European Component Oriented Architecture (ECOA®) Collaboration Programme: Preliminary version of the ECOA Architecture Specification Part 2: Definitions

Dassault Ref No: DGT 2041081-A
Thales DMS Ref No: 69398916-035 --

Issue: 7

Prepared by
Dassault Aviation and Thales DMS

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.*

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

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.

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.

Figures

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.

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 2 of the Architecture Specification, and provides definitions for terms used.

Some of the terms are new and some are defined to ensure there is common understanding of the term as used in the context of ECOA®. Terms are provided in alphabetical order. The reader is encouraged to consult Architecture Specification Part 1 for a more structured introduction to the ECOA® concepts.

Section 6 contains a figure that illustrates the ECOA® terms in the context of a system implementation.

This document is focused on ECOA Core scope but may refer to some extensions to ease the understanding.

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

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.

4 Definitions

For the purpose of this standard, the definitions shown below apply.

Definitions are alphabetically ordered without taking into account any leading “ECOA” acronym.

4.1 Application

An ECOA **Application** is a software system, made of an assembly of ECOA **Components**, and deployed on a **Computing Node**.

An **Application** is an **Executable** that may have sub-**Executables** (**Executables** that will be launched as sons by the **Application** itself).

An **Application** contains one or more **ECOA Component Instances** and associated **Containers**.

An Application is the coarse grain unit of exchange between application developers and system integrators.

4.2 Application Ports

Application 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.

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**

4.3 ECOA Architecture Specification

Specification that defines the essential technical characteristics of **Components** and **ECOA Platforms**.

It is built on a set of fundamental features that constitute the **ECOA Core** and a set of optional features called **Options**. The **Options** are part of the metamodel but are not mandatory for a specific platform to be ECOA compliant.

4.4 Assembly Schema

A specification of a composition of a specific **Application** defined by:

- A set of **Component Instances** with references to their associated **Component Types**
- A set of **Operation Links** between the **Component Instances**.

Note : It is possible to define an Extension to describe a System Assembly of multiple Applications..

4.5 Binding

A language **Binding** defines the way an **ECOA Component Type** will be translated in programming language or technology specific APIs. A **Binding** describes as well the Component/Container and the Container/Component interfaces.

4.6 ECOA Development Process

The process by which **Applications and Components** are designed, implemented, built, verified and managed through-life.

4.7 ECOA Component

Component is a generic concept that may refer to :

- A Component Type
- A Component Instance
- A Component Implementation

4.8 Component Type

The **Component Type** defines the interface of a **Component** in terms of **Component Operations**, **Component Properties, PINFOs, Triggers,** and in terms of kind of Component (STANDARD, EXTERNAL, PERIODIC_TRIGGER_MANAGER, …).

An **ECOA Component** interacts with other **ECOA Components** using the **Component Operations** (i.e. **Events, Request-Response** and **Versioned Data**).

4.9 Component Implementation

A particular realization of an **ECOA Component Type**. There are two kinds of implementations: Concrete Implementations and Composite Implementation.

4.10 Concrete Component Implementation

The software implementing an **ECOA Component Type** in a given programming language.

This software should be re-entrant. Re-entrancy allows a single copy of Component implementation to be used concurrently by many Component instances without them interfering with each other.

This software can be delivered as source code or compiled code.

A Concrete Component Implementation is the fine grain unit of exchange between component developers and software integrators.

4.11 Component Instance

A specific **ECOA Component Implementation** instance of an **ECOA Component**.

Component Operations, for any particular **Component Instance** of an **ECOA Component**, are processed sequentially in a strict FIFO manner - determined by the order in which the initiating action for each **Component Operation** is received by the **Container** instance.

4.12 Component Interface

The interface between a **Component Instance** and a **Container** instance.

It provides the mechanisms for a **Container** instance to invoke **Component Operations**.

See also **Container Interface**.

4.13 Component Operation

A **Component Operation** is a named elaboration of one of a set class of operations, supported by the **Infrastructure**, to send/receive **Events**, make **Request-Responses**, and publish or read **Versioned Data**.

4.14 Component Operation Link

A link defined during design, in an **Assembly Schema**, to specify a connection between several **Component Operations** of **Component Instances**.

Note: Component Operation Links can be logically connected to each other via metadata and implicit links to mimic micro-services approaches

4.15 Component Runtime Lifecycle

A set of states in which a **Component Instance** exists. A **Component Instance** transitions between these states at runtime.

The lifecycle of a **Component Instance** is managed by the **Infrastructure** using the **Lifecycle Events.**

4.16 Composite Component Implementation

Composite Component Implementations (or **Composite Components**) are **Component Implementations**, which are composed of **Components**, which may in turn be **Composite Components.**

4.17 Computing Node

Single processor element onto which **Applications, Executables** and hence **ECOA Components** are executed.

This processor can be multi-core, provided that the cores have access to a shared memory space.

4.18 Computing Platform

The **Computing Platform** is composed of **OS/Middleware** and **Computing Nodes**.

4.19 Container

A Container is the software that provides the operating environment for a Concrete Component Implementation.

The **Container** supports:

- A single thread to invoke the **ECOA Component** entry points as defined by the **Component Interface** according to a defined scheduling policy
- the **Infrastructure Services (**e.g. time, logging and fault management).

The **Container** software has access to the **OS/Middleware Interface**.

4.20 Container Interface

The API made available to the **ECOA Component** providing the ECOA defined **Component Operations** and **the Infrastructure Services**.

See also **Component Interface**.

4.21 Deployment Schema

An allocation of **ECOA Component Instances** to **Executables** and **Tasks** into an **Application.**

Note : : It is possible to define an Extension to describe a System Deployment of multiple Applications.

4.22 Early Validation

A process which can provide an indication that a system will meet its functional and **QoS** requirements prior to availability of **Applications** or **ECOA Platform**.

Early Validation might be applied iteratively, as the design lifecycle proceeds, to obtain more refined results.

4.23 Event

An ECOA **Event** is a one-way discrete interaction between **ECOA Components**, optionally carrying typed parameters.

4.24 Executable

A mechanism that provides spatial partitioning such that code within one **Executable** cannot compromise the operation of another through erroneous or malicious behaviour. Code in one **Executable** cannot directly access (read or write) data in another **Executable**.

An **Executable** contains one or more **ECOA Components** and associated **Container** instance(s).

4.25 External Component

A specific kind of **Component** that is able to communicate with hardware and/or software using interfaces not defined by ECOA. This component has a supplementary thread that may be used to manage active waiting on external interface.

4.26 ECOA Extension

An **ECOA Extension** is a set of features that extend the **ECOA Architecture Specification**. The ECOA Extensions are self-documented, may be standardized by themselves and shall reference the ECOA Architecture Specification.

4.27 Fault Handler

An optional entity being responsible for triggering recovery procedures for Infrastructure errors. Cf. [OPTION FAULT HANDLER]

4.28 Functional Chain

At the Information System Level, a **Functional Chain** is aset of functions working together. In ECOA, these functions are implemented as **Operations** allocated to **Components**.

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.

Each **functional chain** has a maximum response time. This is equal to the sum of all maximum response times of all its functions. This reflects an end-to-end timing requirement for the system.

Functional Chains are derived by the system designer who then allocates functions to **Components**.

4.29 Implicit Link

Implicit link is an ECOA model language artefact that may simplify the **Assembly** writing activity, Cf. Part 7.

4.30 Infrastructure

Everything that provides for the invocation of **ECOA Components**. It includes both the **Platform Integration Code** and the **Computing Platform**.

4.31 Infrastructure Services

Standard **Services** provided by the **Infrastructure** to all **Components**.

An example of an **Infrastructure Service** is the time or log **Services**.

4.32 Inversion-of-Control

Components are passive, i.e. executing only when invoked. **Component Operations** are invoked by the **Container**.

4.33 Legacy Software Architecture

Non-ECOA software architecture (that may be used within, or to support, an **ECOA System**).

4.34 ECOA Librarian

The ECOA Librarian is responsible of:

- Maintaining the ECOA Catalogue of **Components**
- Performing a match-making role, facilitating contact between buyers (system integrators) and vendors (component suppliers),
- Coordinating with Component Suppliers to retrieve information about components

4.35 Lifecycle Events

Events issued by the **Infrastructure** to manage the lifecycle of **ECOA Components** or Events raised by the **Component** itself to act on its own lifecycle state

4.36 ECOA Logical Interface (ELI)

A standardised message protocol, to be defined as an Extension, that defines how separate **ECOA Platforms** may interact over network.

4.37 Logical System

A **Logical System** consists of **Computing Nodes** and network. This allows **Early Validation** to be completed and prediction of the performance of the system, early in the development lifecycle.

This concept may be used in an Extension for multi-nodes systems.

4.38 Metadata

On many elements of the ECOA model 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.

The Metadata mechanism is a simple way to personalize the ECOA standard to specific needs or adapt it to specific tools.

4.39 ECOA Option

An **ECOA Option** is an **ECOA Core** feature that may be modelled through ECOA MetaModel [Part 7] but that is not mandatory to be implemented by an **ECOA Platform** to be an **ECOA compliant Platform**.

4.40 OS/Middleware Interface

The interface between a **Container** and the underlying operating system or middleware.

This interface is independent of **Component Implementation** language.

4.41 PINFO

Persistent Information (PINFO) is a minimal and standard API to allow the retrieval of data that persists when power is cycled.

Data stored using PINFO persists beyond the operating period of an ECOA system and can be a predefined input to an ECOA system.

4.42 ECOA Platform

The hardware and software infrastructure on which **ECOA Components** are hosted.

An **ECOA Platform** consists of one or more collaborating **ECOA Stacks**.

4.43 Platform Integration Code

The code that allows the hosting of **ECOA Components** on a **Computing Platform**.

This includes **Container** instances together with code for managing the **Applications**, **Executables**, and **Computing Nodes**.

4.44 Properties

The **Properties** of a **Component** allow tailoring generic aspects in a data-driven fashion. For example this may specify units, capacity, accuracy, resolution.

Properties are named attributes, with values that can be assigned per **Component**.

4.45 Quality-of-Service

The attributes of an ECOA item that identify the non-functional characteristics of this item, the QoS may be defined trough **Metadata**.

4.46 Reactive Execution Model

Model of execution where the **Container** instance invokes an ECOA Component **Operation** from the queue of activating **Events** or **Request-Responses** as soon as possible after earlier operations of the same Component Instance have been completed.

In the reactive model, an activating operation is processed as soon as the processing resource is given to the Component. In contrast, a non-activating operation is queued until the arrival of an activating Event or Request-Response.

4.47 ECOA Reference Platform

An implementation of the **ECOA Platform** used to develop and validate **Components.**

4.48 Request-Response

A two-way pair of discrete interactions between client and server **ECOA Components**, where the client issues a request, with or without typed parameters, and the server responds (on completion) with a result.

4.49 ECOA Software Platform

The software that implements the **Infrastructure**.

4.50 ECOA Stack

An **ECOA Stack** is the **ECOA Platform Integration Code** and **OS/ Middleware** executing on a single **Computing Node**.

One **ECOA Stack** may communicate with another via the **ECOA Logical Interface**.

4.51 ECOA Standard

The **ECOA Standard** is a modular one, which aggregates several sub-standards that are formally published:

- The **ECOA Architecture Specification** (Core + Options),
- **ECOA Bindings**,
- **ECOA Extensions**.

4.52 ECOA Standard Working Group

The ECOA Standard Working Group is responsible for:

- Defining and maintaining the **ECOA Standard**
- Responding to feedback from ECOA users

4.53 ECOA System

A computing system executing ECOA applications running on one or more **ECOA Platforms**.

4.54 Task

Executables (including the **Application** itself) defines **Tasks** that will host **Component Instances**.

A **Task** defines its priority relatively to other tasks of the application.

4.55 STANDARD Component

A STANDARD Component is a component with no specific feature. Cf. **Component Type** definition.

4.56 Periodic Trigger Manager Component

A design element, implemented by the **Infrastructure**, characterised as a Component that emits an **Event**, at a period specified at design time.

4.57 Trigger

A design element that defines the need to be activated after a specific delay, implemented by the **Infrastructure**, characterised as a specific **Component Operation** that will be associated with a received **Event** (without parameter) of the same **Component**.

4.58 User Context

A data object specific to a **Component Instance**, which together with Warm Start Context is the state data defining an instance of an ECOA Component. This allows the **ECOA Component** to be instantiated more than once if the **ECOA Component** maintains an internal state.

The **user context** holds all the private data that is used:

- by a **Container** instance and the **Infrastructure** to handle the **Component Instance** (**Infrastructure**-level technical data),
- by the **Component Instance** itself to support its functions (user-defined local private data).

The construction for the data structure defining the **user context** is defined by language-specific bindings.

4.59 ECOA Validation Suites

A suite of software that supports confirmation of an **ECOA Platform's** compliance with the **ECOA Standard**.

4.60 Versioned Data

Version Data is a mechanism for sharing data between **Component Instances** over a **Component Operation Link**. For each **Component Operation Link** the **Version Data** mechanism can be configured with or without access control.

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.

4.61 Warm Start Context

An optional (Cf. **[OPTION WARM START CONTEXT])** data object specific to a **Component Instance**, which together with User Context is the state data defining an instance of an ECOA Component. This allows the **ECOA Component** to be restarted in a warm mode with private data it has previously saved.

The **warm start context** holds all the private data that is used:

- by a **Container** instance and the **Infrastructure** to handle the **Component Instance** (**Infrastructure-**level technical data),
- by the **Component Instance** itself to support its functions (user-defined local private data) after its warm restart. The user-defined local private data has been previously saved by the **Component Instance** itself.

The construction for the data structure defining the **warm start context** is defined by language-specific bindings.

4.62 XML Metamodel

XML Metamodel defines the data model used to formally describe ECOA artefacts.

5 Abbreviations

API	Application Programming Interface
ECOA	European Component Oriented Architecture. ECOA® is a registered trademark.
ELI	ECOA® Logical Interface (extension)
OS	Operating System
PINFO	Persistent Information
QoS	Quality of Service
XML	eXtensible Markup Language

6 ECOA Terms in Context

Figure 1 illustrates the ECOA terms in the context of a system implementation.

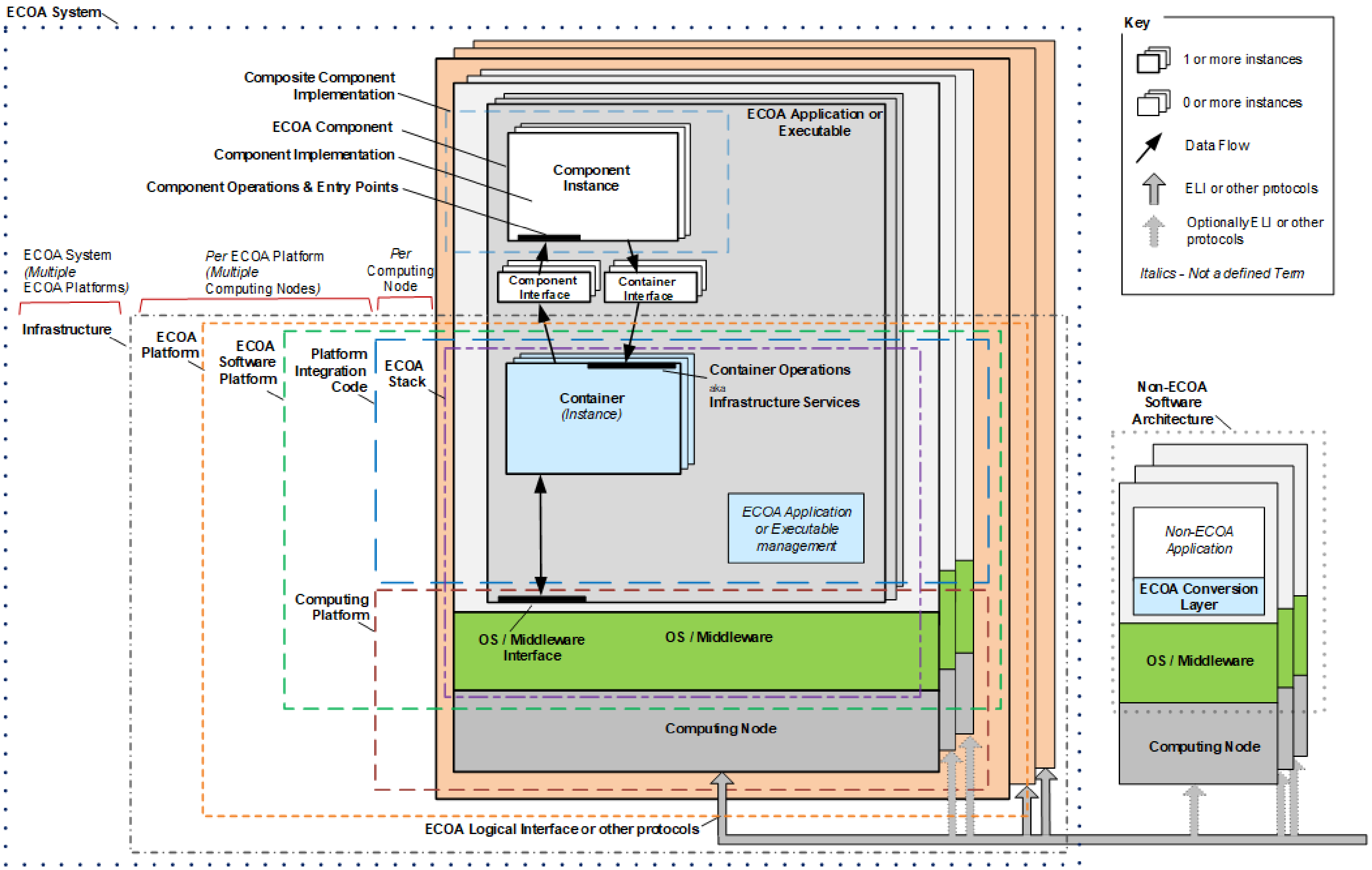

Figure 1 Scope of ECOA Terms within a System Implementation

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.