



News Release

Joint Program Executive Office, Joint Tactical Radio System

Contact: Jeff Mercer

Desk: 619-524-4560 / Mobile: 619-252-2503

james.j.mercer@navy.mil

August 18, 2009

JPEO-NR-2009-05

JPEO JTRS initiates the development of a new Software Communications Architecture (SCA) Release

Joint Program Executive Office for the Joint Tactical Radio System (JPEO JTRS) has initiated the development of a new Software Communications Architecture (SCA) release. The fundamental goal behind this revision is to position the SCA as a specification that is comprehensive yet flexible enough to provide a technical foundation for multiple generations of JTRS and industry products.

To accomplish this flexibility, the proposed SCA enhancements will evolve the specification toward a technology independent representation, notably making Common Object Request Broker Architecture (CORBA) optional. The original version was suitable for larger, multi-channel radios and some of the proposed changes will provide additional flexibility and capability for those platforms. The more encompassing changes will modify the specification to better support the requirements of low-power and single channel radios.

The change proposals and their descriptions for the new version are provided as an attachment to this release. The Software Defined Radio Forum (SDRF) has agreed to participate as the public liaison with JTRS for the development of the new specification. JPEO JTRS has previously collaborated with the SDRF to host a combination JTRS Science and Technology forum (JSTeF) and general working meeting of the SDRF.

###

About JPEO JTRS

The Joint Tactical Radio System, headquartered in San Diego, Calif., was initiated in early 1997 to improve and consolidate the Service's pursuit of separate solutions to replace existing legacy radios in the Department of Defense inventory. The JTRS program has evolved from separate radio replacement programs to an integrated effort to network multiple weapon system platforms and forward combat units where it matters most – the last tactical mile. JTRS will link the power of the Global Information Grid to the warfighter in applying fire effects and achieving overall battlefield superiority.

JTRS is developing an open architecture of cutting edge radio waveform technology that allows multiple radio types (e.g., handheld, aircraft, maritime) to communicate with each other. The goal is to produce a family of interoperable,

modular software-defined radios which operate as nodes in a network to ensure secure wireless communication and networking services for mobile and fixed forces. These goals extend to U.S. allies, coalition partners and disaster response personnel. For more information, please visit <http://jpeojtrs.mil/>

CP #	TITLE	NOTES
S026	Re-factor SCA so that it can be completely tested in an automated fashion	
		<p>The current JTRS Standards are silent regarding language features. Without restriction on language features many developers are using features of C++ which are not appropriate for real-time embedded systems which have memory and processing power constraints</p> <p>Develop a set of language feature guidelines for C and C++ that are based upon best practices for developing real-time embedded software systems that are size, weight, and power constrained. These guidelines would also address application portability as well as performance.</p>
S007	Language Feature Standards for C and C++	
S018	Deletion of Non Waveform SCA Requirements	<p>The non waveform requirements do not affect waveform portability however they have significant impact on cost of terminal development, compliance testing, and terminal boot-up latency. Use, as a basis for requirement removal, the study conducted by General Dynamics under contract to the JTRS JPEO that was completed in April 2002 which looked at the definition of SCA compliance from a Waveform Interface perspective using SCA Version 2.1. Also use the current requirement allocation of the SCA . Objective would be to delete those requirements that do not affect waveform portability and are internal to the JTR Set infrastructure.</p>
S033	Reorganize SCA so that development responsibilities are more self evident (CF developer, WF developer, Device developer, Service developer)	
S047	Develop CORBA/e and CORBA Services wording	
S009	LW AEP CP	LW AEP for Signal Processing (DSP)
S011	SCA Deployment CP	Deployment Optimizations (e.g., port connections)
S019	Remove CORBA specific references from SCA	
S027	Develop backward compatibility strategy	
S032	Decompose CF.idl into a collection of files	

CP #	TITLE	NOTES
		<p>SCA 2.2.2 has a number of features that could be viewed as special cases of more general features:</p> <ol style="list-style-type: none"> 1. Devices, Composite Devices, Resources, and Services are all special cases of Component 2. Components are collected into assemblies or collections, such as application resources collected into an Application, devices into composite devices, all devices running on a processor device, device managers in a domain. 3. DomainManager, DeviceManager, Application/AssemblyController can be thought of as special cases of a manager of a container of sub-objects. <p>While each special case has some unique aspects, there is much in common and while some of that commonality is reflected in the SCA specification as common or similar properties, method calls and interfaces, and required behavior, maximizing the symmetry seems not to have been a strong goal. Thus there are often places where similar cases have unnecessary differences or where hierarchies have limited number of levels when the general case would have been no more difficult.</p>
S013	Architectural Consistency	
S029	Integrate SCA Extensions into SCA	
S034	Remove unnecessary SCA requirements	
S044	Allow for nested applications to be connection endpoints	
S005	POSIX file system accesses	<p>Consider the following changes to this requirement - if the file is not available in the local POSIX file system then the application does the following to access the file: Applications shall perform file access through the CF File interfaces. The application filename syntax is specified in section 3.1.3.4.2.1.</p>
S008	Deployment, Initialization and Configuration CP	<p>clarification of service deployment for initialization and configuration when Lifecycle and PropertySet are supported along with connection behavior.</p> <p>For LW components, this also means AF can manage all capacities offloading responsibilities from Devices.</p>

CP #	TITLE	NOTES
S024	Introduce component model into SCA	
S038	Make Application Factory deployment and configuration more deterministic	
S043	Address Application factory component initialization and configuration requirements	
S046	Enhance external port connectivity	
S002	Domain Profile files	Complete re-vamping of the domain profile. In its current form, it is too complicated and requires extensive deployment experience. Perhaps this can be combined with the initiative to make WFA deployment more deterministic since it was the ultra-flexible WFA deployment scheme that drove some of the complexity into the domain profile in the first place. There should be some consideration given to even moving away from the use of XML for the domain profile and replacing it with something much simpler (e.g. .ini files)
S006	Loadable Device Enhancement	LoadableDevice allows for extendable LoadType that is a numeric value instead of an Enumeration, such that devices that have extended the LoadableDevice interface can support device dependent load types.
S016	Maintain backward compatibility for waveforms while sacrificing compatibility for CF	We have a goal of maintaining a high level of backward compatibility for the new evolved SCA. But absolute backward compatibility will be difficult. Our priority is: 1) backward compatibility for waveforms is highest priority 2) backward compatibility for platform components is lower priority but still high 3) backward compatibility for OE and especially CF is lowest Some current requirements on CF that have no impact on waveforms or even platform components might be eliminated, allowing CFs to implement as desired. Some requirements simply specify how CF components interact internally between each other and have no visibility to platform or waveform components.
S035	Develop static SCA profile	
S037	Expand AEP to include Networking operations, including socket programming.	
S001	SCA Security Supplement	SCA 2.2.2 deleted this document. Create a new, relevant, security document applicable to the broader class of software defined radios.
S004	UUID Format	Determine whether this requirement should be deleted or changed.
S010	Executable Device deployment enhancement CP	Executable Device expanded behavior to handle processing envs or processing collocation behavior.
S012	SCA LW Component CP	Lightweight SCA components (e.g., waveform and devices)

CP #	TITLE	NOTES
S015	DMD and DCD connections	1. The DCD should have an option to specify a DMD. 2. The DMD should have an element (required) to specify the DomainName.
S025	Develop "compliance by inheritance" validation policy	If a software component is produced by an approved SCA-compliant tool, the desire is to not required testing again for such components. Likewise for approved core frameworks, etc.
S042	Formalize Set set up and tear down semantics	
S021	Make descriptor files platform neutral, provide option for XSD inclusion	
S023	Define SCA profiles	
S028	Develop SCA non-GPP operating environment	
S030	Integrate Naming Service and Domain Finder	
S031	Integrate all changes into SCA specification	
S036	Develop program language specific policies on libraries, exceptions and runtime typing	
S040	Develop equivalent SCA Extension for devices	
S041	Remove file operations	
S020	Develop Technology specific mappings	
S039	Develop Appendix D requirements	
S022	Introduce configurable capability within SCA constructs	
S048	Add a discussion on the soon-to-be-standard C++ Boost library	
S049	Standard way for exceptions to be defined and handled	Embedded programmers disdain native language exception handling because of the increase in memory and CPU resources.
S050	PIM definition	Attempt to define the SCA as a PIM and provide appendices for approved PSMs.
S051	One ways	Address how CORBA one-ways or C++ calls without returns should be supported.
S052	Remove Naming Service and add similar behavior like Device Mgr's registerDevice to AppFactory with registerComponent	

CP #	TITLE	NOTES
S053	Remove Device Package Descriptor XML file from Domain Profile	
S054	Allow use of Offline XML parsing tool	
S055	Clarity of service deployment that support resource interfaces	
S056	Introduce JTEL coordinated Argv language within SCA	
S057	Rework SCA Appendix B Signals Function Behavior language to be consistent with SCA Process Model	
S058	Clarify what is meant by pending service connections in the Domain Manager behavior	
S059	Establish standards and guidelines for SCA behavioral aspects.	Centered around establishing standards / best-practices guidelines for 'dynamics' -- e.g., threading policies, performance measurement requirements, etc