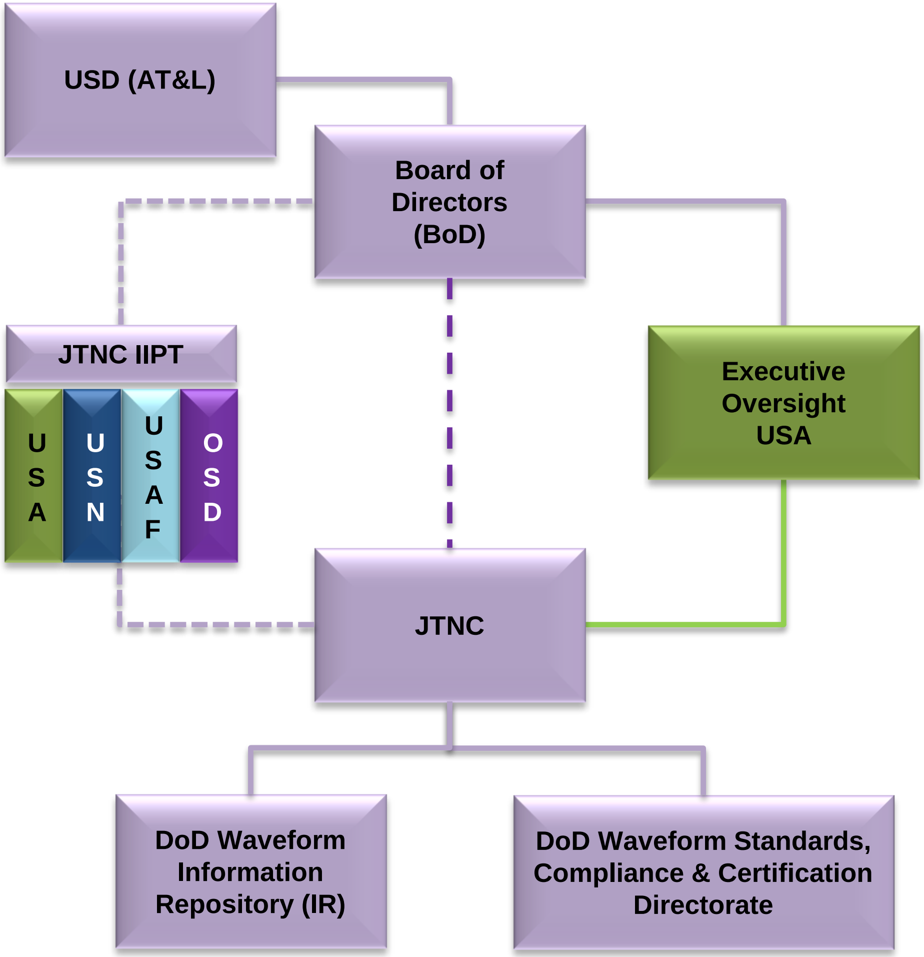
# JTNC Standards Training - Volume 1

**Department of Defense Waveform Standards, Compliance & Certification Directorate**
**19 May 2018**

# Agenda

- **JTNC Overview**
- **Software Communications Architecture (SCA) & Application Program Interfaces (APIs) Overview**
- **SCA Basics**
- **SCA Management & Control Infrastructure**
- **SCA Devices & Services**
- **SCA eXtensible Markup Language (XML) Files**
- **Application Environment Profile (AEP)**
- **SCA Example**
- **Enhancements in SCA 4.1**
- **SCA Wrap-Up**
- **REDHAWK**

# JTNC Overview


USD (AT&L)
Board of Directors (BoD)
JTNC IIPT
USA
USN
USAF
OSD
Executive Oversight USA
JTNC
DoD Waveform Information Repository (IR)
DoD Waveform Standards, Compliance & Certification Directorate

**JTNC Chartered Mission**

**To ensure interoperable, secure, and affordable waveform and wireless communications by recommending standards, conducting compliance and certification analyses in accordance with DoD policies, and maintaining a DoD Waveform Information Repository (IR)**

**JTNC Chartered Vision**

**Interoperable, secure, and affordable waveforms and wireless communications in support of Service, Multi-Service and Coalition forces**

**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# Core Functions

- **DoD Waveform Standards and Software Communications Architecture (SCA)**
  - Provides a validated open systems reference architecture that separates waveform/network manager from the radio set
  - Permits common waveform software to be deployed across multiple vendor's radio sets
- **DoD Waveform IR Management & Configuration Control**
  - Provides a cyber-hardened, DoD-wide waveform library and controlled access for waveforms and associated network managers, operating environment software, models, architectural standards and Application Program Interfaces (APIs)
  - Protects and distributes artifacts based on legal agreements between government and software developers
- **Technical Analysis of DoD Waveform IR products**
  - Compliance: preliminary characterizations regarding meeting gov't standards for interoperability and security. Assessments facilitate preparation for participation in Service-level test events
  - Certification: comprehensive characterization of Waveform IR products as to whether they meet DoD standards and policies for interoperable and secure joint tactical networking
- **Technical Advisor to JTNC Board of Directors (BoD)**
  - Provide subject matter expertise on waveforms and wireless communications as requested or identified in support of DoD, the Services, Program Offices and stakeholders
  - Support the various DoD agencies overseeing the protection of critical technologies of wireless communications exported under commercial and/or Foreign Military Sales (FMS) and licenses

# Collaborative Approach to Open Standards

- **JTNC coordinates with multiple organizations to evolve the SCA & APIs and align with industry standards in support of DoD and coalition efforts**
  - Joint Services
  - Industry
  - Standards Organizations
  - Non-U.S.

- **SCA 4.1 developed within industry forums**
  - Meetings and website portal open to the general public for collaboration on the SCA

**Coordination with industry associations aligns with JTNC's chartered core function to recommend adoption of the SCA and APIs**

# Agenda

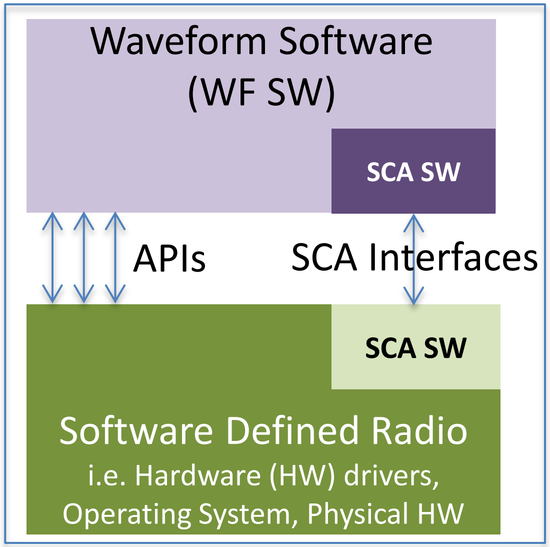
- JTNC Overview
- SCA & APIs Overview
- SCA Basics
- SCA Mgmt. & Control Infrastructure
- SCA Devices & Services
- SCA XML Files
- AEP
- APIs
- SCA Example
- Enhancements in SCA 4.1
- SCA Wrap-Up
- REDHAWK

# SCA & APIs Enable Modular Design and Designate Key Interfaces

- **Software Communications Architecture (SCA)**
  - Defines a component based framework for instantiating, configuring, and managing software running on an SCA-enabled product

- **Tactical Radio Application Programming Interfaces (APIs)**
  - Suite of interface specifications
  - Define the key software interfaces that provide an abstraction of the underlying product-specific software functionality and physical hardware

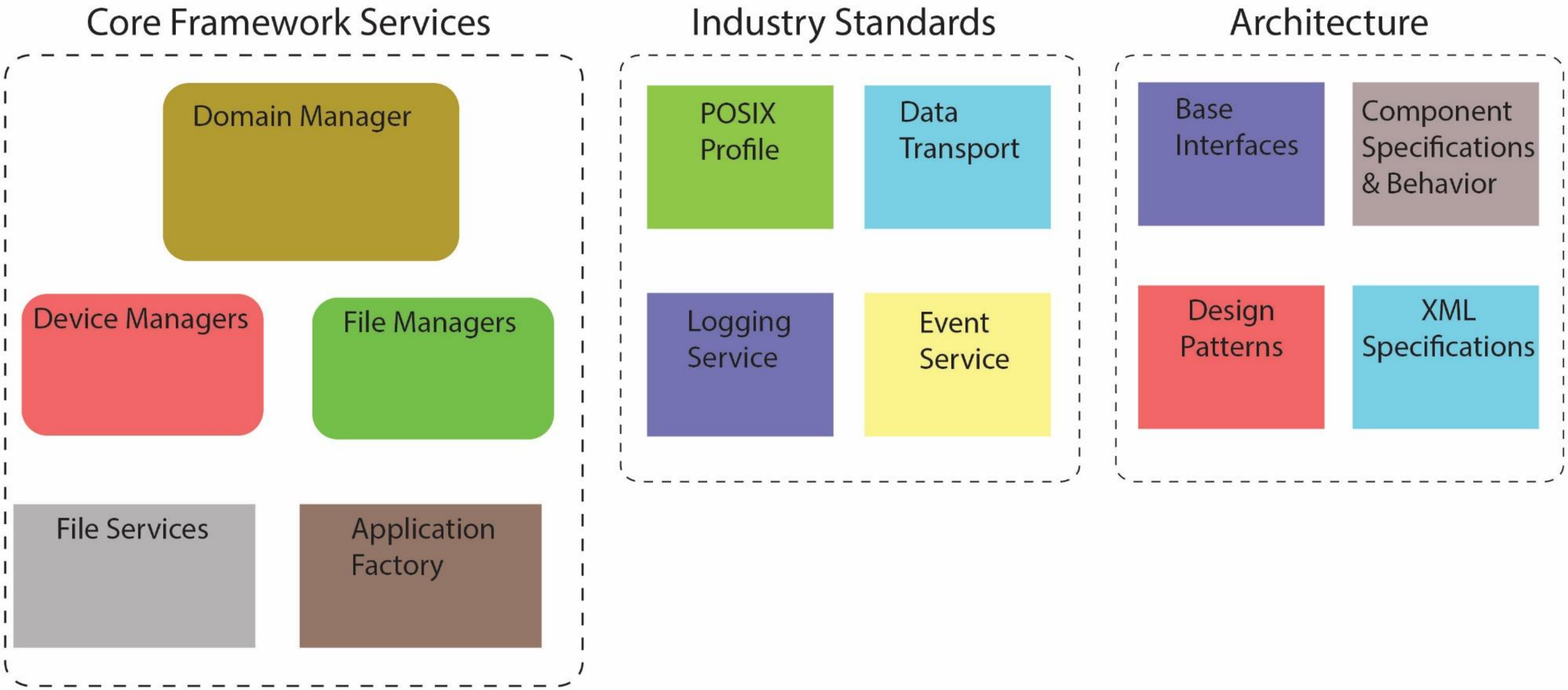
**Example of an SCA & API Enabled Product**



# Agenda

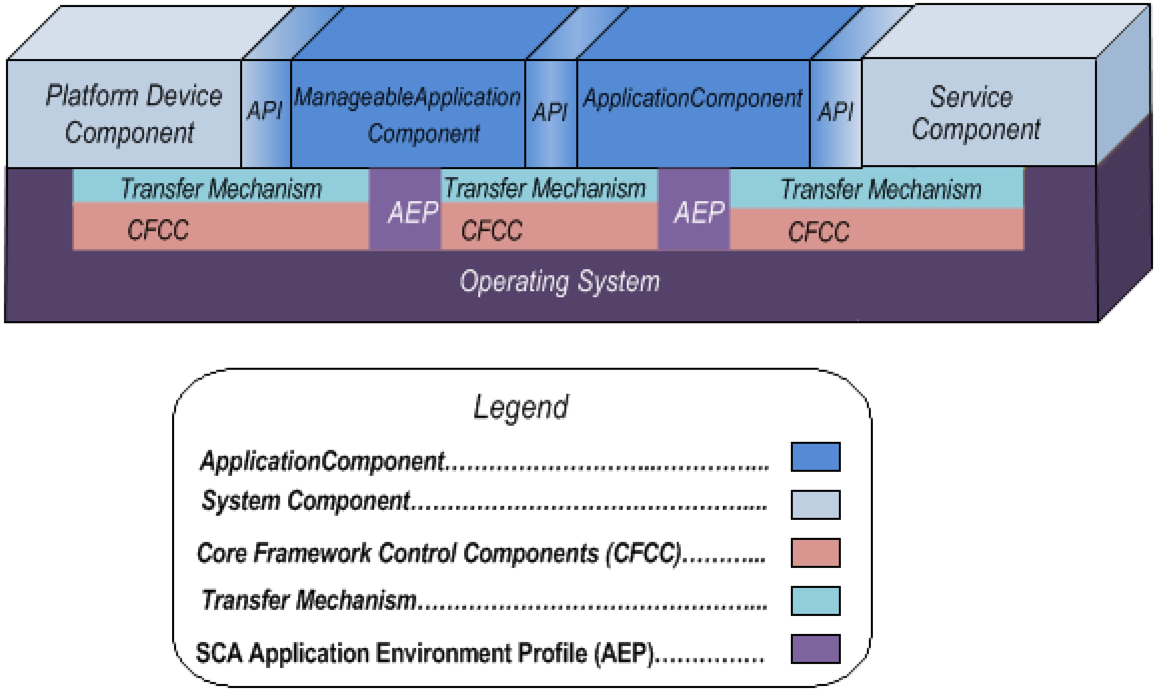
- JTNC Overview
- SCA & APIs Overview
- <u>SCA Basics</u>
- SCA Mgmt. & Control Infrastructure
- SCA Devices & Services
- SCA XML Files
- AEP
- SCA Example
- Enhancements in SCA 4.1
- SCA Wrap-Up
- REDHAWK

# Three Pieces of the Software Communications Architecture (SCA)



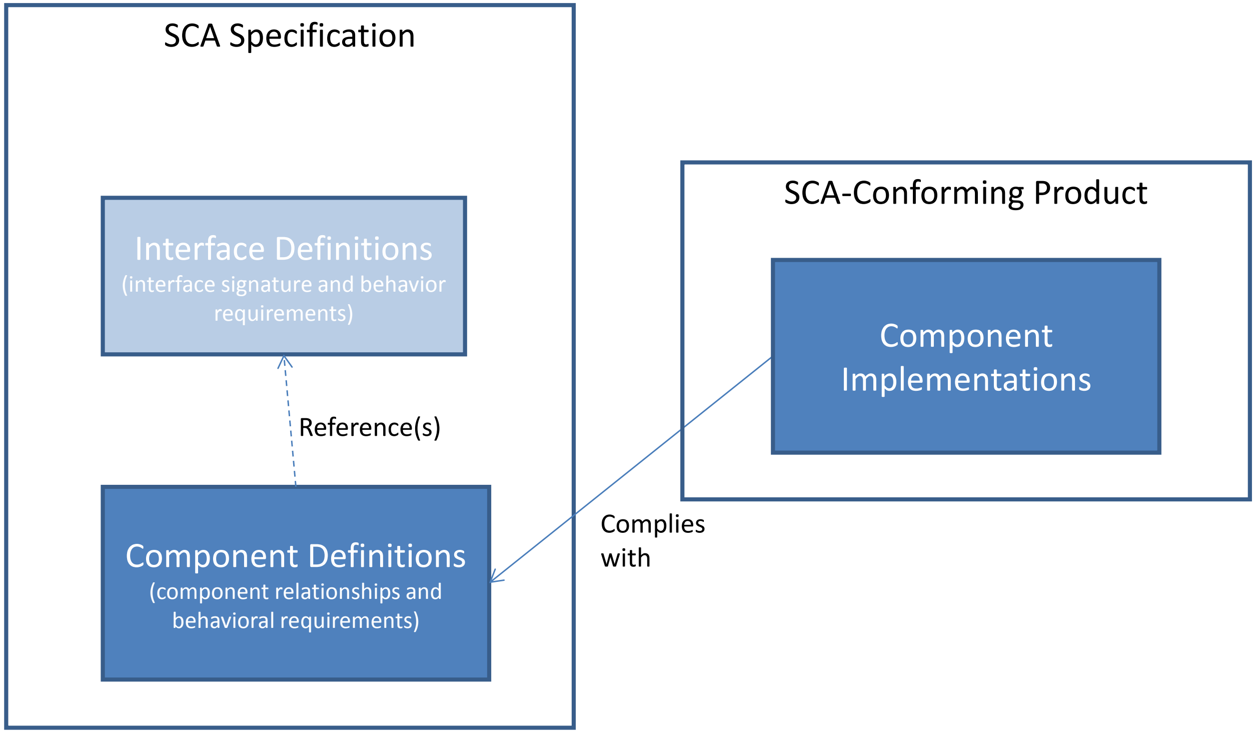
- SCA defines
  - Specific software services to be present
  - Industry standard software on the radio – such as a Portable Operating System Interface (POSIX) operating system and the data transport
  - Basic interfaces and architecture

# SCA 4.1 Layer Model


Platform Device Component
API
ManageableApplication Component
API
ApplicationComponent
API
Service Component
Transfer Mechanism
CFCC
AEP
Transfer Mechanism
CFCC
AEP
Transfer Mechanism
CFCC
Operating System
Legend
ApplicationComponent..........................................
System Component.............................................
Core Framework Control Components (CFCC)...........
Transfer Mechanism.............................................
SCA Application Environment Profile (AEP)...............

- SCA AEP specifies the capabilities of a run-time environment
- Transfer Mechanism permits transports other than Common Object Request Broker Architecture (CORBA)

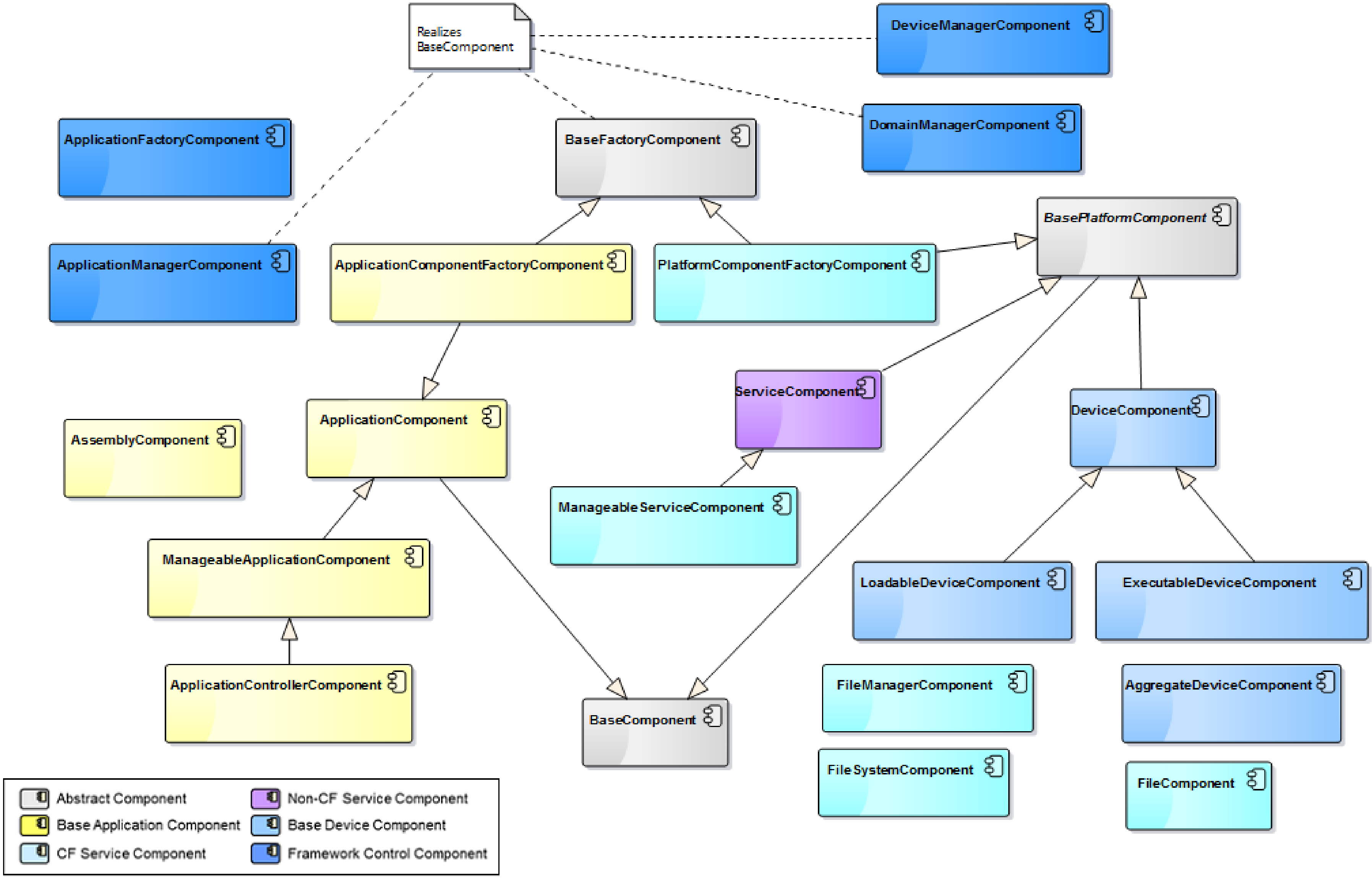
# Component Model of the SCA


SCA Specification
Interface Definitions
(interface signature and behavior requirements)
Reference(s)
Component Definitions
(component relationships and behavioral requirements)
SCA-Conforming Product
Component Implementations
Complies with

- The latest release of the SCA, 4.1, distinguishes between interfaces, component definitions, and component implementations

**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# SCA Component Hierarchy



**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# Manager Component Profiles

Lightweight Profile

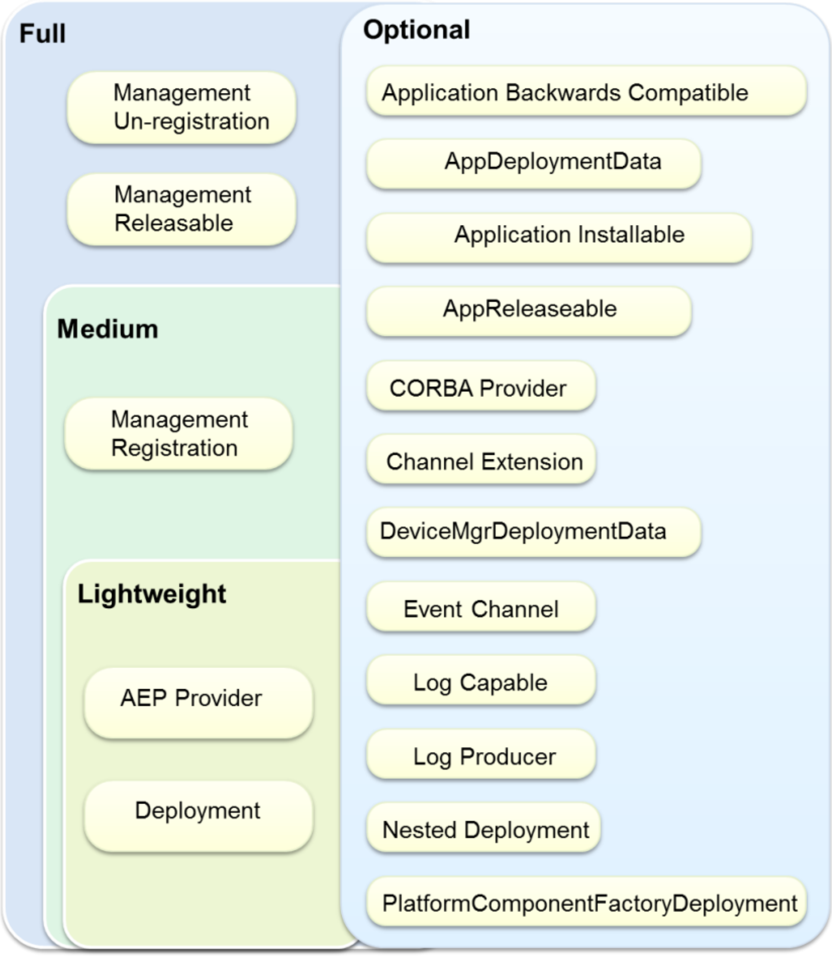
- Provides a minimum set of functionality which is applicable for resource constrained platforms

Medium Profile

- Lightweight, but it introduces a configurable, dynamic aspect to the platform
- May be the most flexible one in that it provides the lightest weight realization of a platform that supports the deployment model introduced in earlier SCA versions

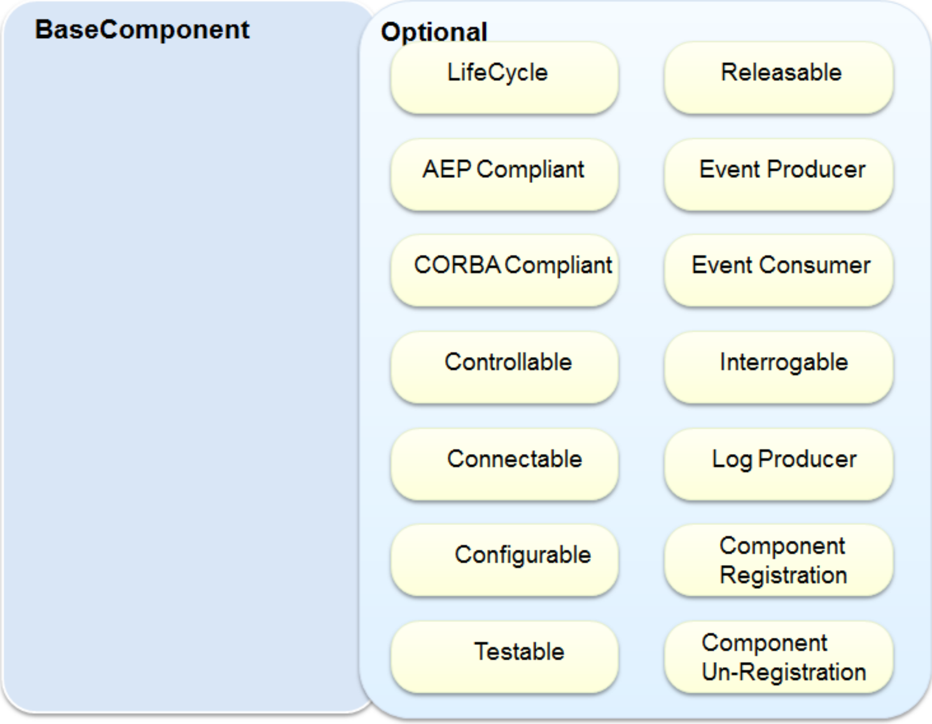
Full Profile

- Provides the full breadth of SCA deployment and management capabilities
- Aligned to support prime power, multi-channel sets



**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# Base Components Units of Functionality

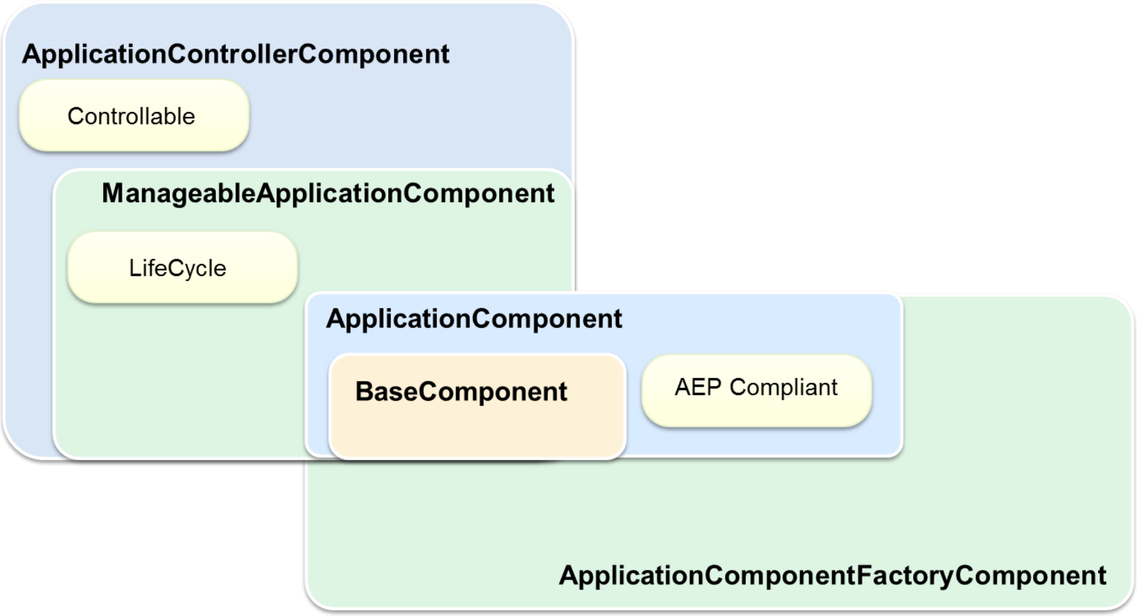


- If a product doesn't want to implement a testable component, then that particular interface is not included in the components
  - Avoids having to 'stub out' functionality that is not implemented
  - Reduces the number of requirements for the software

# Base Units of Functionality

| | |
|---|---|
| • **LifeCycle** | – provides life cycle management capability via the LifeCycle interface |
| • **AEP Compliant** | – adheres to one of the AEPs defined in SCA Appendix B |
| • **Controllable** | – provides a control capability via the ControllableInterface interface |
| • **CORBA Compliant** | – provides a CORBA communication capability that adheres to one of the CORBA profiles defined in SCA Appendix E |
| • **Connectable** | – provides port connection management via the PortAccessor interface |
| • **Configurable** | – provides configure and/or query functionality via the PropertySet interface, or defines configure and/or query properties |
| • **Event Consumer** | – consumes events using the PushConsumer interface |
| • **Event Producer** | – produces events using the PushSupplier interface |
| • **Interrogable** | – provides interrogation capability via a component specified interface (i.e. DeploymentAttributes, ComponentIdentifier, DeviceManagerAttributes, or DeviceAttributes) |
| • **Log Producer** | – produces logs using the LogProducer interface |
| • **Component Registration** | – provides a registration capability via the ComponentRegistry interface |
| • **Component Un-Registration** | – provides an un-registration capability via the FullComponentRegistry interface |
| • **Releasable** | – provides a release capability via the Lifecycle interface |
| • **Testable** | – provides a testing capability via the TestableInterface interface and test properties |

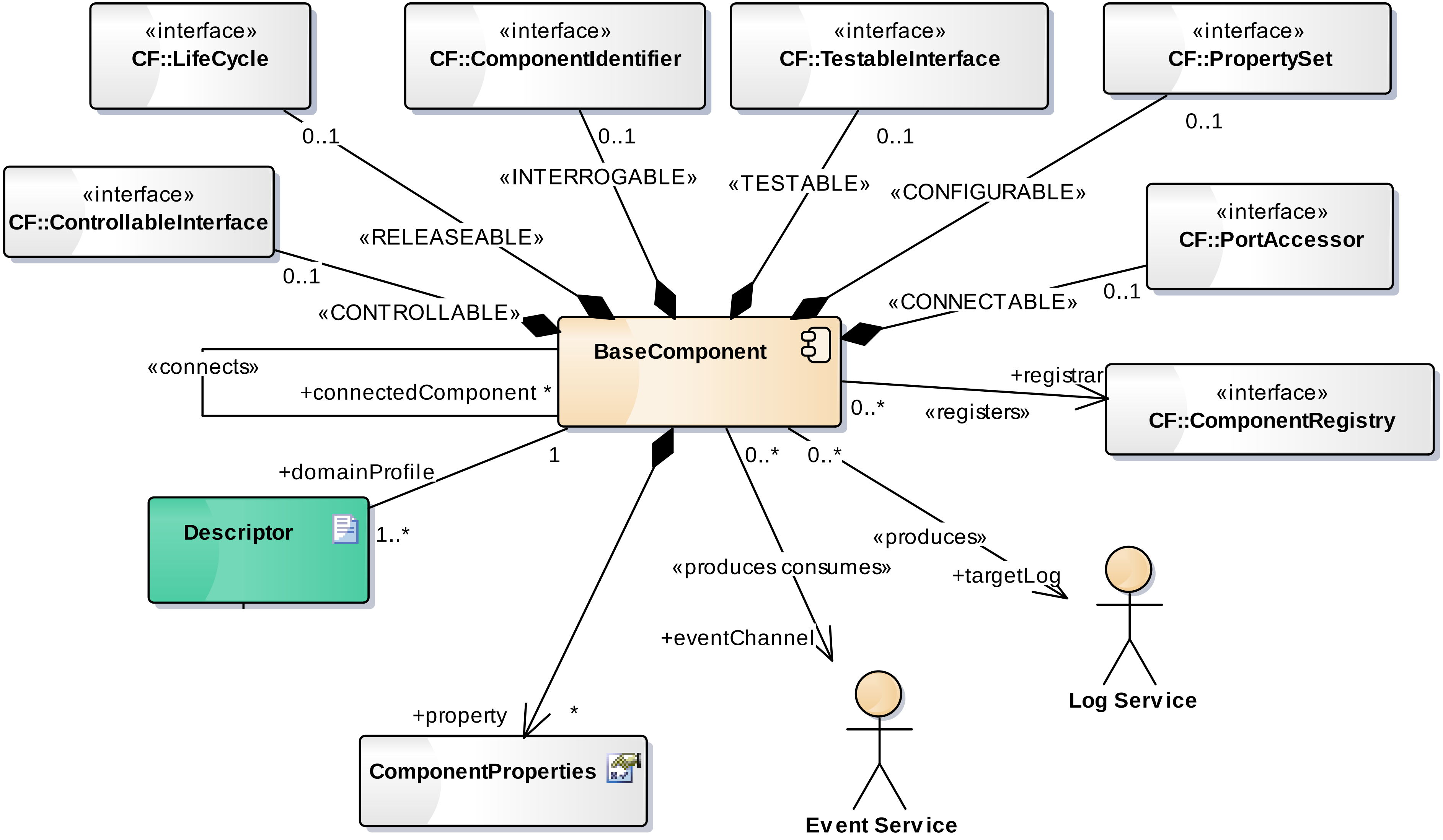
# Application Related Component Units of Functionality (UOF)


ApplicationControllerComponent
Controllable
ManageableApplicationComponent
LifeCycle
ApplicationComponent
BaseComponent
AEP Compliant
ApplicationComponentFactoryComponent

- **ApplicationComponent** extends BaseComponent UOFs with a mandatory AEP UOF
- **ManageableApplicationComponent** extends ApplicationComponent with a mandatory LifeCycle UOF
- **ApplicationControllerComponent** extends ManageableApplicationComponent with a mandatory Controllable UOF

**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# Interfaces for an SCA Software Component



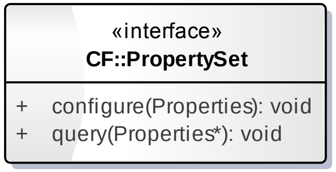
# Property Set Interface

```
struct
DataType
{
  string id;
  any value;
};
```

**DataType** defines a structure, which may be used to specify an id and value pair

The **Properties** is an IDL unbounded sequence of CF::DataType(s), which is used in defining a sequence of id and value pairs.

```
typedef sequence <DataType> Properties;
```



«interface»
**CF::PropertySet**

+ configure(Properties): void
+ query(Properties*): void

The ***configure*** operation shall assign the id/value pair passed in the configProperties argument to the component.

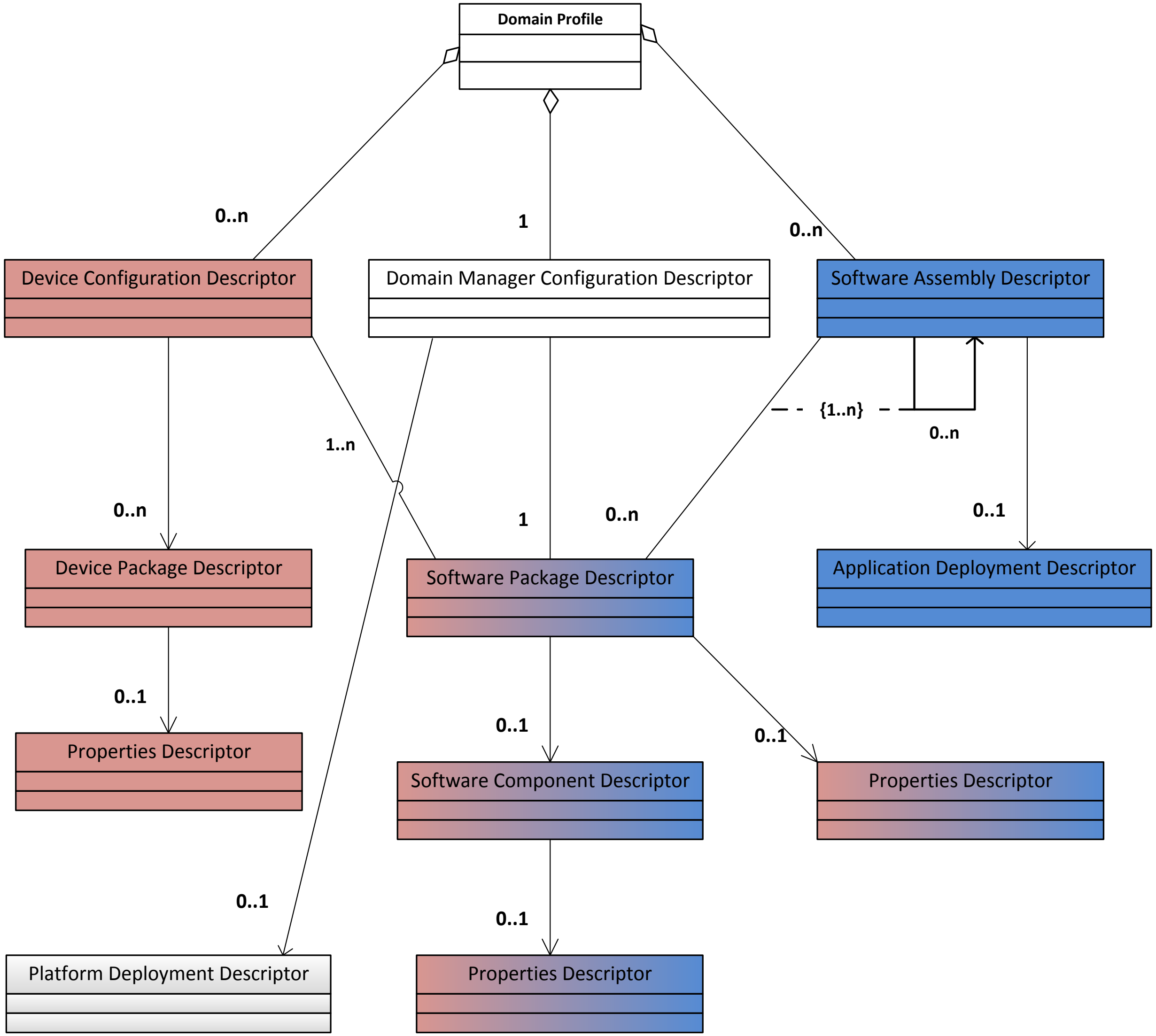
The ***query*** operation shall retrieve a component's requested configuration data.

- If the configProperties are zero size then all component properties shall be returned
- If the configProperties are not zero size then only those id/value pairs specified in the configProperties shall be returned

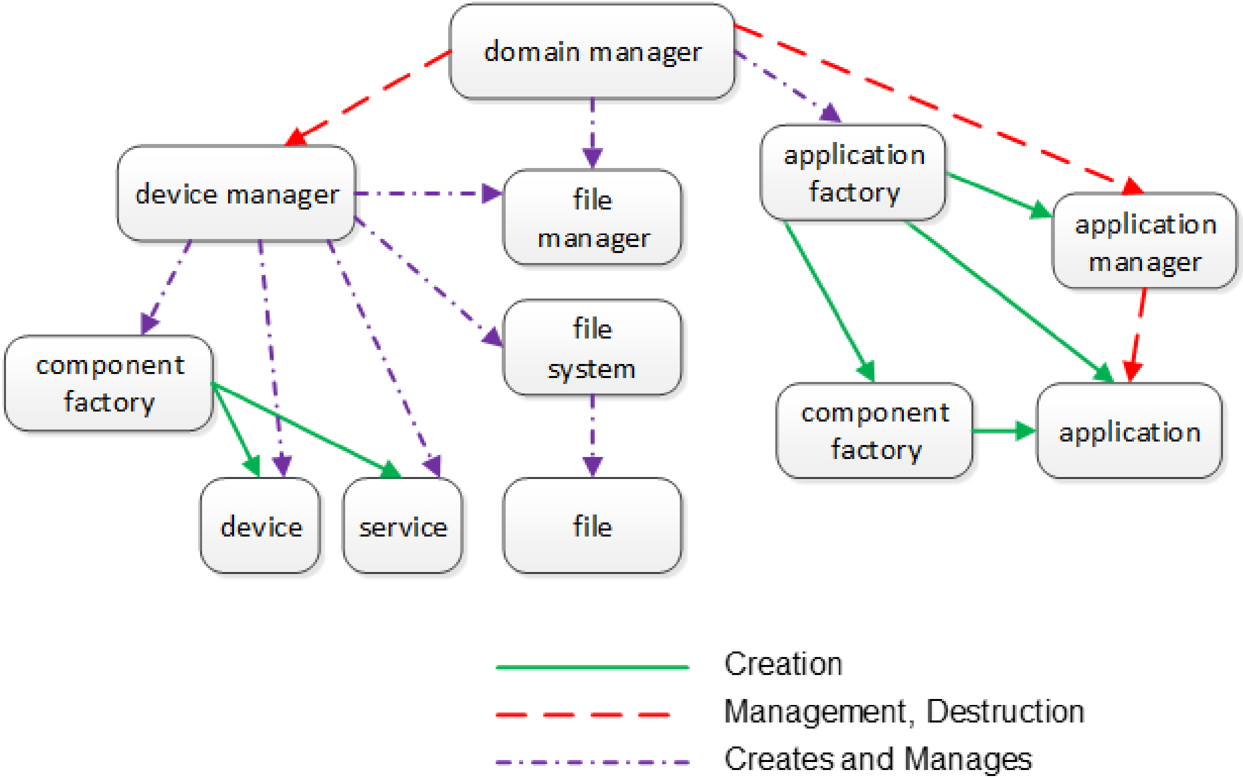
# Software Packages (the Metadata)

SCA application, component, device, service, etc., has XML artifacts which assist the core framework in:

- Connecting the objects
- Launching the object
- Configuring the object


Domain Profile
0..n
1
0..n
Device Configuration Descriptor
Domain Manager Configuration Descriptor
Software Assembly Descriptor
{1..n}
0..n
1..n
0..n
1
0..n
0..1
Device Package Descriptor
Software Package Descriptor
Application Deployment Descriptor
0..1
0..1
0..1
Properties Descriptor
Software Component Descriptor
Properties Descriptor
0..1
0..1
Platform Deployment Descriptor
Properties Descriptor

# SCA Management Hierarchy



- Domain Manager is the central registry for the entire radio
- Device Manager is responsible for managing all devices and services on a processor
- A feature of the SCA is the distributed file system

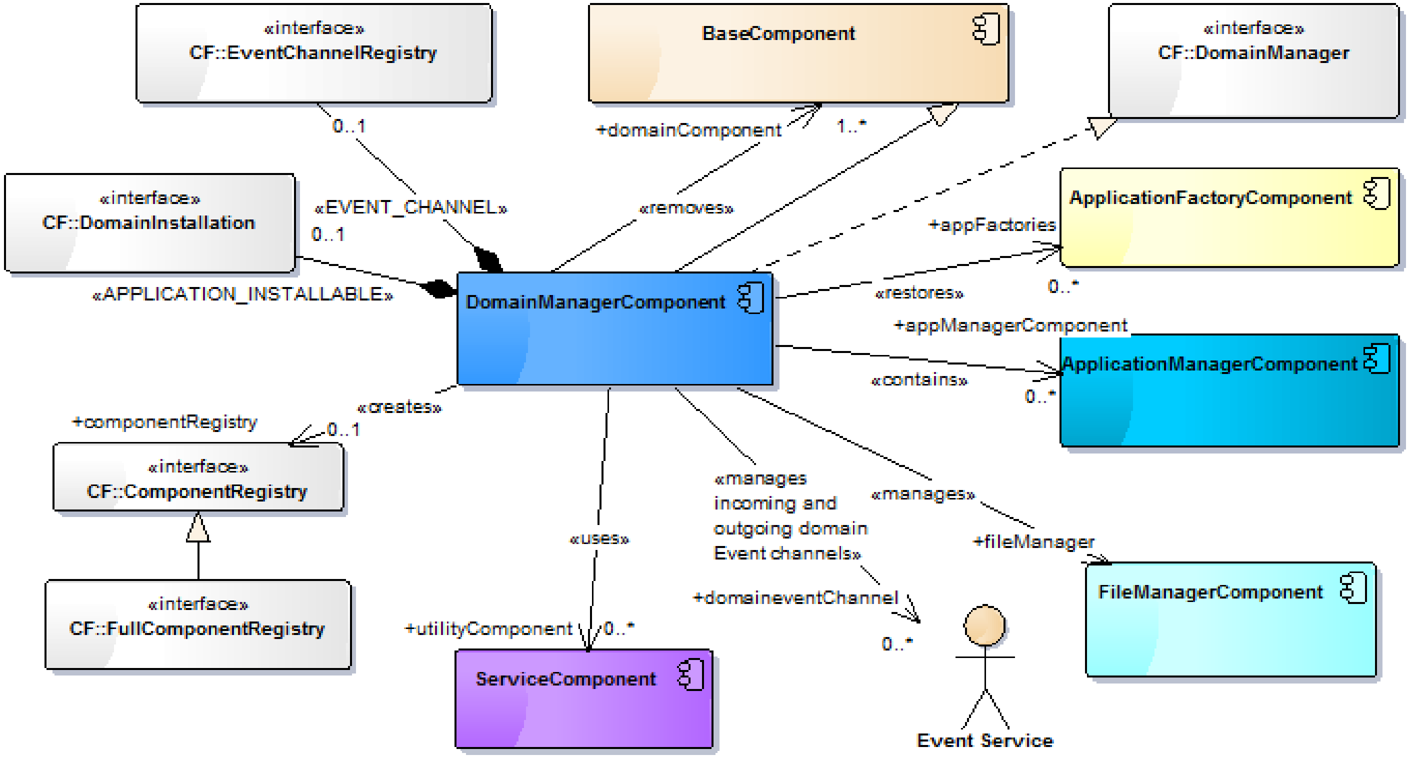
# Agenda

- JTNC Overview
- SCA & APIs Overview
- SCA Basics
- <u>SCA Mgmt. & Control Infrastructure</u>
- SCA Devices & Services
- SCA XML Files
- AEP
- SCA Example
- Enhancements in SCA 4.1
- SCA Wrap-Up
- REDHAWK

# SCA’s Domain Manager

The Domain Manager has three distinct functions:

1) Registration
   - Keeps track of its hardware and software resources
2) Host
   - Allows configuration by the network or operator and provides status
3) Administration
   - Access interfaces of device managers and domain manager's file manager

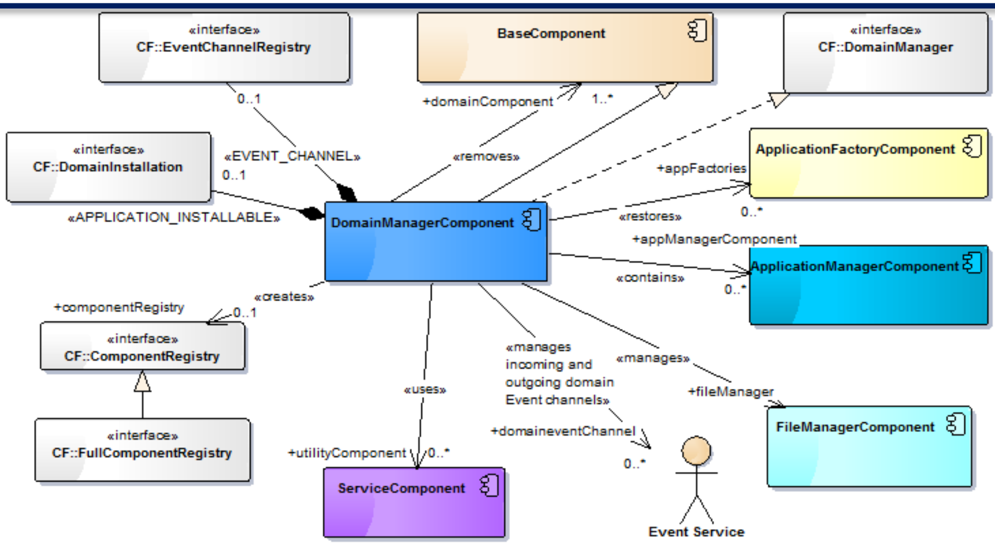


# More About the Domain Manager

A DomainManagerComponent:

- Creates event channels, sends event messages to event channels, disconnects producers and consumers from event channels and may destroy event channels

- Creates and manages FileManagerComponents within the domain



- Restores any ApplicationFactoryComponent(s) instantiated in previous incarnations of the domain

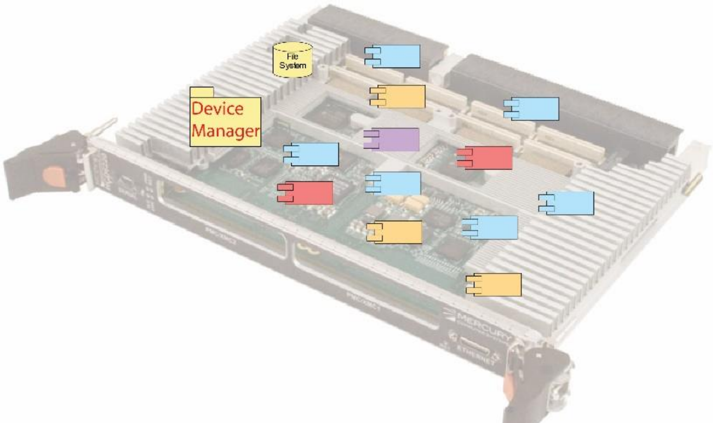
- Contains a set of instantiated applications

- Removes and disconnects, as necessary, components registered within the domain

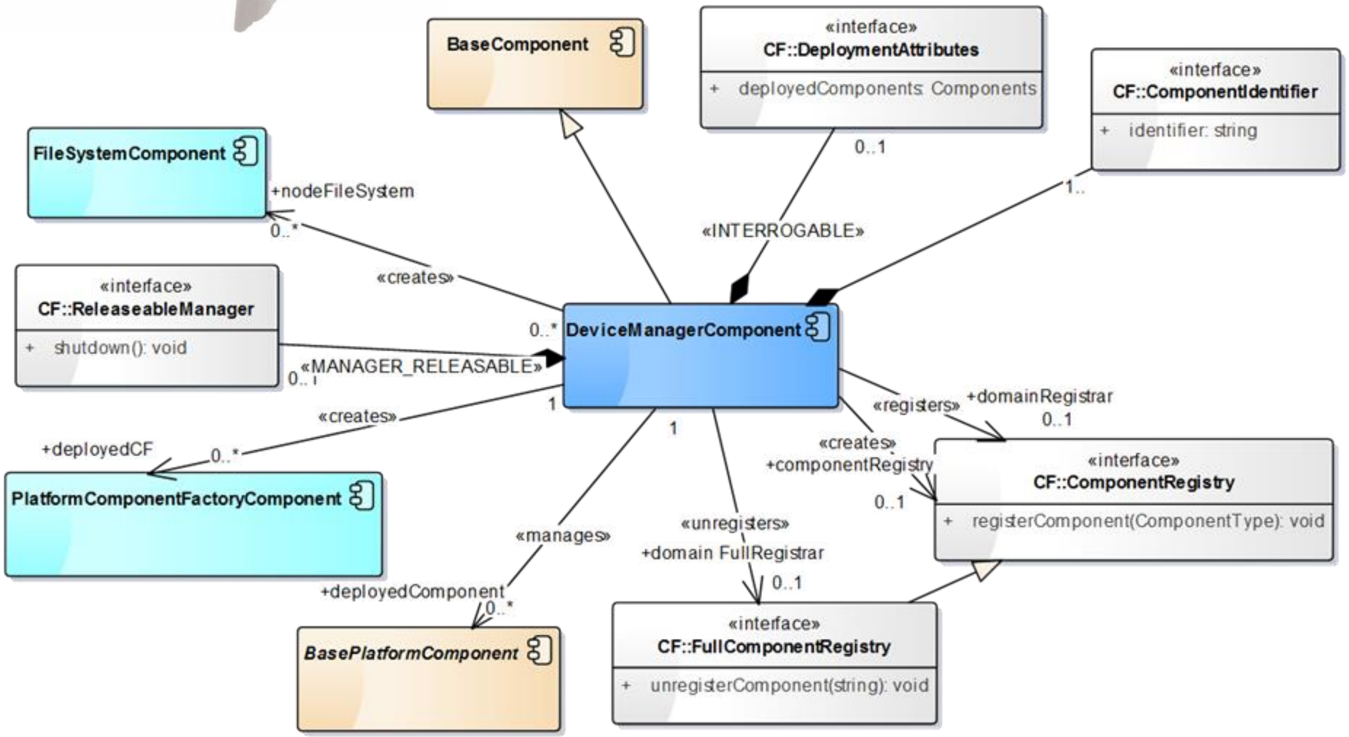
- Utilizes the capabilities provided by ServiceComponent(s) within the domain

- Creates componentRegistries that contain an inventory of the registered components

# SCA’s Device Manager



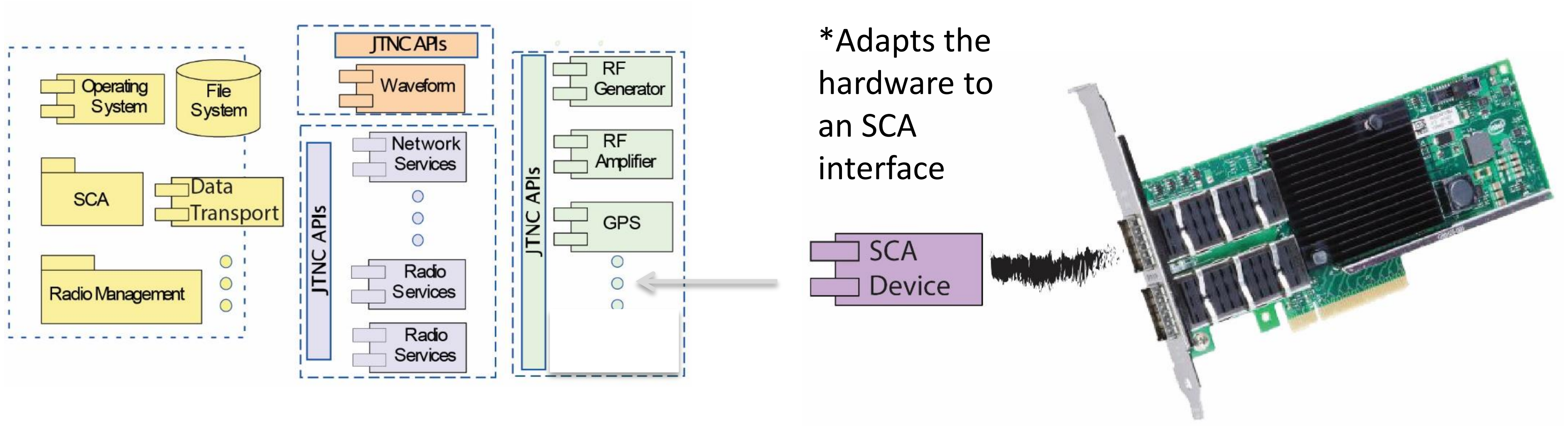
- Device Manager is a delegation of management and control to a single processor
- Creates component registries, file systems, and services upon startup
- As commanded by the Domain Manager, it will create, load, and execute waveform components



# Agenda

- JTNC Overview
- SCA & APIs Overview
- SCA Basics
- SCA Mgmt. & Control Infrastructure
- SCA Devices & Services
- SCA XML Files
- AEP
- SCA Example
- Enhancements in SCA 4.1
- SCA Wrap-Up
- REDHAWK

# SCA Devices

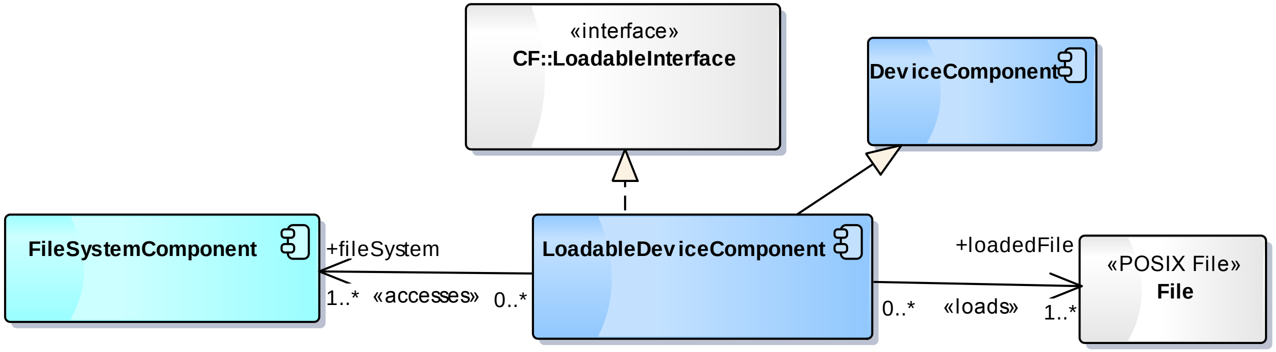


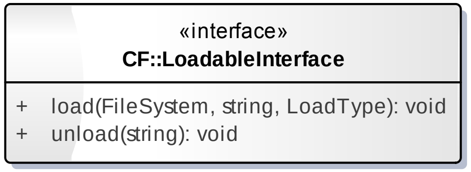
*Adapts the hardware to an SCA interface

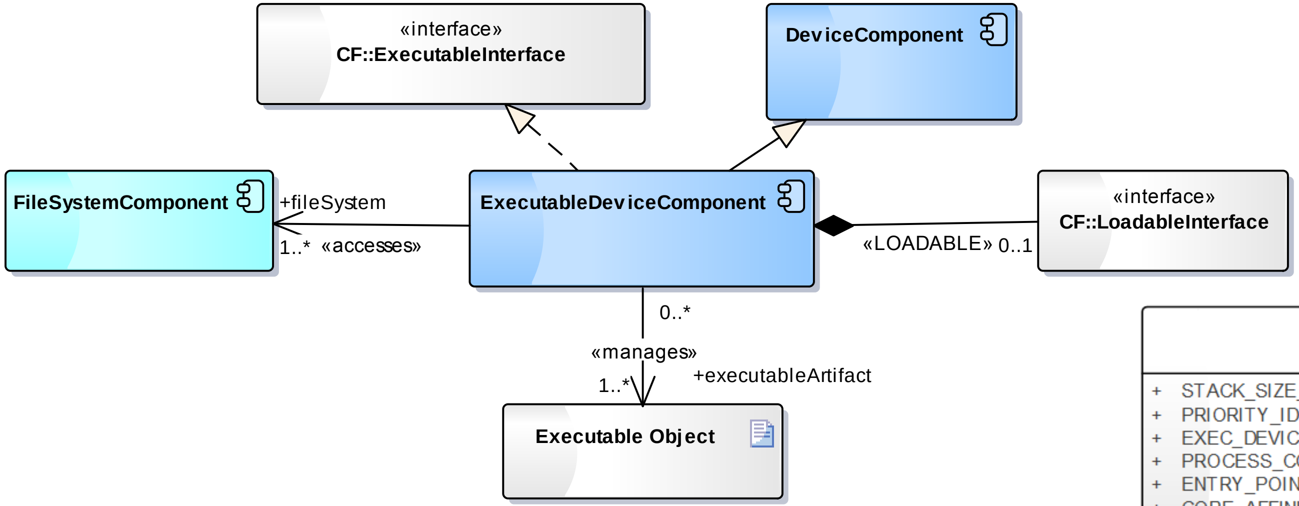
- A Device is a software proxy for a hardware component in the product, e.g.
  - Processors that execute software
  - GPS
  - Ethernet, etc.

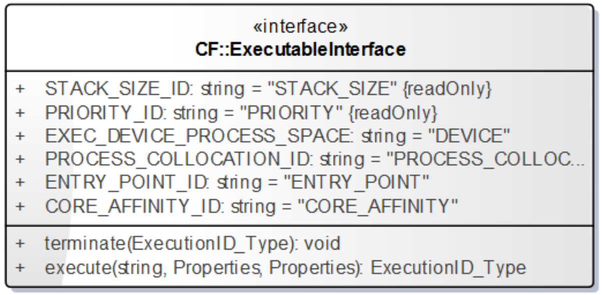
- Capacity management permits the product to use a processor or other device for more than a single waveform

# Additional Types of Devices



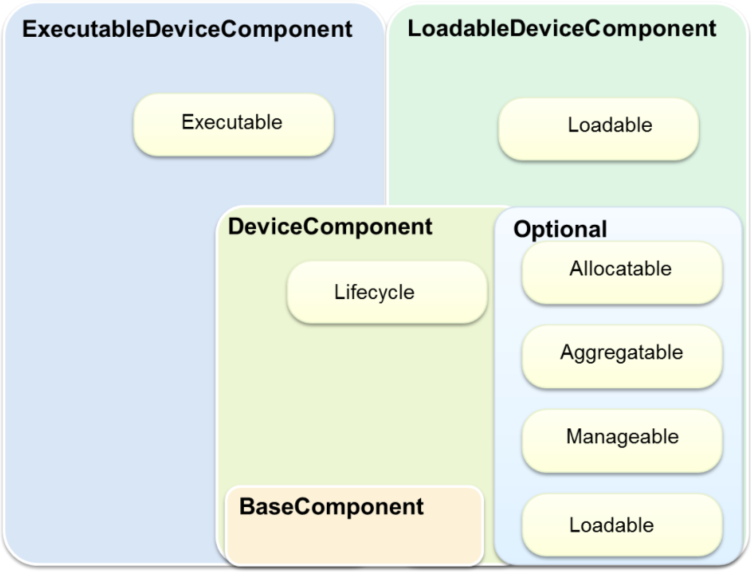





«interface»
CF::ExecutableInterface
+ STACK_SIZE_ID: string = "STACK_SIZE" {readOnly}
+ PRIORITY_ID: string = "PRIORITY" {readOnly}
+ EXEC_DEVICE_PROCESS_SPACE: string = "DEVICE"
+ PROCESS_COLLOCATION_ID: string = "PROCESS_COLLOC...
+ ENTRY_POINT_ID: string = "ENTRY_POINT"
+ CORE_AFFINITY_ID: string = "CORE_AFFINITY"
+ terminate(ExecutionID_Type): void
+ execute(string, Properties, Properties): ExecutionID_Type

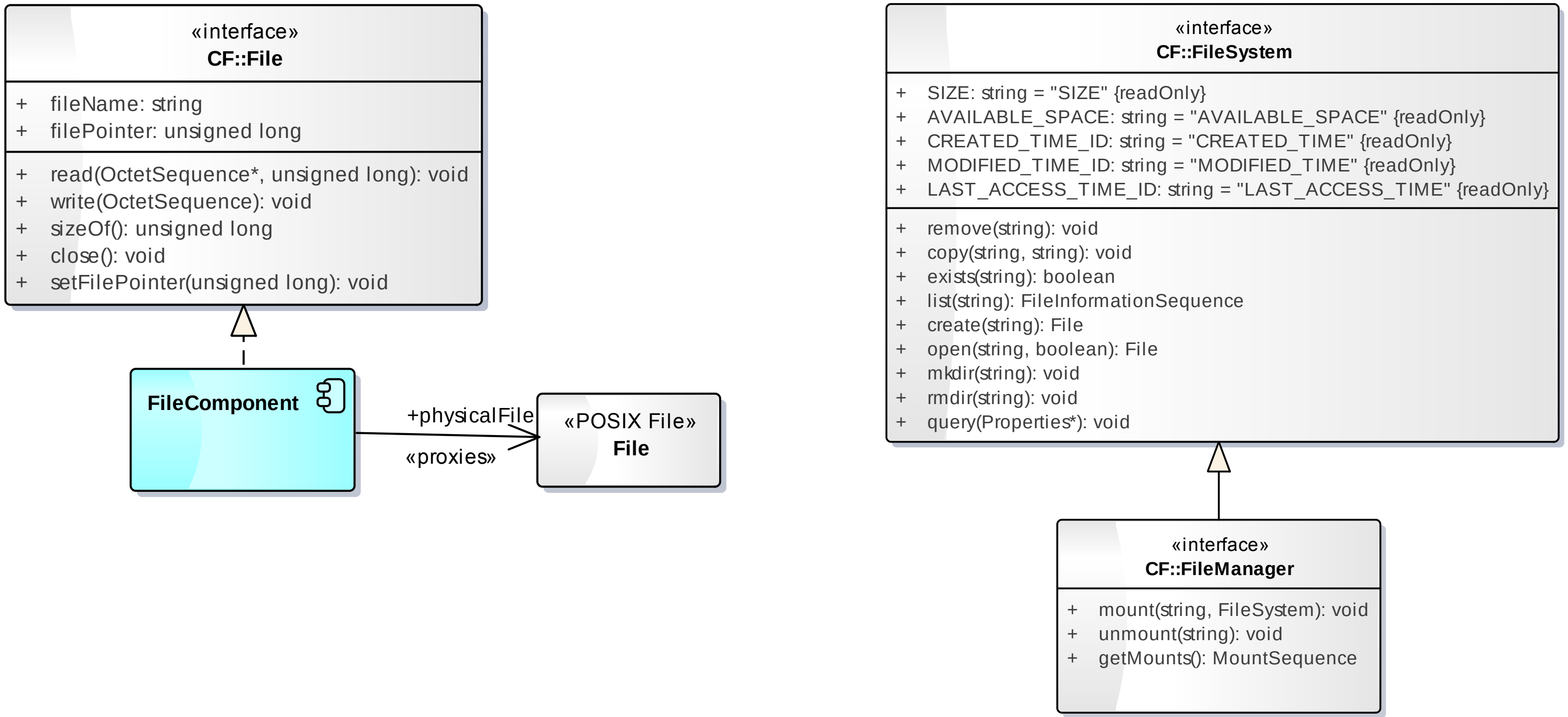
The different devices provide better scalability for the SCA instead of a one-size fits all.

# DeviceComponent Units of Functionality



| | |
|---|---|
| • Loadable | – provides the load management capability via the LoadableInterface interface |
| • Executable | – provides an execution management capability via the ExecutableInterface interface |
| • Aggregatable | – provides an aggregation capability via the AggregateDeviceAttributes interface; a parent composite device with children devices |
| • Allocatable | – provides capacity management via the CapacityManager interface and allocation properties that are managed along with usage state |
| • Manageable | – provides administration capability via the AdministratableInterface interface and administrative state behavior |

# SCA's File Services


«interface»
CF::File
+ fileName: string
+ filePointer: unsigned long
+ read(OctetSequence*, unsigned long): void
+ write(OctetSequence): void
+ sizeOf(): unsigned long
+ close(): void
+ setFilePointer(unsigned long): void
FileComponent
+physicalFile
«proxies»
«POSIX File»
File
«interface»
CF::FileSystem
+ SIZE: string = "SIZE" {readOnly}
+ AVAILABLE_SPACE: string = "AVAILABLE_SPACE" {readOnly}
+ CREATED_TIME_ID: string = "CREATED_TIME" {readOnly}
+ MODIFIED_TIME_ID: string = "MODIFIED_TIME" {readOnly}
+ LAST_ACCESS_TIME_ID: string = "LAST_ACCESS_TIME" {readOnly}
+ remove(string): void
+ copy(string, string): void
+ exists(string): boolean
+ list(string): FileInformationSequence
+ create(string): File
+ open(string, boolean): File
+ mkdir(string): void
+ rmdir(string): void
+ query(Properties*): void
«interface»
CF::FileManager
+ mount(string, FileSystem): void
+ unmount(string): void
+ getMounts(): MountSequence

- File interface provides the ability to read/write files residing within an SCA compliant, distributed file system (modeled after the POSIX/C file interface)
- FileSystem interface defines operations that enable remote access to a physical file system
- Multiple, distributed file systems may be accessed through a file manager

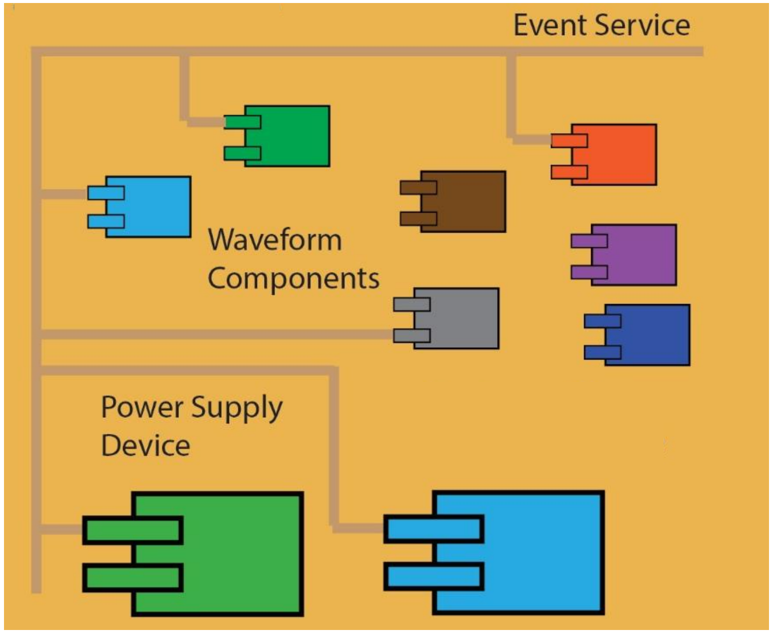
# Log Services

Originally, the SCA defined its own unique log service. We decided later to reference the Object Management Group (OMG) Lightweight Log Service Specification.

```
2015-04-29 18:19:44 TRACE prop_utils:372 - Checking property DCE:2df4cfe4-675c-41ec-9cc8-84dff2f468b3 processor_cores
2015-04-29 18:19:44 TRACE prop_utils:372 - Checking property DCE:8dcef419-b440-4bcf-b893-cab79b6024fb memCapacity
2015-04-29 18:19:44 TRACE prop_utils:372 - Checking property DCE:506102d6-04a9-4532-9420-a323d818ddec mcastnicIngressCapacity
2015-04-29 18:19:44 TRACE prop_utils:372 - Checking property DCE:eb08e43f-11c7-45a0-8750-edff439c8b24 mcastnicEgressCapacity
2015-04-29 18:19:44 TRACE prop_utils:372 - Checking property DCE:3bf07b37-0c00-4e2a-8275-52bd4e391f07 loadCapacityPerCore
2015-04-29 18:19:44 TRACE prop_utils:372 - Checking property DCE:72c1c4a9-2bcf-49c5-bafd-ae2c1d567056 loadCapacity
2015-04-29 18:19:44 TRACE AllocationManager_impl:365 - Device did not match requested processor
```

# Event Services

- Within the SCA, an Event Service is a publish-and-subscribe interface
- The interfaces are defined as the *PushConsumer* and *PushSupplier* interfaces of the CosEventComm module as described in OMG Event Service Specification
- Two channels are defined: Incoming Domain Management and Outgoing Domain Management
- Other channels can be defined by the product integrator

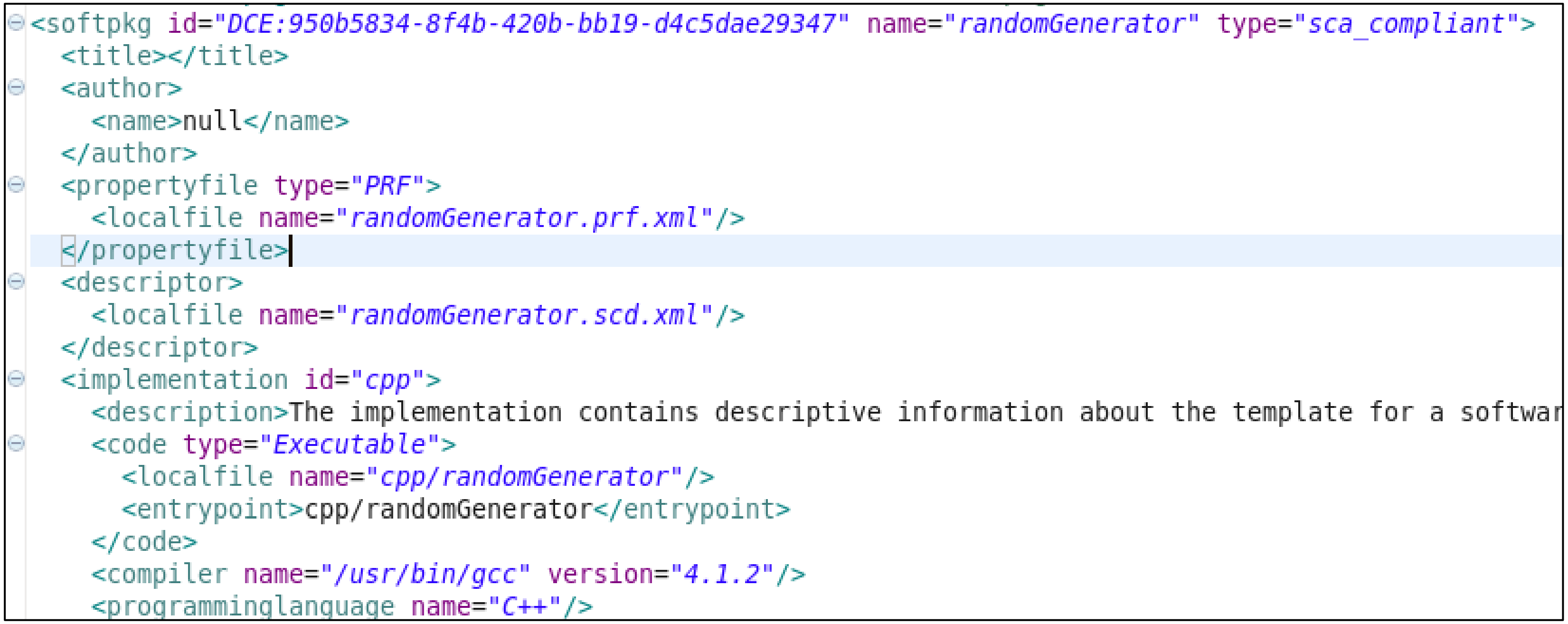


# Agenda

- **JTNC Overview**
- **SCA & APIs Overview**
- **SCA Basics**
- **SCA Mgmt. & Control Infrastructure**
- **SCA Devices & Services**
- **<u>SCA XML Files</u>**
- **AEP**
- **SCA Example**
- **Enhancements in SCA 4.1**
- **SCA Wrap-Up**
- **REDHAWK**

# Software Package Descriptor (SPD)

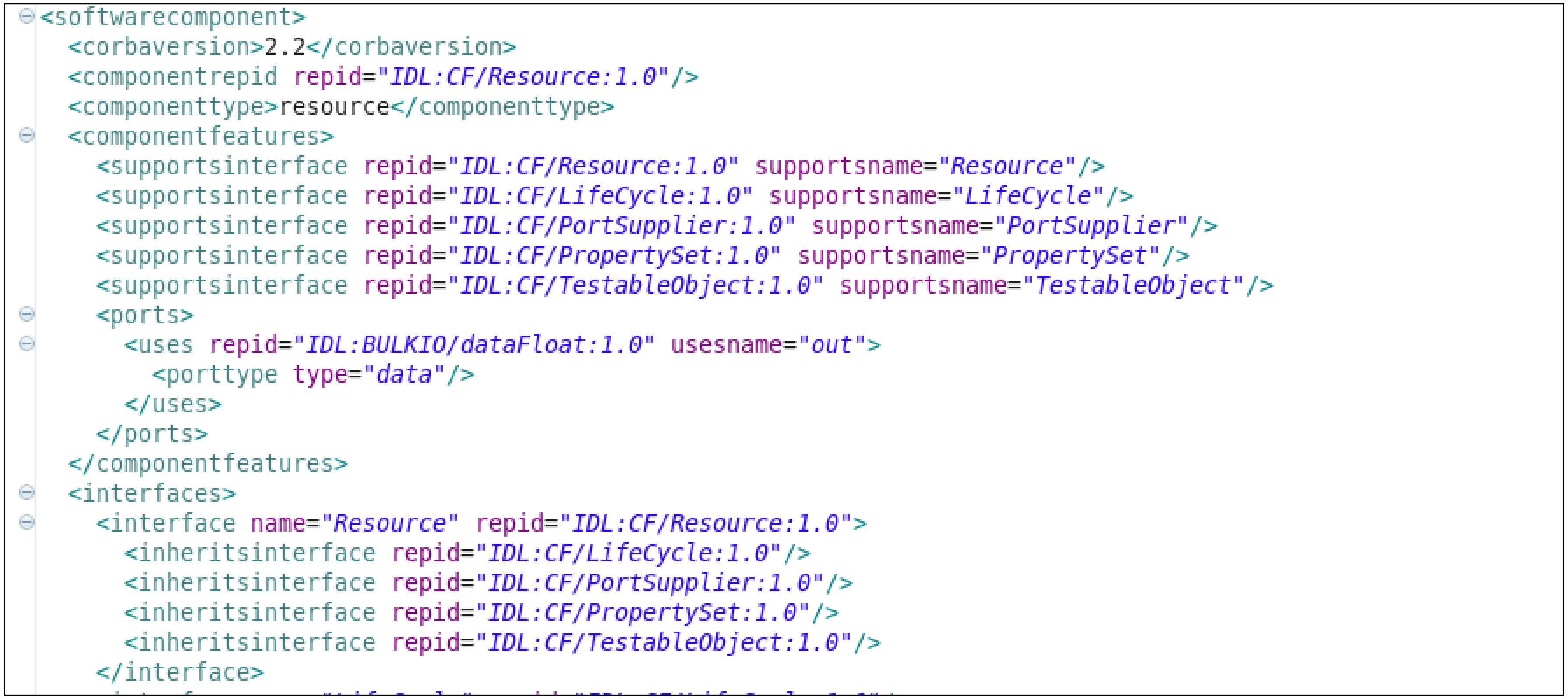
- SPD identifies a software component implementation(s) by providing general information about a software package, e.g.
  - Name
  - Author
  - Property file
  - Code implementation information
  - HW and/or SW dependencies



```
<softpkg id="DCE:950b5834-8f4b-420b-bb19-d4c5dae29347" name="randomGenerator" type="sca_compliant">
  <title></title>
  <author>
    <name>null</name>
  </author>
  <propertyfile type="PRF">
    <localfile name="randomGenerator.prf.xml"/>
  </propertyfile>
  <descriptor>
    <localfile name="randomGenerator.scd.xml"/>
  </descriptor>
  <implementation id="cpp">
    <description>The implementation contains descriptive information about the template for a softwar
    <code type="Executable">
      <localfile name="cpp/randomGenerator"/>
      <entrypoint>cpp/randomGenerator</entrypoint>
    </code>
    <compiler name="/usr/bin/gcc" version="4.1.2"/>
    <programminglanguage name="C++"/>
```

# Software Component Descriptor (SCD)

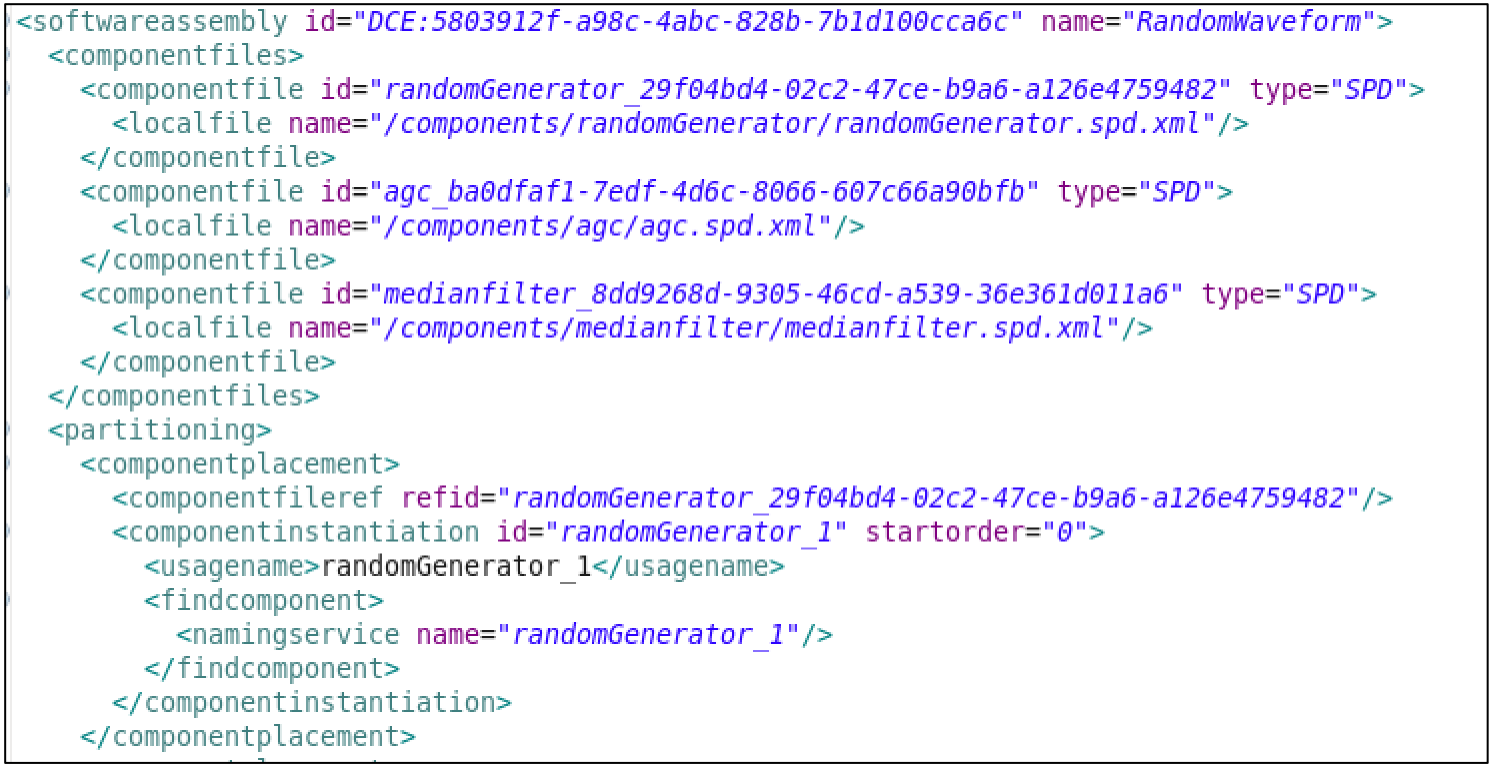
- SCD contains information about a specific SCA software component, e.g.
  - Interfaces that a component provides and/or uses



```
<softwarecomponent>
  <corbaversion>2.2</corbaversion>
  <componentrepid repid="IDL:CF/Resource:1.0"/>
  <componenttype>resource</componenttype>
  <componentfeatures>
    <supportsinterface repid="IDL:CF/Resource:1.0" supportsname="Resource"/>
    <supportsinterface repid="IDL:CF/LifeCycle:1.0" supportsname="LifeCycle"/>
    <supportsinterface repid="IDL:CF/PortSupplier:1.0" supportsname="PortSupplier"/>
    <supportsinterface repid="IDL:CF/PropertySet:1.0" supportsname="PropertySet"/>
    <supportsinterface repid="IDL:CF/TestableObject:1.0" supportsname="TestableObject"/>
    <ports>
      <uses repid="IDL:BULKIO/dataFloat:1.0" usesname="out">
        <porttype type="data"/>
      </uses>
    </ports>
  </componentfeatures>
  <interfaces>
    <interface name="Resource" repid="IDL:CF/Resource:1.0">
      <inheritsinterface repid="IDL:CF/LifeCycle:1.0"/>
      <inheritsinterface repid="IDL:CF/PortSupplier:1.0"/>
      <inheritsinterface repid="IDL:CF/PropertySet:1.0"/>
      <inheritsinterface repid="IDL:CF/TestableObject:1.0"/>
    </interface>
```

# Software Assembly Descriptor (SAD)

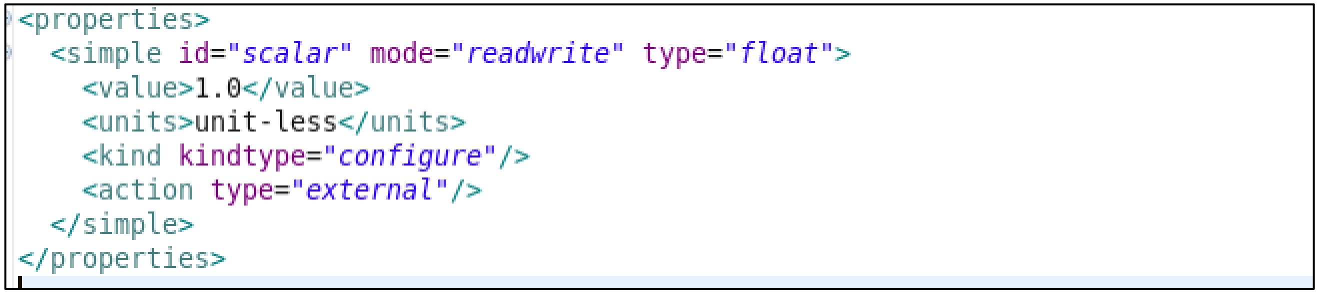
- SAD contains information about the components that make up an application
  - Application factory uses this information when creating an application



```
<softwareassembly id="DCE:5803912f-a98c-4abc-828b-7b1d100cca6c" name="RandomWaveform">
  <componentfiles>
    <componentfile id="randomGenerator_29f04bd4-02c2-47ce-b9a6-a126e4759482" type="SPD">
      <localfile name="/components/randomGenerator/randomGenerator.spd.xml"/>
    </componentfile>
    <componentfile id="agc_ba0dfaf1-7edf-4d6c-8066-607c66a90bfb" type="SPD">
      <localfile name="/components/agc/agc.spd.xml"/>
    </componentfile>
    <componentfile id="medianfilter_8dd9268d-9305-46cd-a539-36e361d011a6" type="SPD">
      <localfile name="/components/medianfilter/medianfilter.spd.xml"/>
    </componentfile>
  </componentfiles>
  <partitioning>
    <componentplacement>
      <componentfileref refid="randomGenerator_29f04bd4-02c2-47ce-b9a6-a126e4759482"/>
      <componentinstantiation id="randomGenerator_1" startorder="0">
        <usagename>randomGenerator_1</usagename>
        <findcomponent>
          <namingservice name="randomGenerator_1"/>
        </findcomponent>
      </componentinstantiation>
    </componentplacement>
```

# Properties Descriptor (PRF)

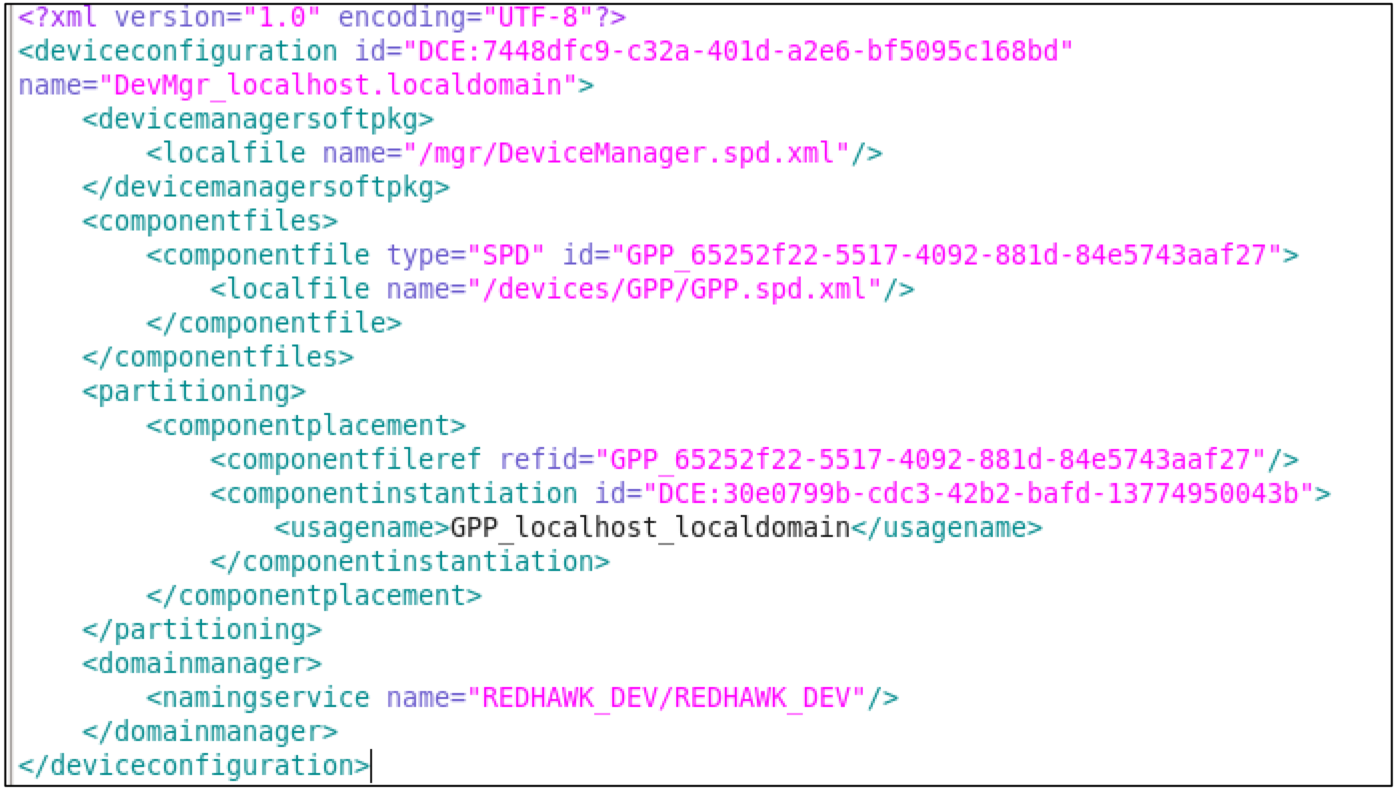
- PRF contains information regarding
  - Properties applicable to a software package
  - Properties of a component, e.g.
    - Configuration
    - Test
    - Execute
    - Allocation types



```
<properties>
  <simple id="scalar" mode="readwrite" type="float">
    <value>1.0</value>
    <units>unit-less</units>
    <kind kindtype="configure"/>
    <action type="external"/>
  </simple>
</properties>
```

# Device Configuration Descriptor (DCD)

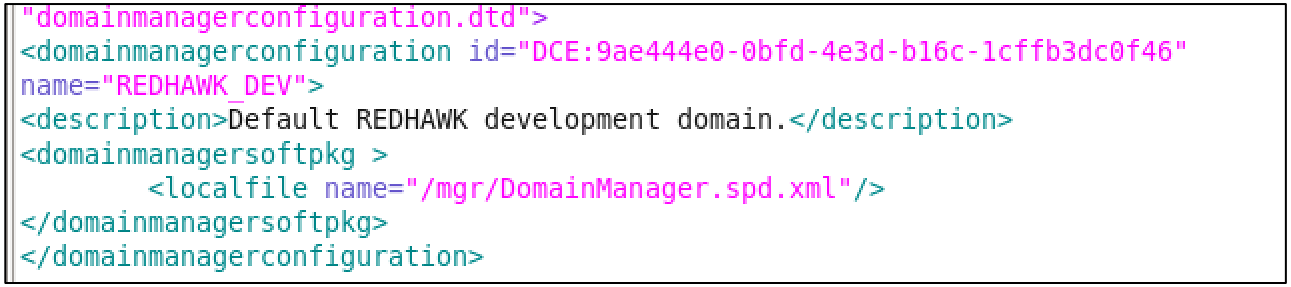
- DCD contains information regarding
  - Devices associated with a device manager
  - How to find the domain manager
  - Configuration information for the components that it deploys



```xml
<?xml version="1.0" encoding="UTF-8"?>
<deviceconfiguration id="DCE:7448dfc9-c32a-401d-a2e6-bf5095c168bd"
name="DevMgr_localhost.localdomain">
    <devicemanagersoftpkg>
        <localfile name="/mgr/DeviceManager.spd.xml"/>
    </devicemanagersoftpkg>
    <componentfiles>
        <componentfile type="SPD" id="GPP_65252f22-5517-4092-881d-84e5743aaf27">
            <localfile name="/devices/GPP/GPP.spd.xml"/>
        </componentfile>
    </componentfiles>
    <partitioning>
        <componentplacement>
            <componentfileref refid="GPP_65252f22-5517-4092-881d-84e5743aaf27"/>
            <componentinstantiation id="DCE:30e0799b-cdc3-42b2-bafd-13774950043b">
                <usagename>GPP_localhost_localdomain</usagename>
            </componentinstantiation>
        </componentplacement>
    </partitioning>
    <domainmanager>
        <namingservice name="REDHAWK_DEV/REDHAWK_DEV"/>
    </domainmanager>
</deviceconfiguration>
```

# Domain Manager Configuration Descriptor (DMD)

- DMD contains configuration information for the domain manager



```
"domainmanagerconfiguration.dtd">
<domainmanagerconfiguration id="DCE:9ae444e0-0bfd-4e3d-b16c-1cffb3dc0f46"
name="REDHAWK_DEV">
<description>Default REDHAWK development domain.</description>
<domainmanagersoftpkg >
        <localfile name="/mgr/DomainManager.spd.xml"/>
</domainmanagersoftpkg>
</domainmanagerconfiguration>
```

# Other XML File Descriptions

Device Package Descriptor (DPD)

- Identifies a class of a device
- Defines specific properties (capacity, serial number, etc.) for this class of device
- Optional, however if it is used the reference to this file will be made from the DCD file

Platform Deployment Descriptor (PDD)

- Identifies the logical relationships between platform resources within the OE's registered services and devices
- Optional, however if it is used the reference to this file will be made from the DMD file
- May be used to exert a greater degree of control over the application deployment process

Application Deployment Descriptor (ADD)

- Contains precedence lists that are used for deploying application instances within a platform
- Optional, however if it is used the reference to this file will be made from a SAD file

# Agenda

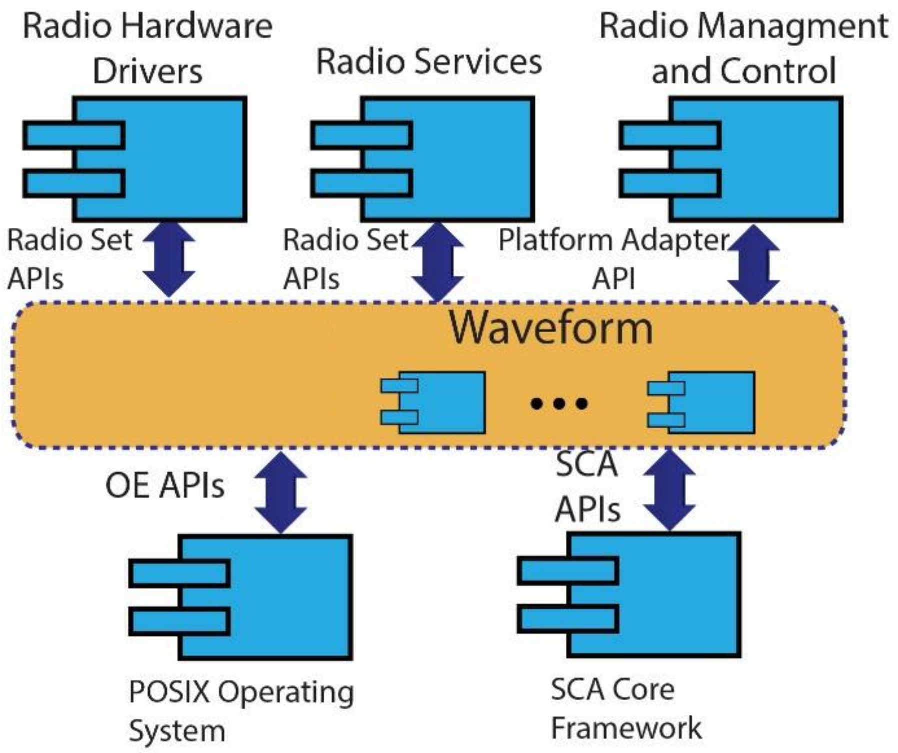
- JTNC Overview
- SCA & APIs Overview
- SCA Basics
- SCA Mgmt. & Control Infrastructure
- SCA Devices & Services
- SCA XML Files
- AEP
- SCA Example
- Enhancements in SCA 4.1
- SCA Wrap-Up
- REDHAWK

# Application Environment Profile (AEP)

POSIX is used as the base operating system specification for the SCA.

Three [3] operating system profiles have been created for SCA 4.1:

1. Application Environment Profile (AEP)
2. Lightweight AEP (LwAEP)
3. Ultra-Lightweight (ULwAEP)



# Sample of Application Environment Profile from SCA 4.1

- Appendix B specifies the POSIX functions supported by an SCA compliant operating system
- Application components are restricted to the functions calls provided by the profile the OS supports
  - AEP
  - LwAEP
  - ULwAEP
- OE components may use a superset of the POSIX functionality, e.g.
  - Services
  - Devices
  - SCA Core Framework

| Option | AEP | LwAEP | ULwAEP |
|---|---|---|---|
| {_POSIX_ASYNCHRONOUS_IO} | MAN | NRQ | NRQ |
| {_POSIX_CHOWN_RESTRICTED} | NRQ | NRQ | NRQ |
| {_POSIX_CLOCK_SELECTION} | NRQ | NRQ | NRQ |
| {_POSIX_FSYNC} | MAN | NRQ | NRQ |
| {_POSIX_MAPPED_FILES} | NRQ | NRQ | NRQ |
| {_POSIX_MEMLOCK_RANGE} | MAN | NRQ | NRQ |
| {_POSIX_MEMLOCK} | MAN | NRQ | NRQ |
| {_POSIX_MEMORY_PROTECTION} | NRQ | NRQ | NRQ |
| {_POSIX_MESSAGE_PASSING} | MAN | PRT | PRT |
| {_POSIX_MONOTONIC_CLOCK} | NRQ | NRQ | NRQ |
| {_POSIX_NO_TRUNC} | PRI | NRQ | NRQ |
| {_POSIX_PRIORITIZED_IO} | NRQ | NRQ | NRQ |
| {_POSIX_PRIORITY_SCHEDULING} | NRQ | NRQ | NRQ |
| {_POSIX_REALTIME_SIGNALS} | MAN | NRQ | NRQ |
| {_POSIX_SAVED_IDS} | NRQ | NRQ | NRQ |
| {_POSIX_SEMAPHORES} | MAN | MAN | MAN |
| {_POSIX_SHARED_MEMORY_OBJECTS} | NRQ | NRQ | NRQ |
| {_POSIX_SYNCHRONIZED_IO} | MAN | NRQ | NRQ |
| {_POSIX_THREAD_ATTR_STACKADDR} | MAN | MAN | MAN |
| {_POSIX_THREAD_ATTR_STACKSIZE} | MAN | MAN | MAN |
| {_POSIX_THREAD_CPUTIME} | NRQ | NRQ | NRQ |
| {_POSIX_THREAD_PRIO_INHERIT} | MAN | MAN | MAN |
| {_POSIX_THREAD_PRIO_PROTECT} | MAN | NRQ | NRQ |
| {_POSIX_THREAD_PRIORITY_SCHEDULING} | MAN | MAN | MAN |
| {_POSIX_THREAD_PROCESS_SHARED} | NRQ | NRQ | NRQ |
| {_POSIX_THREAD_SAFE_FUNCTIONS} | MAN | NRQ | NRQ |
| {_POSIX_THREAD_SPORADIC_SERVER} | NRQ | NRQ | NRQ |
| {_POSIX_TIMEOUTS} | MAN | MAN | MAN |
| {_POSIX_TIMERS} | MAN | MAN | MAN |
| {_POSIX_TRACE_EVENT_FILTER} | NRQ | NRQ | NRQ |
| {_POSIX_TRACE_LOG} | NRQ | NRQ | NRQ |
| {_POSIX_TRACE} | NRQ | NRQ | NRQ |
| {_POSIX_VDISABLE} | NRQ | NRQ | NRQ |

**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

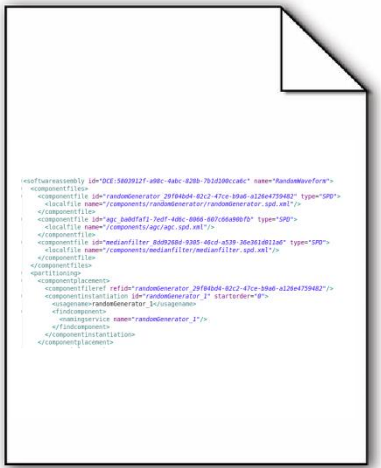
# Agenda

- JTNC Overview
- SCA & APIs Overview
- SCA Basics
- SCA Mgmt. & Control Infrastructure
- SCA Devices & Services
- SCA XML Files
- AEP
- <u>SCA Example</u>
- Enhancements in SCA 4.1
- SCA Wrap-Up
- REDHAWK

# SCA Starting Up a Waveform Application

1

The application factory reads the Software Assembly Descriptor which describes the waveform as a schematic



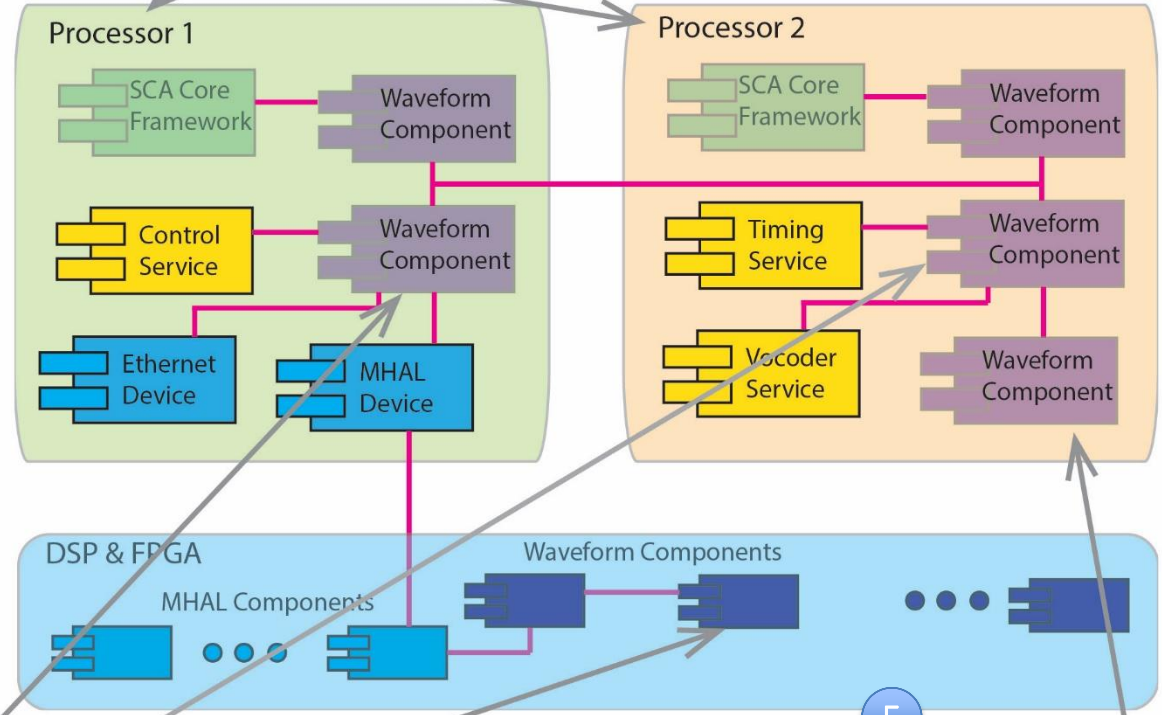
```
<softwareassembly id="DCE:5803912f-a98c-4abc-828b-7b1d100cca6c" name="RandomWaveform">
  <componentfiles>
    <componentfile id="randomGenerator_29f04bd4-02c2-47ce-b9a6-a126e4759482" type="SPD">
      <localfile name="/components/randomGenerator/randomGenerator.spd.xml"/>
    </componentfile>
    <componentfile id="agc_ba0dfaf1-7edf-4d6c-8066-607c66a90bfb" type="SPD">
      <localfile name="/components/agc/agc.spd.xml"/>
    </componentfile>
    <componentfile id="medianfilter_8dd9268d-9305-46cd-a539-36e361d011a6" type="SPD">
      <localfile name="/components/medianfilter/medianfilter.spd.xml"/>
    </componentfile>
  </componentfiles>
  <partitioning>
    <componentplacement>
      <componentfileref refid="randomGenerator_29f04bd4-02c2-47ce-b9a6-a126e4759482"/>
      <componentinstantiation id="randomGenerator_1" startorder="0">
        <usagename>randomGenerator_1</usagename>
        <findcomponent>
          <namingservice name="randomGenerator_1"/>
        </findcomponent>
      </componentinstantiation>
    </componentplacement>
```

2

The application factory locates all the executable and metadata files required to launch the waveform

4

The SCA's application factory commands the executable devices to launch the individual waveform components and then connects them together



3

The application factory determines which processors have sufficient capacity and resources to host the individual executables

5

The application factory configures all of the waveform components with the parameters and values required for waveform operation

**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# Agenda

- **JTNC Overview**
- **SCA & APIs Overview**
- **SCA Basics**
- **SCA Mgmt. & Control Infrastructure**
- **SCA Devices & Services**
- **SCA XML Files**
- **AEP**
- **SCA Example**
- **<u>Enhancements in SCA 4.1</u>**
- **SCA Wrap-Up**
- **REDHAWK**

# SCA 4.1 Differentiating Factors

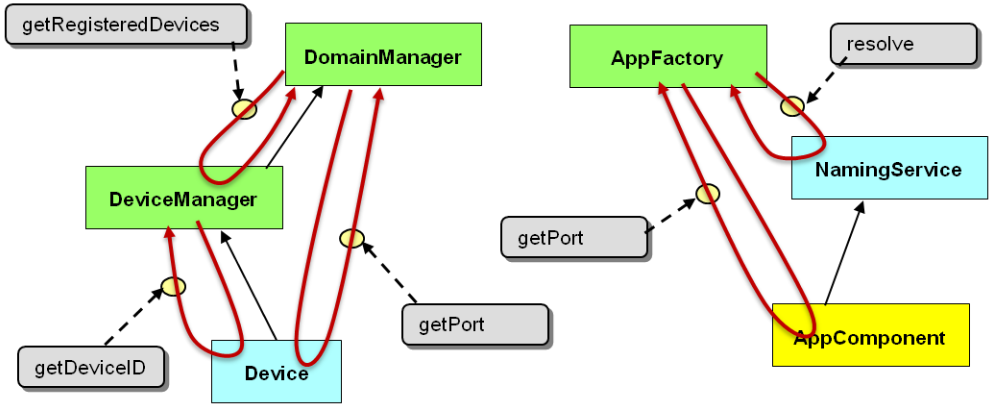
- Technologies and features of SCA 4.1
  - Cyber-hardened by architecture; Push-vs.-Pull & Principle of Least Privilege
  - Scalable for different form factors
  - Multi-core processor support
  - External connections to other framework architectures
  - Middleware agnostic
  - Backwards compatible
- Industry provided much of the technology and design of SCA 4.1
- SCA 4.1 is a Mandated standard in the DoD IT Standards Registry (DISR) as of 30-November-2017

# SCA 4.1 Push Model

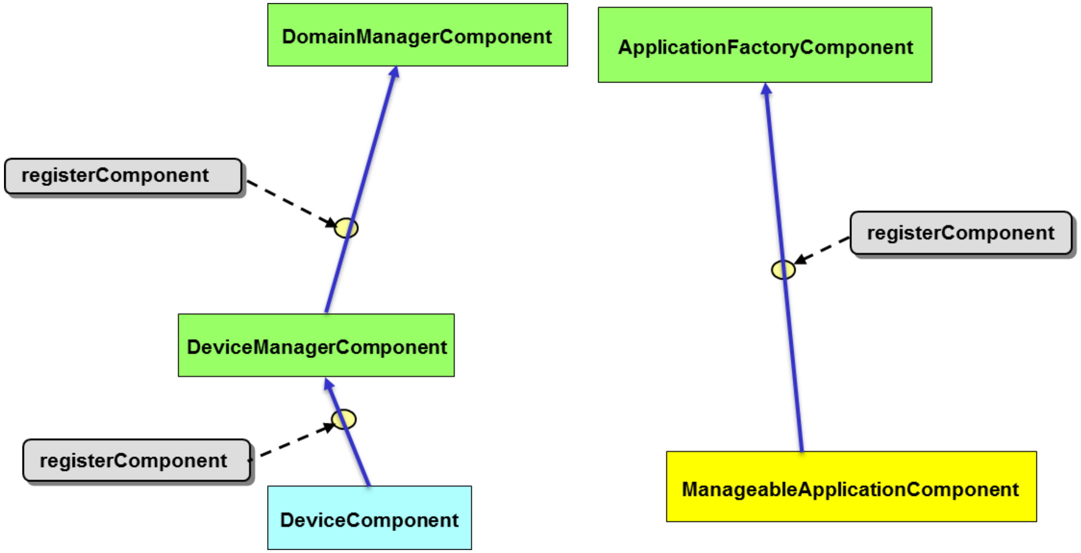
- Pull model requires the Core Framework to query the devices and components within the radio

- Push model permits more secure boots/launches and decreases startup time

SCA 2.2.2 Pull Model Registration

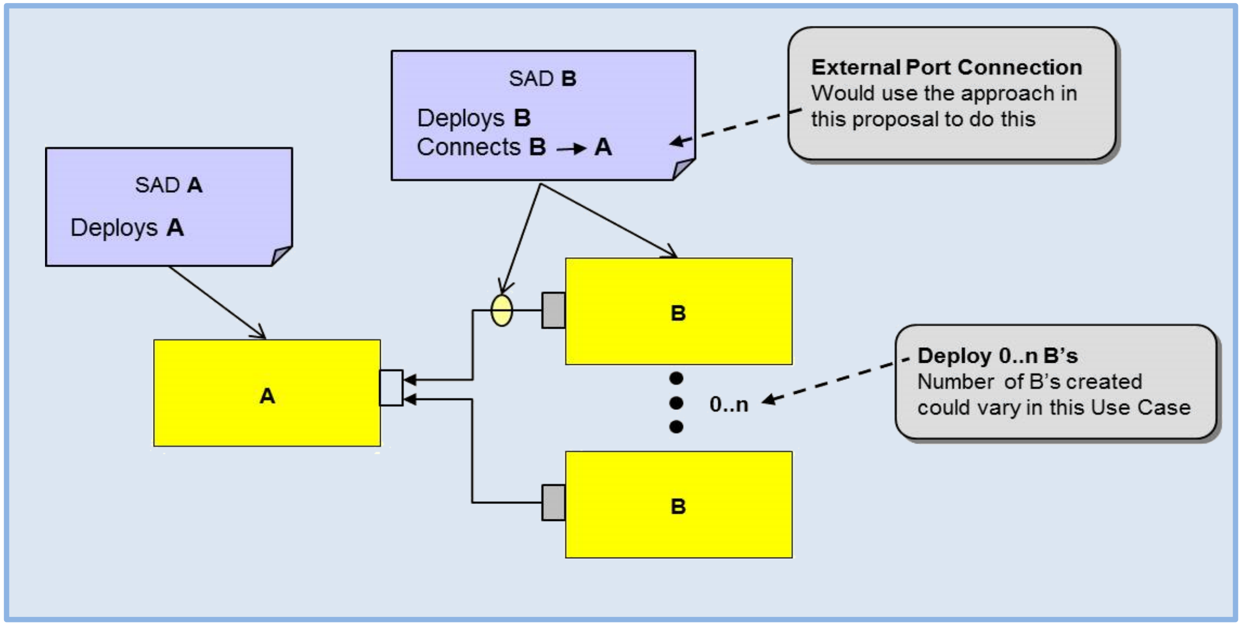


SCA 4.1 Push Model Registration



# SCA 4.1 Connects to External App and Devices

- SCA 4.1 introduces an intra-application connection mechanism that allows the framework to connect multiple applications
  - Ideal for handling communication to external apps seamlessly via the Android presentation layer
  - Facilitates rapid integration and expansion of application capabilities
  - Direct connection to the application



**External Port Connection**
Would use the approach in this proposal to do this

**Deploy 0..n B's**
Number of B's created could vary in this Use Case

**DISTRIBUTION STATEMENT A.** Approved for public release: distribution unlimited. (15 May 2018)

# Agenda

- **JTNC Overview**
- **SCA & APIs Overview**
- **SCA Basics**
- **SCA Mgmt. & Control Infrastructure**
- **SCA Devices & Services**
- **SCA XML Files**
- **AEP**
- **SCA Example**
- **Enhancements in SCA 4.1**
- **<u>SCA Wrap-Up</u>**
- **REDHAWK**

# SCA Takeaways

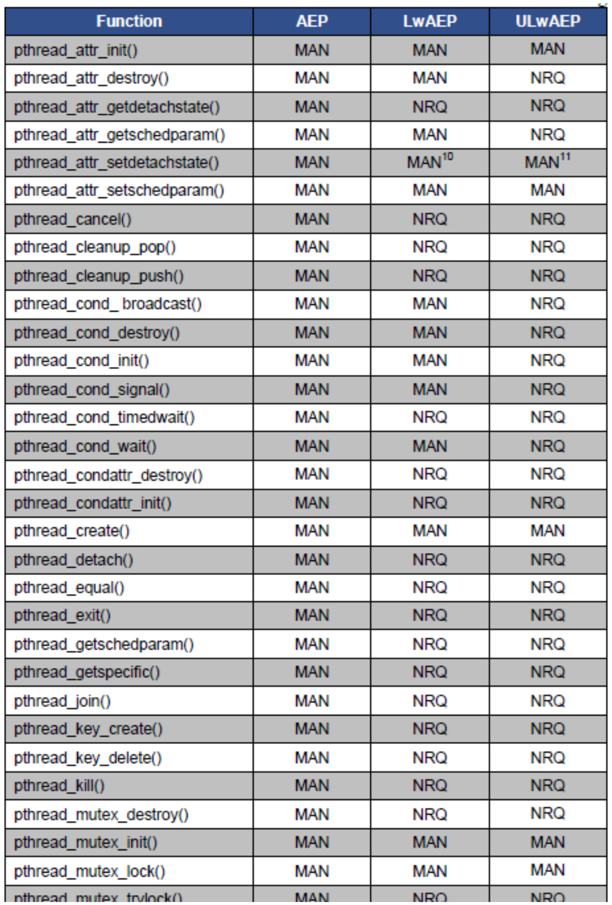
- SCA is part of a set of open systems standards
- SCA does not dictate whether product is an open or proprietary solution
- Specification is validated and mature with over 425,000 delivered SCA radios
- SCA evolves with software technology and changing missions

**SCA open systems architecture facilitates competition in DoD acquisition**

# Alignment Activities with other Standards

- JTNC released SCA 4.1 operating system (OS) profiles
  - Conducted comparative analysis with other frameworks
  - Identified areas of alignment with other framework's OS profiles
  - Enables SCA 4.1 applications to execute on an OS that is conformant  to another framework's OS profile
- SCA 4.1 provides connections to external frameworks
  - Enables SCA 4.1 software components to exchange information with those of another framework

## SCA 4.1 Application Environment Profiles

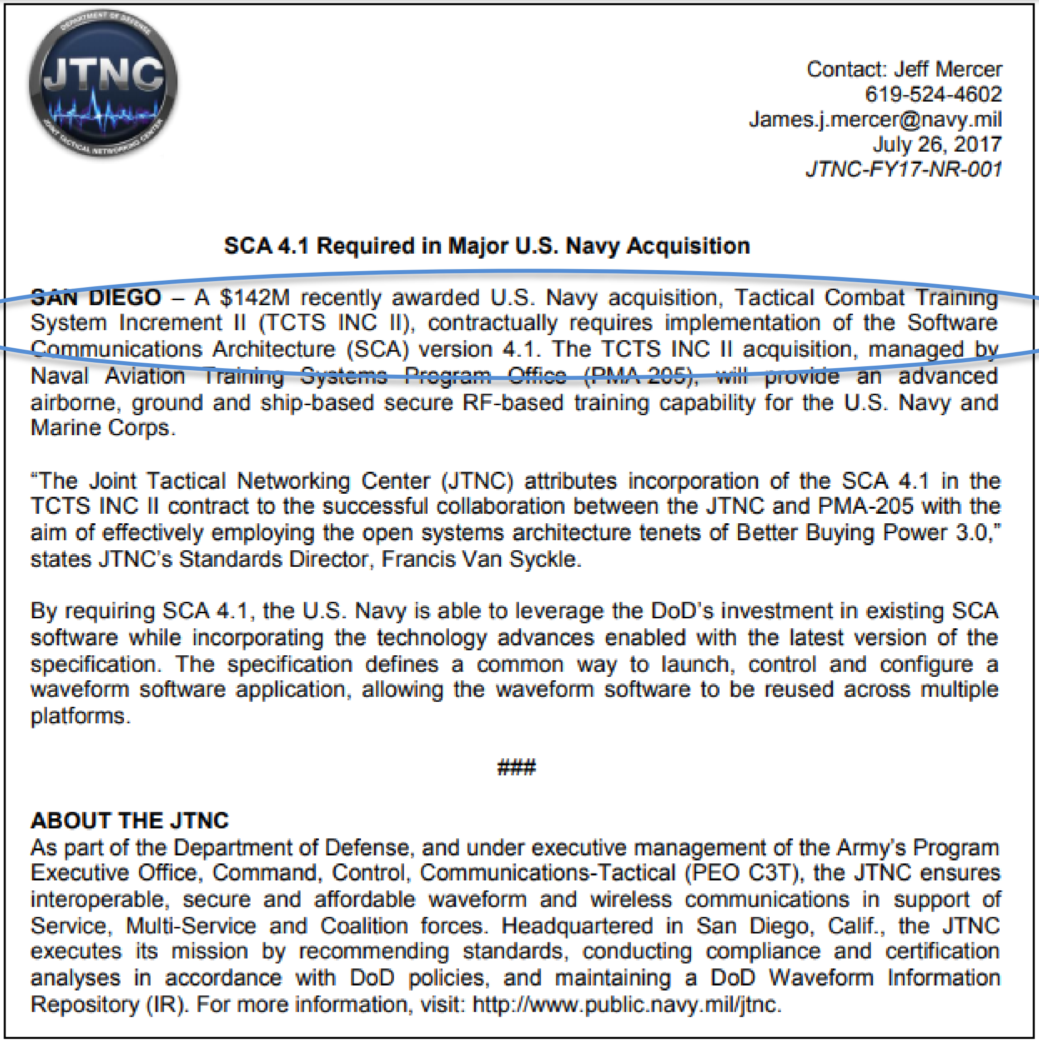


| Function | AEP | LwAEP | ULwAEP |
|---|---|---|---|
| pthread_attr_init() | MAN | MAN | MAN |
| pthread_attr_destroy() | MAN | MAN | NRQ |
| pthread_attr_getdetachstate() | MAN | NRQ | NRQ |
| pthread_attr_getschedparam() | MAN | MAN | NRQ |
| pthread_attr_setdetachstate() | MAN | MAN[10] | MAN[11] |
| pthread_attr_setschedparam() | MAN | MAN | MAN |
| pthread_cancel() | MAN | NRQ | NRQ |
| pthread_cleanup_pop() | MAN | NRQ | NRQ |
| pthread_cleanup_push() | MAN | NRQ | NRQ |
| pthread_cond_ broadcast() | MAN | MAN | NRQ |
| pthread_cond_destroy() | MAN | MAN | NRQ |
| pthread_cond_init() | MAN | MAN | NRQ |
| pthread_cond_signal() | MAN | MAN | NRQ |
| pthread_cond_timedwait() | MAN | NRQ | NRQ |
| pthread_cond_wait() | MAN | MAN | NRQ |
| pthread_condattr_destroy() | MAN | NRQ | NRQ |
| pthread_condattr_init() | MAN | NRQ | NRQ |
| pthread_create() | MAN | MAN | MAN |
| pthread_detach() | MAN | NRQ | NRQ |
| pthread_equal() | MAN | NRQ | NRQ |
| pthread_exit() | MAN | NRQ | NRQ |
| pthread_getschedparam() | MAN | NRQ | NRQ |
| pthread_getspecific() | MAN | NRQ | NRQ |
| pthread_join() | MAN | NRQ | NRQ |
| pthread_key_create() | MAN | NRQ | NRQ |
| pthread_key_delete() | MAN | NRQ | NRQ |
| pthread_kill() | MAN | NRQ | NRQ |
| pthread_mutex_destroy() | MAN | NRQ | NRQ |
| pthread_mutex_init() | MAN | MAN | MAN |
| pthread_mutex_lock() | MAN | MAN | MAN |
| pthread_mutex_trylock() | MAN | NRQ | NRQ |

**Alignment enables DoD systems to conform to the SCA and other frameworks simultaneously - increasing software reuse and cost savings**

# JTNC Alignment Activities Realized in Recent DoD Acquisition

- SCA 4.1 Contractual Requirement in $142M NAVAIR Acquisition



Contact: Jeff Mercer
619-524-4602
James.j.mercer@navy.mil
July 26, 2017
*JTNC-FY17-NR-001*

**SCA 4.1 Required in Major U.S. Navy Acquisition**

**SAN DIEGO** – A $142M recently awarded U.S. Navy acquisition, Tactical Combat Training System Increment II (TCTS INC II), contractually requires implementation of the Software Communications Architecture (SCA) version 4.1. The TCTS INC II acquisition, managed by Naval Aviation Training Systems Program Office (PMA-205), will provide an advanced airborne, ground and ship-based secure RF-based training capability for the U.S. Navy and Marine Corps.

"The Joint Tactical Networking Center (JTNC) attributes incorporation of the SCA 4.1 in the TCTS INC II contract to the successful collaboration between the JTNC and PMA-205 with the aim of effectively employing the open systems architecture tenets of Better Buying Power 3.0," states JTNC's Standards Director, Francis Van Syckle.

By requiring SCA 4.1, the U.S. Navy is able to leverage the DoD's investment in existing SCA software while incorporating the technology advances enabled with the latest version of the specification. The specification defines a common way to launch, control and configure a waveform software application, allowing the waveform software to be reused across multiple platforms.

###

**ABOUT THE JTNC**

As part of the Department of Defense, and under executive management of the Army's Program Executive Office, Command, Control, Communications-Tactical (PEO C3T), the JTNC ensures interoperable, secure and affordable waveform and wireless communications in support of Service, Multi-Service and Coalition forces. Headquartered in San Diego, Calif., the JTNC executes its mission by recommending standards, conducting compliance and certification analyses in accordance with DoD policies, and maintaining a DoD Waveform Information Repository (IR). For more information, visit: http://www.public.navy.mil/jtnc.

http://www.public.navy.mil/jtnc_new/News/SCA_4_1_in_Major_US_Navy_Acquisition.pdf

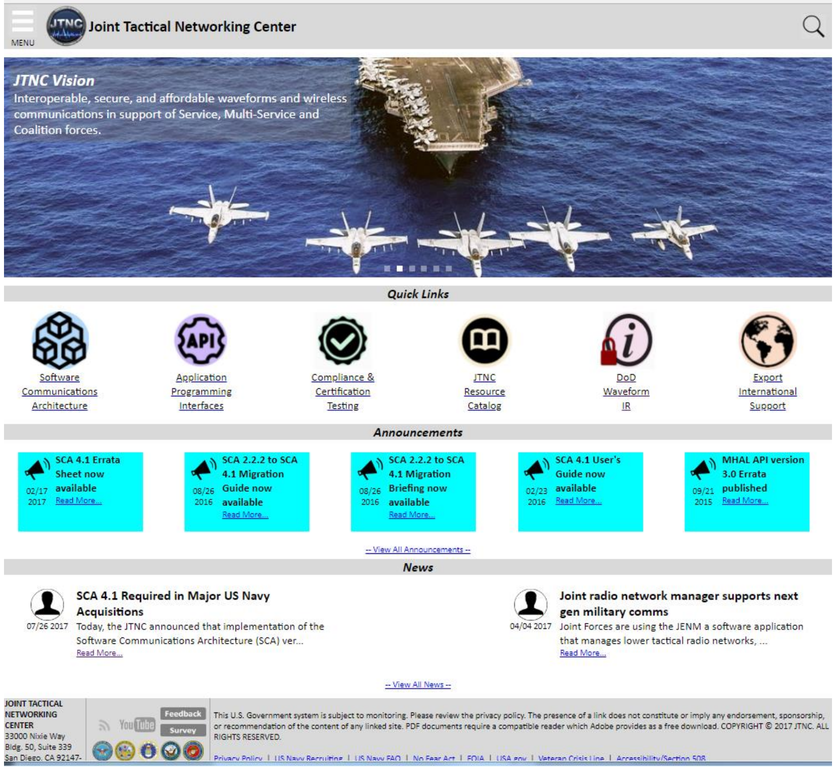
# Additional Material

- SCA & APIs
  - Standards & Specifications: http://www.public.navy.mil/jtnc_new/Pages/home.aspx

- SCA 4.1 Benefits and Features <*New*>
  - Paper: http://www.public.navy.mil/jtnc/PapersBriefsReports/SCA_4.1_Features___Benefits%20_v1a.pdf

- Detailed SCA 2.2.2 to SCA 4.1 migration brief and companion paper
  - Brief: http://www.public.navy.mil/jtnc/PapersBriefsReports/SCA_222_to_41_Migration.pdf
  - Paper: http://www.public.navy.mil/jtnc/PapersBriefsReports/SCA_222_to_SCA_41_Migration_Guidev0.1.pdf

- Webinar Recording: SCA 2.2.2 to SCA 4.1 Product Migration Guide Briefing and Discussion
  - https://youtu.be/5j4prVMiWZg


MENU
Joint Tactical Networking Center

*JTNC Vision*

Interoperable, secure, and affordable waveforms and wireless communications in support of Service, Multi-Service and Coalition forces.

**Quick Links**

Software Communications Architecture
Application Programming Interfaces
Compliance & Certification Testing
JTNC Resource Catalog
DoD Waveform IR
Export International Support

**Announcements**

| | |
|---|---|
| 02/17 2017 | **SCA 4.1 Errata Sheet now available** Read More... |
| 08/26 2016 | **SCA 2.2.2 to SCA 4.1 Migration Guide now available** Read More... |
| 08/26 2016 | **SCA 2.2.2 to SCA 4.1 Migration Briefing now available** Read More... |
| 02/23 2016 | **SCA 4.1 User's Guide now available** Read More... |
| 09/21 2015 | **MHAL API version 3.0 Errata published** Read More... |

-- View All Announcements --

**News**

07/26 2017 **SCA 4.1 Required in Major US Navy Acquisitions**
Today, the JTNC announced that implementation of the Software Communications Architecture (SCA) ver...
Read More...

04/04 2017 **Joint radio network manager supports next gen military comms**
Joint Forces are using the JENM a software application that manages lower tactical radio networks, ...
Read More...

-- View All News --

JOINT TACTICAL
NETWORKING
CENTER
33000 Nixie Way
Bldg. 50, Suite 339
San Diego, CA 92147-

This U.S. Government system is subject to monitoring. Please review the privacy policy. The presence of a link does not constitute or imply any endorsement, sponsorship, or recommendation of the content of any linked site. PDF documents require a compatible reader which Adobe provides as a free download. COPYRIGHT © 2017 JTNC. ALL RIGHTS RESERVED.

Privacy Policy | US Navy Recruiting | US Navy FAQ | No Fear Act | FOIA | USA.gov | Veteran Crisis Line | Accessibility/Section 508

# Agenda

- **JTNC Overview**
- **SCA & APIs Overview**
- **SCA Basics**
- **SCA Mgmt. & Control Infrastructure**
- **SCA Devices & Services**
- **SCA XML Files**
- **AEP**
- **SCA Example**
- **Enhancements in SCA 4.1**
- **SCA Wrap-Up**
- **REDHAWK**

# What is REDHAWK?

REDHAWK is a software package that supports the design, development, deployment, management, upgrade, and recycling of real-time, network-enabled software defined radios.

The REDHAWK software package is comprised of four major pieces:

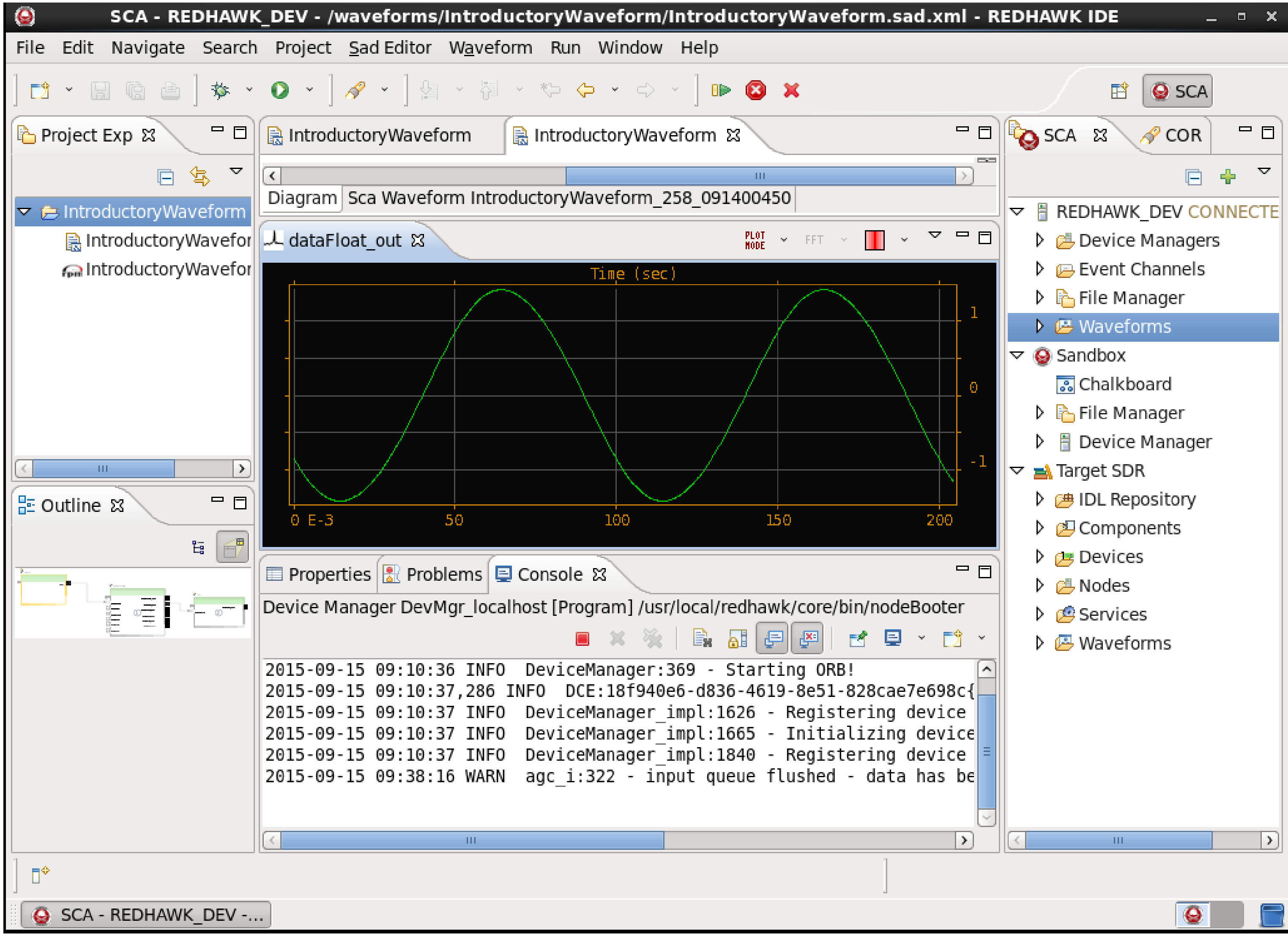
- A set of programs to manage distributed deployment of software applications
- A set of tools that allow developers to easily create software that is deployable within the REDHAWK environment
- A set of tools for introspecting a running REDHAWK system.
- A set of signal processing building blocks that developers can compose into larger, customized applications

REDHAWK takes care of the complicated “under the hood” hardware/software integration challenges so that developers can focus on application development.

REDHAWK also defines data interfaces, hardware management, and configuration management.

# REDHAWK's Integrated Development Environment (IDE)

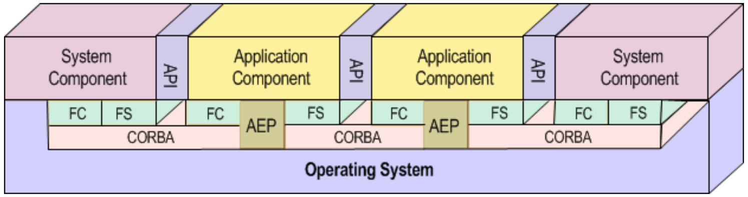
REDHAWK and similar commercial IDEs enable the development and installation of software on an asset.



```
2015-09-15 09:10:36 INFO  DeviceManager:369 - Starting ORB!
2015-09-15 09:10:37,286 INFO  DCE:18f940e6-d836-4619-8e51-828cae7e698c{
2015-09-15 09:10:37 INFO  DeviceManager_impl:1626 - Registering device
2015-09-15 09:10:37 INFO  DeviceManager_impl:1665 - Initializing device
2015-09-15 09:10:37 INFO  DeviceManager_impl:1840 - Registering device
2015-09-15 09:38:16 WARN  agc_i:322 - input queue flushed - data has be
```

# Relationship Between REDHAWK and the SCA

SCA 2.2.2



- REDHAWK originally based upon SCA 2.2.2
- Has some SCA 4.1 features – others are planned
- There are differences between SCA & REDHAWK, e.g.
  - BulkIO – REDHAWK standardizes interfaces
  - Logging – SCA specifies the OMG logging service, whereas REDHAWK uses the Apache log4cxx package
  - AEP – REDHAWK uses a larger runtime environment profile
  - Miscellaneous differences in component registration and device capacity

# Thank you

**Contact Info:**

**Joint Tactical Networking Center**
**Department of Defense Waveform Standards,**
**Compliance & Certification Directorate**
**JTNC_Standadards@navy.mil**
**http://www.public.navy.mil/jtnc**