European Component Oriented Architecture (ECOA) Collaboration Programme: Volume IV: Common Terminology

BAE Ref No: IAWG-ECOA-TR-012
Dassault Ref No: DGT 144487-B

Issue: 2

Prepared by
BAE Systems (Operations) Limited and Dassault Aviation

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

1 Table of Contents

2 Table of Figures

3 Abbreviations

APEX	Application Express
API	Application Program Interface
ASAAC	Allied Standards Avionics Architecture Council
ASC	Application Software Component
ECOA	European Component Oriented Architecture
ELI	ECOA Logical Interface
OS	Operating System
POSIX	Portable Operating System Interface
QoS	Quality-of-Service
XML	Extensible Markup Language

4 Introduction

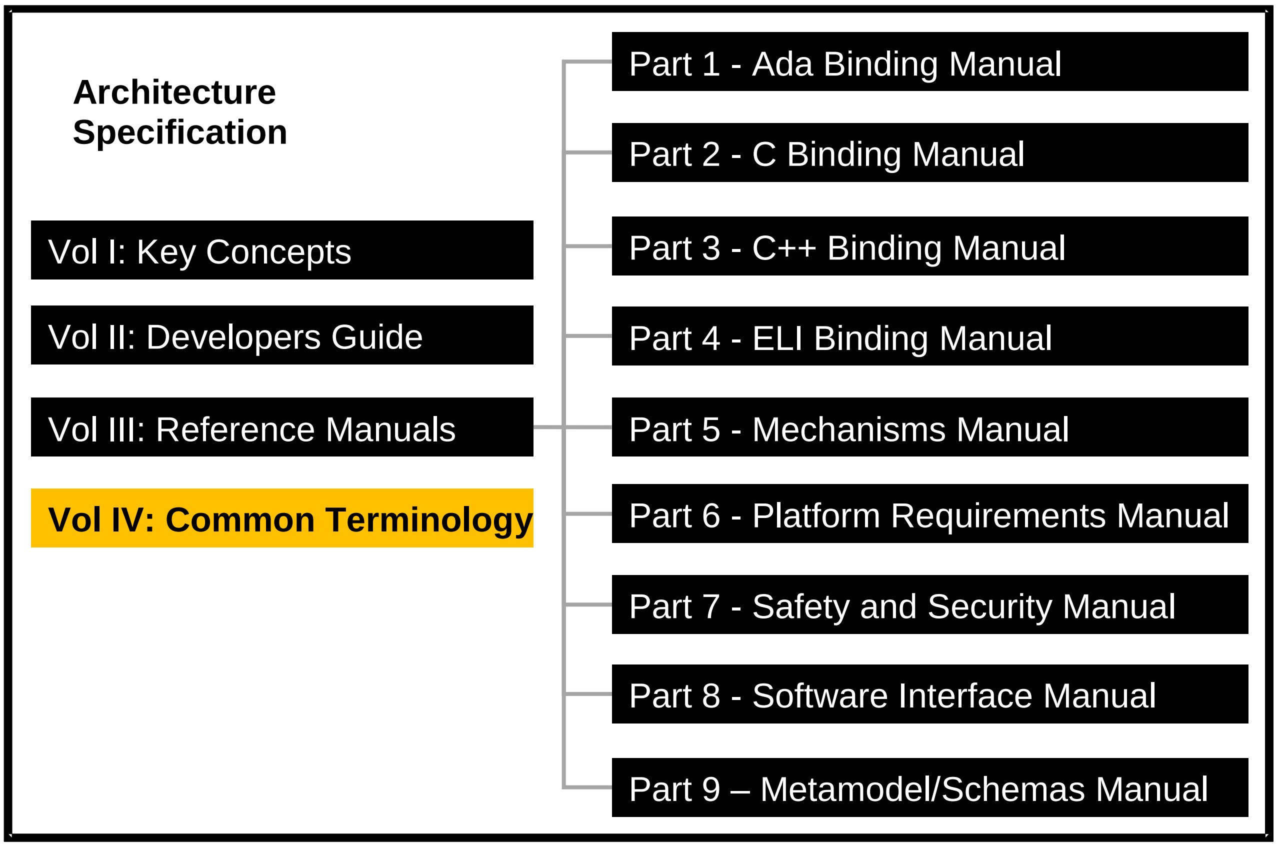
Architecture Specification
Vol I: Key Concepts
Vol II: Developers Guide
Vol III: Reference Manuals
Vol IV: Common Terminology
Part 1 - Ada Binding Manual
Part 2 - C Binding Manual
Part 3 - C++ Binding Manual
Part 4 - ELI Binding Manual
Part 5 - Mechanisms Manual
Part 6 - Platform Requirements Manual
Part 7 - Safety and Security Manual
Part 8 - Software Interface Manual
Part 9 – Metamodel/Schemas Manual

Figure 1 – ECOA Documentation

The Architecture Specification provides the definitive specification for creating ECOA-based systems. It describes the standardised programming interfaces and data-model that allow a developer to construct an ECOA-based system. The details of the other documents comprising the rest of the Architecture Specification can be found in Section 0.

The Architecture Specification consists of four volumes, as shown in Figure 1:

- Volume I: Key Concepts
- Volume II: Developer's Guide
- Volume III: Reference Manuals
- Volume IV: Common Terminology

This document is Volume IV of the ECOA Architecture Specification, and describes the common terminology used within ECOA.

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 the Key Concepts document for a more structured introduction to the ECOA concepts.

5 ECOA Specific Definitions

Where not explicitly prefixed with ECOA, the definitions are specific to the ECOA context, rather than any general definition.

ECOA Agency	The **ECOA Agency** is responsbile for, but not limited to, performing the following roles: • Define and maintain the **ECOA Standard** • Maintain catalogues of **ASCs** • Coordinate and facilitate cooperation between stakeholders • Provide certification / regulation guidance, with respect to the **ECOA Standard**, to customers and suppliers • Verify that an ECOA **ASC** is correct from two points of view: o compliance with the ECOA Reference Platform o consistency with the Reference Domain Architecture. NOTE: the ECOA Agency does not yet exist.
Application Software Component	An **Application Software Component** (**ASC**) is the unit of exchange between software developers and/or integrators. It has the following properties: • Provides **Services** • May in turn require **Services** of other **ASCs** • Conforms to ECOA **Inversion-of-Control** principles • Requires a **Container** to invoke its operations and provide linkage to its required **Services**. • May be tailored to provide specific behaviour using **Properties**. An **ASC** is sometimes to referred to as a Component where its meaning is readliy apparent from the context.
Application Software Component Definition	An **Application Software Component Definition** specifies the identity of: • Provided **Services** • Required **Services** • Provided **QoS** for the Provided **Services** • Required **QoS** for the Required **Services** • Defined **Properties** of the **ASC**. Note: There may be more than one implementation for a given **Application Software Component Definition**.

Application Software Component Implementation	An implementation of an **ASC** which conforms to a given **Application Software Component Definition**. An **Application Software Component Implementation** includes: • **Application Software Component Implementation Description** • Code that implements the provided **Services**. An **ASC** Implementation can be exchanged.
Application Software Component Implementation Description	The description of the **Application Software Component Implementation**. The description includes: • References to any code libraries used • The **Module Types**, **Module Implementations** and **Module Instances** that form the **Application Software Component Implementation** • **Module Operation Links** for: o the provided **Service Operations** o required **Service Operations** o any **ECOA Module** to **ECOA Module** interactions internal to the **ASC**.
Application Software Component Instance	An instance of an **Application Software Component Implementation**, which will be independently deployed.
Assembly Schema	A specification of a composition of **ASCs** defined by: • A set of **Application Software Component Instances** with references to their associated **Application Software Component Definitions** • A set of **Service Links** between the **Application Software Component Instances.**
ECOA Business Model	The definition of how the overall ECOA concept is expected to operate in the business sense to achieve the commercial benefits. For example it may cover the following areas: • Commercial • Legal • Regulatory • Technical conformance.
ECOA Component Development Process	The process by which **ASCs** are designed, implemented, built, verified and managed through-life.

ECOA Compliant Platform	An **ECOA Platform** which is fully compliant with the **ECOA Standard.**
Component Runtime Lifecycle	A set of states in which an **Application Software Component Instance** may exist. An **ASC** will make transitions between these states at runtime.
Composite Component	**Composite Components** resemble **ASCs** externally, but are composed from **ASCs**, which may in turn be **Composite Components**.
Computing Node	Single processor element onto which **Protection Domains** and hence **ECOA Modules** are allocated.
Computing Platform	The **Computing Platform** is composed of **OS/Middleware** and **Computing Nodes**.
Container	A **Container** is the software that provides the operating environment for an **ECOA Module** or a set of **ECOA Modules.** The **Container** supports: • multiple threads to invoke the **ECOA Modules'** entry points as defined by the **Module Interface** according to a defined scheduling policy • the **Container Operations** defined in the **Container Interface** which includes the ECOA **Infrastructure Services**. A **Container** may contain one or more **ECOA Modules** which are implementing the **Service Operations** of one or more **ASCs**. The **Container** software has access to the **OS/Middleware Interface**.
Container Interface	The API made available to the **ECOA Module** providing the ECOA defined **Container Operations**. See also **Module Interface**.

Container Operation	**Container Operations** are made available to an **ECOA Module** through the **Container Interface**, and can be used to: • Interact with **ECOA Modules** implementing the same **ASC** • Interact with **ECOA Modules** implementing other **ASCs** • Access **Infrastructure Services** (e.g. time, logging and fault management) The API name and parameters are instantiated from a language-specific template that includes information such as **Module Implementation** name and parameters.
Context	A data object specific to a **Module Instance**, which allows the **ECOA Module** to be instantiated more than once. The **context** holds all the private data that is used : • by a **Container** instance and the **Infrastructure** to handle the **Module Instance** (**Infrastructure-**level technical data), • by the **Module Instance** itself to support its functions (user-defined local private data). The construction for the data structure defining the **context** is defined by language-specific bindings.
ECOA Conversion Layer	Software that adapts a legacy application to make it compatible with the **ECOA Logical Interface** (ELI). This enables the legacy software to interact with the rest of an **ECOA System**.
Deployment Schema	An allocation of **ECOA Modules** to **Protection Domains**, **Protection Domains** to **Computing Nodes**. Also specifies the logging policy to be applied.
Driver Component	An **ASC** that provides **Services** to communicate with hardware and/or software using interfaces not defined by ECOA.
Dynamic Discovery	Runtime identification / selection of a service or data provider. The specification of Dynamic Discovery in ECOA is immature and will be addressed in further stages of the ECOA programme.
Dynamic Trigger	A design element, implemented by the **Infrastructure**, characterised as a Module that accepts an initiating **Event** and emits, after the period defined by the initiating **Event**, a delayed **Event**.

Early Validation	A process which can provide an indication that a system will meet its functional and **QoS** requirements prior to availability of **ASCs** or **ECOA Platform**. **Early Validation** might be applied iteratively, as the design lifecycle proceeds, to obtain more refined results.
ECOA Ecosystem	The **ECOA Ecosystem** comprises, but is not limited to the following: • Library of **ASCs** • ECOA stakeholders • **ECOA Standard** and Process guidelines.
Event	An ECOA **Event** is a one-way discrete interaction between **ECOA Modules**, optionally carrying typed parameters.
Functional Chain	At the Information System Level, a **Functional Chain** is an ordered set of functions working together. In ECOA, these functions are implemented as **Service Operations** allocated to **ASCs**. 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 Chain** are derived by the system designer who then allocates functions to **ASCs**.
Infrastructure	Everything that provides for the invocation of **ECOA Modules**. It includes both the **Platform Integration Code** and the **Computing Platform**.
Infrastructure Services	Standard **Services** provided by the **Infrastructure** to all **ASCs**. These may be implemented locally or remotely. An example of an **Infrastructure Service** is the time **Services**.
Insertion Policy	The specification of how an **ASC** is inserted into an **ECOA System**. The insertion policy will include: • The specification of the **ASC's** offered **Quality-of-Service** (**QoS**) and the expected **QoS** of its required **Services** • The specification of entry points • The specification of resource requirements (e.g. memory) • Specification of an **ASC's** scheduling requirements, including static or priority scheduling parameters.

Inversion-of-Control	**ASCs** are passive, i.e. executing only when invoked. **ASC Module Operations** are invoked by the **Container** in accordance with the **ASC's** scheduling policy.
Legacy Software Architecture	Non-ECOA software architecture (that may be used within, or to support, an **ECOA System**).
Lifecycle Commands	Commands passed as **Events** managed by the **Infrastructure** to handle the lifecycle of **ECOA Modules** and **ASCs**.
ECOA Logical Interface	The standardised message protocol that defines how separate **ECOA Platforms** interact across a communication links. It may optionally be used as the message protocol between **Protection Domains** on the same **ECOA Stack** or between **ECOA Stacks** within the same **ECOA Platform**. The message protocol may be implemented using any suitable transport layer.
Logical System	A **Logical System** consists of **Protection Domains**, **Computing Nodes** and network. This allows **Early Validation** to be completed and prediction of the performance of the system, early in the development lifecycle.
ECOA Module	An **ASC** is implemented by one or more **ECOA Modules**. **Module Operations**, for any particular instance of an **ECOA Module**, are processed sequentially in a strict FIFO manner - determined by the order in which the initiating action for each **Module Operation** is received by the **Container** instance. An **ECOA Module** interacts with other **ECOA Modules** using the ECOA defined interactions (i.e. **Events, Request-Response** and **Versioned Data**).
Module Deadline	The maximum time by which a **Module Instance** should have responded by, regardless of which entry point is invoked, which, given sufficient resource, would guarantee all the response time constraints (maximum response times and minimum inter-arrival times) of all the **ASC** functions in which the **Module Instance** is involved. The **Module Deadline** is provided by the **ASC** supplier.
Module Implementation	The software implementing an **ECOA Module**. This software must be re-entrant.

Module Instance	An instance of an **ECOA Module**.
Module Interface	The interface between a **Module Instance** and **Container** instance. It provides the mechanisms for a **Container** instance to invoke **Module Operations**. See also **Container Interface**.
Module Operation	A **Module 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**. A **Service Operation** Is implemented by a **Module Operation**. **Module Operations** for **Module Instances** within the same **Component Instance** may be wired together without reference to any **Service Operation**.
Module Operation Link	A link defined during design, to specify a connection between any of the following: • a **Service Operation** and a **Module Operation**. • a **Service Operation** and a **Container Operation** • a **Container Operation** and a **Module Operation**
Module Runtime Lifecycle	A set of states in which a **Module Instance** exists. A **Module Instance** transitions between these states at runtime. The lifecycle of a **Module Instance** can be managed by a **Supervision Module** Instance using the **Lifecycle Commands**, provided they are both within same **ASC instance.**
Module Type	The **Module Type** defines the interface of a **Module Implementation** in terms of **Module Operations**, **Container Operations**, Module Properties and whether it is an **ECOA Supervision Module**.
OS/Middleware Interface	The interface between the **Container** and the underlying operating system or middleware. This interface is independent of **Application Software Component Implementation** language. Examples are POSIX, APEX or ASAAC APOS.

ECOA Platform	The hardware and software infrastructure on which an **ECOA Modules** are hosted. An **ECOA Platform** consists of one or more collaborating **ECOA Stacks**.
Platform Integration Code	The code that allows the hosting of **ECOA Modules** on a **Computing Platform**. This includes **Container** instances together with code for managing the **Protection Domains**, **Computing Nodes** and Platform.
Properties	The **Properties** of an **ASC** 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 **ASC** Instance and subsequently read at runtime by **Module Instances** to access the values relevant to the **ASC** instance. At this stage of ECOA, it is envisaged that **Properties** will be set statically at design-time. *In future lifecycle services may be able to modify them at runtime (during a reset, for example)*.
Protection Domain	A mechanism that provides spatial and potentially temporal partitioning such that code within one **Protection Domain** cannot compromise the operation of another through erroneous or malicious behaviour. Code in one **Protection Domain** cannot directly access (read or write) data in another **Protection Domain**. A **Protection Domain** contains one or more **ECOA Modules** and associated **Container** instance(s).
Quality-of-Service	The attributes of an **ASC** that identify the non-functional characteristics of provided **Services** and places requirements on the non-functional characteristics of required **Services**.
Reactive Execution Model	Model of execution where the **Container** instance invokes an ECOA **Module Operation** from the queue of activating **Events** or **Request-Responses** as soon as possible after earlier operations of the same Module Instance have been completed. See also **Rhythmic Execution Model.**
ECOA Reference Platform	An implementation of the **ECOA Platform** developed by, or for, the **ECOA Agency** to develop and validate **ASCs**.

Request-Response	A two-way pair of discrete interactions between client and server **ECOA Modules**, where the client issues a request, with or without typed parameters, and the server responds (on completion) with a result.
Rhythmic Execution Model	This where the **Container** considers the queued (FIFO) **Module Operations** at a specified periodic rate. If the queue contains an activating **Module Operation** this will be invoked together with non-activating **Module Operations** that precede it in the queue. Only one activating **Module Operation** is invoked in a single period. See also **Reactive Execution Model.**
Service	A **Service** is a named and published set of one or more operations (**Service Operations**) that are offered by a provider and may be utilised by a client.
Service Definition	The definition of a **Service**, including: • **Service** identifier • Set of **Service Operations** **Service Definition**s will be referenced in an **Application Software Component Definition** to specify provided and required **Services**.
Service Instance	An instance of a **Service**. The same **Service** may be provided by multiple instances of an **ASC** or by different **ASCs**.
Service Link	A system design level connection that links a **Service** required by one **ASC** to a **Service** provided by another **ASC**. A **Service**, provided or required by an **ASC**, may have multiple **Service Links**, which through a ranking system define alternative system connectivity in support of reconfiguration..
Service Operation	A **Service Operation** defined in a **Service Definition**. A **Service** is implemented by one or more **Service Operations**. A **Service Operation** is identified as either **Request-Response**, **Event** or **Versioned Data**.

ECOA Software Platform	The software that implements the **Infrastructure**.
ECOA Specification	Specification that defines the essential technical characteristics of **ASCs** and **ECOA Platforms**.
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**.
ECOA Standard	A formal published subset of the **ECOA Specification**.
ECOA Supervision Module	An **ECOA Supervision Module** has the responsibility of managing an **ASC,** including the management of other **ECOA Modules** that make up that **ASC**. The **ECOA Supervision Module** has additional operations in the **Container Interface** in order to enable it to achieve this.
ECOA System	A computing system executing ECOA applications running on one or more **ECOA Platforms**.
Timestamps	Information provided by the **Infrastructure** which indicates when data was written and events, requests and responses were sent.
ECOA Toolset	At a minimum, the **ECOA Toolset** enables the generation of skeleton application source code, together with its required **Platform Integration Code**, based on XML descriptions.
Trigger Instance	A design element, implemented by the **Infrastructure**, characterised as a Module that emits an **Event**, at a period specified at design time.
ECOA Validation Suites	A suite of software that supports confirmation of an **ECOA Platform's** compliance with the **ECOA Standard**.

Versioned Data	**Version Data** is a mechanism for making versions of locally held data sets available to **Module Instances** throughout an **ECOA System**. This is achieved through the publication and distribution of data sets to identified subscribers. Readers work on local copies of the data that remain consistent throughout a read transaction. Writers are able to modify data locally before committing or cancelling any updates to end a transaction.

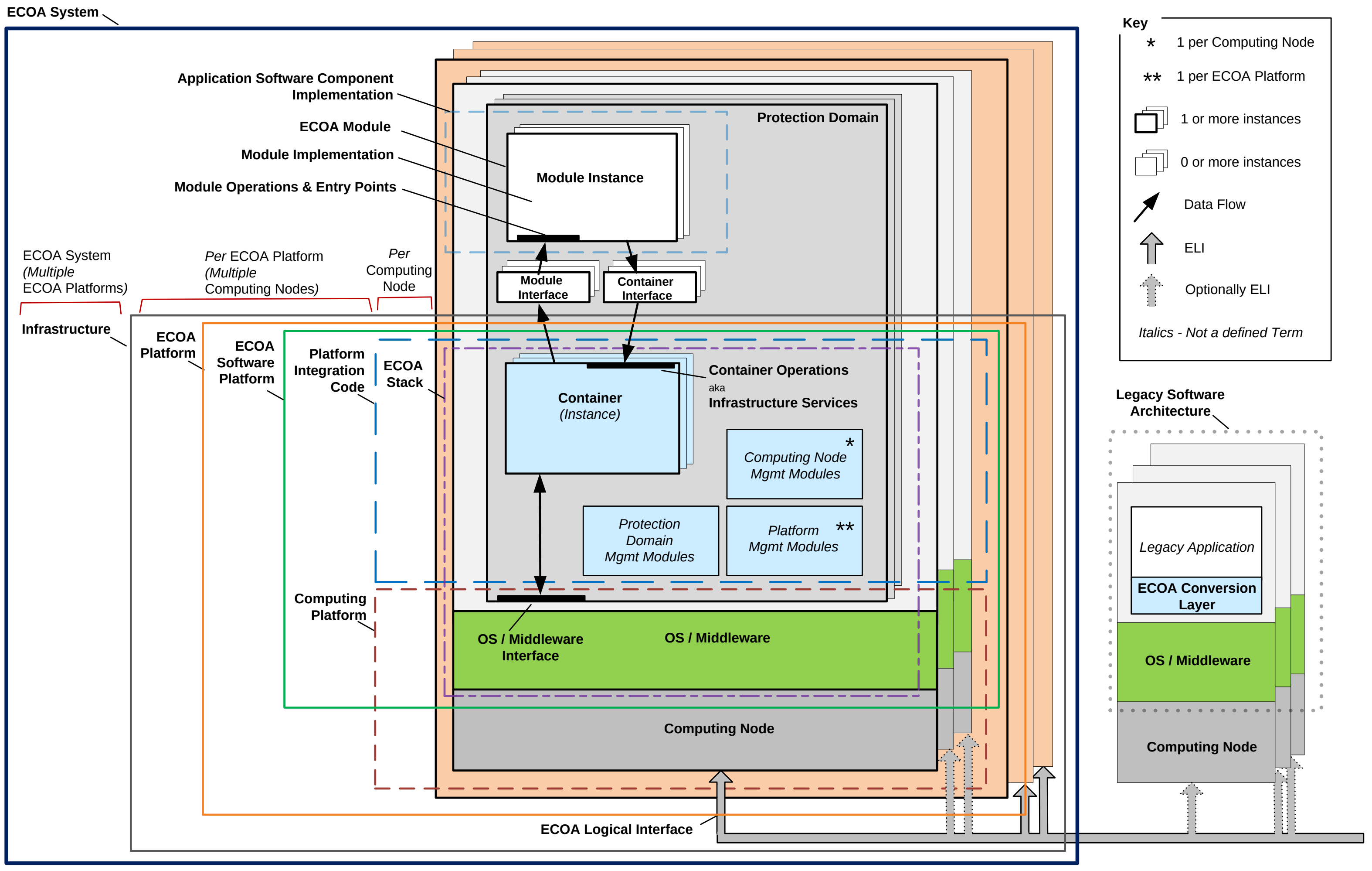

Figure 2 – Scope of ECOA Terms within a System Implementation

6 References

Ref.	Document Number	Version	Title
1.	IAWG-ECOA-TR-001	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume I Key Concepts
2.	IAWG-ECOA-TR-002	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume II Developers Guide
3.	IAWG-ECOA-TR-003	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 1: Ada Binding Reference Manual
4.	IAWG-ECOA-TR-004	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 2: C Binding Reference Manual
5.	IAWG-ECOA-TR-005	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 3: C++ Binding Reference Manual
6.	IAWG-ECOA-TR-006	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 4: ELI and Transport Binding Reference Manual
7.	IAWG-ECOA-TR-007	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 5: Mechanisms Reference Manual
8.	IAWG-ECOA-TR-008	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 6: Platform Requirements Reference Manual
9.	IAWG-ECOA-TR-009	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 7: Approach to Safety and Security Reference Manual
10.	IAWG-ECOA-TR-010	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 8: Software Interface Reference Manual
11.	IAWG-ECOA-TR-011	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 9: Metamodel and XSD Schemas Reference Manual

Table 1 - Table of ECOA references