Face 3.1 Enhancements
What Could Possibly Go Wrong?

A White paper jointly authored by Collins Aerospace, LDRA and Lynx Software Technologies

Authors
Jeffry A. Howington, Strategic Pursuits and Advanced Technology, Collins Aerospace
Ehsan Salehi, Field Applications Engineer, LDRA
Arun Subbarao, VP Engineering, Lynx Software Technologies

Why FACE?

It's hard to escape the headlines around Modular Open Systems Approach (MOSA), open standards, and individual initiatives such as those from The Open Group FACE™ Consortium, the creators of the FACE Technical Standard. In 2004, the United States government started a large effort to define and implement MOSA as a requirement within military systems with the definitive document titled *MOSA Principles defined in OSD Open Systems Joint Task Force Program Manager's Guide: A MOSA to Acquisition*. Since then, we've seen a progression in policy guidance that raised the profile of MOSA and its applicability within military systems to enable success on the battlefield while lowering acquisition costs and promoting innovation.

There are various drivers behind this including:

1) The implementation of multi-domain operations.
2) Working within limited budgets while quickly delivering increasing levels of capability into new and enduring weapons' systems.

Eventually this policy has made its way into National Defense Authorization Act language that was first

signed into law for fiscal year 2017 and again for 2021. In turn this has flowed into requirements at the PEO and program manager levels under the *Tri-Services Memo: MOSAs for Our Weapon Systems is a Warfighting Imperative* issued in 2019.

The FACE Technical Standard – as one of the named open standards endorsed by U.S. policy – was recently said by U.S. Army Brigadier General Rob Barrie as "integral to MOSA success by enabling modularity and promoting software reuse." Programs such as the U.S. Army's Future Vertical Lift call out the FACE Standard as a requirement.

UNITED STATES NAVY FIGHTER WEAPONS SCHOOL
PARACHUTE PACKING INSPECTION DATA

Figure 1 - The Collins Aerospace Advanced Flight Deck - utilizing a Modular Open Systems Approach (MOSA) to rapidly deploy new technology upgrades when needed to overcome the latest threats

In today's digitized avionics, most of the functional capability is implemented in software. 70% to 90% of that functionality depends on the selected equipment and how extensively it relies on displays only for the human machine interface. Most of the development and upgrade costs for commercial and military systems are now dominated by that software.

Unfortunately, we have seen plenty of examples where software creates most of the headline grabbing cost and schedule overruns on some development and upgrade programs. The reality is that software development is hard. Testing and safety certifying that software is hard. The need for greater functionality and adapting that software to new situations is constant. And the reality is that the old way of getting that software into aircraft fleets – which traditionally has taken years if not decades – now needs to be completed in weeks or days, if not hours. The FACE Consortium started its work in 2010 focused on developing an approach aimed at improving the modularity and reuse of software with the idea of reducing the cost and time to field avionics capability.

The FACE Approach tackles not just the technical aspects, but the business aspects for multiplatform adoption as well. After all, modularity and standards have been around for a while in the technical realm. Without addressing the business drivers, the result over time will be the use of standards within a single program, a military service, a company, or some other defined boundary which used those standards in their own way. The practice of MOSA in that way becomes more of "my own system approach" rather than what it should be – "modular open system approach."

Those systems in reality ended up being closed, even if they were based on an existing standard. If only one program used it, it was, in effect, a closed system.

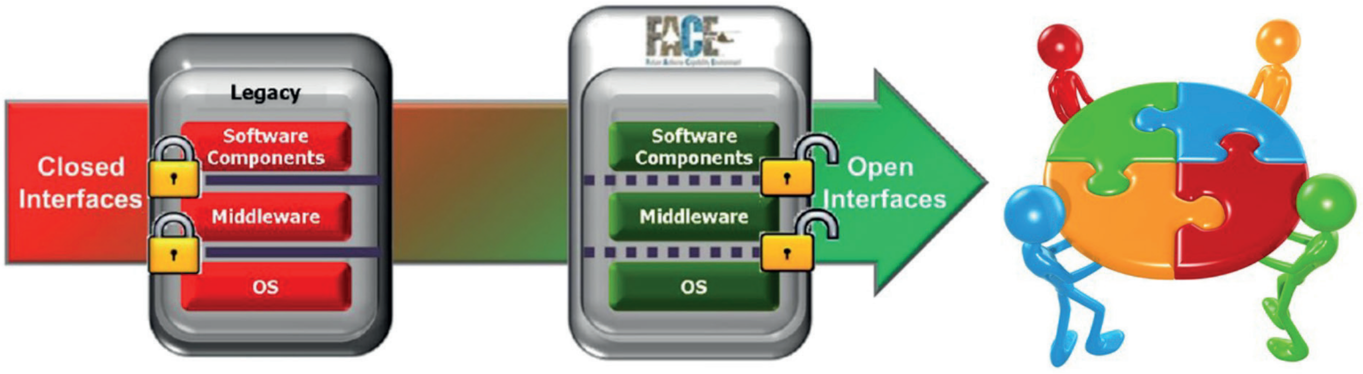


Figure 2 – Transition to open interfaces using the FACE Technical Standard allows greater component reuse[1]

The FACE Consortium addressed that problem by identifying well accepted standards that really mattered – existing standards such as POSIX and ARINC 653 for operating system interfaces, ARINC 661 and OpenGL for graphics interfaces, and so on. They also established an architectural framework by which software components that conform to the FACE Standard can reside. The important aspect of that framework was to abstract the hardware away from those software components so now those components have more ability to be integrated into different systems.

By codifying a modular architecture, standard interfaces, data models, and conformance criteria into a common operating environment and reusable components, we now have the means to share capabilities not only across platforms, but also across the military services and avionics vendors as well.

The business benefits were that customers did not need to redo development of a capability just to place it on a different aircraft. Components from different programs could be reused, and new ones obtained components from a wider variety of suppliers. This increases the competition for delivering innovative solutions.

With open interfaces and the data rights to go with it, an integrator has more understanding about how the component fits within the framework that resides on the aircraft. Framework components such as middleware and operating systems provided by companies like Lynx are hugely important to enable the FACE Approach to work. Those companies need open interfaces to compete effectively.

Modularity and interchangeability - The vision of FACE

In short, this standardization of approaches for using open standards within military avionics systems promises to lower implementation costs, accelerate development, ensure robust architecture and consistently high-quality software implementation, and maximize opportunities for reuse. The FACE standard embraces these ideals by providing a modular reference architecture based on segments that can be integrated to meet final system requirements.

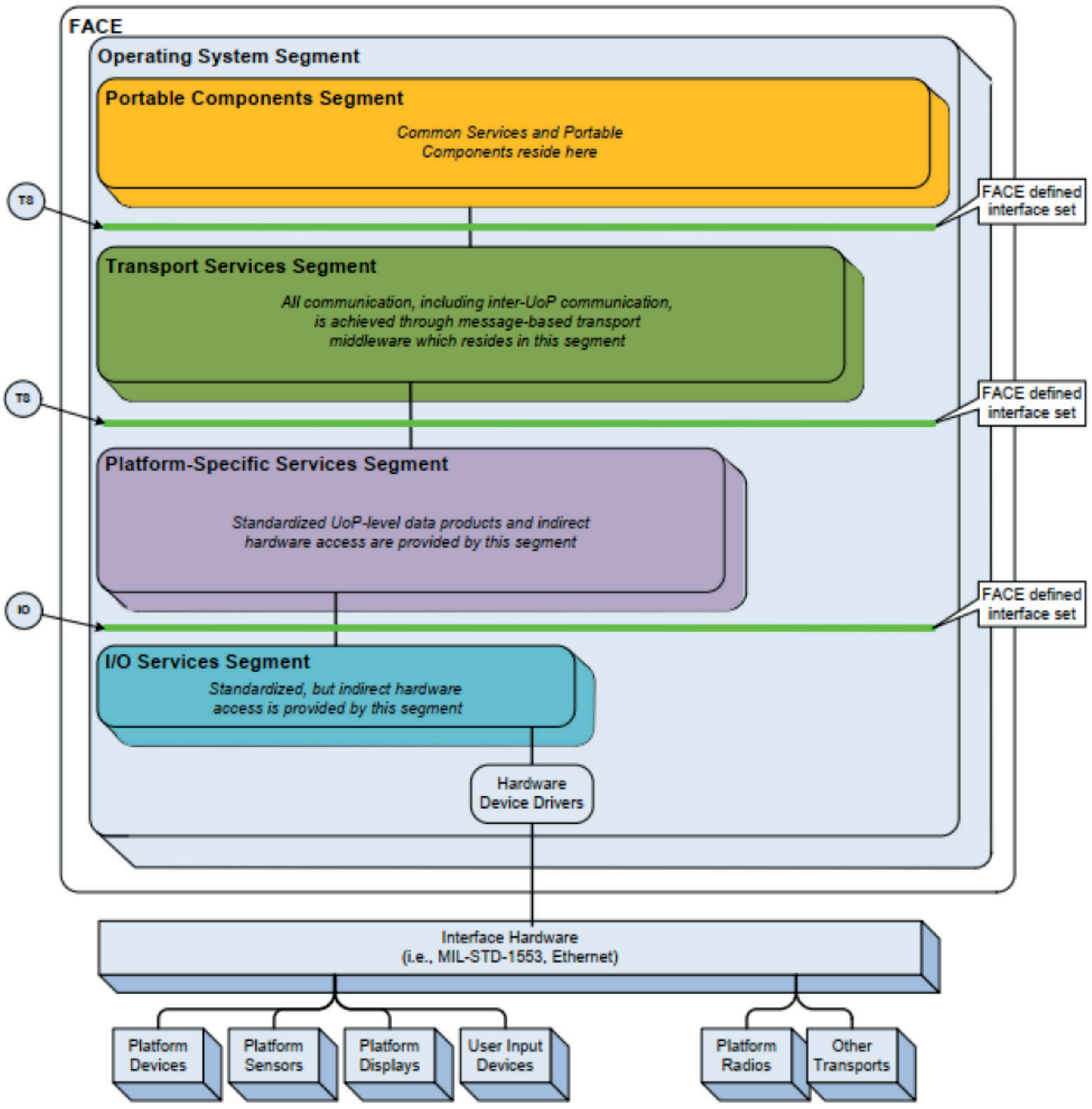

FACE
Operating System Segment
Portable Components Segment
Common Services and Portable Components reside here
TS
FACE defined interface set
Transport Services Segment
All communication, including inter-UoP communication, is achieved through message-based transport middleware which resides in this segment
TS
FACE defined interface set
Platform-Specific Services Segment
Standardized UoP-level data products and indirect hardware access are provided by this segment
IO
FACE defined interface set
I/O Services Segment
Standardized, but indirect hardware access is provided by this segment
Hardware Device Drivers
Interface Hardware (i.e., MIL-STD-1553, Ethernet)
Platform Devices
Platform Sensors
Platform Displays
User Input Devices
Platform Radios
Other Transports

Figure 3 - FACE Architectural Segments

The content of each of those segments can and will vary depending on the preference of the system architect and the demands of the system under development. Despite those variations, modularity is ensured because the FACE standard defines the logical interfaces between the segments.

Each segment consists of one or more components. Each of those components must be shown to be fully conformant to the FACE Standard applicable to the segment to which it contributes.

A FACE Conformance Test Suite is used for that purpose, and a certification of conformance is awarded following successful reviews by a FACE Verification Authority and a FACE Certification Authority, and the registration of that success on the FACE Registry with the FACE Library Administrator.

Any component that has been certified in this way is known as a “Unit of Conformance” or “UoC”.

FACE Operating System Segment

For example, foundational system services reside in the FACE Operating System Segment (OSS). These services are provided by OSS UoCs and include the control of access to the computing platform, support for the execution of all FACE UoCs, and the hosting of operating systems interfaces. As a leading provider of real-time operating systems and separation kernel hypervisors, the OSS is a primary area of focus for Lynx Software Technologies.

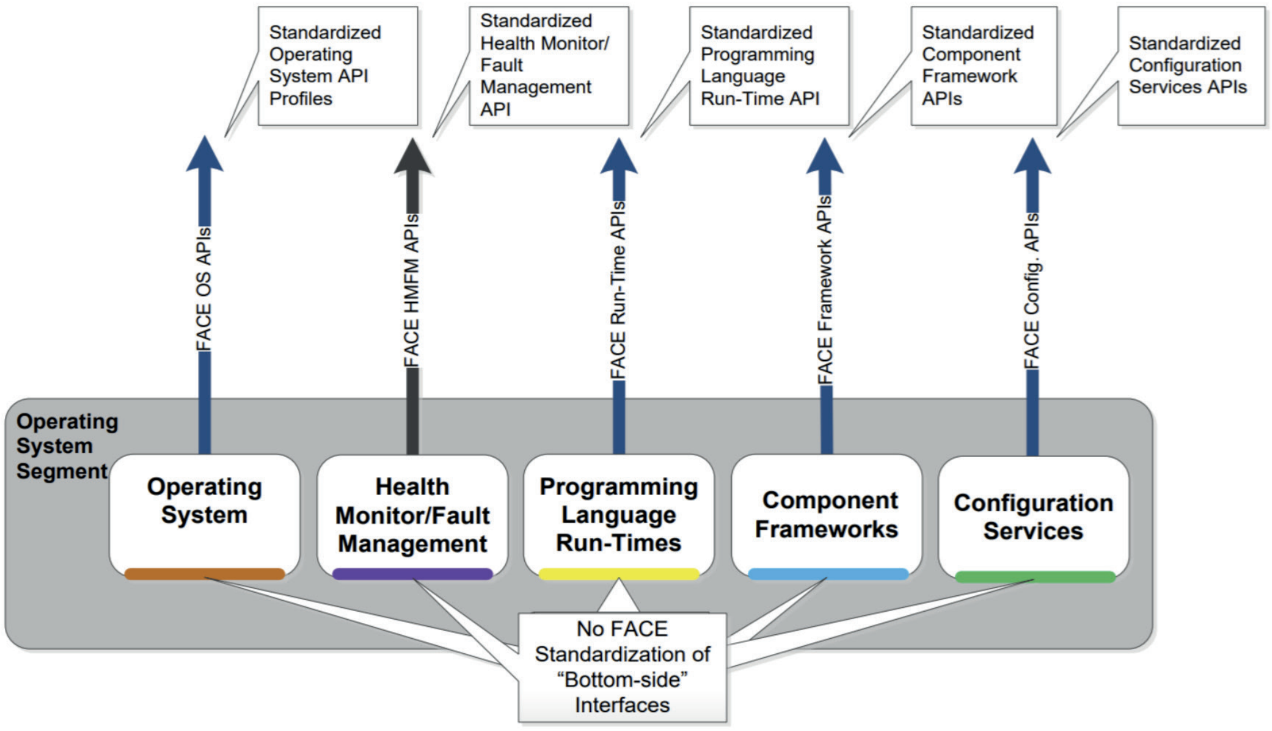

Standardized Operating System API Profiles
Standardized Health Monitor/ Fault Management API
Standardized Programming Language Run-Time API
Standardized Component Framework APIs
Standardized Configuration Services APIs
FACE OS APIs
FACE HMFM APIs
FACE Run-Time APIs
FACE Framework APIs
FACE Config. APIs
Operating System Segment
Operating System
Health Monitor/Fault Management
Programming Language Run-Times
Component Frameworks
Configuration Services
No FACE Standardization of "Bottom-side" Interfaces

Figure 4 - Operating System Segment with FACE

The OSS is where the foundational element of FACE conformant systems is laid out. The platform software that resides directly on hardware runs here. The benefit of writing this code using FACE APIs is that it becomes far easier to migrate this code between different systems and different hardware. When FACE standard was being defined, the consortium chose to harness existing standards such as POSIX and ARINC, both of which have withstood the tests of time. Note, that there are different profiles associated with the OSS, namely:

1. **Security Profile:** Most restricted Application Programming Interfaces (APIs) and, therefore the smallest attack service
2. **Safety Profile:** A superset of the security profile, intended for applications that require safety certification
3. **General-Purpose Profile:** Widest set of APIs to support rich applications that do not need a real-time or deterministic response

Developers pick the profile that best addresses the need for their application.

It is important to note that both the Safety and Security profiles include time and space partitioning. This originates from the ARINC 653 standard and delivers hard isolation between applications running on a platform. This means that a problem in one area of a system remains sandboxed in that area and doesn't cascade into other areas.

The FACE standard has gone through several iterations. For versions 1.x and 2.x of the standard, the primary focus was on ensuring the right subset of APIs were provided to the correct profiles. As things have evolved, additional interfaces have been added, and this is why we feel that the inclusion of Life cycle management, Component State Persistence (which allows data to be stored and retrieved) and Transport Protocol Module (allows movement of data between TSS residing on different domains or different hardware), combined with the definition of Abstract classes for FACE Interfaces which simplifies the creation of object oriented designs are strong reasons for OEMs to migrate to version 3.x of the specification.

Over the last ten years, there has been considerable advances running multiple operating systems on a single processor through CPU virtualization. While popularized and universally adopted in the IT world of computing, the underlying hardware has features that are just as appealing to embedded engineers concerned with robustness and predictability.

In traditional platform software design, each processor hosts a single OS kernel responsible for managing memory allocation, execution scheduling, interrupt routing, exception handling, peripheral control, and bus multiplexing. But now, virtualization enabled multicore hardware can accommodate many kernels, with each kernel allocated subsets of resources of varying types and sizes. Therefore, multiple independent software runtimes can be implemented on a single device without the interference of a common kernel and consequently common mode failure hazards. Such capabilities enhance fundamental architectural properties relating to safety and security concerns.

From a security perspective, the use of built-in CPU virtualization features to isolate hardware security functions, and separate application runtime services from hardware control interfaces, goes a long was in assuring system robustness. Such design techniques eliminate commonly exploited threat vectors that result in security policy bypass, privilege escalation, and loss of CPU control altogether.

From a safety perspective, upholding threshold design principles—predictability, integrity, and high availability etc. —is greatly aided using virtualization partitioning features such as:

- DMA channel isolation
- Shared last level cache partitioning
- Memory bus bandwidth allocation
- Independent interrupt, event, and exception handling

The ability for software partitions to be fortified and controlled with greater fidelity at hardware levels aligns perfectly with FACE ideals. The diagram below introduces the notion of a “Hardware Partitioning Segment” fulfilled by a hypervisor to the FACE reference architecture. The depiction shows a hypervisor isolating two sets of software on two different CPU cores. Each set is configured with FACE conformant components. Each set of software is granted greater partitioning properties over a single OS hosted design where the device drivers and internal services are separated.

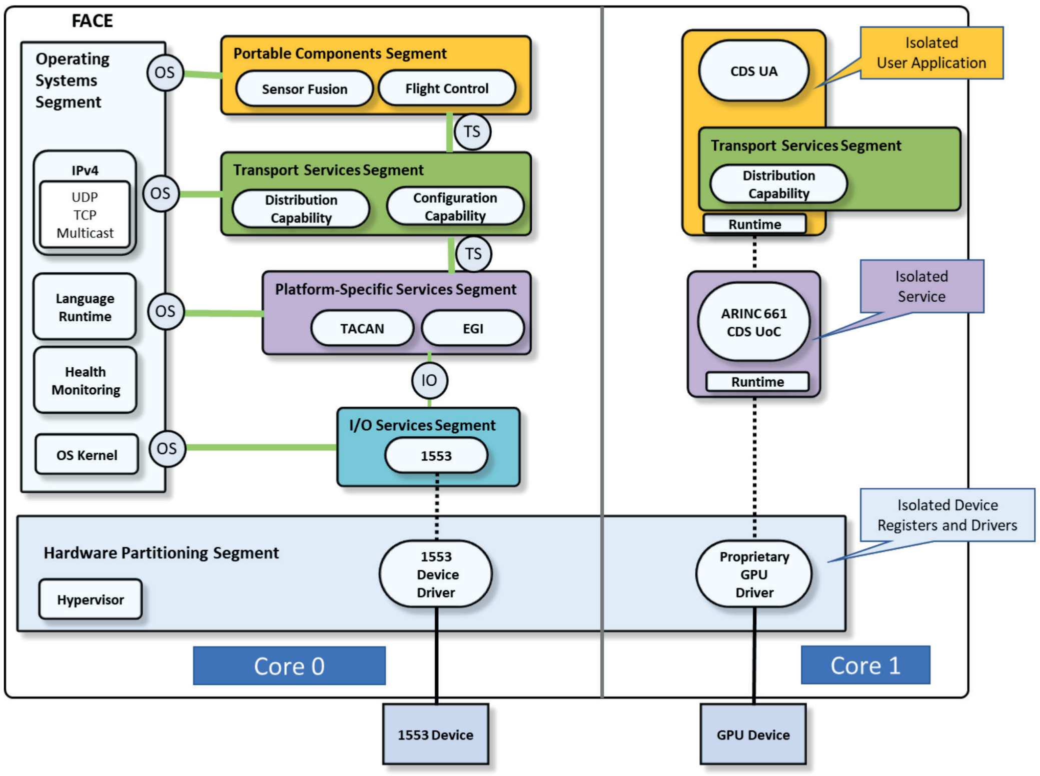


Figure 5 - Example FACE configuration with CPU virtualization assisted hardware partitioning segment

This can be further extended by adding another segment. Introducing another class of technology and layer of software beneath an OS may seem counterproductive to real-time and safety conscious developers wary of complexity. But the hardware partitioning and multiplexing capabilities presented by CPU virtualization offers the opportunity to encapsulate and map subsets of runtime features for critical tasks on a processor that hosts applications simultaneously with inherently richer runtime and service dependencies.

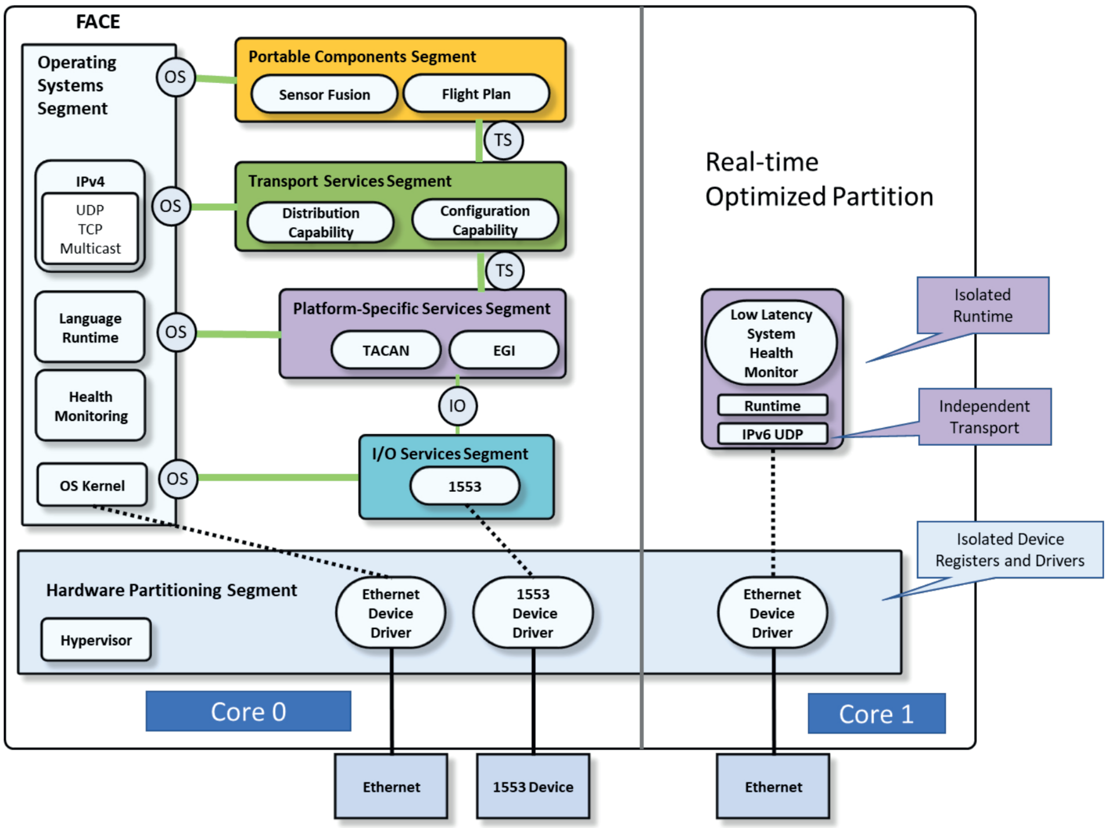


Figure 6 - Example FACE configuration with independent real-time partition

As a tangible example, suppose a vehicle control health monitor application, such as a high frequency majority voting CBIT (Continuous Built-In Test) needed for TMR (Triple Modular Redundancy) error detection must run alongside a flight planning application on a multicore processor. Using a hypervisor-based solution rather than implementing both applications concurrently on the same OS sharing the same network stack and kernel allows the health monitor application (Figure 6) to be:

- Mapped to a separate CPU core
- Mapped to a separate Ethernet MAC
- Run according to an independent thread scheduling algorithm
- Isolated from orthogonal interrupts and blocking semaphores
- Isolated from DMA and OS kernel memory access errors
- Run on an optimized, minimalistic POSIX compliant runtime environment

The result is an ideal scenario for a real-time programmer looking to simplify analysis of worst-case execution time (WCET). Yet at the LRU (Line Replaceable Unit) level, the platform retains the ability to host applications with richer TSS (Transport Services Segment) and OSS (Operating System Segment) capability requirements less concerned about timing and integrity hazards.

Military programs are often stuck with Board-Support Package (BSP) Non-Recurring Engineering (NRE) costs that could be avoided if internal platform software were more portable. Low-level code modules (particularly drivers) are notoriously problematic in providing valued properties of reuse and interoperability.

Standardizing OS internal kernel interfaces is not practical due to their unique design and (in many cases) proprietary nature. However, several classes of device drivers that are naturally independent from core services and require minimal OS feature support (such as the file system) can be isolated by a hypervisor and integrated with applications over standard Inter-process Communication (IPC) interfaces.

It is demonstrable that devices can be controlled independently from operating systems and integrated with other components without embedding proprietary OS dependencies. Consider an OpenGL UA application that simply needs drivers with access to the GPU device interface. Or consider a self-contained MIL-STD-1553 Service with TSS compatible I/O interfaces made available to PCS (Portable Component Segment) applications.

Instead of relying on OS implementations of resource mapping and IPC transports, the TSS layer and local application runtime software can have sufficient capabilities to locate dependent modules and integrate with the use of standard hypervisor provided interfaces and services. Such an approach can even follow the FACE Unit of Conformance packaging requirements.

While the focus here has been on the OSS, Figure 7 shows how a complete FACE conformant system can be constructed:

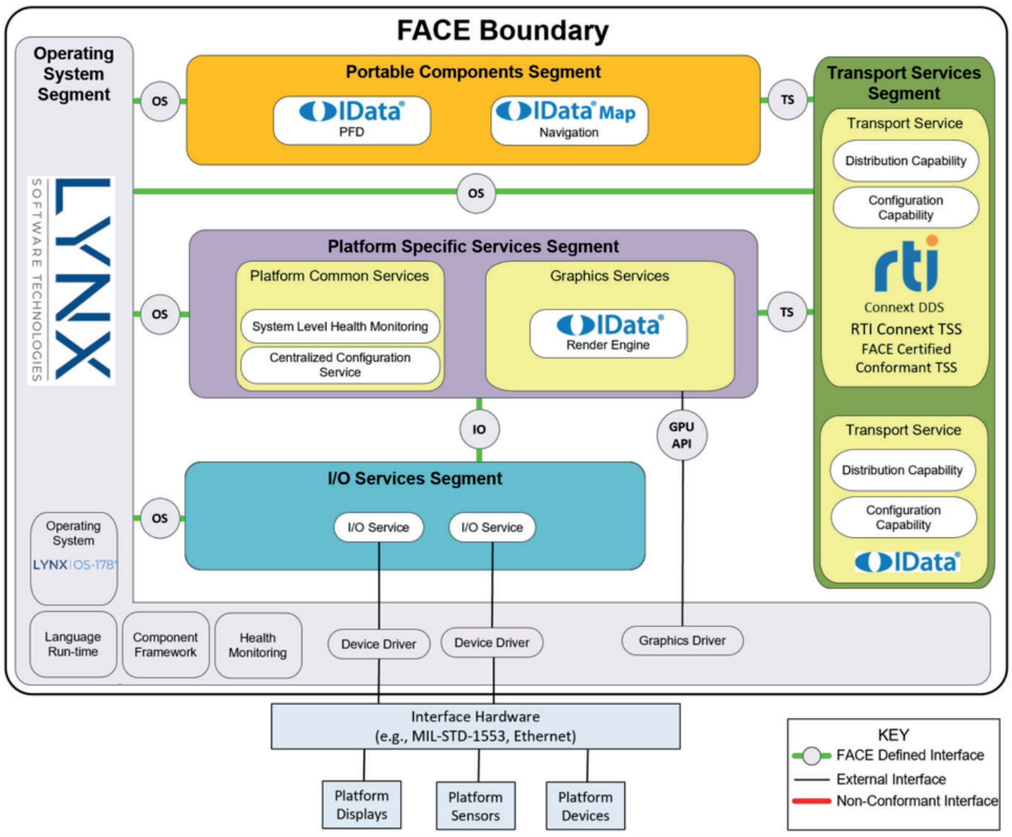


Figure 7 - Example FACE Implementation

FACE Conformance Program

The FACE Conformance Program provides the conformance criteria and processes necessary to assure that UoCs are developed according to the FACE Technical Standard. These criteria and processes require the Verification, Certification, and Registration of the UoC.

FACE Verification: Determining the conformance of the UoC to FACE Technical Standard requirements. Verification is handled through a Verification Authority (VA), such as LDRA Certification Services (LCS).

FACE Certification: Applying for FACE Conformance Certification once verification has been completed. Certification is processed through the FACE Certification Authority (CA).

FACE Registration: Publicly listing the FACE Certified UoC in the FACE Registry.

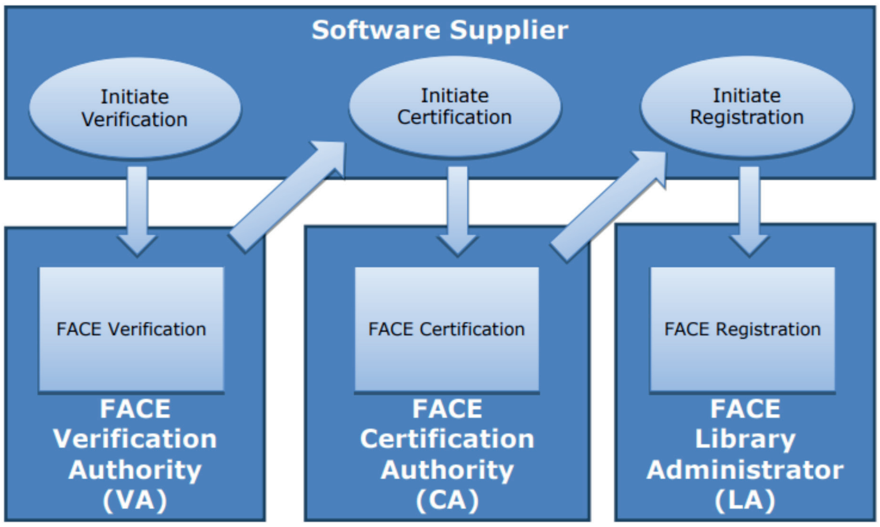
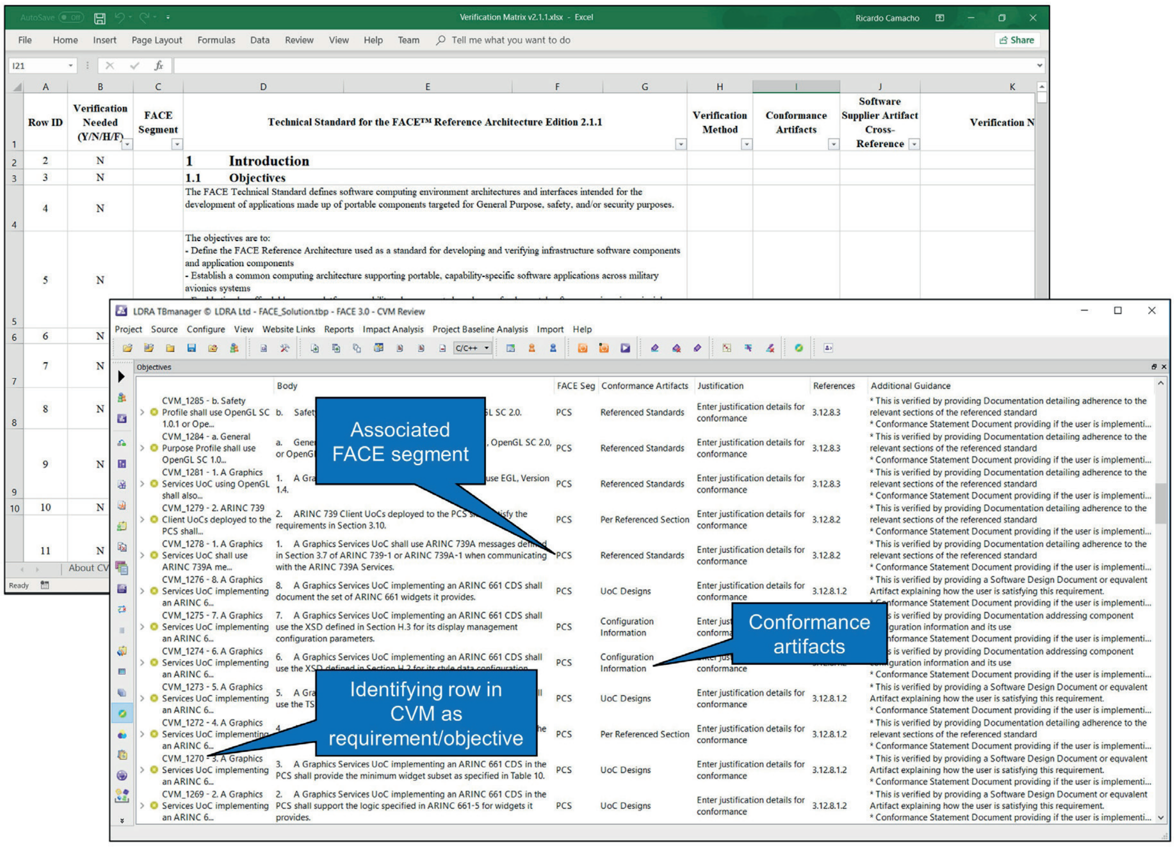


Figure 8 - Interactions between a FACE UoC software supplier, and FACE Authorities and Administrators

The FACE Technical Standard represents the de facto reference for the certification of a UoC. In support of that standard, the FACE Consortium have also created a Conformance Verification Matrix (CVM). It clarifies the Conformance Requirements that a product must meet in order to be certified as FACE conformant, along with the specific techniques to be used to verify each of those requirements.

It is prudent for software suppliers to confer with their chosen Verification Authority from the outset. At the time of writing there are five to choose from, one of which is LCS - the LDRA Certification Services team. The UoC can only be FACE conformant if it is based upon a FACE conformant architecture. Failure to meet the architectural requirements of FACE can often lead to major changes further down the line, with inevitable cost and time implications.

The addition of these demands to the functional requirements of the application and any functional safety standards all makes for a project management headache – especially when tests are failed, or functional requirements change. Automating traceability of requirements helps to relieve that pain point (Figure 9).



Row ID	Verification Needed (Y/N/H/F)	FACE Segment	Technical Standard for the FACE™ Reference Architecture Edition 2.1.1	Verification Method	Conformance Artifacts	Software Supplier Artifact Cross-Reference	Verification N
2	N		1 Introduction				
3	N		1.1 Objectives				
4	N		The FACE Technical Standard defines software computing environment architectures and interfaces intended for the development of applications made up of portable components targeted for General Purpose, safety, and/or security purposes.				
5	N		The objectives are to: - Define the FACE Reference Architecture used as a standard for developing and verifying infrastructure software components and application components - Establish a common computing architecture supporting portable, capability-specific software applications across military avionics systems				
6	N						
7	N						
8	N						
9	N						
10	N						
11	N						

Objectives	Body	FACE Seg	Conformance Artifacts	Justification	References	Additional Guidance
CVM_1285 - b. Safety Profile shall use OpenGL SC 1.0.1 or Ope...	b. Safet... L SC 2.0.	PCS	Referenced Standards	Enter justification details for conformance	3.12.8.3	* This is verified by providing Documentation detailing adherence to the relevant sections of the referenced standard * Conformance Statement Document providing if the user is implementi...
CVM_1284 - a. General Purpose Profile shall use OpenGL SC 1.0...	a. Gene... , OpenGL SC 2.0, or OpenG...	PCS	Referenced Standards	Enter justification details for conformance	3.12.8.3	* This is verified by providing Documentation detailing adherence to the relevant sections of the referenced standard * Conformance Statement Document providing if the user is implementi...
CVM_1281 - 1. A Graphics Services UoC using OpenGL shall also...	1. A Gra... use EGL, Version 1.4.	PCS	Referenced Standards	Enter justification details for conformance	3.12.8.3	* This is verified by providing Documentation detailing adherence to the relevant sections of the referenced standard * Conformance Statement Document providing if the user is implementi...
CVM_1279 - 2. ARINC 739 Client UoCs deployed to the PCS shall...	2. ARINC 739 Client UoCs deployed to the PCS s... isfy the requirements in Section 3.10.	PCS	Per Referenced Section	Enter justification details for conformance	3.12.8.2	* This is verified by providing Documentation detailing adherence to the relevant sections of the referenced standard * Conformance Statement Document providing if the user is implementi...
CVM_1278 - 1. A Graphics Services UoC shall use ARINC 739A me...	1. A Graphics Services UoC shall use ARINC 739A messages defined in Section 3.7 of ARINC 739-1 or ARINC 739A-1 when communicating with the ARINC 739A Services.	PCS	Referenced Standards	Enter justification details for conformance	3.12.8.2	* This is verified by providing Documentation detailing adherence to the relevant sections of the referenced standard * Conformance Statement Document providing if the user is implementi...
CVM_1276 - 8. A Graphics Services UoC implementing an ARINC 6...	8. A Graphics Services UoC implementing an ARINC 661 CDS shall document the set of ARINC 661 widgets it provides.	PCS	UoC Designs	Enter justification details for conformance	3.12.8.1.2	* This is verified by providing a Software Design Document or equvalent Artifact explaining how the user is satisfying this requirement. * Conformance Statement Document providing if the user is implementi...
CVM_1275 - 7. A Graphics Services UoC implementing an ARINC 6...	7. A Graphics Services UoC implementing an ARINC 661 CDS shall use the XSD defined in Section H.3 for its display management configuration parameters.	PCS	Configuration Information	Enter just... conform...		...s is verified by providing Documentation addressing component ...guration information and its use ...nformance Statement Document providing if the user is implementi...
CVM_1274 - 6. A Graphics Services UoC implementing an ARINC 6...	6. A Graphics Services UoC implementing an ARINC 661 CDS shall use the XSD defined in Section H.2 for its style data configuration	PCS	Configuration Information	... conformance		...s is verified by providing Documentation addressing component ...guration information and its use * Conformance Statement Document providing if the user is implementi...
CVM_1273 - 5. A Graphics Services UoC implementing an ARINC 6...	5. A Gra... ll use the TS...	PCS	UoC Designs	Enter justification details for conformance	3.12.8.1.2	* This is verified by providing a Software Design Document or equvalent Artifact explaining how the user is satisfying this requirement. * Conformance Statement Document providing if the user is implementi...
CVM_1272 - 4. A Graphics Services UoC implementing an ARINC 6...	4. ... he	PCS	Per Referenced Section	Enter justification details for conformance	3.12.8.1.2	* This is verified by providing Documentation detailing adherence to the relevant sections of the referenced standard * Conformance Statement Document providing if the user is implementi...
CVM_1270 - 3. A Graphics Services UoC implementing an ARINC 6...	3. A Graphics Services UoC implementing an ARINC 661 CDS in the PCS shall provide the minimum widget subset as specified in Table 10.	PCS	UoC Designs	Enter justification details for conformance	3.12.8.1.2	* This is verified by providing a Software Design Document or equvalent Artifact explaining how the user is satisfying this requirement. * Conformance Statement Document providing if the user is implementi...
CVM_1269 - 2. A Graphics Services UoC implementing an ARINC 6...	2. A Graphics Services UoC implementing an ARINC 661 CDS in the PCS shall support the logic specified in ARINC 661-5 for widgets it provides.	PCS	UoC Designs	Enter justification details for conformance	3.12.8.1.2	* This is verified by providing a Software Design Document or equvalent Artifact explaining how the user is satisfying this requirement. * Conformance Statement Document providing if the user is implementi...

Figure 9 - Automating Conformance Verification Matrix (CVM) management with the TBmanager component of the LDRA tool suite

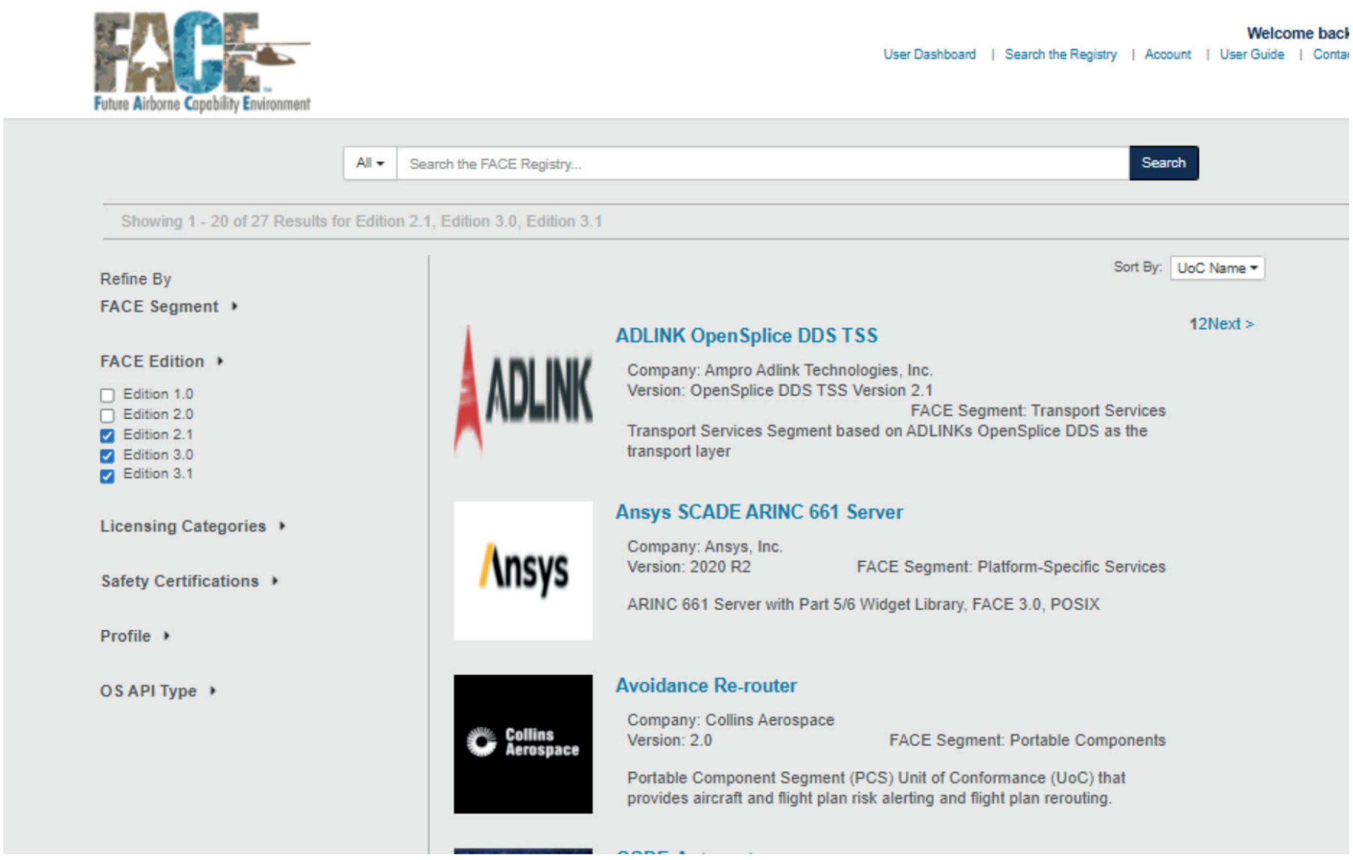
The verification audit process

Verification audit will likely involve a series of interactions between the supplier and the Verification Authority (VA). Verification is confirmed when the VA has audited the development process and approved the evidential artifact showing that all FACE Technical standard requirements applicable to the UoC have been satisfied. Once the VA approves the UoC, the software supplier may contact the FACE Certification Authority (CA).

Whereas there are several VAs to choose from, there is a single nominated FACE CA. The CA will audit the VA process to ensure all the process are in alignment with FACE conformance policy and will then issue the Conformance Certificate.

The last step is for the Library Administrator to register the UoC certified product to the FACE Registry. This responsibility is undertaken by Library Administrator (LA), a trusted third party selected by the FACE Steering committee.

As illustrated in Figure 10, all the certified UoCs are publicly visible. It follows that any software that is not listed in the FACE registry cannot be FACE Conformant.


FACE Future Airborne Capability Environment

Welcome back

User Dashboard | Search the Registry | Account | User Guide | Conta

All ▾ Search the FACE Registry... Search

Showing 1 - 20 of 27 Results for Edition 2.1, Edition 3.0, Edition 3.1

Refine By

FACE Segment ▸

FACE Edition ▸

- [ ] Edition 1.0
- [ ] Edition 2.0
- [x] Edition 2.1
- [x] Edition 3.0
- [x] Edition 3.1

Licensing Categories ▸

Safety Certifications ▸

Profile ▸

OS API Type ▸

Sort By: UoC Name ▾

12Next >

ADLINK OpenSplice DDS TSS

Company: Ampro Adlink Technologies, Inc.
Version: OpenSplice DDS TSS Version 2.1
FACE Segment: Transport Services
Transport Services Segment based on ADLINKs OpenSplice DDS as the transport layer

Ansys SCADE ARINC 661 Server

Company: Ansys, Inc.
Version: 2020 R2
FACE Segment: Platform-Specific Services
ARINC 661 Server with Part 5/6 Widget Library, FACE 3.0, POSIX

Avoidance Re-router

Company: Collins Aerospace
Version: 2.0
FACE Segment: Portable Components
Portable Component Segment (PCS) Unit of Conformance (UoC) that provides aircraft and flight plan risk alerting and flight plan rerouting.

Figure 10 - A screenshot showing the FACE registry

FACE Conformance Verification Process

FACE conformance is a complicated and time-consuming process, and automation is key to ensuring cost-effective product development. An integrated tool suite capable of spanning the whole development process is likely to offer the most effective solution (Figure 11). The expertise that resulted in LDRA's successful application to become a VA is also reflected in the LDRA tools developed to verify and validate UoCs.

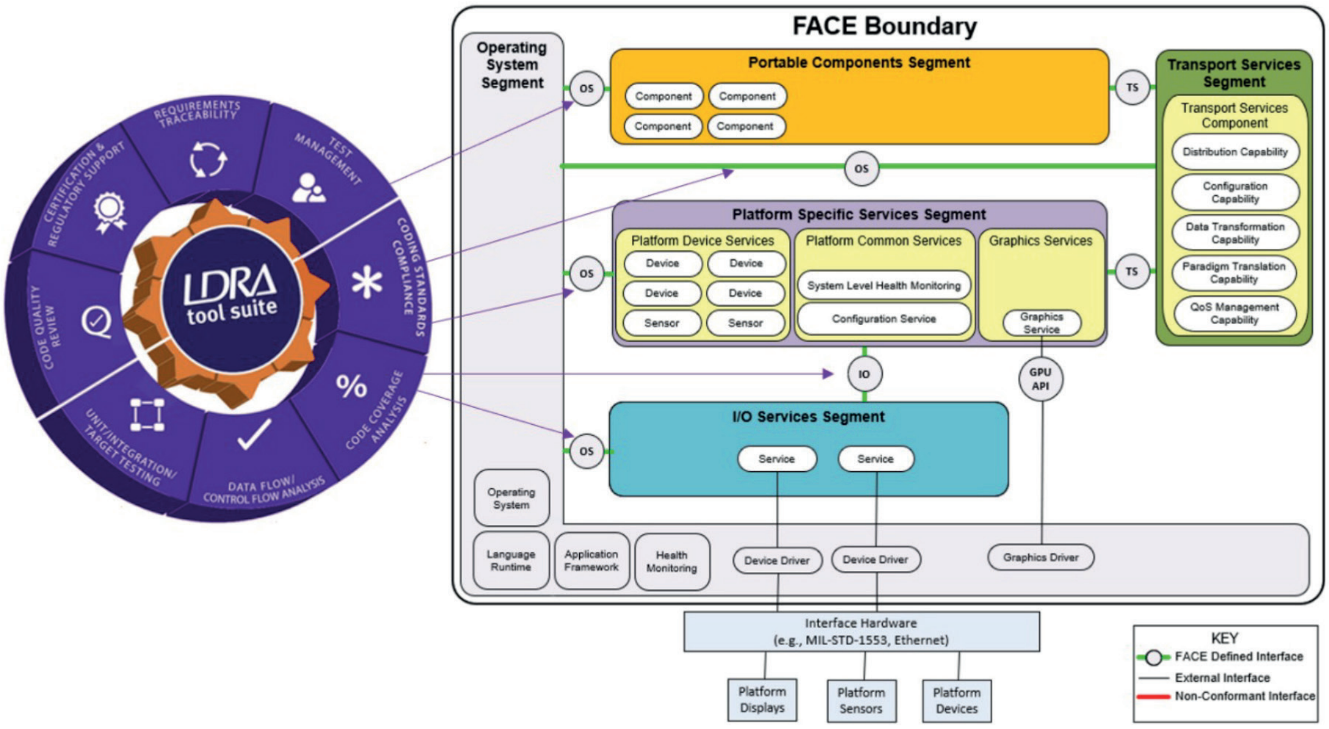
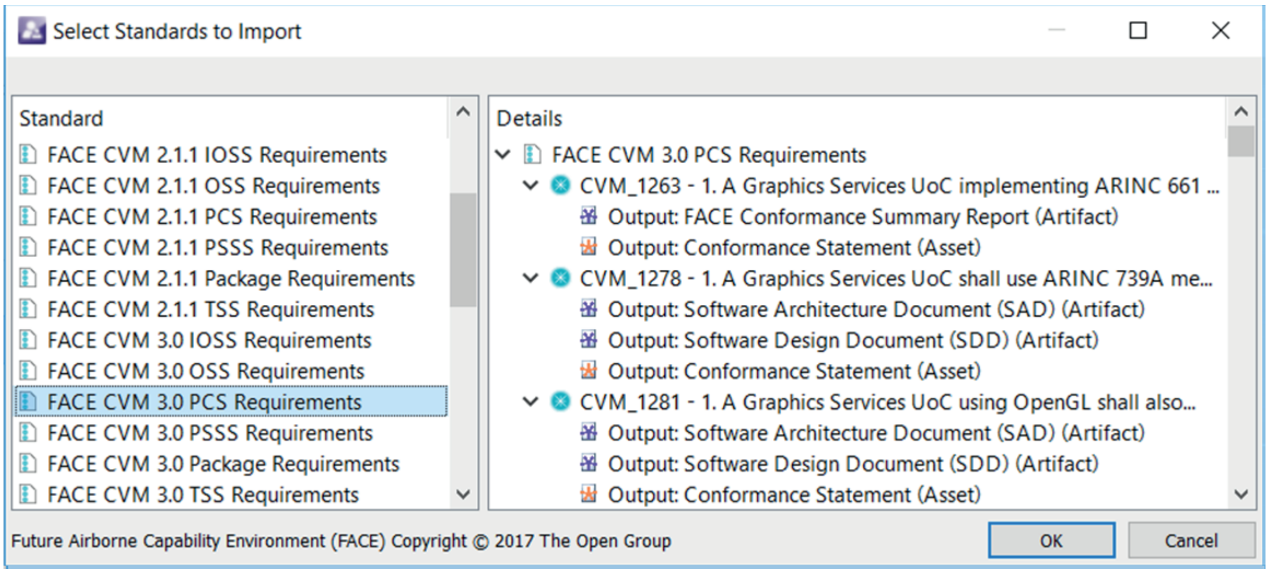


Figure 11 - The LDRA tool suite spans the entire FACE software development lifecycle, irrespective of segment

There are FACE segment definitions available within the TBmanager component of the LDRA tool suite reflecting each of the FACE architectural segments (IOSS, OSS, PCS, PSSS, or TSS) and each of the FACE editions. Importing a FACE segment definition artifact into TBmanager initiates the creation of a project and tree containing a complete set of objectives. Each objective correlates to a requirement defined in the FACE technical standard (Figure 12).


Select Standards to Import

Standard
- FACE CVM 2.1.1 IOSS Requirements
- FACE CVM 2.1.1 OSS Requirements
- FACE CVM 2.1.1 PCS Requirements
- FACE CVM 2.1.1 PSSS Requirements
- FACE CVM 2.1.1 Package Requirements
- FACE CVM 2.1.1 TSS Requirements
- FACE CVM 3.0 IOSS Requirements
- FACE CVM 3.0 OSS Requirements
- FACE CVM 3.0 PCS Requirements
- FACE CVM 3.0 PSSS Requirements
- FACE CVM 3.0 Package Requirements
- FACE CVM 3.0 TSS Requirements

Details
- FACE CVM 3.0 PCS Requirements
  - CVM_1263 - 1. A Graphics Services UoC implementing ARINC 661 ...
    - Output: FACE Conformance Summary Report (Artifact)
    - Output: Conformance Statement (Asset)
  - CVM_1278 - 1. A Graphics Services UoC shall use ARINC 739A me...
    - Output: Software Architecture Document (SAD) (Artifact)
    - Output: Software Design Document (SDD) (Artifact)
    - Output: Conformance Statement (Asset)
  - CVM_1281 - 1. A Graphics Services UoC using OpenGL shall also...
    - Output: Software Architecture Document (SAD) (Artifact)
    - Output: Software Design Document (SDD) (Artifact)
    - Output: Conformance Statement (Asset)

Future Airborne Capability Environment (FACE) Copyright © 2017 The Open Group

Figure 12 - Importing a FACE segment definition artifact into the TBmanager component of the LDRA tool suite

A TBmanager project created in this way will not only include the requirements from the FACE CVM that are relevant to the selected FACE segment. It will be supplemented by “Software Supplier Artifact – Cross Reference” placeholders for proof of conformance evidential artifacts. Complementary template artifacts include:

- Interface Control Document (ICD)
- Interface Design Description (IDD)
- Interface Requirements Specification (IRS)
- Software Architecture Document (SAD)
- Software Design Document (SDD)
- Software Product Specification (SPS)
- Software Requirements Specification (SRS)
- Software Test Plan (STP)
- Software Test Report (STR)
- Software User Manual (SUM)

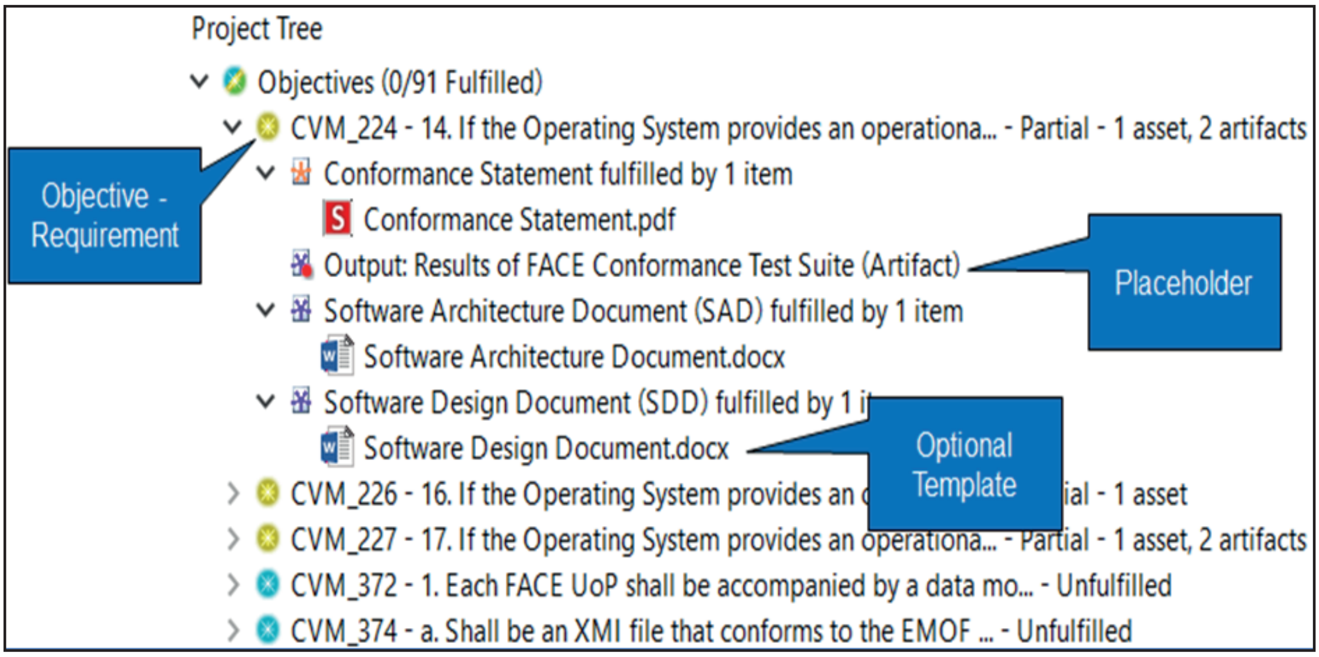

Project Tree
- Objectives (0/91 Fulfilled)
  - CVM_224 - 14. If the Operating System provides an operationa... - Partial - 1 asset, 2 artifacts
    - Conformance Statement fulfilled by 1 item
      - Conformance Statement.pdf
    - Output: Results of FACE Conformance Test Suite (Artifact)
    - Software Architecture Document (SAD) fulfilled by 1 item
      - Software Architecture Document.docx
    - Software Design Document (SDD) fulfilled by 1 i
      - Software Design Document.docx
  - CVM_226 - 16. If the Operating System provides an ial - 1 asset
  - CVM_227 - 17. If the Operating System provides an operationa... - Partial - 1 asset, 2 artifacts
  - CVM_372 - 1. Each FACE UoP shall be accompanied by a data mo... - Unfulfilled
  - CVM_374 - a. Shall be an XMI file that conforms to the EMOF ... - Unfulfilled

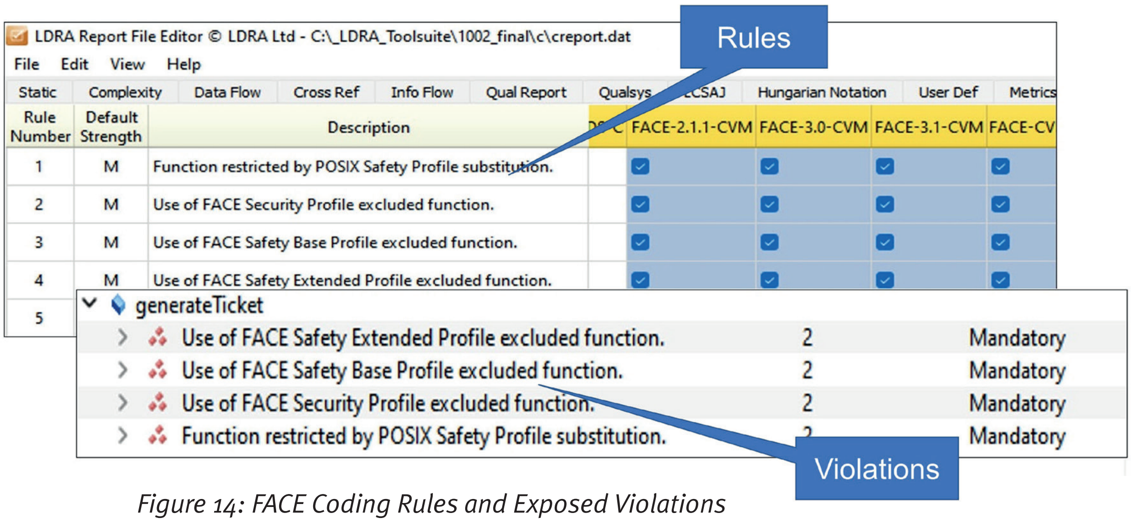
Figure 13 - FACE Requirements are represented as “Objectives” in the TBmanager component of the LDRA tool suite

FACE and static analysis

Static analysis tools contribute by means of an automated "inspection" of the source code. Static analysis techniques can be used to address most the FACE requirements deemed to be "Fully Tested Requirements".

The FACE Technical Standard restricts the use of certain API calls while requiring others. Non-conformant calls listed in the FACE Technical Standard and other FACE coding violations can be detected statically. For example, the checks for adherence to specific sections of the POSIX API ensure that the UoC function signatures are syntactically correct, enforcing proper use of certain language constructs.

The application of static analysis tools complements the static analysis element of the FACE Conformance Test Suite (CTS) by being part of the day-to-day development lifecycle, finding violations early at the developer's desktop and at a time of their choosing - prior to compilation of the code, or during the build phase.



Rule Number	Default Strength	Description	[illegible]	FACE-2.1.1-CVM	FACE-3.0-CVM	FACE-3.1-CVM	FACE-CV
1	M	Function restricted by POSIX Safety Profile substitution.		✓	✓	✓	✓
2	M	Use of FACE Security Profile excluded function.		✓	✓	✓	✓
3	M	Use of FACE Safety Base Profile excluded function.		✓	✓	✓	✓
4	M	Use of FACE Safety Extended Profile excluded function.		✓	✓	✓	✓
5							

generateTicket

Use of FACE Safety Extended Profile excluded function.	2	Mandatory
Use of FACE Safety Base Profile excluded function.	2	Mandatory
Use of FACE Security Profile excluded function.	2	Mandatory
Function restricted by POSIX Safety Profile substitution.	2	Mandatory

Figure 14: FACE Coding Rules and Exposed Violations

By then analyzing the code using the official FACE CTS conformance utility prior to submission to the VA, it can be confirmed that non-conformances have been dealt with during the development phase.

The FACE Conformance Test Suite can be configured and invoked from TBmanager which in turn will also establish traceability and facilitate regression testing. The association of TBmanager objectives with the obligatory FACE CTS evidential artifacts exposes any verification gaps and allows them to be addressed.

Conformance: An example from Collins Aerospace

The importance of earning a FACE Conformance Certificate for the Mission FMS software from Collins Aerospace cannot be overstated. Being able to certify conformance to a technical standard is one of the five key principles of MOSA as defined by the OSD Open Systems Joint Task Force. The fact that the FACE Conformance certification process is conducted using an independent assessment is the best way to assure customers that a product conforms against the requirements found within the FACE Technical Standard. The answer to preventing the "what could go wrong" with using the FACE standard comes down to assuring complete 100% conformance to requirements. Otherwise, the potential strongly exists for reuse to be denied and integration costs to rise because an interface mismatch can occur.

Collins Aerospace is committed to FACE Conformance and is proud to have earned the very first FACE Conformance Certification in 2016 for the Mission FMS, the first to earn a second certificate for the Localizer Performance and Vertical Guidance Calculator product. Today we hold the largest number of listings in the FACE Library. We've found that these certificates provide an outstanding marketing vehicle for showing proof that we are open system software providers and MOSA adherences. We've weaved this practice into our corporate DNA and it is in no small part of our current success.

Figure 15 – Collins Aerospace Tactical Combat Training System (TCTS) Increment II program

Here is an example of our winning business through adherence to MOSA and the FACE open standard. The Tactical Combat Training System (TCTS) Increment II program is one important example of using open standards to provide multi domain Live, Virtual and Constructive (LVC) training capabilities to modern warfighters.

The underlying system (called the Joint Secure ACMI System (JSAS)) utilizes an open systems architecture incorporating FACE™ and other industry interface standards to allow interfacing with a variety of monitoring and debriefing systems.

It's scalable - enables training scenarios from individual to large force, home stationed to deployed, and across service, allied and coalition security boundaries. It's ready now. Collins Aerospace achieved Milestone C for Tactical Combat Training System Increment II in June of 2021 – so we are moving into production with our team members at Leonardo DRS.

In addition to the initial $142 million contract with the U.S. Navy, this technology is moving toward fulfilling U.S. Air Force and United Kingdom Royal Air Force and Navy needs with the P6 Combat Training System.

Summary

As a testament to the celebrated success of FACE (Future Airborne Capability Environment), mandatory conformance requirements have flowed down for nearly every applicable military program since the publication of FACE 2.0. Since then, the standard has been enhanced still further, and there are strong reasons for OEMs to migrate to version 3.x of the specification.

FACE presently views virtualization as merely a coarse-grained consolidation tool. The enhancement of the FACE standard with an additional Hardware Specification Segment promises to allow virtualization to deliver on core FACE principles, particularly where hard real-time control must be accommodated alongside less time critical applications. Virtualization enables the simplified analysis of worst-case execution time (WCET) for that critical application, while retaining the ability to host applications with less demanding timing and integrity requirements on the same platform.

Developing software in accordance with any standard is hard. The benefits of FACE come at the cost of additional overhead on developers required to meet its various requirements. Working with a FACE VA from the beginning of the project is likely to minimize any nasty surprises later.

The concurrent application of project requirements, coding standards, FACE, and functional safety standards provide a logistical challenge for any project manager. Automating that traceability is a boon, especially in the face of changing functional requirements, or failed tests.

Static analysis tools contribute by means of an automated “inspection” of the source code. Static analysis techniques can be used to address most the FACE requirements deemed to be “Fully Tested Requirements”. The application of static analysis tools complements the static analysis element of the FACE Conformance Test Suite (CTS) by being part of the day-to-day development lifecycle. Violations can be found early, at the developer’s desktop and at a time of their choosing - prior to compilation of the code, or during the build phase.

There is undoubtedly a lot to take on here. But ultimately, the proof of the principles lies in the ability to win business through their application. That ability is ably demonstrated by the success of Collins’ Tactical Combat Training System (TCTS) Increment II program, whose underlying system utilizes an open systems architecture incorporating FACE™ and other industry interface standards.

[1] Collins Aerospace acknowledges The Open Group, L.L.C. for permission to include text/figures derived from its copyrighted FACE™ Business Guide 2.0. FACE, Future Airborne Capability Environment, and the FACE logo are trademarks of The Open Group in the United States and other countries.

ldra.com collinsaerospace.com lynx.com