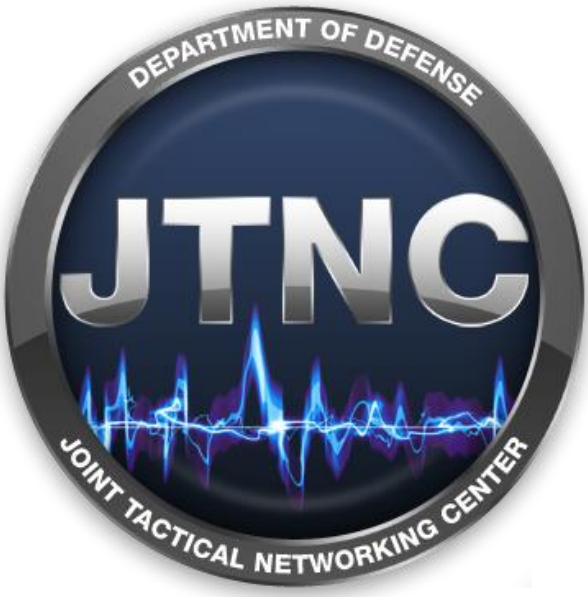
SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION VERSION 4.1 FEATURES AND BENEFITS



18 January 2018
Version: 1.0

Prepared by:

Joint Tactical Networking Center
33000 Nixie Way
San Diego, CA 92147-5110

UNCLASSIFIED

REVISION SUMMARY

Version	Revision
1.0	Original document

TABLE OF CONTENTS

1 INTRODUCTION

The Software Communications Architecture (SCA) 2.2.2 [2] has been very successful with a multitude of implementations and over 400K communications platforms deployed to the field. Given the proliferation of 2.2.2 products there have been questions regarding whether the effort should be expended to use SCA 4.1 [1] and if existing products should be transitioned to align with the new specification.

SCA 2.2.2 was published over ten years ago and in the time span since its publication the landscape of Software Defined Radio (SDR), technologies, system requirements and threats have changed. SCA 4.1 is an evolutionary specification, it doesn't fundamentally change the core capabilities of SCA 2.2.2, but it incorporates lessons learned over the past decade which enhance the framework's performance and better situate it to accommodate future changes.

SCA 4.1 provides a baseline set of capabilities that are geared towards enhancing security, improving infrastructure performance, maximizing waveform portability and providing a high degree of design flexibility and extensibility. Usage of the least privilege pattern and push model registration are two of the cybersecurity modifications that enhance the reliability and resilience of software-based platforms by making them less susceptible to vulnerabilities or intrusions. Within the realm of performance, boot time improvements, e.g. registration, allow platforms to enter the net and start processing data more rapidly. The efficiency updates, partially represented by enhanced process co-location and multicore support, allow SCA 4.1 components to utilize the underlying resources and infrastructure more effectively. The expanded flexibility allows for newer technologies, architectural approaches and features to be incorporated in a standardized manner. The improvements when taken as a whole, provide system designers and developers with additional tools that allow for innovative capabilities such as higher bandwidth waveforms, component migration to less expensive development platforms or a reduction of the physical hardware footprint.

The intent of this paper is to highlight the technical features and enhancements that are included within SCA 4.1 and describe how the inclusion of those capabilities would benefit an SCA system stakeholder relative to that of an equivalent SCA 2.2.2 implementation. Although this paper highlights the key new SCA 4.1 features, it is not intended to be a primer for the specification. New users might find that the briefing materials on the Joint Tactical Networking Center (JTNC) public web page are a better starting point to learn about the SCA.

1.1 INFORMATIVE REFERENCES

The following documents are referenced within this specification or used as guidance material in its development.

[1] Software Communications Architecture Specification, Version 4.1, 20 August 2015.

[2] Software Communications Architecture Specification, Version 2.2.2, 15 May 2006.

[3] Joint Tactical Networking Center Application Programming References. Retrieved 15 DEC 2017 from http://www.public.navy.mil/jtnc/Pages/resources.aspx?filter=cat-api

[4] SCA 2.2.2 to 4.1 Migration, Joint Tactical Networking Center Standards. Retrieved 15 DEC 2017 from http://www.public.navy.mil/jtnc/PapersBriefsReports/SCA_222_to_41_Migration.pdf

2 BENEFITS OF SCA 4.1 TECHNICAL FEATURES

This section highlights the benefits that can be realized when SCA 4.1 is implemented.

2.1 BENEFITS OF UPGRADING BOTH THE OE AND APPLICATIONS

2.1.1 Leveraging Industry Standards Reduces Costs

SCA 4.1 includes a Unified Modeling Language (UML) representation of each component in a model which can be integrated directly within an organization's development process. The models are more rigorous than those contained within SCA 2.2.2 and better represent the responsibilities and relationships of the system elements that they represent.

The annotated models convey additional guidance to the development community regarding how compliant products should be built, thus enhancing the uniformity of artifacts built in accordance with the specification. The presence of well-defined models enables organizations that already use modeling within their development processes, to use the standardized SCA models within their projects. It also provides a jump start for those who do not use models, to incorporate them within their engineering practices.

SCA 4.1 components, when coupled with the Tactical Radio Application Programming Interfaces (APIs) [3] and products modeled by the development organization, provide a foundation for building and deploying flexible products that can be very cost efficient over their lifespan. SCA 4.1 defines modular, loosely coupled components that can be easily replaced, removed or enhanced. The reasons for transitioning to SCA 4.1 could be as varied as a technology refresh or the incorporation of new algorithms to provide a more secure replacement. The adoption of component based development principles enables development efficiencies to be realized within or across organizations when a software repository is maintained which contains reusable blocks of functionality.

The standards compliant nature of the SCA 4.1 models allows them to be used within Commercial Off-The-Shelf (COTS) tools that comply with the underlying standards. The use of those tools can be optimized and streamline the development process when they utilize the model extension capabilities to support the SCA 4.1 constructs.

2.1.2 Improved Guidance for Development Responsibilities

SCA 2.2.2 is primarily interface-based with behavioral requirements interspersed within the interface descriptions. Sections 3.2 (Applications) and 3.3 (Logical Device) are behaviorally oriented but neither comprehensive nor fully coordinated with the interface definitions. This lack of demarcation makes it more difficult for developers to distinguish between the capabilities driven by the interfaces and the supplemental run-time features that need to be provided by a product implementation. SCA 4.1 highlights this separation through its utilization of distinct sections for interfaces and components.

Within SCA 2.2.2, developers had to make a significant time investment learning the intricacies of the specification in order to build SCA compliant systems. SCA 4.1 components clearly identify the features that need to be provided as part of an implementation, decompose the products to be implemented into their constituent parts and describe their relationships with other system elements.

2.1.3 Improved Representation of Component Roles and Responsibilities

The revised document structure in SCA 4.1 allows developers to focus on the portions of the specification that are most relevant to their projects. The separation of components and interfaces also assists in the software development life cycle process. The new representation facilitates the allocation of requirements to each software component in the system, which can assist in creating development cost and resource estimates.

2.1.4 Removal of the CORBA Mandate Enables Technology Adoption

SCA 4.1 is expressed in a technology neutral model which allows developers to receive the overarching benefits of SCA using whatever technology is most appropriate for their implementation. The model permits customization at multiple layers. For example, if a determination is made that eXtensible Markup Language (XML) descriptors are not feasible for a platform, a specific technology and grammar can be substituted that allows equivalent configurations, connections and properties to be defined and communicated. Similarly, if a determination is made that an alternative distributed communications mechanism, e.g. Data Distribution Service (DDS), should be used instead of Common Object Request Broker Architecture (CORBA), a set of rules can be developed that provide a security aware SDR relevant profile that fulfills the high level embedded system and distributed communication SCA objectives.

The SCA 4.1's Platform Independent Model (PIM) expands the specification's applicability to additional form factors and architectures. As the underlying technology transitions, the essential SCA deployment and management functionality will be preserved and the accumulated knowledge of how to design and build portable components will be documented and shared. Tradeoffs and design alternatives, such as the introduction of a single processor architecture with software enforced separation to extend battery life, will need to be assessed on a case by case basis to identify the optimal target architecture.

CORBA, Portable Operating System Interface (POSIX®), C, C++ and Document Type Definitions (DTD) are the only technologies currently defined within SCA 4.1, but additional ones are being sought and evaluated for incorporation within the specification. The inclusion of additional targets will increase the ability of SCA compliant products to align with platform specific requirements.

2.1.5 Units of Functionality Benefit Product Cost, Flexibility, Product Alignment and Security

The units of functionality (UOFs) provide an additional path for communication between specification and system developers. Each UOF provides a logical grouping of requirements that are related to a single capability or block of functionality within the specification. The groupings can be leveraged throughout the software development life cycle as they represent capabilities or blocks of functionality that can be worked concurrently.

2.1.6 IDL Decomposition Reduces Product Size

SCA 4.1 formalizes a capability that existed implicitly within SCA 2.2.2. The SCA 2.2.2 Interface Definition Language (IDL) is contained in a monolithic file that includes all of the interface and

® POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.

type definitions. Left unspoken was how that file should be used within the context of product development. For example, if a waveform was being developed then the expectations regarding how the IDL core framework and Device definitions should be treated were unspecified. The developer could preserve the file in its entirety, which would result in a large amount of code that would have to be developed or stubbed and never used within the application. Another alternative would be to delete the non-essential constructs from the IDL file, which addresses the shortcomings of the other alternative, but is time consuming, error prone and could result in a non-compliant product.

SCA 4.1 partitions the IDL content into logical groupings across multiple IDL files. This allows the developer to include only the needed files, thus aligning the IDL directly with a product. The result of this alignment is an executable that is as small as possible and minimizes the amount of any unused code.

2.1.7 AEP Modifications Improve Security and Product Alignment

The Application Environment Profile (AEP) was updated to better reflect alignment with commercially available RTOS products, development experience and current thinking on function vulnerabilities. RTOS Standards have evolved since the publication of SCA 2.2.2 and the functions that were introduced, deprecated or modified within the newer implementations were evaluated during the SCA 4.1 specification development process. Through consultations with developers, vendors and other specification groups the AEP was modified to reflect a set of functions that are commonly available and useful to SDR developers. During the specification development process, more than 20 operations were added to the specification and approximately 15 were removed. The majority of those removals were performed because the underlying operations were known to be inefficient or contained vulnerabilities. In addition, the specification clarifies an SCA 2.2.2 ambiguity by providing clear direction regarding the usage of the Standard C libraries within Application implementations.

2.1.8 Lightweight & Ultra-Lightweight CORBA Provides Improved Access to Resource Constrained Platforms

The introduction of the Lightweight & Ultra-Lightweight CORBA profiles extends the applicability of the SCA to additional form factors such as handheld devices. The profiles restrict the availability of CORBA features which in turn leads to smaller and in some cases, better performing middleware packages. For products that use CORBA, this capability provides a means to streamline the middleware footprint on a platform.

2.1.9 Lightweight & Ultra-Lightweight AEP Provides Improved Access to Resource Constrained Platforms

The addition of the new Real-Time Operating System (RTOS) profiles (i.e. Lightweight & Ultra-Lightweight AEPs) extends the applicability of the SCA model to additional form factors such as handheld devices. These new profiles restrict the availability of operating system features which in turn leads to smaller and in some cases, better performing operating systems. The reduced set of operations can also be identified as the target on less resource constrained processors. Applying the smaller profiles in such settings provides an additional benefit in that it enhances component portability across processor families.

2.1.10 Lightweight Components Reduce Component Costs

SCA 4.1 lightweight components enhance a framework user's ability to align component needs and requirements with the services provided by SCA. The utilization of lightweight components can reduce the cost and improve the security awareness of each developed component. A control capability is a constituent part of any SDR, but it is likely that the entire collection of BaseComponent functionality is not applicable to all scenarios. Lightweight components are realized within a component through the use of UOFs to include or exclude blocks of functionality. For example, if a component does not need to provide a built-in test capability, then that capability can be excluded at the model level. From that point, there is a ripple effect across the product life cycle as those requirements would not need to be implemented, tested or maintained which in turn lowers development cost and time. Excluding unnecessary code has the second order benefit of reducing the size of the developed artifacts. Alternatively, SCA interfaces can be excluded from a product by stubbing out the unnecessary functions. Stubs provide the benefit of reducing software lifecycle costs, however they also introduce unused code into the baseline which may be interpreted as an Information Assurance (IA) issue.

2.1.11 Lightweight Managers Reduce Core Framework Costs

The SCA 4.1 lightweight managers provide the same benefits as lightweight components except that those savings are targeted towards the Operating Environment (OE) infrastructure components. Scaling back OE capabilities can introduce portability issues if the candidate applications make extensive use of features that are not incorporated within the baseline, so current and future usage scenarios should be considered prior to using this capability.

2.1.12 Least Privilege Pattern Improves Cyber-Security

SCA 4.1 applies the principle of least privilege to all of its information exchanges. This principle only permits components to access information that is required for their stated purpose. The removal of the naming service in favor of the component registries is an example of this policy. The component registry interface has a single operation to register a component; no additional capabilities exist within the interface for a component to query the registry for information about itself or other domain components.

Least privilege enhances a system's IA posture as it prevents potentially destructive components from collecting and consequently acting upon system information. This reduction of attack surfaces within the radio also reduces development time and bill-of-material costs for SCA 4.1 implementations.

2.1.13 Late Registration Support Provides Additional Flexibility and Security

SCA 4.1 clarifies and corrects the description of how the framework can introduce new OE components within an operating platform. A practical scenario for this capability is the introduction of plug and play devices within a platform. This capability enhances SCA platform flexibility, allowing compliant implementations to incorporate new functionality and alternative deployment strategies in a uniform manner, based on environmental conditions or requirements.

2.1.14 Revised Parsing Guidance Benefits Performance and Security

SCA 4.1 clarifies guidance on XML parsing and as such affords platforms with the opportunity to enhance performance and security. When platform developers are confident that domain profile files do not have to be parsed as part of application installation or platform boots, architectural

trade-offs can be made to emphasize product speed, size or flexibility. For example, performance can be optimized through the use of preprocessed domain profile files during application installation. When on board parsing is not performed as part of the deployment process, there exists the possibility of not executing the parser or excluding it from the platform.

2.1.15 Static Configurations Improve Performance and Security

Static configuration is a more extensive application of relaxed parsing. Static configurations, when properly implanted, provide even more performance and security improvements. Using this approach, the platform's entire configuration can be determined off board and reflected in the installed platform image. While this limits a platform's flexibility, it minimizes framework processing during the deployment process and provides a deployment alternative that might be optimal for specific scenarios.

2.2 BENEFITS INDEPENDENT OF APPLICATION UPGRADE STATUS

2.2.1 Channels Improve Systems Modeling and Deployment Performance

SCA channels are logical collections of devices and services that can be used as part of a communications path. This concept can improve systems modeling because it affords a system designer with a mechanism that is easy to understand and model to specify the OE resources that a waveform will use.

SCA 2.2.2 deploys applications by comparing their properties with those of all the components registered within the system for candidate targets. The SCA 4.1 capability optimizes the process by restricting the search space to only those elements defined within the channel.

2.2.2 Services Facilitate System Modeling and Improve Portability

The SCA 2.2.2 specification identified naming, log and event as its only formal services. In addition, the specification did not define a Service construct or data type. Therefore, if a system designer wished to include additional services there was no clear strategy to use for these types of components. Consequently, such elements were inappropriately represented as devices or applications and this characterization did not allow for an accurate capture of their logical meaning.

SCA 4.1 includes a formal component, ServiceComponent, that represents both SCA and non-SCA compliant services. This construct provides a standardized entity that can be incorporated within designs, consistently utilized across implementations and accurately evaluated for compliance.

2.2.3 Push Model Registration Increases System Performance and Security

Push model registration allows a registering component to determine the timeline for its registration. Within push model registration, the optimal strategy is for the registering component to aggregate all of its information relative to registration and then send or push it the registrar.

Prototyping has demonstrated that the SCA 4.1 push model registration feature has a significantly positive impact on performance. The improvements represent a transition from an approach where the SCA deployment machinery takes several steps to retrieve application information to one where the application assembles all of its information and then passes the complete bundle to the registry. This enhancement was designed to minimize application impacts and 95% of the required changes are contained within the OE.

Push model registration also enhances a platform's cyber-security posture. The presence of one versus multiple information exchanges reduces the opportunity for message intercepts. Push model

registration removes the inherent vulnerabilities of the globally available CORBA Naming Service and replaces it with a local repository that is encapsulated within the bounds of the infrastructure. The component registries that provide entry points to the domain repository can be modeled in several ways which allow for additional layers of security, e.g. registry location in a manner that eliminates registration calls across security boundaries. Lastly, since the registration repository is not directly part of a system's externally visible components, external domain consumers or other platform applications are not able to retrieve component registration information.

2.2.4 Device Push Registration Improves Architectural Consistency

Device push registration has the same performance and security benefits as application push registration. From a performance perspective, this is an important boot up time feature because it expedites availability of the platform's OE. Incorporating device push model registration also improves the consistency of the specification because it allows the same approach to be used for both OE and application registration. This consistency benefits organizations that implement both types of components as it allows for better code reuse.

2.2.5 Multicore Allocation Provides a Tech Refresh and Increased Performance

This enhancement provides a systems integrator with the ability to leverage the full capabilities of their hardware components. As processors have matured they have become more powerful and better able to execute commands in parallel. The multicore enhancement allows a system designer to have deterministic control over their deployment infrastructure so that components can be deployed across the available processor cores.

Multicore allocation allows system designers to enhance performance by more efficiently distributing execution across the available processing platforms, reduce costs by fully utilizing existing computing resources, or save power by requiring fewer processors.

2.2.6 Nested Applications Improve Deployment Security and Performance

Nested applications are beneficial from a performance and security perspective and can also provide a strategic framework for a product's evolution from SCA 2.2.2 to 4.1. Nested applications allow a subset of an application's deployment and instantiation processes to be managed separately. From a performance perspective, this allows a portion of an application (e.g. a reusable collection of components) to be packaged and leveraged across multiple implementations. These same techniques can be employed iteratively to transition an application from one version of the specification to the other. For example, the developing organization could have a nested application that is the SCA 2.2.2 portion of the waveform. As those components are migrated to SCA 4.1 they could be relocated from the nested application to the containing application. From a security viewpoint, nested applications can be used to create more IA aware applications. Nesting can be used to constrain portions of an application to certain processing elements or upon instantiation either minimize or eliminate crossings between enclaves.

2.2.7 External Applications Enhance Reusability

External applications provide an alternate mechanism to incorporate reusable blocks of code within an application or a mix of components that span multiple specification versions. However, the significant benefit of the external application capability is that it allows connectivity to external agents, e.g. user interfaces.

The SCA 2.2.2 Software Assembly Descriptor (SAD) file provided a means to declare external ports but the framework did not contain requirements to establish its connections. Consequently, external, i.e. non-SCA, services or capabilities were developed to perform a function that should have been contained within the framework. Having a means to establish external application connectivity allows the SCA to have a common approach to connect SCA applications with external user interfaces such as those on Android devices or applications built with another framework. A bridge or adapter could allow applications built using different methodologies to coexist on a single platform yet share the same underlying hardware.

2.2.8 ComponentType Structure Streamlines Product Implementation

The introduction of the ComponentType structure is an internal enhancement that unifies the architecture and provides a common baseline that can be used as the system evolves. Since all SCA components use this data type, a higher percentage of the code can take advantage of the inherent commonality as specific behaviors are processed. Consequently, more of the code base can be multi-purposed to serve several component types. The unification of the code also assists in the evolution of those components as new functionality can be introduced across components with only one change instead of several.

2.2.9 Pluggable Protocols Enhance Platform Flexibility

Pluggable or extensible protocols are supported by the CORBA Platform Specific Model (PSM) and the SCA 4.1 Platform Independent capability. Usage of custom or alternate protocols can enhance system performance in terms of latency or available bandwidth. Several CORBA implementations exist that provide optimizations which take advantage of conditions, such as processor co-location, to improve the performance of the system's CORBA calls. In addition, those products provide the option of using protocols other than Transmission Control Protocol and Internet Protocol (TCP-IP) for distributed calls. Using alternative transport mechanisms allows the ORBs to approach performance characteristics similar to those of sockets while providing the location transparency and support for heterogeneous technologies that are typical of CORBA. Depending on their structure and content, some alternate protocols can provide greater security than out of the box CORBA or other distributed component technologies.

3 SUMMARY

Adoption of SCA 4.1 does not invalidate or negate the investments made in SCA 2.2.2 products. SCA 4.1 is backwards compatible in that it is able to deploy and manage SCA 2.2.2 applications. The updated specification is evolutionary in the sense that its core model, mission and technical foundation has remained the same. The modifications introduce new capabilities (e.g. multicore allocation) that expand the specification's comprehensiveness or change how existing functions are accomplished (e.g. push model registration) to improve its efficiency and security. In transition scenarios, the degree of commonality between the versions allows existing SCA 2.2.2 products to provide the bulk of an SCA 4.1 implementation [4].

SCA 4.1 is the byproduct of a decade of SDR implementation experience and outstanding collaboration between Government and Industry stakeholders. The specification preserves key features such as application portability through the specification of the AEP and decoupled development of platform hardware and software through well-defined component interfaces. It also introduces several new features which improve system security, performance, flexibility and usability.

4 ACRONYMS

Abbreviation	Definition
AEP	Application Environment Profile
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
DDS	Data Distribution Service
DTD	Document Type Definition
IA	Information Assurance
IDL	Interface Definition Language
JTNC	Joint Tactical Networking Center
OE	Operating Environment
PIM	Platform Independent Model
POSIX®	Portable Operating System Interface
PSM	Platform Specific Model
RTOS	Real-Time Operating System
SAD	Software Assembly Descriptor
SCA	Software Communications Architecture
SDR	Software Defined Radio
TCP-IP	Transmission Control Protocol (TCP) and Internet Protocol (IP)
UML	Unified Modeling Language
UOF	Unit of Functionality
XML	eXtensible Markup Language

® POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.