributes (fifoSize, etc.) of the derived operation links will have all their attributes set to their default value. If some other value if desired, then normal Links shall be used instead of Implicit Links.

6.1.4.6.4 Algorithm

The algorithm by which Operation Links are derived from Implicit Operation Links is the following:

```
manual_ops := the list of all instanciated operations in normal Operation Links
(EventLink, DataLink, RequestLink elements)
For each element IL of type ImplicitLinks:
   auto_ops := Empty list
   For each element OS of type OperationSet in IL:
      For each instance of the assembly, including "extern", which name matches OS.instance:
         For each operation which name starts with OS.prefix:
            If this operation is not in manual_ops, add it to auto_ops
   If all operations in auto_ops are not compatible, raise an error
   If nature=event, create an EventLink with all operations of auto_ops
   If nature=data, create a DataLink with all operations of auto_ops
   If nature=request, for each client of auto_ops, create a RequestLink with this client and all the servers of auto_ops
```

Notes

- An instanciated operation is an operation on a given instance. It is defined by a couple (instance name, operation name).
- The EventLinks, DataLinks, RequestLinks are created without attributes (i.e. all attributes have their default value).
- The case of RequestLinks is special because of the constraint of having only one client per RequestLink.

6.1.5 Data Types

Data types are “portable types”, that have an equivalent in all programming languages supported by ECOA, and are used to describe information transmitted between components through operation links. By using these types, information can be serialized for transmission according to network protocols (defined through Extensions or Platform specific features). The way they are bound to a given programming language, is left to the platform provider based on language bindings (Architecture Specification Part 4).

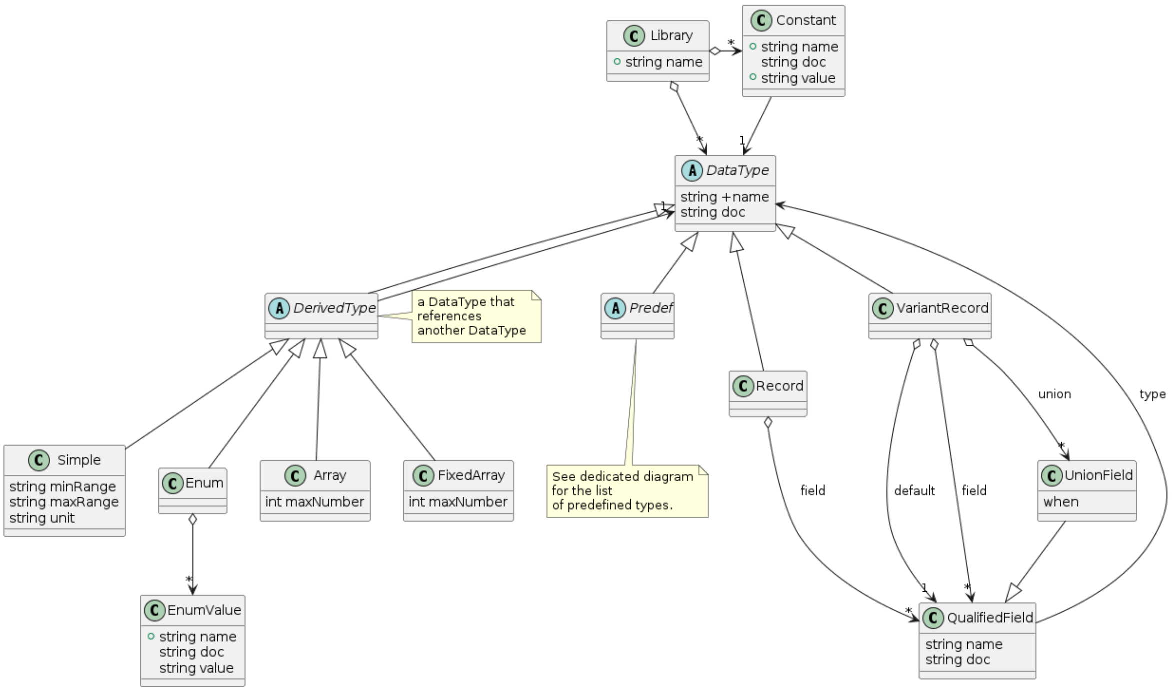


Figure 8 DataType definition

A **DataType** is a language-neutral type definition. It is used as a shared definition, and is referenced by VersionedData operations, and by Parameters of Events and RequestResponses operations. It is also used to type Properties. A comment can describe the DataType.

6.1.5.1 Libraries

A DataType definition is located in a **Library**. A Library is a set of DataTypes and Constants.

A Library may use types defined in another Library.

Cyclic dependencies between libraries are not allowed.

6.1.5.2 Constants

A **Constant** is a remarkable integer, floating-point or single-character value, identified with a name and located in a given Library. A Constant may be of any of the Simple or Basic DataTypes except boolean8 Basic type and Simple types based on boolean8.

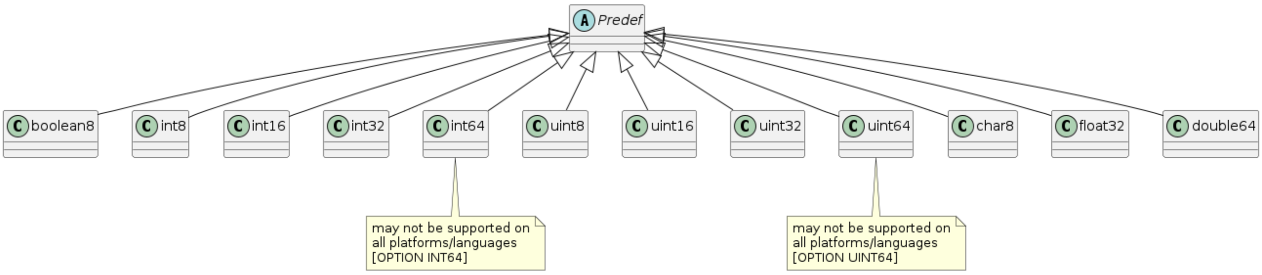

A Predef
C boolean8
C int8
C int16
C int32
C int64
C uint8
C uint16
C uint32
C uint64
C char8
C float32
C double64
may not be supported on all platforms/languages [OPTION INT64]
may not be supported on all platforms/languages [OPTION UINT64]

Figure 9 Predefined Data Types

6.1.5.3 Predefined types

A **Predefined** type belongs to a fixed list of predefined types: boolean8, char8, int8, int16, int32, int64, uint8, uint16, uint32, uint64, float32, double64. 8-bit characters are encoded in ASCII. For boolean8, the value 1 means TRUE while the value 0 means FALSE. float32 and double64 comply with theIEEE754 standard. The type uint64 may not be supported by all platforms/languages (e.g. Java has no corresponding efficient type).

6.1.5.4 User-defined types

A **Simple** type is defined to give a meaningful name to a basic or another simple type. It can define range limits, a precision and a unit. Each limit can be a literal numeric, a single-character or a reference to a symbolic constant. The unit is functional (not used in any way by the infrastructure) and expressed as a string (e.g. ‘second’). The precision is functional and expressed as a literal numeric. The precision is functionally consistent with the unit.

An **Enum** type shall be a basic integer type, or a simple type derived from a basic integer type and defines the list of authorized values, **EnumValue**, each of which has a symbolic name. Each value can be a literal numeric, or a reference to a symbolic constant.

An **Array** defines a variable-capacity array, whose maximum capacity is fixed. All elements are of the same type. The maximum capacity can be a literal numeric, or a reference to a symbolic constant.

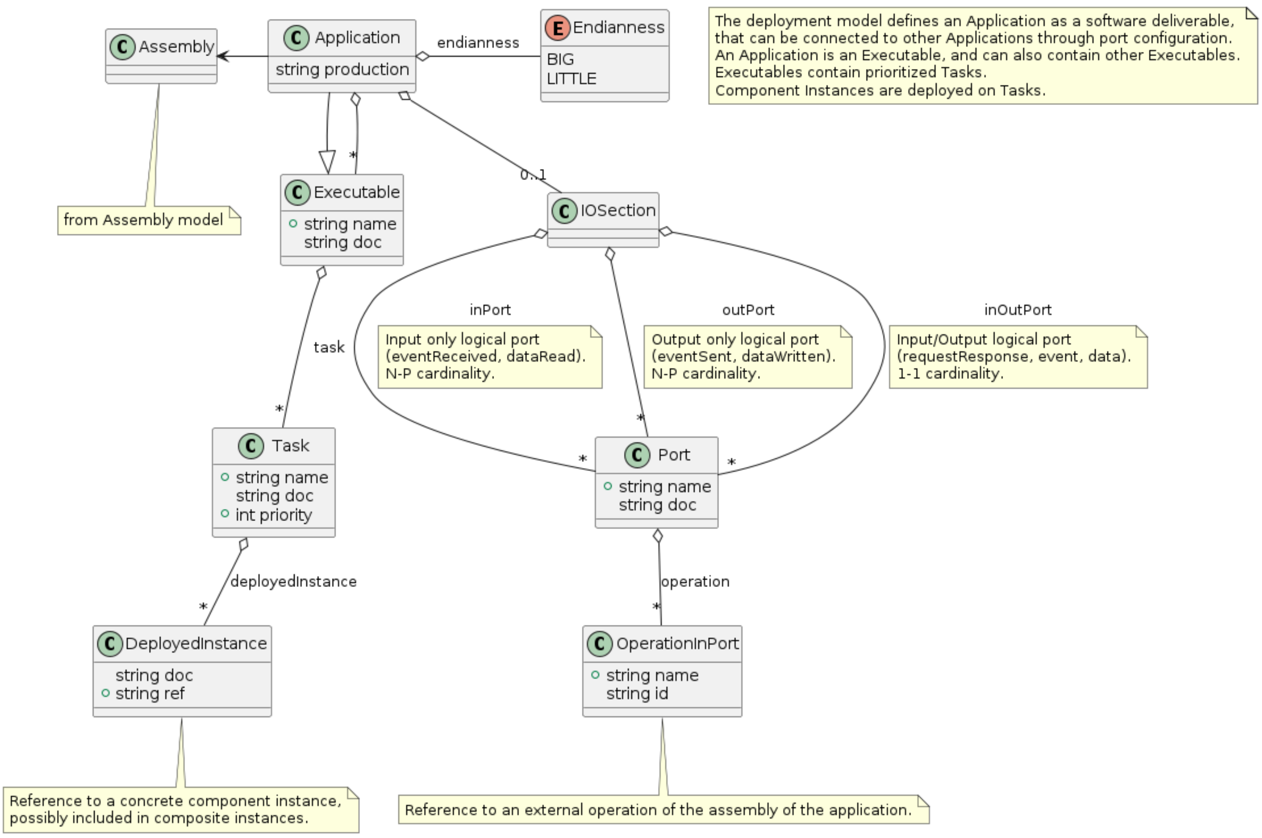
A **FixedArray** defines a fixed-capacity array. All elements are of the same type. The capacity can be a literal numeric, or a reference to a symbolic constant.

A **Record** is a structure with named **Fields**, of any type.

A **VariantRecord** is like a Record, with a special field called the selector (of boolean, integer or enum type). Some of the fields, **Union**, of a VariantRecord are optional: they are valued only when the selector has a certain value (given by the attribute “when”).

Nested types are not allowed; i.e. it is not possible to define local types specific to a given field. All types used at field level must be defined prior to the record definition.

6.1.6 Deployment



The deployment model defines an Application as a software deliverable, that can be connected to other Applications through port configuration. An Application is an Executable, and can also contain other Executables. Executables contain prioritized Tasks. Component Instances are deployed on Tasks.

from Assembly model

Input only logical port (eventReceived, dataRead). N-P cardinality.

Output only logical port (eventSent, dataWritten). N-P cardinality.

Input/Output logical port (requestResponse, event, data). 1-1 cardinality.

Reference to a concrete component instance, possibly included in composite instances.

Reference to an external operation of the assembly of the application.

Figure 10 Deployment metamodel

A **Deployment** model refers to an Assembly model: it describes how an **Application** will be deployed on a computing platform (logicial one or physical one).

An **Application** is an **Executable** that may have sub-Executables (Executables that will be launched as sons by the Application itself). A Executable offers spatial isolation (memory protection). It corresponds to the concept of process.

In each Executable (including the Application itself) the architect may define **Tasks** that will host component instances (**DeployedInstances**).

A Task has a unique **name** that may be used during debug activities. To facilitate debugging, the scope of unicity is the Application.

A Task defines its **relativePriority** relatively to other tasks of the application. The higher the value, the higher the priority.

The order of the list of DeployedInstances is significant: in case an event is received simultaneously by several components having the same priority, it determines the order in which these components will be activated.

Only concrete component instances may be deployed (not composite instances), and composite instances are referenced (**ref** attribute) through a dot notation : A1 component instance of the C2 composite instance will be references as C2.A1 in the deployment file.

Ports are groups of **Operations**, that will be connectable to another application. The entire list of **Operations** referenced by the **Port** will be connected when the **Port** itself will be connected. How exactly the connection is done is not specified by the ECOA Core Architecture Specification. The connection may be done through a modelisation of a multi-application assembly extension, or by configuration of the Application (at run time or design time). This configuration depends on the protocols used (e.g. IP addresses and port numbers for UDP and TCP, sockets names for Unix-type IPC, URLs for HTTP or gRPC, etc.)

There are 3 kinds of **Ports** that have different cardinalities and may reference different kinds of operations:

- **InPort** : May refer to **EventReceived** and **DataRead** operations and may be connected to 0 to n **OutPorts**
- **OutPort** : May refer to **EventSent** and **DataWritten** operations and may be connected to 0 to n **InPorts**
- **InOutPort** : May refer to every kind of **Operations** and may be connected to 0 to 1 **InOutPort**

Each **OperationInPort** element references an Operation of the Component that defines the interface of the Application.

It also defines an optional id that will be used to identify the operation sent/received on the given port.

The id is optional because:

- When a port contains only one operation, there is no need to identify the operation on the wire. Each message sent/received on this port will correspond to this operation.
- The infrastructure may allow to define this id elsewhere, by configuration of the Application, without the need for recompilation.
- In the metomodel the id is an integer, which is sufficient for the identification of an operation inside a port. However, some transport protocols identify the operations by litteral names (strings that appear in the message), or other non-integer identifiers. In this case, there is no requirement to allocate integer ids to operations; the operation names may be mapped directly to the identifier (by configuration table or by protocol-specific rules).
    - For example, if the gRPC protocol is used, ECOA operations can be mapped to gRPC's methods (or RPCs), which are defined in the context of "services" (which themselves are organized in packages), with a "path" that is transmitted in messages, in the form "/ {*package and service name*} / {method name}".

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6.1.7 Cross-cutting aspects

This section deals with cross-cutting aspects, that are applicable in several places of the metamodels.

6.1.7.1 Documentation

On all significant elements of the metamodel, a child element named 'doc' can be optionally added, in order to hold "comments" or "documentation", i.e. free text intended to describle the element.This text is not significant for tools, but can be used for example to generate documentation from models.

For the exact list of places where documentation can be added, please refer to the XSD schemas.

6.1.7.2 Metadata

On elements of the metamodel where documentation can be added, child elements named 'meta' can be optionally added, in order to hold "metadata", i.e. information intended to describe, in a more structured way than the free text, some properties about the model element.

Each 'meta' element has a name (a string) and a value (a string).

The list of metadata names is completely open. It is not defined in the ECOA Architecture Specification. No particular semantic is defined.

As it is not standardized, metadata shall not prescribe or modify in any significant way, anything in the behaviour of the software execution. Otherwise, the portability of components on different platforms would be compromised.

Metadata can be used to define, for example:

- Tracability information with other models, which can be higher-level ("upstream") or "same-level" models.
- Tracability with requirements
- "Quality of Service" (QoS) attributes
- etc.

Extensions of the ECOA Standard may define and describe QoS attributes (meaning of the metadata, definition of a fixed 'name' and constraints on the corresponding 'value').

6.1.7.3 Rules on logical names

Operation, operation parameter and component names must follow the naming conventions for identifiers used in the most common programming languages: a name being a sequence of letters, figures and underscores, beginning with a letter. Typically, “-“ must not be used in ECOA names.

The attributes "name" in all elements of the model shall be compliant with the following rules:

- A name is made of alphanumeric characters ([A-Za-z0-9]).
- A name shall not start with a digit.
- A name shall not contain consecutive underscore characters (__).
- A name shall be no longer than 64 characacers.
- A name shall avoid clashes with reserved words of the programming languages supported by ECOA.

Rationale : Some name may appear in the source code of component implementations. Some languages, like Ada, forbid consecutive underscore characters. Most languages forbid identifiers starting with a digit.

Some languages or compilers impose restrictions on the length of identifiers. Having bounded-size identifiers can facilitate the implementation of tools using the ECOA models.

6.2 Concrete Metamodel

6.2.1 Rules on XML writing

Certain rules need to be followed to ensure that the XML is consistent and correct. The following rules may be checked through normal validation requirements of the XML relative to its XSD schema or by specific tooling.

- Information names used within XML files are case sensitive. If the name of one item is used many times, character strings used for that name shall use the same case sensitivity.
- The parsing of XML files is done in one pass; i.e. items need to be defined before they are used. For example, the type for a field in a structure shall be defined before the definition of the structure.
- Each component type name must be unique within the assembly schema.
- Each component instance name must be unique within the assembly schema.
- Each operation name must be unique within each component definition.
- Each parameter name must be unique within each operation definition.
- Operation, operation parameter and component names must follow the rules and recommandations defined in §6.1.7.3.

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6.2.2 Schemas

Table 1 summarizes the ECOA schemas, which are presented in full in Section Schemas.

Table 1 ECOA Defined Schemas

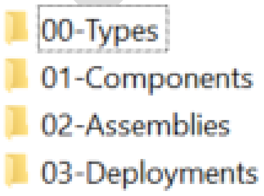
Filename	Description
DataTypes.xsd	Describes the syntax for defining ECOA types constructed from the ECOA basic types.
ComponentType.xsd	Describes the syntax for defining ECOA component type (contract).
Implementation.xsd	Describes the syntax for defining an implementation of an ECOA Component Type.
Assembly.xsd	Describes the syntax for defining an assembly of an ECOA Component Type instances and links between those instances.
Deployment.xsd	Describes the syntax for defining an deployment of ECOA Component instances declared in an Assembly on Executables and Tasks.
Workspace.xsd	Describes directories used for one given ECOA application

ECOA Schemas and XML files are fully compliant with the W3C XML Standards. They are validated with the following files:

- XMLSchema.xsd that describes the Schema for XML Schemas. Origin of the file used for the purpose of ECOA: http://www.w3c.org/2001/XMLSchema.xsd
- xml.xsd that describes the XML namespace, in a form suitable for import by other schema documents. Origin of the file used for the purpose of ECOA: http://www.w3c.org/2001/xml.xsd

6.2.3 Standard ECOA project folders tree and Filename Conventions

The standard folders organization for an ECOA project is the following one :



00-Types
01-Components
02-Assemblies
03-Deployments

Figure 11 - Standard ECOA project

00-Types directory contains all the ECOA types definitions

01-Components directory contains all the components types, and the different implementations for those component types (models which describe components interfaces), unless the workspace configuration file

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

specifies otherwise. The implementation directories also contain the source code and/or the precompiled object files for the component.

02-Assemblies directory contains one or more assembly files (models which instantiate components and define links between them)

03-Deployments directory contains one or more deployment files

04-Integration [Optional] directory contains all generated code, build files, as well as outputs of building process.

In order to ensure the unicity of names or namespace in an application, the default project organization is mainly based on folder names (the FileSystem will ensure the name unicity).

Depending on the platform, the PINFO mechanism may be based on files. These files may be located in the workspace; if so, their relative location in the project is free, and file paths must be relative paths, defined from the root of the project.

Note that an ECOA project directory may contain many applications (one application per deployment), as well as many components.

The following table describes a standard ECOA project tree in details.

An ECOA project tree may also contain additional directories not described in the ECOA Architecture Specification.

Table 2 ECOA Standard Project

Directory	Sub-directory 1	Sub-directory 2	Files	XSD
00-Types	N/A	N/A	<name_of_ library>.types.xml	DataTypes.xsd
01-Components	<name_of_component_type>	N/A	<name_of_component_type>.comp.xml	ComponentType.xsd
01-Components	<name_of_component_type>	<name_of_implementation>	<name_of_component_type>.<name_of_implementation >.impl.xml	Implementation.xsd
01-Components	<name_of_component_type>	<name_of_implementation>	<name_of_component_type>.<name_of_implementation >.assembly.xml	Assembly.xsd
02-Assemblies	N/A	N/A	<name >.assembly.xml	Assembly.xsd
03-Deployment	N/A	N/A	<name >.deployment.xml	Deployment.xsd
04-Integration	<name_of_deployment>	N/A	Generated files and folder specific to platform provider.	N/A
(root of workspace)	N/A	N/A	workspace.xml	Workspace.xsd

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

6.2.4 Workspace configuration file

The standard ECOA project folders tree can be modified by an optional file called "*workspace configuration file*".

The workspace configuration file allows to redefine the location of :

1) library files, i.e. the files that are, by default: 00-Types/<name_of_library>.types.xml,
2) component type directories, i.e. the directories that are, by default: 01-Components/<name_of_component_type>. When the location of a componenttype directory is redefined, all implementation directories (which are its sub-directories) are redefined accordingly.

The workspace configuration file is optional.

The workspace configuration file, if it exists, is always named “workspace.xml”, and is always in directory <project_root>.

It contains a list of mappings from a logical name to a file or directory.

For example:

```
<workspace>
  <library name=”LIB1” dir=”08-SpecialTypes/LIB1.types.xml” />
  <!-- look in another directory, relative to project root -->
  <componenttype name=”C1” dir=”09-OtherComponents/C” />
  <!-- component renaming: C2 is in fact in a directory named myC -->
  <componenttype name=”C2” dir=”01-Components/myC” />
  <!-- env variables  -->
  <componenttype name=”C3” dir=”${VAR}/C3” />
  <componenttype name=”C4” dir=”$(VAR)/C4” />
</workspace>
```

Path names may be relative or absolute.

When a path name is relative, it is always relative to the project root.

The syntax for paths may be Unix-like or windows-like (with “/” as separator, or “\”).

All paths are relative to 01-Components, except if they start with “/” or a drive name followed by “\” or by “\\” on windows (after variable substitution).

In case of name conflict between the default location and the workspace file (e.g. a component name is defined in the file and a directory with the same name exists in 01-Components), the name in the configuration file is used.

It is not allowed to have several definitions with the same name in the workspace configuration file.

6.2.5 XSD Files

6.2.5.1 File "inc/common.xsd"

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="quali
fied">
    <xsd:simpleType name="Name">
        <xsd:annotation>
            <xsd:documentation>
Name that can be used as an identifier in ECOA models
and in the source code of ECOA components.
Note: names must not start with an underscore nor a digit
names must not have two successive underscores
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9]*(_[A-Za-z0-9]+)*_?" />
            <xsd:maxLength value="64"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="TypeName">
        <xsd:annotation>
            <xsd:documentation>
Name of a data type inside a library, without any qualification.
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="Name" />
    </xsd:simpleType>
    <xsd:simpleType name="PossiblyQualifiedName">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9]*(_[A-Za-z0-9]+)*_?(\.[A-Za-
z][A-Za-z0-9]*(_[A-Za-z0-9]+)*_?)?" />
            <xsd:maxLength value="129"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="TypeQName">
        <xsd:annotation>
            <xsd:documentation>
Name of a data type, possibly prefixed by the name of the library that defines it.
If the referenced type is a predefined type or defined in the same library,
it can be referenced by "TYPE_NAME".
If the referenced type is defined in another library, it must be referenced by
"LIBRARY_NAME.TYPE_NAME".
</xsd:documentation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        </xsd:annotation>
        <xsd:restriction base="PossiblyQualifiedName" />
    </xsd:simpleType>
    <xsd:simpleType name="OperationName">
        <xsd:annotation>
            <xsd:documentation>
Name of an operation (data, event or service).
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="Name" />
    </xsd:simpleType>
    <xsd:element name="doc" type="xsd:string">
        <xsd:annotation>
            <xsd:documentation>
Dedicated element for documentation.
</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:simpleType name="positive-decimal">
        <xsd:annotation>
            <xsd:documentation>
The positive-decimal type specifies a positive
decimal value (including 0).
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:decimal">
            <xsd:minInclusive value="0" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="strictly-positive-decimal">
        <xsd:annotation>
            <xsd:documentation>
The strictly-positive-decimal type specifies a strictly positive
decimal value (excluding 0).
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:decimal">
            <xsd:minExclusive value="0" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="positiveInt">
        <xsd:restriction base="xsd:unsignedInt">
            <xsd:minInclusive value="1" />
        </xsd:restriction>
    </xsd:simpleType>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    <xsd:simpleType name="TriggerName">
        <xsd:annotation>
            <xsd:documentation>
Name of a trigger.
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="Name" />
    </xsd:simpleType>
    <xsd:complexType name="MetaData">
        <xsd:annotation>
            <xsd:documentation>
Element of metadata, used for tracability to higher-level models.
Not used by the execution platform.
</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="name" type="xsd:NCName" use="required">
        </xsd:attribute>
        <xsd:attribute name="value" type="xsd:string" use="required">
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="QualifiedField">
        <xsd:annotation>
            <xsd:documentation>
Association of an identifier and a typename.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
        <xsd:attribute name="type" type="TypeQName" use="required">
            <xsd:annotation>
                <xsd:documentation>
This attribute value references an already defined type name.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="name" type="Name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Type name.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    </xsd:complexType>
</xsd:schema>
```

6.2.5.2 File "DataTypes.xsd"

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ecoa.technology/DataTypes/3.0"
targetNamespace="http://www.ecoa.technology/DataTypes/3.0"
elementFormDefault="qualified" >
    <xsd:include schemaLocation="inc/common.xsd" />
    <xsd:group name="library_types">
        <xsd:annotation>
            <xsd:documentation>
List of all possible data types.
</xsd:documentation>
        </xsd:annotation>
        <xsd:choice>
            <xsd:element name="simple" type="CT_simple">
                <xsd:annotation>
                    <xsd:documentation>
A simple is an alias for another simple, an enum or a base type.
It allow defining a unit (for documentary purpose only), as well as min and max ra
nge.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="record" type="CT_record">
                <xsd:annotation>
                    <xsd:documentation>
A record follows the same definition as a structure in C language.
It is a sequence of fields, each one having a name and a type.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="field">
                    <xsd:annotation>
                        <xsd:documentation>
Field of a record, with an association of a name and a type.
</xsd:documentation>
                    </xsd:annotation>
                    <xsd:selector xpath="field" />
                    <xsd:field xpath="@name" />
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                </xsd:unique>
            </xsd:element>
            <xsd:element name="variantRecord" type="CT_variant_record">
                <xsd:annotation>
                    <xsd:documentation>
A variant record follows the same definition as ADA variant records, which is a sp
ecial type of discriminated
record where the presence of some elements depends on the value of the discriminan
t (called "the selector").
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="fieldunion">
                    <xsd:annotation>
                        <xsd:documentation>
Element of the variant record, that can be either a field or a union, with a name
and a type.
                        </xsd:documentation>
                    </xsd:annotation>
                    <xsd:selector xpath="field|union" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
                <xsd:unique name="when">
                    <xsd:annotation>
                        <xsd:documentation>
Selector in the variant record.
</xsd:documentation>
                    </xsd:annotation>
                    <xsd:selector xpath="union" />
                    <xsd:field xpath="@when" />
                </xsd:unique>
            </xsd:element>
            <xsd:element name="array" type="CT_array">
                <xsd:annotation>
                    <xsd:documentation>
Array definition allows specifying the maximum number of elements in the array, as
 well as the type of those
elements.
When an array is sent over the network, or written to memory, only the current num
ber of elements in the array
will be affected.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="fixedArray" type="CT_fixed_array">
                <xsd:annotation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                    <xsd:documentation>
Fixed array definition allows specifying the maximum number of elements in the arr
ay, as well as the type of
those elements.
In opposition to array type, when a fixed array is sent over the network, or writt
en to memory, all its
elements are sent even though some of them had never been assigned.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="enum" type="CT_enum">
                <xsd:annotation>
                    <xsd:documentation>
A ECOA enumerate follows the same definition as a C enumerate, except that it allo
ws defining the base type
of the values.
This is mandatory for ECOA to know the size of this enumerate, once written to mem
ory.
An enumerate can contain values of predefined integer types.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="value">
                    <xsd:annotation>
                        <xsd:documentation>
Field of the enumerate, that is defined with a name and optionally a valnum.
</xsd:documentation>
                    </xsd:annotation>
                    <xsd:selector xpath="value" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
                <xsd:unique name="valnum">
                    <xsd:annotation>
                        <xsd:documentation>
Numerical value of the enumerate field.
</xsd:documentation>
                    </xsd:annotation>
                    <xsd:selector xpath="value" />
                    <xsd:field xpath="@valnum" />
                </xsd:unique>
            </xsd:element>
            <xsd:element name="constant" type="CT_constant">
                <xsd:annotation>
                    <xsd:documentation>
A symbolic constant defining a numeric constant that can be used an an array size,
 or as a range limit in a simple type.
```

```
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:choice>
    </xsd:group>
    <xsd:complexType name="CT_library">
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:group ref="library_types" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="E_predef">
        <xsd:annotation>
            <xsd:documentation>
Predefined ECOA types.
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="boolean8" />
            <xsd:enumeration value="int8" />
            <xsd:enumeration value="int16" />
            <xsd:enumeration value="int32" />
            <xsd:enumeration value="int64">
                <xsd:annotation>
            <xsd:documentation>
This type may not be supported on all platforms/languages ([OPTION INT64]).
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="uint64">
                <xsd:annotation>
            <xsd:documentation>
This type may not be supported on all platforms/languages ([OPTION UINT64]).
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="uint8" />
            <xsd:enumeration value="uint16" />
            <xsd:enumeration value="uint32" />
            <xsd:enumeration value="char8" />
            <xsd:enumeration value="float32" />
            <xsd:enumeration value="double64" />
        </xsd:restriction>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    </xsd:simpleType>
    <xsd:complexType name="CT_type">
        <xsd:annotation>
            <xsd:documentation>
Base type of all ECOA data types.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="TypeName" use="required">
            <xsd:annotation>
                <xsd:documentation>
This attribute value is the key used to reference this type.
The value shall be unique in the library scope.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_derived_type">
        <xsd:annotation>
            <xsd:documentation>
Base type of all ECOA data types that rely on a previous type definition.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_type">
                <xsd:attribute name="type" type="TypeQName" use="required">
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_simple">
        <xsd:annotation>
            <xsd:documentation>
A type based on any other type (renaming case), with a name, min/max constraints,
and a unit.
Base type shall only be a predefined type or another simple type.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_derived_type">
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                <xsd:attribute name="minRange" type="ConstantReferenceOrValue" use
="optional">
                    <xsd:annotation>
                        <xsd:documentation>
Minimal range value for this type.
This value shall be equal or superior to the minimal range value for the base type
.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="maxRange" type="ConstantReferenceOrValue" use
="optional">
                    <xsd:annotation>
                        <xsd:documentation>
Maximal range value for this type.
This value shall be inferior or equal to the maximal range value for the base type
.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="unit" type="xsd:string" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
Use of International System units is recommended.
This attribute is only used as a documentation.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_enum">
        <xsd:annotation>
            <xsd:documentation>
Enumerated type.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_derived_type">
                <xsd:sequence>
                    <xsd:element name="value" type="CT_value" minOccurs="1" maxOcc
urs="unbounded">
                        <xsd:annotation>
                            <xsd:documentation>
Definition of a valid value for this enumerated type.
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_value">
        <xsd:annotation>
            <xsd:documentation>
Value and label of an enumerated type item.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="Name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Label of the item.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="valNum" type="ConstantReferenceOrIntegerValue">
            <xsd:annotation>
                <xsd:documentation>
Numerical value of the item.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_fixed_array">
        <xsd:annotation>
            <xsd:documentation>
Constant size array.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_derived_type">
                <xsd:attribute name="maxNumber" type="ConstantReferenceOrPositiveI
ntegerValue" use="required">
                    <xsd:annotation>
                        <xsd:documentation>
```

```
Maximum number of elements in the array.
When a fixed array is sent over the network, or written to memory, all its element
s are sent, even those which
had never been assigned.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_array">
        <xsd:annotation>
            <xsd:documentation>
Variable size array.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_derived_type">
                <xsd:attribute name="maxNumber" type="ConstantReferenceOrPositiveI
ntegerValue" use="required">
                    <xsd:annotation>
                        <xsd:documentation>
Maximum number of elements in the array.
When an array is sent over the network, or written to memory, only the current num
ber of elements in the array
are sent.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_union_field">
        <xsd:annotation>
            <xsd:documentation>
Optional field associated to a specific value of the variant in the definition of
a variantrecord type.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="QualifiedField">
                <xsd:attribute name="when" type="xsd:string" use="required">
                    <xsd:annotation>
                        <xsd:documentation>
Expression of the condition.
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
The value of this attribute can be an integer value or an already defined enumerat
ed type item name if the
defined selectType is a enumerated type.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_record">
        <xsd:annotation>
            <xsd:documentation>
Sequence of fields, each one having a name and a type.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_type">
                <xsd:sequence>
                    <xsd:element name="field" type="QualifiedField" minOccurs="0"
maxOccurs="unbounded" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_variant_record">
        <xsd:annotation>
            <xsd:documentation>
Special type of discriminated record: sequence of conditional fields.
The condition is based on a specific attribute called selector.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_type">
                <xsd:sequence>
                    <xsd:element name="field" type="QualifiedField" minOccurs="0"
maxOccurs="unbounded" />
                    <xsd:element name="union" type="CT_union_field" minOccurs="1"
maxOccurs="unbounded" />
                    <xsd:element name="default" type="QualifiedField" minOccurs="0
" maxOccurs="1" />
                </xsd:sequence>
                <xsd:attribute name="selectName" type="Name" use="required">
                    <xsd:annotation>
                        <xsd:documentation>
Name of the selector.
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="selectType" type="TypeQName" use="required">
                    <xsd:annotation>
                        <xsd:documentation>
Type of the selector.
This attribute value references an already defined type name.
The referenced type shall be an enumerated one or an integer one.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_constant">
        <xsd:annotation>
            <xsd:documentation>Constant definition</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_type">
                <xsd:attribute name="type" type="TypeQName" use="required" />
                <xsd:attribute name="value" type="ConstantReferenceOrValue" use="r
equired" />
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:simpleType name="ConstantReferenceOrValue">
        <xsd:annotation>
            <xsd:documentation>
Use of a constant reference or of any constant value.
Note: the ConstantValue type is replaced by its flattened union members.
</xsd:documentation>
        </xsd:annotation>
        <xsd:union
            memberTypes="ConstantReference xsd:double xsd:integer CharConstantType
 HexaCharConstantType" />
    </xsd:simpleType>
    <xsd:simpleType name="ConstantReference">
        <xsd:annotation>
            <xsd:documentation>
Constant reference
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
```

```
            <xsd:pattern value="%([A-Za-z][A-Za-z0-9_]*\.)?[A-Za-z][A-Za-z0-
9_]*%" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="ConstantReferenceOrPositiveIntegerValue">
        <xsd:annotation>
            <xsd:documentation>
Use of a constant or of a positive integer value.
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern
                value="%([A-Za-z][A-Za-z0-9_]*\.)?[A-Za-z][A-Za-z0-9_]*%|[0-
9]+" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="ConstantReferenceOrIntegerValue">
        <xsd:annotation>
            <xsd:documentation>
Use of a constant or of an integer value.
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern
                value="%([A-Za-z][A-Za-z0-9_]*\.)?[A-Za-z][A-Za-z0-9_]*%|(\+|-
)?[0-9]*" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="CharConstantType">
        <xsd:annotation>
            <xsd:documentation>Character Constant Type</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="'\p{IsBasicLatin}'" />
            <xsd:length value="3" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="HexaCharConstantType">
        <xsd:annotation>
            <xsd:documentation>
This type shall be used as the base type for
any XML schema attribute or element that contains a hex-encoded
char value.
This hex-encoded binary value contains the mandatory
string 0x followed by a sequence of 1 or 2 characters 0-9 and
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
a-f.
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="0x[0-9A-Fa-f][0-9A-Fa-f]|0x[0-9A-Fa-f]" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:element name="library" type="CT_library">
        <xsd:annotation>
            <xsd:documentation>
A library allows to define types in a unique implementation space, shared between
components.
</xsd:documentation>
        </xsd:annotation>
        <xsd:key name="librarytypekey">
            <xsd:selector xpath="*" />
            <xsd:field xpath="@name" />
        </xsd:key>
    </xsd:element>
</xsd:schema>
```

6.2.5.3 File "ComponentType.xsd"

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ecoa.technology/ComponentType/3.0"
targetNamespace="http://www.ecoa.technology/ComponentType/3.0"
elementFormDefault="qualified" >
    <xsd:include schemaLocation="inc/common.xsd" />
    <xsd:simpleType name="E_component_kind">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STANDARD">
                <xsd:annotation>
                    <xsd:documentation>
Defines a standard component, single threaded, without special capabilities.
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="PERIODIC_TRIGGER_MANAGER">
                <xsd:annotation>
                    <xsd:documentation>
Defines a PERIODIC_TRIGGER_MANAGER component, fully generated component that allow
s to send periodic events.
</xsd:documentation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="DYNAMIC_TRIGGER_MANAGER">
                <xsd:annotation>
                    <xsd:documentation>
Defines a DYNAMIC_TRIGGER_MANAGER component, fully generated component that allows
 to send events with a delay.
This may not be supported on all platforms [OPTION DYNAMIC TRIGGER].
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="EXTERNAL">
                <xsd:annotation>
                    <xsd:documentation>
Defines an EXTERNAL component, used to manage I/O.
External components are deployed on two separate threads.
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="SUPERVISOR">
                <xsd:annotation>
                    <xsd:documentation>
Defines a SUPERVISOR component dedicated to the supervision of the whole applicati
on.
This may not be supported on all platforms [OPTION SUPERVISION].
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="CT_operation" abstract="true">
        <xsd:annotation>
            <xsd:documentation>
An operation is the greatest common denominator between events, request_responses
and data.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="OperationName" use="required">
            <xsd:annotation>
                <xsd:documentation>
```

```
Name of the operation.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_event" abstract="true">
        <xsd:annotation>
            <xsd:documentation>
Specificities of an EVENT operation.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_operation">
                <xsd:sequence>
                    <xsd:element name="parameter" type="QualifiedField" minOccurs=
"0" maxOccurs="unbounded">
                        <xsd:annotation>
                            <xsd:documentation>
Parameter of the operation.
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_sent_event">
        <xsd:annotation>
            <xsd:documentation>
Definition of a SENT event.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_event">
                <xsd:attribute name="period" type="strictly-positive-
decimal" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
Period of the event (in milliseconds).
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="delay" type="positive-decimal" default="0.0">
                    <xsd:annotation>
                        <xsd:documentation>
```

```
Initial delay (in milliseconds). must be lower than or equal to the period, no mat
ter how the period
is specified.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_received_event">
        <xsd:annotation>
            <xsd:documentation>
Definition of a RECEIVED event.
NOTA: there is no additional information, considering a CT_event.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_event" />
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_data" abstract="true">
        <xsd:annotation>
            <xsd:documentation>
Definition of a DATA operation.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_operation">
                <xsd:attribute name="type" type="TypeQName" use="required">
                    <xsd:annotation>
                        <xsd:documentation>
Type memorized in the data.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="maxVersions" type="positiveInt" default="1">
                    <xsd:annotation>
                        <xsd:documentation>
Maximum number of simultaneously accessed data versions.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="notifying" type="xsd:boolean" default="false"
>
                    <xsd:annotation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                        <xsd:documentation>
True if the component wants to be notified through a dedicated event every time th
e corresponding data is
modified.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_written_data">
        <xsd:annotation>
            <xsd:documentation>
Specificities of WRITTEN data.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_data">
                <xsd:attribute name="writeOnly" type="xsd:boolean" default="false"
>
                    <xsd:annotation>
                        <xsd:documentation>
True iff data has to be accessed with write access only (thus, no read one).
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_read_data">
        <xsd:annotation>
            <xsd:documentation>
Specificities of READ data.
NOTA: there is no additional information, considering a CT_data.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_data" />
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_properties">
        <xsd:annotation>
            <xsd:documentation>
Definition of the properties of a component type.
</xsd:documentation>
```

```
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="property" type="QualifiedField" minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Definition of an property of the component type.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CT_pinfo">
        <xsd:annotation>
            <xsd:documentation>
Definition of a string identifier
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unboded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="Name">
            <xsd:annotation>
                <xsd:documentation>
Value of the string identifier
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="writable" type="xsd:boolean" default="false">
            <xsd:annotation>
                <xsd:documentation>
If the true, the contents of the pinfo can be modified by the component with an API.
By default, a pinfo is read only.
This may not be supported on all platforms [OPTION PINFO WRITE].
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_pinfos">
        <xsd:annotation>
            <xsd:documentation>
Definition of the post-production properties of a component type.
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="pinfo" type="CT_pinfo" minOccurs="0" maxOccurs="unb
ounded">
                <xsd:annotation>
                    <xsd:documentation>
Definition of a post-production property of the component type.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:group name="component_type_declarations">
        <xsd:sequence>
            <xsd:element name="properties" type="CT_properties" minOccurs="0" maxO
ccurs="1">
                <xsd:annotation>
                    <xsd:documentation>
component parameterization with properties whose values are fixed at the instantia
tion.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="propertyname">
                    <xsd:selector xpath="property" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
            </xsd:element>
            <xsd:element name="pinfos" type="CT_pinfos" minOccurs="0" maxOccurs="1
">
                <xsd:annotation>
                    <xsd:documentation>
component post-
production parameterization with properties whose values are fixed at the instanti
ation.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="pinfo_name">
                    <xsd:selector xpath="pinfo" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
            </xsd:element>
            <xsd:element name="variables" type="CT_variables" minOccurs="0" maxOcc
urs="1">
                <xsd:annotation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                    <xsd:documentation>
Definition of supervision variables and their value domain.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="variablename">
                    <xsd:selector xpath="variable" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
            </xsd:element>
        </xsd:sequence>
    </xsd:group>
    <xsd:complexType name="CT_component_type">
        <xsd:annotation>
            <xsd:documentation>Definition of a component type.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:group ref="component_type_declarations" />
            <xsd:element name="operations" type="CT_operations" minOccurs="1" maxO
ccurs="1">
                <xsd:annotation>
                    <xsd:documentation>Operations that characterize the component
contract.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="triggers" type="CT_triggers" minOccurs="0" maxOccur
s="1">
                <xsd:annotation>
                    <xsd:documentation>Dedicated section for triggers declaration.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="kind" type="E_component_kind" default="STANDARD">
            <xsd:annotation>
                <xsd:documentation>By default a ECOA component is a STANDARD one.<
/xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="componentType" type="CT_component_type">
        <xsd:annotation>
            <xsd:documentation>
```

```
Definition of a ECOA component type.
</xsd:documentation>
        </xsd:annotation>
        <xsd:key name="operationkey">
            <xsd:selector xpath="operations/*" />
            <xsd:field xpath="@name" />
        </xsd:key>
    </xsd:element>
    <xsd:group name="all_operations">
        <xsd:choice>
            <xsd:element name="dataRead" type="CT_read_data" minOccurs="0">
                <xsd:annotation>
                    <xsd:documentation>
Read access to a DATA.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="dataWritten" type="CT_written_data" minOccurs="0">
                <xsd:annotation>
                    <xsd:documentation>
Write (+read) access to a DATA.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="eventReceived" type="CT_received_event" minOccurs="
0">
                <xsd:annotation>
                    <xsd:documentation>
Arrival of an event.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="param_event_received">
                    <xsd:selector xpath="parameter" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
            </xsd:element>
            <xsd:element name="eventSent" type="CT_sent_event" minOccurs="0">
                <xsd:annotation>
                    <xsd:documentation>
Emission (production) of an EVENT.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="param_event_sent">
                    <xsd:selector xpath="parameter" />
                    <xsd:field xpath="@name" />
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                </xsd:unique>
            </xsd:element>
            <xsd:element name="requestSent" type="CT_request_sent" minOccurs="0">
                <xsd:annotation>
                    <xsd:documentation>
Call of a request_response.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="param_request_sent">
                    <xsd:selector xpath="parameter|out" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
            </xsd:element>
            <xsd:element name="requestReceived" type="CT_request_received" minOccu
rs="0">
                <xsd:annotation>
                    <xsd:documentation>
Supplying of a request_response.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="param_request_received">
                    <xsd:selector xpath="parameter|out" />
                    <xsd:field xpath="@name" />
                </xsd:unique>
            </xsd:element>
        </xsd:choice>
    </xsd:group>
    <xsd:complexType name="CT_operations">
        <xsd:group ref="all_operations" maxOccurs="unbounded" />
    </xsd:complexType>
    <xsd:complexType name="CT_trigger">
        <xsd:annotation>
            <xsd:documentation>
A trigger is an artefact that can be declared by a component, with
an associated event for trigger back.
Such a declaration enables generation of methods to set and cancel
alarms, i.e., give a triggerDelay and the associated declared event will
be sent upon delay expiration.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        <xsd:attribute name="name" type="TriggerName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the trigger.
    </xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="event" type="OperationName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the event that will be received upon trigger expiration.
Such an event shall exist, shall be declared as received
and shall have no parameter.
    </xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_triggers">
        <xsd:annotation>
            <xsd:documentation>
Definition of triggers.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="trigger" type="CT_trigger" minOccurs="0" maxOccurs=
"unbounded" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CT_request_response" abstract="true">
        <xsd:annotation>
            <xsd:documentation>
Specificities of a request_response operation.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_operation">
                <xsd:sequence>
                    <xsd:element name="parameter" type="QualifiedField" minOccurs=
"0" maxOccurs="unbounded">
                        <xsd:annotation>
                            <xsd:documentation>
request_response input parameter.
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="out" type="QualifiedField" minOccurs="0" ma
xOccurs="unbounded">
                        <xsd:annotation>
                            <xsd:documentation>
request_response output parameter.
    </xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_request_received">
        <xsd:annotation>
            <xsd:documentation>
Specificities of a PROVIDED request_response.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_request_response">
                <xsd:attribute name="immediate" type="xsd:boolean" default="false"
>
                    <xsd:annotation>
                        <xsd:documentation>
If true, the response is immediate, i.e. sent when the execution of the entry poin
t of the receiver component ends.
If false, the response can be postponed; the component may send it at any time, fr
om any entry point.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="maxConcurrentRequests" type="xsd:unsignedInt"
 default="8">
                    <xsd:annotation>
                        <xsd:documentation>
Maximum number of requests that the server can handle at any time. Meaningful only
 when immediate=false.
If immediate=true, only one request can be handled at the time.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_request_sent">
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        <xsd:annotation>
            <xsd:documentation>
Specificities of a REQUIRED request_response.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="CT_request_response">
                <xsd:attribute name="isSynchronous" type="xsd:boolean" default="fa
lse">
                    <xsd:annotation>
                        <xsd:documentation>
A request is synchronous or asynchronous from the viewpoint of the
component that calls it (i.e., the client).
    </xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="timeout" type="xsd:decimal" default="0">
                    <xsd:annotation>
                        <xsd:documentation>
Maximum authorized time (in milliseconds) for the call of a synchronous
or asynchronous request_response. Zero or a negative value means infinite timeout.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="maxConcurrentRequests" type="positiveInt" def
ault="8">
                    <xsd:annotation>
                        <xsd:documentation>
Maximum number of requests that can be pending (i.e. waiting for response or timeo
ut) at any given moment.
Meaningful only for asynchronous requests.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_variables">
        <xsd:annotation>
            <xsd:documentation>
Definition of SYSTEM variables associated to a component of
SUPERVISOR type.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
            <xsd:element name="variable" type="QualifiedField" minOccurs="0" maxOc
curs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Definition of supervision variable.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

6.2.5.4 File "Implementation.xsd"

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ecoa.technology/Implementation/3.0"
targetNamespace="http://www.ecoa.technology/Implementation/3.0"
elementFormDefault="qualified" >
    <xsd:include schemaLocation="inc/common.xsd" />
    <xsd:element name="implementation" type="CT_implementation">
        <xsd:annotation>
            <xsd:documentation>
Root node of implementation model.
This model contains the implementation specific information.
</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <!-- C programming language -->
    <xsd:simpleType name="CFullName">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9_]*">
                <xsd:annotation>
                    <xsd:documentation>
Unique prefix for global identifiers, preventing name conflicts in C language and
in object files.
This string will be used as a prefix for every method and attribute name of the co
rresponding component.
</xsd:documentation>
                </xsd:annotation>
            </xsd:pattern>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="CT_language">
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element name="binaryDescription" type="CT_bindesc" minOccurs="0"
maxOccurs="1">
                <xsd:annotation>
                    <xsd:documentation>
If this element is present, then the component is assumed to be provided in compil
ed form and not in source form.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="incDir" type="CT_extra" minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Optional directory to be added to compilation includes for this
component.
$(PROJECT_ROOT) may be used to refer to the root of the ECOA workspace.
$(X) will be expanded to the value of the make variable X, or the
environment variable X.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="srcDir" type="CT_extra" minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Optional directory to be added to compilation source path for this
component.
$(PROJECT_ROOT) may be used to refer to the root of the ECOA workspace.
$(X) will be expanded to the value of the make variable X, or the
environment variable X.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="compilationFlags" type="CT_extra"
                minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Optional compilation flags specific to this component.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
            <xsd:element name="linkFlags" type="CT_extra"
                minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Optional compilation link flags specific to this component.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="additionalJar" type="CT_extra"
                minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Optional JAR files needed for this implementation.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="APIType" type="xsd:string" use="optional">
            <xsd:annotation>
                <xsd:documentation>
An optional string identifying the type of the language binding used by
this implementation.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="APIVersion" type="xsd:string" use="optional">
            <xsd:annotation>
                <xsd:documentation>
An optional string identifying the version of the language binding used by
this implementation.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="stack" type="xsd:unsignedInt"
            use="optional" default="32768">
            <xsd:annotation>
                <xsd:documentation>
Maximum stack size, in bytes, used by the implementation, considering
all entry points of the component contract.
This value will be used to calculate the thread stack
needs.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="externalStack"
```

```
            type="xsd:unsignedInt" use="optional" default="32768">
            <xsd:annotation>
                <xsd:documentation>
Maximum stack size, in bytes, used by the implementation of the
external thread of an 'EXTERNAL' component.
This value will be used to calculate the thread stack
needs.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_languagec">
        <xsd:complexContent>
            <xsd:extension base="CT_language">
                <xsd:attribute name="fullName" type="CFullName"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>
Name used as a prefix for identifiers (types, functions) in the C code.
All C implementations used in the same ECOA system must have
different fullnames.
Recommendation: if component type name is X.Y.Z, this should be X_Y_Z.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="filePrefix" type="xsd:string" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
The name to use when building file names for this component implementation.
If not defined, use the value of 'fullName'.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- -->
    <xsd:complexType name="CT_implementation">
        <xsd:annotation>
            <xsd:documentation>
Defines everything about a given component implementation, that is needed for the
integration of this implementation in an application (i.e., programming language,
etc.).
</xsd:documentation>
        </xsd:annotation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an 'as is' basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    <xsd:sequence>
      <xsd:element ref="doc" minOccurs="0" />
      <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
      <xsd:choice>
        <xsd:annotation>
          <xsd:documentation>
  Defines the programming language and all language-specific details,
  required for the use of this implementation in a system.
  </xsd:documentation>
        </xsd:annotation>
        <xsd:element name="language.c" type="CT_languagec" />
        <xsd:element name="language.cpp" type="CT_languagecpp" />
        <xsd:element name="language.ada" type="CT_languageada" />
        <xsd:element name="language.rust" type="CT_languagerust" />
        <xsd:element name="language.java" type="CT_languagejava" />
        <xsd:element name="language.python" type="CT_languagepython" />
        <xsd:element name="composite" type="CT_composite" />
      </xsd:choice>
      <xsd:element name="option" type="CT_option" minOccurs="0" maxOccurs="u
nbounded">
        <xsd:annotation>
          <xsd:documentation>
Feature of the software execution platform, needed by this component implementatio
n.
An option that does not appear in the list is not needed.
An option that appears in the list without an explicit value is needed.
An option that appears in the list with an explicit value is needed only if value=
true.
</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_composite" />
  <!-- C++ programming language -->
  <xsd:simpleType name="CppNamespace">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[A-Za-z:][A-Za-z0-9_:]*">
      </xsd:pattern>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="CT_languagecpp">
    <xsd:complexContent>
      <xsd:extension base="CT_language">
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                <xsd:attribute name="namespace" type="CppNamespace"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>
The C++ namespace containing the component code (e.g., company::project::component
).
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="filePrefix" type="xsd:string" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
The name to use when building file names for this component implementation.
If not defined, use the value of 'namespace' with '::' replaced by '_'.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- Java programming language -->
    <xsd:simpleType name="JavaPackageName">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9_\.]*">
            </xsd:pattern>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="CT_languagejava">
        <xsd:complexContent>
            <xsd:extension base="CT_language">
                <xsd:attribute name="packageName" type="JavaPackageName"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>
The name of the Java package (e.g., company.project.component) containing the code
 of the
component.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- Ada programming language -->
    <xsd:simpleType name="AdaPackageName">
```

```
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9_\.]*">
            </xsd:pattern>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="CT_languageada">
        <xsd:complexContent>
            <xsd:extension base="CT_language">
                <xsd:attribute name="packageName" type="AdaPackageName"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>
The name of the Ada package containing the code of the
component (ex: project.component).
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- Rust programming language -->
    <xsd:simpleType name="RustPackageName">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z0-9_-]+">
                <xsd:annotation>
                    <xsd:documentation>
Reference: https://doc.rust-lang.org/cargo/reference/manifest.html#the-name-field
</xsd:documentation>
                </xsd:annotation>
        </xsd:pattern>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="CT_languagerust">
        <xsd:complexContent>
            <xsd:extension base="CT_language">
                <xsd:attribute name="packageName" type="RustPackageName"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>
The name of the Rust package containing the code of the component.
This package must contain a root crate of type 'library', implementing the compone
nt's entry points.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- Python programming language -->
    <xsd:simpleType name="PythonPackageName">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9_\.]*">
            </xsd:pattern>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="CT_languagepython">
        <xsd:complexContent>
            <xsd:extension base="CT_language">
                <xsd:attribute name="packageName" type="PythonPackageName"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>
The name of the Python package containing the code of the component.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CT_extra">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="production" type="xsd:string">
                    <xsd:annotation>
                        <xsd:documentation>
Production for which this element is applicable to.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="CT_bindesc">
        <xsd:sequence>
            <xsd:element name="objectFile" type="CT_extra" maxOccurs="unbounded">
                <xsd:annotation>
                <xsd:documentation>
Filename of a binary file implementing the component. The filename may contain pat
h information
relative to the location of the implementation, e.g. "binaries/module.o"
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
relates a file 'module.o' located in a subdirectory 'binaries' of the directory containing this file.
Separators are '/'.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="userContextSize" type="xsd:unsignedInt">
            <xsd:annotation>
                <xsd:documentation>
Size in bytes of the component's user context.</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="warmStartContextSize" type="xsd:unsignedInt" default="0">
            <xsd:annotation>
                <xsd:documentation>
Size in bytes of the component's warm start context.</xsd:documentation>
                </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CT_option">
        <xsd:attribute name="name" type="Name"/>
        <xsd:attribute name="value" type="xsd:boolean" default="true"/>
    </xsd:complexType>
</xsd:schema>
```

6.2.5.5 File "Assembly.xsd"

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ecoa.technology/Assembly/3.0"
targetNamespace="http://www.ecoa.technology/Assembly/3.0"
elementFormDefault="qualified" >
    <xsd:include schemaLocation="inc/common.xsd" />
    <xsd:element name="assembly" type="AS_assembly">
        <xsd:annotation>
            <xsd:documentation>
Assembly Root node. The assembly defines instances of components and operation links between instances.
</xsd:documentation>
        </xsd:annotation>
        <xsd:key name="instancekey">
            <xsd:selector xpath="instance" />
            <xsd:field xpath="@name" />
        </xsd:key>
    </xsd:element>
    <xsd:complexType name="AS_assembly">
        <xsd:annotation>
            <xsd:documentation>
Definition of an assembly of instantiated components.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element name="instance" type="AS_instance" minOccurs="0" maxOccur
s="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Defines an instance of a component.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="links" type="AS_links">
                <xsd:annotation>
                    <xsd:documentation>
Defines links between operations of instantiated components.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="componentType" type="xsd:NCName" use="optional">
            <xsd:annotation>
                <xsd:documentation>
ComponentType model describing the logical interface of the whole system / composi
te of components.
Only needed if "extern" instance is used in the toplevel assembly.
This attribute value must reference an existing componentType model.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="AS_instance">
        <xsd:annotation>
            <xsd:documentation>
Definition of an instance of component.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element name="propertyValue" type="AS_member_value" minOccurs="0"
 maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Initialization of properties defined in component model.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="pinfoValue" type="AS_member_value" minOccurs="0" ma
xOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Initialization of post-
compilation attributes defined in component model via Pinfo mechanism.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="variableInit" type="AS_identified_member_value"
                minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Numerical variable (defined by a SUPERVISOR component) used as activation conditio
n for certain connections (cf. 'when' elements).
The values of variables can be modified during the execution by supervisor compone
nts.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="variableAlias" type="AS_variable_alias"
                minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Makes accessible a numerical variable defined internally, outside of the component
.
This element can only be declared by SUPERVISOR components.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:NCName" use="required">
            <xsd:annotation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                <xsd:documentation>
Name of the instance. This attribute value must be unique in the assembly.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="componentType" type="xsd:NCName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Reference to the associated component model.
This attribute value must reference an already defined componentType.
The name of a componentType is defined by the name a the component.xml parent directory (Cf. 01-Components).
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="implementation" type="xsd:NCName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the component implementation used for this instance.
This attribute must reference an already defined component implementation.
The component implementation name is the implementation.xml parent directory name.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="AS_member_value">
        <xsd:annotation>
            <xsd:documentation>
Association between the name of a member (property, pinfo) and a value.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:NCName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the member.
This must match the name of a property or pinfo defined in the associated component type.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        <xsd:attribute name="value" type="xsd:string" use="required">
            <xsd:annotation>
                <xsd:documentation>
Value of the member for this specific component instance.
If the value starts with special character '$', it refers to a property or pinfo d
efined by the enclosing composite.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="AS_operation_link">
        <xsd:annotation>
            <xsd:documentation>
Base type for datalink, eventlink, requestlink.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
        <xsd:attribute name="id" type="positiveInt" use="optional">
            <xsd:annotation>
                <xsd:documentation>
Unique Id of the operation link.
This Id may be fixed by user in order to guarantee code generation stability.
If not fixed by user, a unique Id will be generated.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="AS_default_data_value">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" type="xsd:string" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
Type is optional when two different contexts of use allow correct interpretation o
f the value.
If not, then the type should be made explicit.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:simpleContent>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    </xsd:complexType>
    <xsd:complexType name="AS_data_link">
        <xsd:annotation>
            <xsd:documentation>
Link between DATA operations.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_operation_link">
                <xsd:sequence>
                    <xsd:element name="writer" type="AS_op_ref_write" minOccurs="0
" maxOccurs="unbounded" />
                    <xsd:element name="reader" type="AS_op_ref_read" minOccurs="0"
 maxOccurs="unbounded" />
                    <xsd:element name="defaultValue" type="AS_default_data_value"
minOccurs="0" maxOccurs="1">
                        <xsd:annotation>
                            <xsd:documentation>
The value that consumers will read while no production has occurred.
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="uncontrolledAccess" type="xsd:boolean" defaul
t="false">
                    <xsd:annotation>
                        <xsd:documentation>
Optimization directive.
Data with "uncontrolledAccess" exists in a single version at any time.
Special care should be taken about exclusive read and write cycles.
Data has to be identified as optimizable all along the path of
inner assemblies of composite components to actually be optimized.
If this attribute value is true then ECOA will not protect anymore this data again
st concurrent accesses.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_event_link">
        <xsd:annotation>
            <xsd:documentation>
Link between EVENT operations.
</xsd:documentation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_operation_link">
                <xsd:sequence>
                    <xsd:element name="sender" type="AS_op_ref_send" minOccurs="1"
 maxOccurs="unbounded">
                        <xsd:annotation>
                            <xsd:documentation>
Identification of the event sender.
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="receiver" type="AS_op_ref_receive" minOccur
s="1" maxOccurs="unbounded">
                        <xsd:annotation>
                            <xsd:documentation>
Identification of the event receiver.
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_links">
        <xsd:annotation>
            <xsd:documentation>
Definition of the links of the assembly.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="dataLink" type="AS_data_link" />
                <xsd:element name="eventLink" type="AS_event_link" />
                <xsd:element name="requestLink" type="AS_request_response_link" />
            </xsd:choice>
            <xsd:element name="implicitLinks" type="AS_implicitLinks" minOccurs="0
" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref" abstract="true">
        <xsd:annotation>
            <xsd:documentation>
Contribution to a link (as producer, consumer, server or client).
</xsd:documentation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="when" type="AS_when_condition"
                minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
Expression of inclusion/exclusion of an operation in a link.
The operation belongs to the link only if the specified variable has the specified
 value.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="operation" type="OperationName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Reference to an operation of the componentType model associated to the instance by
 the "instance" attribute.
This attribute value must reference a valid operation name defined in the associat
ed componentType.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="instance" type="Name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Reference to a component instance then the attribut value must be an existing inst
ance name
(ie one of the instances defined in this assembly), or to the assembly itself then
 the attribute value
must be "extern".
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref_read">
        <xsd:annotation>
            <xsd:documentation>
Contribution to a data link (as reader).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_op_ref">
                <xsd:attribute name="activating" type="xsd:boolean" default="true"
>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                    <xsd:annotation>
                        <xsd:documentation>
True if the notification event should activate the thread hosting the instance.
Only meaningful if notifying=true on the component's operation.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="fifoSize" type="xsd:unsignedInt" default="8">
                    <xsd:annotation>
                        <xsd:documentation>
Max number of notification events waiting to be processed.
Only meaningful if notifying=true on the component's operation.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref_write">
        <xsd:annotation>
            <xsd:documentation>
Contribution to a data link (as writer).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_op_ref_read">
                <xsd:attribute name="reference" type="xsd:boolean" default="true">
                    <xsd:annotation>
                        <xsd:documentation>
If a 'dataWritten' operation is connected to several datalinks, then there must be
 one and only one datalink
for which reference=true on this operation.
The 'current value' is always taken from this datalink.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref_send">
        <xsd:annotation>
            <xsd:documentation>
Contribution to an event link (as sender).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_op_ref"></xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref_receive">
        <xsd:annotation>
            <xsd:documentation>
Contribution to an event link (as receiver).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_op_ref">
                <xsd:attribute name="activating" type="xsd:boolean" default="true"
>
                    <xsd:annotation>
                        <xsd:documentation>
If true, the event cause the activation of the receiver component.
If false, the event will be added into the FIFO and executed on the next activatin
g event reception.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="fifoSize" type="xsd:unsignedInt" default="8">
                    <xsd:annotation>
                        <xsd:documentation>
Max number of operations waiting to be processed.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_identified_member_value">
        <xsd:annotation>
            <xsd:documentation>
Add of a numerical identifier to a member.
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_member_value">
                <xsd:attribute name="id" type="xsd:unsignedInt" use="optional" />
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_request_response_link">
```

```
        <xsd:annotation>
            <xsd:documentation>
Link between RequestResponse operations. Exactly one client; one nominal server +
backup servers (cf. 'when' element).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_operation_link">
                <xsd:sequence>
                    <xsd:element name="client" type="AS_op_ref_client"
                        minOccurs="1" maxOccurs="1" />
                    <xsd:element name="server" type="AS_op_ref_server"
                        minOccurs="0" maxOccurs="unbounded" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref_client">
        <xsd:annotation>
            <xsd:documentation>
Contribution to a RequestResponse link (as client).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_op_ref">
                <xsd:attribute name="callbackActivating" type="xsd:boolean" defaul
t="true">
                    <xsd:annotation>
                        <xsd:documentation>
True if the callback should activate the thread hosting the instance.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="fifoSize" type="xsd:unsignedInt" default="8">
                    <xsd:annotation>
                        <xsd:documentation>
Max number of responses waiting to be processed.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_op_ref_server">
        <xsd:annotation>
```

```
            <xsd:documentation>
Contribution to a RequestResponse link (as server).
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="AS_op_ref">
                <xsd:attribute name="activating" type="xsd:boolean" default="true"
>
                    <xsd:annotation>
                        <xsd:documentation>
True if the request should activate the thread hosting the instance.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="fifoSize" type="xsd:unsignedInt" default="8">
                    <xsd:annotation>
                        <xsd:documentation>
Max number of requests waiting to be processed.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AS_variable_alias">
        <xsd:annotation>
            <xsd:documentation>
Association between the internal and the external names of a same variable.
</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="name" type="xsd:NCName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Internal name of the variable.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="alias" type="xsd:NCName" use="required">
            <xsd:annotation>
                <xsd:documentation>
External name of the variable.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
```

```
  <xsd:complexType name="AS_when_condition">
    <xsd:attribute name="instance" type="Name" use="required">
      <xsd:annotation>
        <xsd:documentation>
Name of the instance.
</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="variable" type="Name" use="required">
      <xsd:annotation>
        <xsd:documentation>
Name of the variable.
</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="value" type="xsd:string" use="required">
      <xsd:annotation>
        <xsd:documentation>
Value of the variable.
</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="AS_implicitLinks_operations">
    <xsd:annotation>
      <xsd:documentation>
Select some operations on some component instances.
</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="instance" type="xsd:string" use="required">
      <xsd:annotation>
        <xsd:documentation>
The name of an instance of this assembly, or a pattern matching several instances.
The character "*" matches any sequence of characters.</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="prefix" type="xsd:string" default="">
      <xsd:annotation>
        <xsd:documentation>
A prefix that is used to filter operations. The name of the operation must start w
ith the prefix value.
Moreover, the prefix is removed from the operation's name when comparing with othe
r operation names.</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    </xsd:complexType>
    <xsd:complexType name="AS_implicitLinks">
        <xsd:annotation>
            <xsd:documentation>
Create automatically links, based on operation names.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element name="operations" type="AS_implicitLinks_operations" minO
ccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="activating" type="xsd:boolean" default="true">
            <xsd:annotation>
                <xsd:documentation>
Value of the corresponding attribute on all created links.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
</xsd:complexType>
</xsd:schema>
```

6.2.5.6 File "Deployment.xsd"

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ecoa.technology/Deployment/3.0"
targetNamespace="http://www.ecoa.technology/Deployment/3.0"
elementFormDefault="qualified" >
    <xsd:include schemaLocation="inc/common.xsd" />
    <xsd:simpleType name="Endianness">
        <xsd:annotation>
            <xsd:documentation>
Ordering of bytes in words (BIG or LITTLE).
</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="BIG" />
            <xsd:enumeration value="LITTLE" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="E_start_mode">
```

```
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="NONE">
                <xsd:annotation>
                    <xsd:documentation>
The plaftorm does not initialize nor start components.
</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="FAST">
            <xsd:annotation>
                <xsd:documentation>
The plaftorm initializes and starts all components without any order constraints.
</xsd:documentation>
            </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="SYNCHRONIZED">
        <xsd:annotation>
            <xsd:documentation>
The platform initializes all components, then starts all components.
</xsd:documentation>
        </xsd:annotation>
    </xsd:enumeration>
</xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="DE_task">
        <xsd:annotation>
            <xsd:documentation>
Set of instances of components that are not processed in parallel.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element minOccurs="0" ref="doc" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="deployedInstanc
e" type="DE_deployed_instance">
                <xsd:annotation>
                    <xsd:documentation>
List of deployed instances in the task.
The order is significant: when several instances are activated by the same operati
on,
they will be activated in the order defined here.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        </xsd:sequence>
        <xsd:attribute name="name" type="Name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the task (unique in the application).
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="relativePriority" type="positiveInt" use="optional">
            <xsd:annotation>
                <xsd:documentation>
Priority associated to the task (in the scope of the application).
The higher the value, the higher the priority.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="DE_deployed_instance">
        <xsd:annotation>
            <xsd:documentation>
Definition of an instance deployed in an executable.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element minOccurs="0" ref="doc" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
        </xsd:sequence>
        <xsd:attribute name="ref" type="Name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the instance, with regards to its
declaration in the Assembly model.
The name of an instance inside a composite is "C_I",
where C is the name of the composite instance,
and I is the name of the instance in the Assembly model.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="DE_executable">
        <xsd:annotation>
            <xsd:documentation>
Set of instances of components.
</xsd:documentation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element minOccurs="0" ref="doc" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="task" type="DE_
task">
                <xsd:annotation>
                    <xsd:documentation>
List of tasks (i.e. threads) defined in the executable.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="name" type="Name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the executable (unique in the application).
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="DE_application">
        <xsd:complexContent>
            <xsd:extension base="DE_executable">
                <xsd:annotation>
                    <xsd:documentation>
Definition of a deployed application.
</xsd:documentation>
                </xsd:annotation>
                <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" name="executa
ble" type="DE_executable">
                        <xsd:annotation>
                            <xsd:documentation>
List of application executables.
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="external_io" ty
pe="IO_section">
                        <xsd:annotation>
                            <xsd:documentation>
'external_io' node is optional.
Indeed, applications may not have to communicate with external systems.
```

```
</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="assembly" type="PossiblyQualifiedName" use="r
equired">
                    <xsd:annotation>
                        <xsd:documentation>
Logical name of the Assembly model referenced by this deployed application model.
If the name is not qualified, the assembly is found in 02-Assemblies.
If the name is qualified (C.I), the assembly is the internal assembly of implement
ation
I of component C (which must be a composite).
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="production" type="xsd:string" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
Identification of the target platform for the code of this application shall be co
mpiled
(determines compilation chain, options, executable format, endianness, etc.)</xsd:
documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="endianness" type="Endianness" default="BIG">
                    <xsd:annotation>
                        <xsd:documentation>
Endianness used in network messages
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="start_mode" type="E_start_mode" default="NONE
">
                    <xsd:annotation>
                        <xsd:documentation>
Mode for automatic startup of components.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="application" type="DE_application">
        <xsd:annotation>
            <xsd:documentation>
```

```
Application root node.
</xsd:documentation>
        </xsd:annotation>
        <xsd:key name="execnamekey">
            <xsd:selector xpath="application/executable|application" />
            <xsd:field xpath="@name" />
        </xsd:key>
        <xsd:key name="instnamekey">
            <xsd:selector xpath="application/*/deployedInstance|application/deploy
edInstance" />
            <xsd:field xpath="@ref" />
        </xsd:key>
    </xsd:element>

    <!-- external_io part -->
    <xsd:complexType name="IO_interface">
        <xsd:annotation>
            <xsd:documentation>
Interface used to communicate with external systems.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="doc" minOccurs="0" />
            <xsd:element name="meta" type="MetaData" minOccurs="0" maxOccurs="unbo
unded" />
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="operation" type
="IO_operation">
                <xsd:annotation>
                    <xsd:documentation>
Defines the technical details of an operation that is at the boundary of the syste
m.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="name" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the logical port for network communication.
</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleType>
                <xsd:restriction base="xsd:NCName">
                    <xsd:maxLength value="27" />
                </xsd:restriction>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="IO_interface_in">
        <xsd:complexContent>
            <xsd:extension base="IO_interface">
                <xsd:annotation>
                    <xsd:documentation>
Interface used to communicate with external systems with a thread listening for in
puts.
</xsd:documentation>
                </xsd:annotation>
                <xsd:sequence></xsd:sequence>
                <xsd:attribute name="relativePriority" type="positiveInt" use="opt
ional">
                    <xsd:annotation>
                        <xsd:documentation>
Relative (in the application) priority associated to the thread managing port list
ening.
By default, priority is the highest one.
</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="IO_operation">
        <xsd:attribute name="name" type="xsd:NCName" use="required">
            <xsd:annotation>
                <xsd:documentation>
Name of the operation. This attribute value must reference an operation defined in
 the assembly.
</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="id" type="xsd:string" use="optional">
            <xsd:annotation>
                <xsd:documentation>
Identifier of the operation (at least unique in the port).
Is optional at application level, since it may be defined by application configura
tion.
Scope of unicity may depend on the platform and protocol used.
Type (e.g. integers, free string, constraints) may depend on the platform and prot
ocol used.
</xsd:documentation>
```

```
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="IO_section">
        <xsd:annotation>
            <xsd:documentation>
Informations about network interfaces to provide or to communicate with.
</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="inPort" type="IO_interface_in">
                <xsd:annotation>
                    <xsd:documentation>
Input only logical port (eventReceived, dataRead). N-P cardinality.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="op_id_inPort">
                    <xsd:selector xpath="operation" />
                    <xsd:field xpath="@id" />
                </xsd:unique>
            </xsd:element>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="outPort" type="IO_interface">
                <xsd:annotation>
                    <xsd:documentation>
Output only logical port (eventSent, dataWritten). N-P cardinality.
</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="inOutPort" type="IO_interface_in">
                <xsd:annotation>
                    <xsd:documentation>
Input/Output logical port (request_response, event, data). 1-1 cardinality.
</xsd:documentation>
                </xsd:annotation>
                <xsd:unique name="op_id_inOutPort">
                    <xsd:selector xpath="operation" />
                    <xsd:field xpath="@id" />
                </xsd:unique>
            </xsd:element>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
    </xsd:complexType>
</xsd:schema>
```

6.2.5.7 File "Workspace.xsd"

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ecoa.technology/Workspace/3.0"
targetNamespace="http://www.ecoa.technology/Workspace/3.0"
elementFormDefault="qualified" >

  <xsd:element name="workspace" type="CT_workspace">
    <xsd:annotation>
      <xsd:documentation>
Customisation of the workspace structure: a mapping from logical names to file/dir
ectory locations.
</xsd:documentation>
    </xsd:annotation>
  </xsd:element>

  <xsd:complexType name="CT_workspace">
    <xsd:sequence>
      <xsd:element name="library" type="CT_name_to_file" minOccurs="0" maxOccurs="
unbounded"/>
      <xsd:element name="componentType" type="CT_name_to_dir" minOccurs="0" maxOcc
urs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="CT_name_to_dir">
    <xsd:attribute name="name" type="xsd:NCName" use="required">
      <xsd:annotation>
        <xsd:documentation>
The logical name of a ECOA entity.
</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="dir" type="xsd:string" use="required">
      <xsd:annotation>
        <xsd:documentation>
The directory where the files of the entity are located.
The path can be absolute or relative (to the ECOA project root).
The path may contain environment variables, with syntax $(VAR) or ${VAR}.
</xsd:documentation>
```

Without prejudice to the property rights relating to the ECOA AS6 Standard, the information in this document relating to the changes envisaged for the transition from the ECOA AS6 Standard to the ECOA AS7 Standard is the intellectual property of Dassault Aviation and Thales DMS France SAS. The information set out in this document is provided solely on an ‘as is’ basis and co-developers of this specification make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="CT_name_to_file">
    <xsd:attribute name="name" type="xsd:NCName" use="required">
      <xsd:annotation>
        <xsd:documentation>
The logical name of a ECOA entity.
</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="file" type="xsd:string" use="required">
      <xsd:annotation>
        <xsd:documentation>
The file where the entity is located.
The path can be absolute or relative (to the ECOA project root).
The path may contain environment variables, with syntax $(VAR) or ${VAR}.
</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>

</xsd:schema>
```