European Component Oriented Architecture (ECOA®) Collaboration Programme: Architecture Specification Part 2: Definitions

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

Issue: 6

Prepared by
BAE Systems (Operations) Limited and Dassault Aviation

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

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

Contents

0 Introduction v

1 Scope 1

2 Warning 1

3 Normative References 1

4 Definitions 2

Application Software Component 2

Application Software Component Definition 2

Application Software Component Implementation 3

Application Software Component Implementation Description 3

Application Software Component Instance 3

Assembly Schema 3

ECOA Component Development Process 3

ECOA Compliant Platform 3

Composite Component 4

Computing Node 4

Computing Platform 4

Container 4

Container Interface 4

Container Operation 4

ECOA Conversion Layer 5

Deployment Schema 5

Driver Component 5

Dynamic Trigger 5

Early Validation 5

Event 5

Fault Handler 5

Functional Chain 6

Infrastructure 6

Infrastructure Services 6

Insertion Policy 6

Inversion-of-Control 6

Legacy Software Architecture 6

ECOA Librarian 7

Lifecycle Events 7

ECOA Logical Interface 7

Logical System 7
ECOA Module 7
Module Implementation 7
Module Instance 8
Module Interface 8
Module Operation 8
Module Operation Link 8
Module Runtime Lifecycle 8
Module Type 8
OS/Middleware Interface 9
PINFO 9
ECOA Platform 9
Platform Integration Code 9
Properties 9
Protection Domain 9
Quality-of-Service 10
Reactive Execution Model 10
ECOA Reference Platform 10
Request-Response 10
Service 10
Service Definition 10
Service Instance 10
Service Link 11
Service Operation 11
ECOA Software Platform 11
ECOA Specification 11
ECOA Stack 11
ECOA Standard 11
ECOA Standard Working Group 11
ECOA System 12
Trigger Instance 12
User Context 12
ECOA Validation Suites 12
Versioned Data 12
Warm Start Context 13
Wire 13
XML Metamodel 13

5 Abbreviations 13

6 ECOA Terms in Context 14

Figures

Figure 1 Scope of ECOA Terms within a System Implementation 15

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.

1 Scope

This Architecture Specification specifies a uniform method for design, development and integration of software systems using a component oriented approach.

2 Warning

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

3 Normative References

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

Architecture Specification Part 11	IAWG-ECOA-TR-031 / DGT 154934 Issue 6 Architecture Specification Part 11 – High Integrity Ada Language Binding
ISO/IEC 8652:1995(E) with COR.1:2000	Ada95 Reference Manual Issue 1
ISO/IEC 9899:1999(E)	Programming Languages – C
ISO/IEC 14882:2003(E)	Programming Languages C++
SPARK_LRM	SPARK – The SPADE Ada Kernel (including RavenSPARK) Issue 7.3

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 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 referred to as a Component where its meaning is readily apparent from the context.

4.2

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**.

4.3

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 between stakeholders

4.4

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:
  - the provided Service Operations
  - any required Service Operations
  - any **ECOA Module** to **ECOA Module** interactions internal to the **ASC**.

4.5

Application Software Component Instance

An instance of an **Application Software Component Implementation**, which will be independently deployed.

4.6

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**.

4.7

ECOA Component Development Process

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

4.8

ECOA Compliant Platform

An **ECOA Platform** which is fully compliant with the **ECOA Standard.**

However, this does not preclude the fact that an ECOA compliant platform may simultaneously host both ECOA and non ECOA applications.

4.9

Composite Component

Composite Components resemble **ASCs** externally, but are composed from **ASCs**, which may in turn be **Composite Components**.

4.10

Computing Node

Single processor element onto which **Protection Domains** and hence **ECOA Modules** are allocated.

4.11

Computing Platform

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

4.12

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**.

4.13

Container Interface

The API made available to the **ECOA Module** providing the ECOA defined **Container Operations**.

See also **Module Interface**.

4.14

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.

4.15

ECOA Conversion Layer

Software that adapts Non-ECOA Applications to make it compatible with the **ECOA Logical Interface** (ELI).

This enables the legacy software to interact with the rest of an **ECOA System**.

4.16

Deployment Schema

An allocation of **ECOA Modules** to **Protection Domains**, **Protection Domains** to **Computing Nodes**. Also specifies the logging policy to be applied.

4.17

Driver Component

An **ASC** that provides **Services** to communicate with hardware and/or software using interfaces not defined by ECOA.

4.18

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**.

4.19

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.

4.20

Event

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

4.21

Fault Handler

An entity being responsible for triggering recovery procedures for Infrastructure errors. This entity can either be implemented within the **Infrastructure**, or be implemented as an **ASC** depending on **ECOA Platform** procurement requirements.

4.22

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

4.23

Infrastructure

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

4.24

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**.

4.25

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.

4.26

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.

4.27

Legacy Software Architecture

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

4.28

ECOA Librarian

The ECOA Librarian is responsible for:

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

4.29

Lifecycle Events

Events issued by the **Infrastructure** to manage the lifecycle of **ECOA Modules**.

4.30

ECOA Logical Interface

The standardised message protocol that defines how separate **ECOA Platforms** interact across 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.

4.31

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.

4.32

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**).

4.33

Module Implementation

The software implementing an **ECOA Module**. This software should be re-entrant. Re-entrancy allows a single copy of module implementation to be used concurrently by many module instances without them interfering with each other.

4.34

Module Instance

An instance of an **ECOA Module**.

4.35

Module Interface

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

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

See also **Container Interface**.

4.36

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**.

4.37

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**

4.38

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** is managed by the **Infrastructure** using the **Lifecycle Events.**

4.39

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 **Fault Handler**.

4.40

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.

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 Modules** 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 Modules** on a **Computing Platform**.

This includes **Container** instances together with code for managing the **Protection Domains**, **Computing Nodes** and Platform.

4.44

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.

Properties are set statically at design-time.

4.45

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).

4.46

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**.

4.47

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.

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

4.48

ECOA Reference Platform

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

4.49

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.

4.50

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.

4.51

Service Definition

The definition of a **Service**, including:

- **Service** identifier
- Set of **Service Operations**

Service Definitions will be referenced in an **Application Software Component Definition** to specify provided and required **Services**.

4.52

Service Instance

An instance of a **Service**.

The same **Service** may be provided by multiple instances of an **ASC** or by different **ASCs**.

4.53

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 by an **ASC**, may have multiple **Service Links**.

4.54

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 a **Request-Response**, **Event** or **Versioned Data**.

4.55

ECOA Software Platform

The software that implements the **Infrastructure**.

4.56

ECOA Specification

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

4.57

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**.

4.58

ECOA Standard

A formal published subset of the **ECOA Specification**.

4.59

ECOA Standard Working Group

The ECOA Standard Working Group is responsible for:

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

4.60

ECOA System

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

4.61

Trigger Instance

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

4.62

User Context

An optional data object specific to a **Module Instance**, which together with Warm Start Context is the state data defining an instance of an ECOA module. This allows the **ECOA Module** to be instantiated more than once if the **ECOA Module** 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 **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 **user context** is defined by language-specific bindings.

4.63

ECOA Validation Suites

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

4.64

Versioned Data

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

With access control, the **ECOA Infrastructure** ensures a concurrency-safe read-write paradigm by 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.

Without access control, the data repository is accessed directly by all readers and writers. Sharing data without access control is possible only between **Module Instances** of the same **Application Software Component Instance** and provided these **Module Instances** are deployed in the same **Protection Domain**. Concurrent access between these **Module Instances** must be managed at application level under the responsibility of the Component Supplier.

Access control is enabled by default for each **Module Operation Link** that uses the **Versioned Data** mechanism, unless explicitly disabled by the Component Supplier.

4.65

Warm Start Context

An optional data object specific to a **Module Instance**, which together with User Context is the state data defining an instance of an ECOA module. This allows the **ECOA Module** to be instantiated more than once (if the **ECOA Module** maintains an internal state) and 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 **Module Instance** (**Infrastructure-**level technical data),
- by the **Module 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 **Module Instance** itself.

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

4.66

Wire

An SCA term which corresponds to a Service link in the Assembly Schema.

4.67

XML Metamodel

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

5 Abbreviations

APEX	Application Express
API	Application Programming Interface
ASAAC	Allied Standards Avionics Architecture Council
ASC	Application Software Component
ECOA	European Component Oriented Architecture. ECOA® is a registered trademark.
ELI	ECOA® Logical Interface
OS	Operating System
PINFO	Persistent Information
POSIX	Portable Operating System Interface
SCA	Service Component Architecture
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.

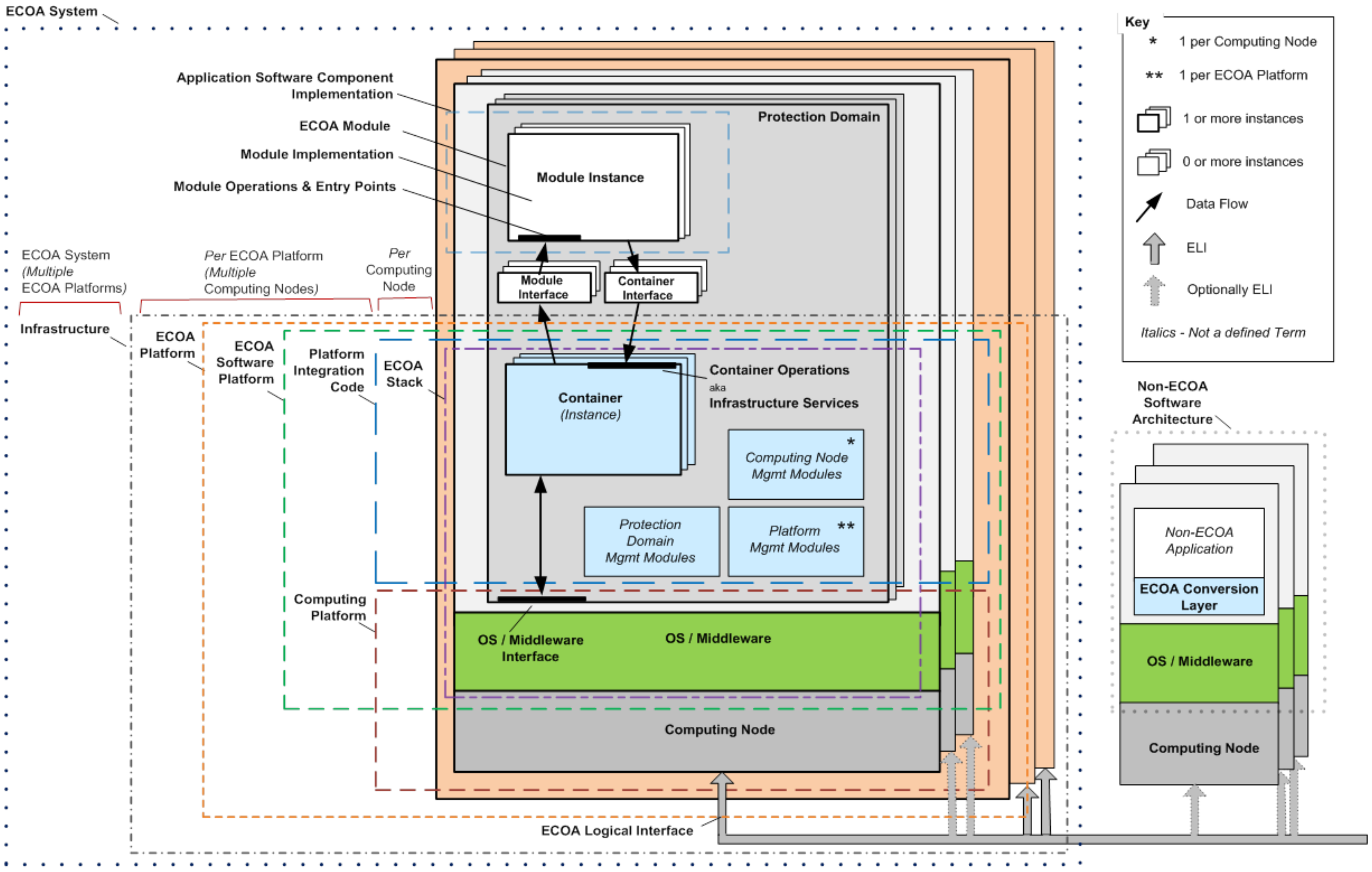

Figure 1 Scope of ECOA Terms within a System Implementation