



AEROSPACE INFORMATION REPORT

AIR5315™**REV. A**

Issued	1998-04
Reaffirmed	2011-04
Stabilized	2018-05

Superseding AIR5315

Overview and Rationale for GOA Framework Standard

RATIONALE

The Technical Committee which originally created this document no longer exists.

STABILIZED NOTICE

This document has been declared "Stabilized" by the SAE AS-2, Embedded Computing Systems Committee and will no longer be subjected to periodic reviews for currency. Users are responsible for verifying references and continued suitability of technical requirements. Newer technology may exist.

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2018 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
<http://www.sae.org>

SAE WEB ADDRESS:

Full document details: [SAE AIR5315A](#)

SAE values your input. To provide feedback on this Technical Report, please visit <http://standards.sae.org/AIR5315A>

Documents
Document Sharing Hub

FOREWORD

This SAE Aerospace Information Report (AIR) was developed as a supplement to SAE AS4893 Generic Open Architecture (GOA) Framework standard. This AIR provides an overview and rationale of the GOA Framework. This AIR was prepared under the direction of:

Chuck Roark Chairman, AS-5 Committee
Bosch Telecom, Inc.
P.O. Box 742466
Dallas, TX 75374

TABLE OF CONTENTS

1. SCOPE 3

1.1 Purpose 3

1.2 Intended Audience..... 3

1.3 Document Structure..... 4

2. REFERENCES 4

2.1 Standards 4

2.2 Specifications..... 4

2.2.1 Government Specifications 4

2.3 Other Publications 5

3. GOA FRAMEWORK DESCRIPTION 5

3.1 Purpose of GOA Framework Structure 5

3.2 Baseline Document Overview - SGOAA..... 6

3.3 Basic Concepts and Perspective 6

TABLE OF CONTENTS (Continued)

4. GOA FRAMEWORK DEVELOPMENT APPROACH.....	16
4.1 Evolution from SGOAA	16
4.2 Issues and Resolution.....	16
4.3 Types of Companies and Agencies Involved	19
5. EXAMPLES.....	19
5.1 Pi-Bus Mapping to GOA Framework.....	19
5.2 GOA Stack Definition and Interface Control Documents (ICDs).....	20
5.2.1 Application Interfaces.....	20
5.2.2 System Services Interfaces	22
5.2.3 Resource Access Services and Physical Resources Interfaces.....	22
6. COMPARISON TO OTHER MODELS.....	22
6.1 POSIX Comparison.....	23
6.2 ISO OSI Comparison	23
6.3 TAFIM Comparison	24
6.4 PCMCIA Comparison.....	26
7. COMPLIANCE	30
APPENDIX A	31
A.1 Background.....	31
A.2 Objectives	31
A.3 Approach.....	33
A.4 Results	35
FIGURE A1- Mission and Operational Requirements for all Vehicles	35
FIGURES	
FIGURE 1 - GOA Framework	7
FIGURE 2 - Different Ways of Viewing GOA Framework	11
FIGURE 3 - Example Inter-Backplane Communication	15
FIGURE 4 - Recursion Using the GOA Framework.....	15
FIGURE 5 - Example GOA Stack and ICDs	21
FIGURE 6 - Comparison of GOA Framework with POSIX and OSI Reference Models	23
FIGURE 7 - DoD Technical Reference Model (from TAFIM v3.0).....	25
FIGURE 8 - PCMCIA Architecture.....	28
FIGURE 9 - Mapping of PCMCIA Architecture to GOA Framework - Alternative 1	28
FIGURE 10 - Mapping of PCMCIA Architecture to GOA Framework - Alternative 2	30

1. SCOPE:

This document is a companion document to SAE AS4893 “Generic Open Architecture (GOA) Framework Standard” and provides an overview and rationale for SAE AS4893. The GOA Framework establishes an architectural framework to assist in the application of open systems interface standards to the design of specific hardware/software systems. The GOA Framework standard is intended for use by both system designers and system implementers in the development of open systems architectures. It is intended that domain specific guidelines be developed to provide clarification for application of the GOA Framework.

The Generic Open Architecture (GOA) Framework was initially developed by the SAE to provide a framework which could be used to classify interfaces needed in airborne avionics systems. At the time of the development of the GOA Framework, development of such a classification was considered crucial to the application of open systems standards to military avionics. However, it was recognized during the development of the GOA that the GOA Framework might be applicable to domains other than avionics. For that reason the framework is entitled Generic Open Architecture instead of the original name, Generic Open Avionics Architecture (GOAA).

This document is one of several documents contained within the GOA document set. “Introduction to the GOA Family of Document Set”, SAE AIR5314, provides an overview of the GOA Family of Documents. SAE AIR5314, the GOA Framework Standard, and this document make up the domain independent set of documents within the GOA Family of Documents. These domain independent documents are augmented with domain specific sets of document. The Domain specific sets of documents consist of a Catalog of Preferred Standards document, Rationale for Catalog of Preferred Standards Document, and GOA Guidance Document for each domain. The first domain specific set of documents within the GOA Family of Documents being developed is for the aviation domain.

1.1 Purpose:

The purpose of this document is two-fold:

1. provide detailed explanation of the key concepts contained within the GOA Framework standard.
2. provide rationale for key concepts on which the GOA Framework standard is based.

1.2 Intended Audience:

The intended audience for this document is anyone that has an interest in using or understanding the GOA Framework standard. This typically would include system engineers, hardware engineers, software engineers, engineering managers, project managers, academia, and procurement personnel. This document assumes the reader is familiar with the GOA Framework, SAE AS4893.

1.3 Document Structure:

This document is organized as follows:

- Section 1: Scope - provides an introduction and purpose to this document.
- Section 2: References - provides documents referenced within this document.
- Section 3: GOA Framework Description - provides a description of the GOA Framework.
- Section 4: GOA Framework Development Approach - provides an overview of how the GOA Framework was developed.
- Section 5: Examples - provides several examples of the application of the GOA Framework.
- Section 6: Comparison With Other Models - compares the GOA Framework with other well-known reference models.
- Section 7: Compliance - discusses issues of compliance with the GOA Framework.
- Appendix A: Overview of the Space Generic Open Avionics Architecture (SGOAA) – the baseline documents for the GOA Framework.

2. REFERENCES:

2.1 Standards:

- | | |
|-------------|--|
| ISO | International Organization for Standardization 7498: Information Processing Systems - Open Systems Interconnection - Basic Reference Model, 1984 |
| POSIX91 | "Draft Guide to the POSIX Open Systems Environment", P1003.0/D14, IEEE Computer Society, November 1991 |
| SAE AS4893 | "Generic Open Architecture (GOA) Framework", SAE AS4893, SAE AS5, Jan 1996 |
| SAE AIR5314 | "Introduction to the GOA Family of Document Set", SAE AIR5314, SAE AS5, [TBD] |

2.2 Specifications:

2.2.1 Government Specifications:

- | | |
|---------|---|
| TAFIM97 | "Technical Architecture Framework for Information Management", Defense Information Systems Agency, Version 3.0, April, 1997. |
| SGOAA | [SGOAA]Wray, R.B. and Stovall, J.R., "Space Generic Open Avionics Architecture Standard Specification", LESC-30354-C (NASA CR-188290), Lockheed Engineering & Sciences Company, June 1994 |

2.3 Other Publications:

BOE91	Flanagan, Rich and Van Ausdal, Art, "SATWG Flight Data System Architecture Specification Outline" briefing, 25 October 1991
GD90A	General Dynamics "Space Avionics Requirements Study", 21 October 1990 as briefed to the SATWG
Levine	Levine, L., PCMCIA Primer, M&T Books, 1995
PRU90	Pruett, D., "Avionics Software Open System Environment Reference Model", JSC, March 1990.
WRA91	Wray, R.B., "Requirements Analysis Notebook for the Flight Data Systems Definition in the Real-time Systems Engineering Laboratory (RSEL)," Job Order 60-430 for the JSC, NASA-CR-185698, LESC- 29702, December 1991.
WRA93	Wray, R.B. and Stovall, J.R., "Space Generic Open Avionics Architecture (SGOAA) Reference Model Technical Guide", Job Order 60-430, Contract NAS9-17900 for the JSC, NASA CR-188246, LESC-30347, April 1993. (Note: some of the figures in this reference have been superseded by the figures in revision C of the standard.)

3. GOA FRAMEWORK DESCRIPTION:

3.1 Purpose of GOA Framework Standard:

The purpose of SAE AS4893 is to provide a framework to identify interface classes for applying open systems to the design of a specific hardware/software system. This framework is used to define an abstract architecture based on a generic set of interfaces. The generic set of system interfaces facilitate identification of critical interfaces.

It is intended that the GOA Framework be specialized for varying domains. A domain specific implementation of the GOA Framework will increase the chance that components/capabilities produced independently will "plug and play" and evolve affordably. The GOA Framework provides a basis for commonality for both vendors and users of components/capabilities. Application of the GOA Framework imposes constraints on implementations that increase the likelihood that independently produced products will interoperate.

Application of the GOA Framework together with the appropriate open system interface standards is expected to provide the following benefits to future programs:

- Provide the basis for establishing a set of specifications, standards and procedures that will become common to all elements of a major system.
- Ensure that future systems can be upgraded and maintained with minimal redesign impact to the existing system by identifying the interfaces required to enable modular replacement of hardware and software.
- Promote availability of multiple sources of needed software and hardware, especially commercial off-the-shelf components.

3.1 (Continued):

- Provide a pool of hardware and software modules for multiple program commonality and re-use.
- Insure access to the architecture and its design documentation for any vendor or agency desiring to propose new uses and applications, and to facilitate competition to contain cost growth.
- Aid in the definition of an open architecture for the system.

3.2 Baseline Document Overview – SGOAA:

The Space Generic Open Avionics Architecture (SGOAA) was the base document for the GOA Framework. Appendix A provides a synopsis and rationale for the SGOAA.

3.3 Basic Concepts and Perspective:

This section presents basic concepts and perspectives on the GOA Framework. The GOA Framework is depicted in Figure 1 for reference. A definition of the GOA Framework can be found in SAE AS4897. The overriding reason for the definition of the GOA Framework was to provide an architectural framework to aid in the identification of critical components and the interfaces between these critical components. The identification of such interfaces supports reuse and seamless upgrades that address obsolescence, the need for increased functionality, and leveraging emerging technology.

A system consists of a topology of nodes (or modules), some of which may be used for processing, i.e., containing one or more processors. A node is a self-contained assembly of electronic components and circuitry. A typical avionic system is comprised of multiple nodes, each individually addressable via a backplane bus. An instantiation of the framework associated with a particular processing node is referred to as a "GOA stack". As will be discussed later, there are views of the system in which a "full" GOA stack view is not appropriate - what is appropriate are the 1L-4L logical peer-to-peer (same level) interfaces and the 1D direct physical interface between GOA stacks. This latter view is a type of "black box" view of a system while a full GOA stack view is appropriate when viewing a portion of the system that contains a processor together with the software that executes on the processor.

The GOA Framework is defined by nine interface classes and four GOA layers. One of the GOA layers, the Systems Services Layer, contains two types of sublayers - Operating System and eXtended Operating System (XOS). The following provides an overview and reasons for the chosen classes and layering.

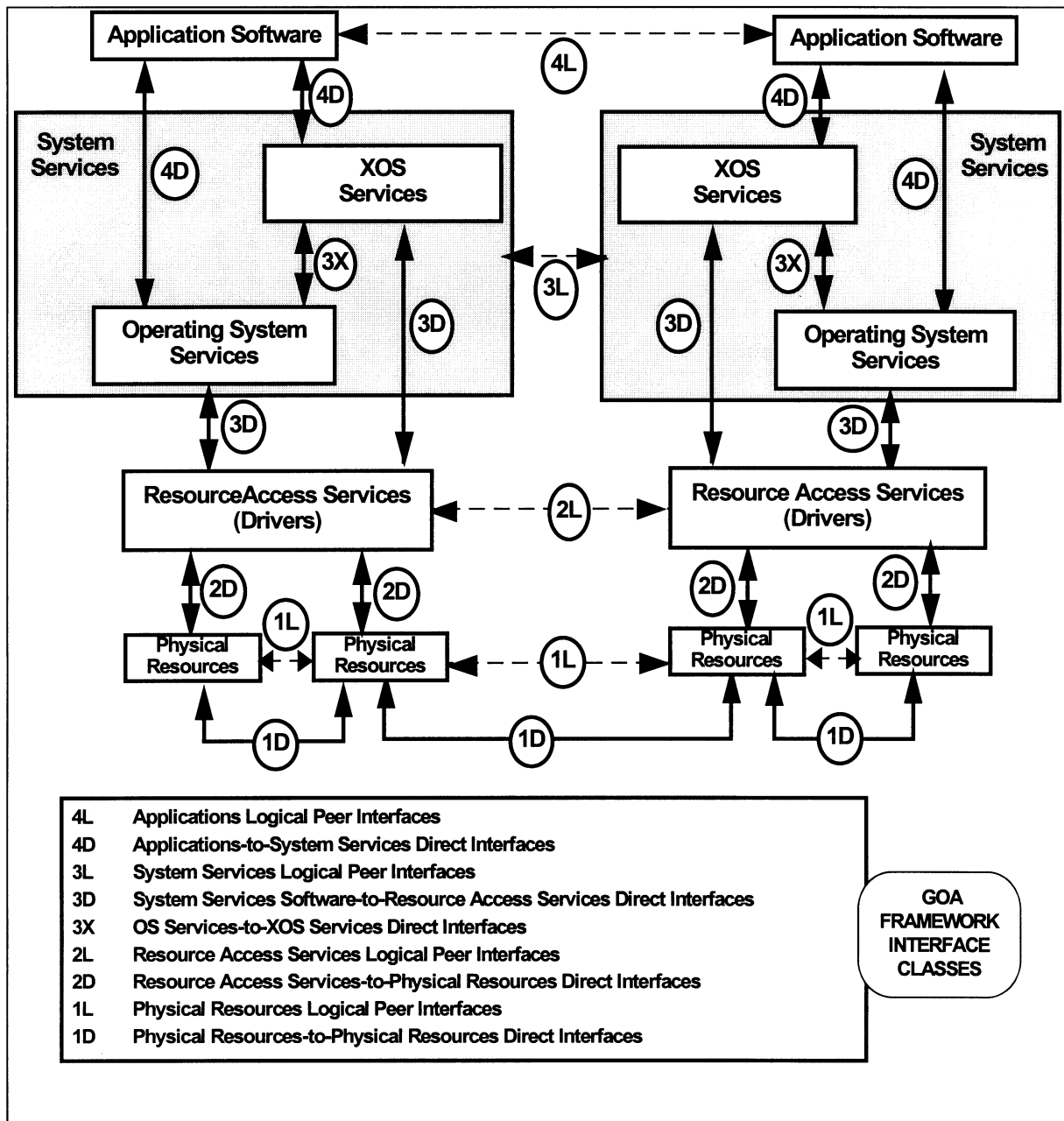


FIGURE 1 - GOA Framework

3.3 (Continued):

The GOA Framework interfaces are based on two types of interfaces: logical and direct interfaces. A logical interface defines requirements for peer-to-peer interchange of data. A logical interface identifies senders and receivers, the type of data to be exchanged, when the data is to be exchanged, and formats if appropriate. A direct interface identifies the characteristics of the information transfer medium/facility. No information transfer actually takes place as part of a logical interface; instead, information transfer takes place as part of one or more direct interfaces. Simply stated, logical interfaces define what information is transferred, and direct interfaces define how the information is transferred.

One example of a logical interface is the definition of interface requirements for exchange of latitude and longitude information from a Navigation application to a Stores Management application. The logical interface defines information that would become the basis for an Interface Control Document (ICD) for Navigation to Stores Management communication. Another example of a logical interface is the message header used by a Communications XOS service to encapsulate message data. This message header contains information that is interpreted by the peer Communications XOS service entities in order to make a message transfer occur. For example, logical designations of the message destination(s) and type might appear in such a header. The definition of the message header is a logical interface that tells the two peer Communications XOS services how to initialize and interpret the message header when a message transfer actually takes place.

Direct interfaces specify how components in one layer invoke services in the layer immediately below it within the same GOA stack or how the physical layer in one GOA stack "physically" interfaces with the physical layer of another GOA stack (e.g. between processor modules). Logical interfaces occur as the result of invocations of direct interfaces that support communications. For example, consider the latitude and longitude logical interface example presented in the previous paragraph. If message-passing communication is used, the actual transfer of the latitude and longitude information would occur as the result of the Navigation application invoking a "send" operation in the System Services layer. The invocation of the send routine would cause the system to physically transfer the data to make it available to the Stores Management application. The Stores Management application would then invoke a "receive" System Services operation to actually get the message containing latitude and longitude information. These send and receive routines are example of direct interfaces. However, there are also direct interfaces that have nothing to do with communications - these direct interfaces just invoke a service such as reading the time, allocating or deallocating memory, etc. The interface specification of a component within a GOA layer designates the direct interfaces to that GOA layer. For example, POSIX defines a standard Operating System component interface; bus standards define the electrical characteristics of the bus.

Thus, logical interfaces are considered important to define communication and synchronization requirements between peer-to-peer components. Direct interfaces are important for defining the interface to services (including communication services that make logical interfaces possible.)

Before one can define interfaces, one must know what is being interfaced. The following provides an overview of these components - the GOA layers and some rationale for choosing the GOA layers.

3.3 (Continued):

The Application Software layer contains the platform level application specific code (or function). The application layer within a specific GOA stack can contain multiple software application components. The requirements for communication between different application components (whether within the same or different GOA stacks) is a logical interface. As noted above, the communication occurs through direct interfaces. For message passing paradigms, the communication occurs by invoking message-passing services. If the interface is through shared memory (instead of message passing), the direct interface is simply a load or store operation, and the logical interface is the association of the address and size of the data. The 4D interfaces, i.e., Application Program Interfaces (APIs), are critical since they are important from a reuse perspective and a seamless upgradeability point-of-view. APIs support reuse by having the same Systems Services implemented the same way on different hardware platforms. Seamless upgrades can be supported by 4D interfaces since they abstract the Applications Software from the lower GOA layers. In particular, the Systems Service layer abstracts the Applications Software from changes in the Physical Resources layer, such as processor modifications/upgrades.

The Physical Resources layer is the bottom-most layer and provides the interface between different physical components. The Physical Resources layer within a specific GOA stack can contain multiple Physical Resource components. The following are examples of Physical Resource components that might appear in the same GOA layer: Pi-Bus interface unit for intermodule data communication, TM-Bus interface unit for intermodule test and maintenance, IEEE-488 interface unit for debug, and CPU device for program execution. The 1D direct and 1L logical interfaces are clearly critical interfaces from a black box point-of-view. These interfaces include the physical and data link level definitions of busses where the electrical/mechanical requirements are direct (1D) interfaces and how the bits are interpreted are logical (1L) interfaces.

The System Services Layer is provided to encapsulate common services accessed by the Application Software Layer. The Operating System (OS) Service is the minimal component within the System Services Layer. XOSServices designate other components contained within the Systems Services Layer besides the OS. This layer is important since it allows an abstract view of common services required by applications. As discussed in section 4.2 Issues and Resolution, the 3X interface provides an interface between XOS and the OS services to provide either capabilities not included in the OS 4D interface (i.e., privileged operations) or optimized 4D interface capabilities. The optimization is for performance and could occur through an interface that has reduced checking or has less call overhead (such as avoiding a system trap). The logical 3L interface provides System Service Layer peer-to-peer interface communications interfaces. For example, the interpretation of the fields within the communications header added to a message between two applications is an example of a 3L interface. It should be noted that, even though not explicitly mentioned in SAE AS4893, an XOS Service may use 4D interfaces to invoke the Operating System or other XOS Services resident within the same GOA stack.

3.3 (Continued):

The Resource Access Layer contains components that directly access the hardware. This layer includes classical device drivers and memory mapped I/O definition translation, for example. The GOA Framework supports the idea that this layer abstracts the remaining layers from the Physical Resources. The 3D interface is provided between the System Services Layer and the Resource Access Layer to allow the non-processor dependent (i.e., non-I/O dependent) algorithms within the Systems Services Layer to be abstracted from specific Physical Resource Layer implementations. This supports OS and XOS portability between target platforms. The Joint Integrated Avionics Working Group (JIAWG) Input/Output Built-In-Test Description Specification (IOBIDS) specification and the Uniform Device Driver Interface (UDI) interface are example of 3D interfaces. The 2D interface defines the direct interface between the Resource Access Services Layer and the Physical Resource Layer. This is typically the direct interface between software and hardware. Memory mapped I/O definitions and processor instruction-set architecture (ISA) definitions are two examples of a 2D interface. The 2L interfaces are Resource Access Services peer-to-peer interfaces. An example of a 2L interface is the interpretation of error information returned by the Resource Access Service layer in the destination GOA stack for a data communication activity between two stacks.

The GOA Framework's inherent flexibility allows a system to be viewed from several different perspectives:

- Application point-of-view
- Systems software point-of-view
- "Classical" hardware point-of-view.

Except for 1D direct physical interfaces, the direct interfaces are between contiguous GOA layers within the same GOA stack residing on the same hardware module. Logical interfaces can be between peer GOA layers residing:

- within the same hardware module
- in different hardware modules within the same backplane
- in different hardware modules within different backplanes within the same system
- in different hardware modules within two subsystems, e.g., Mission Computer and Vehicle Management System (VMS)
- between different hardware modules in different systems (e.g., aircraft and a ground based system).

As shown in Figure 2, only the logical interfaces and 1D direct interface are of interest when considering interfaces between GOA layers in different hardware modules. In other words, if one has two entities considered as black boxes (i.e., the internals of each entity are not visible to the other), only the logical interfaces and the direct physical interface are visible. However, when one looks at a complete GOA stack executing on a hardware module, all the GOA layers are visible and hence the direct interfaces between adjoining layers are of interest.