

---

# SCA Training for Developers and Testers



**16 - 20 September 2002**

Copyright © 2002, Raytheon Company.  
All Rights Reserved

# AGENDA

---

- Day 1 Technical Overview of the Software Communications Architecture
- Day 2 Security and Waveform API Supplements
- Day 3 Developing SCA-Compliant Waveforms / Applications and Device / DeviceManager
- Day 4 Developing Application XML Configuration Files
- Day 5 SCA Porting, Testing and Evaluation



# Points Of Contact

---

This SCA Training Class is being presented by **Raytheon** of Fort Wayne, Indiana on behalf of the JTRS Joint Program Office.

## Points of Contact:

Mr. Jerry Bickle	Gerald_L_Bickle @ raytheon.com	(260) 429-6280	Sr. Principal SW Engr.
Mr. Bruce Baker	Bruce_A_Baker @ raytheon.com	(260) 429-4467	Sr. Principal SW Engr.
Mr. Jimmie Marks	Jimmie_T_Marks @ raytheon.com	(260) 429-6422	Sr. Principal SW Engr.
Mr. Rick Giardina	Rick_Giardina @ raytheon.com	(260) 429-7369	Adv. Comm. Program Manager
Mr. Jim Brown	James_J_Brown @ raytheon.com	(260) 429-6120	JTRS Step 2 Program Manager

---

# SCA Training for Developers and Testers

## Day 1: Technical Overview of the Software Communications Architecture





# Day 1 AGENDA

---

- JTRS Background
  - JTRS Objectives
  - Digital Radio Evolution
  - JTRS Program History
- What the SCA Attempts To Do
- What the SCA Consists Of
  - SW Architecture Overview
  - HW Architecture Overview
  - Intro to API Supplement
  - Intro to Security Supplement
- How the SCA Fits Into a Procurement / System Design

# JTRS Background



# JTRS Program Mission

---

*Acquire  
a Family of  
Affordable,  
High-capacity  
Tactical Radio Systems  
to Provide  
Interoperable  
LOS/BLOS and Wireless,  
Mobile Network  
C4I Capabilities  
to the Warfighters.*

# JTRS Objectives

---

- Meet Warfighter Requirements With Systems
  - Built to a common open architecture
  - That leverage & keep up with commercial technology
  - That permit insertion of new capabilities through software or module replacement
- Reduce the Logistics Tail
  - Reduce the number of different types of radios
    - 25 to 30 families to 1 family
  - Maintain a competitive environment across all program phases
    - initial acquisition, upgrade, support
  - Re-use application programs (radio waveforms)



# JTRS Target - a Software Radio

---

- Its Characteristics Can Be Altered Allowing it to Operate with Any Legacy Waveform. Thus a Single Unit Can Accommodate Multi-Service and Multi-National Capabilities.
- It Can Be Upgraded or Reprogrammed using standardized APIs. New or Improved Capabilities Can Be Incorporated Easily.
- JTRS Therefore Provides a Previously Unavailable Capability for Joint, Civilian-military, Multi-national and Coalition Operations.



# The JTRS Program

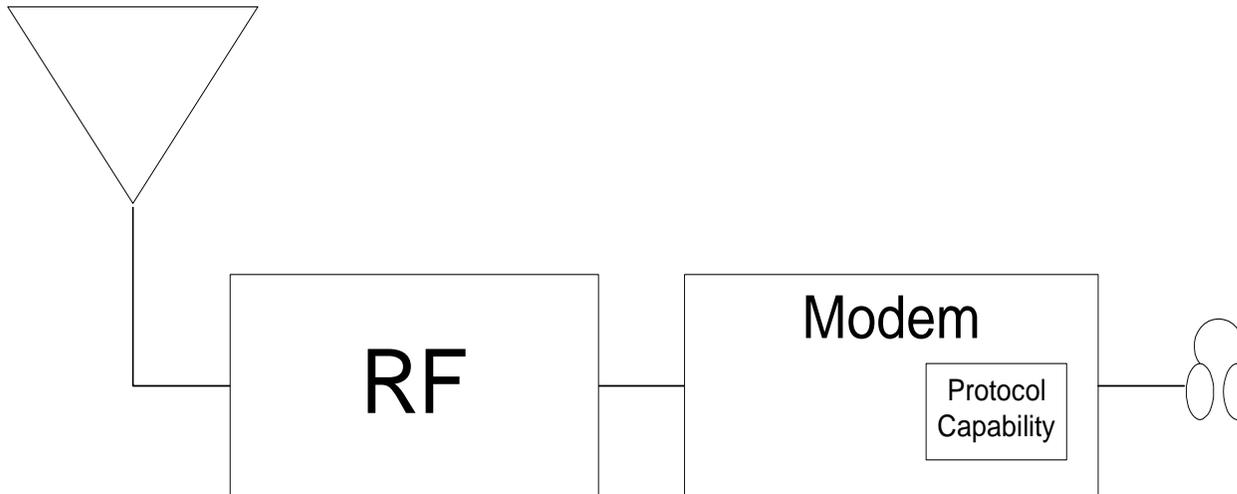
---

- JTRS the Architecture
  - An open-standard rule book
  - Focus of current efforts
    - industry led activity to develop & validate
- JTRS the Product
  - Hardware and software built to the architecture
  - Cluster procurements
- JTRS the Capability
  - Interoperable, reprogrammable, upgradeable, multi-channel, multi-mode, flexible communications
    - flexible data pipes

# Digital Radios

# Evolution to Digital Radios

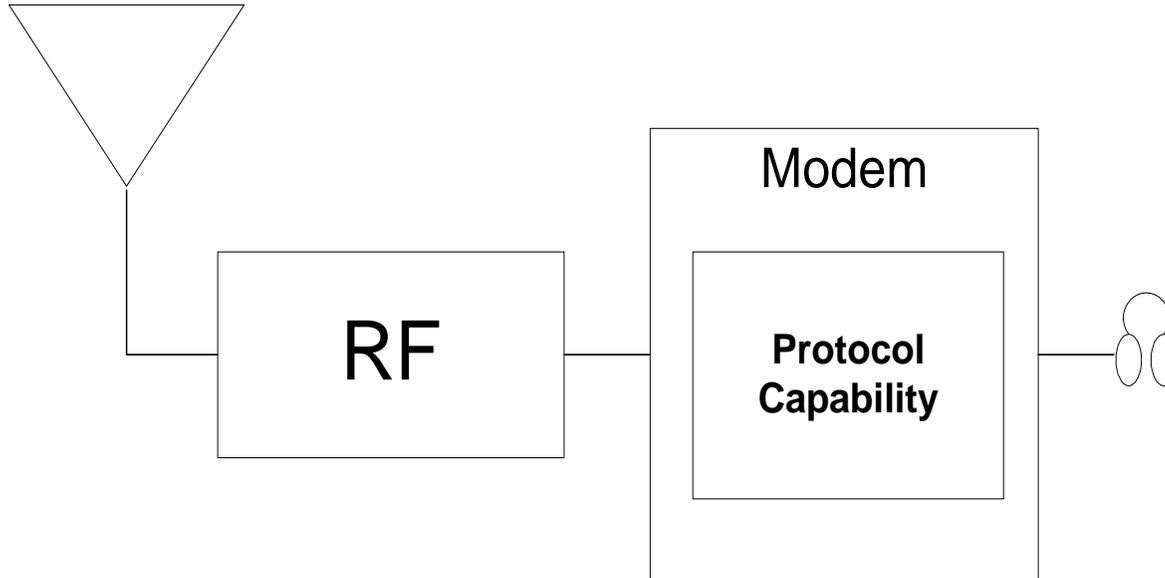
- Radio Prior to Digital Processing
  - Modem protocol capability low
  - No reprogrammability



- Hardware Based
- Simplistic Protocols

# Evolution to Digital Radios (cont'd)

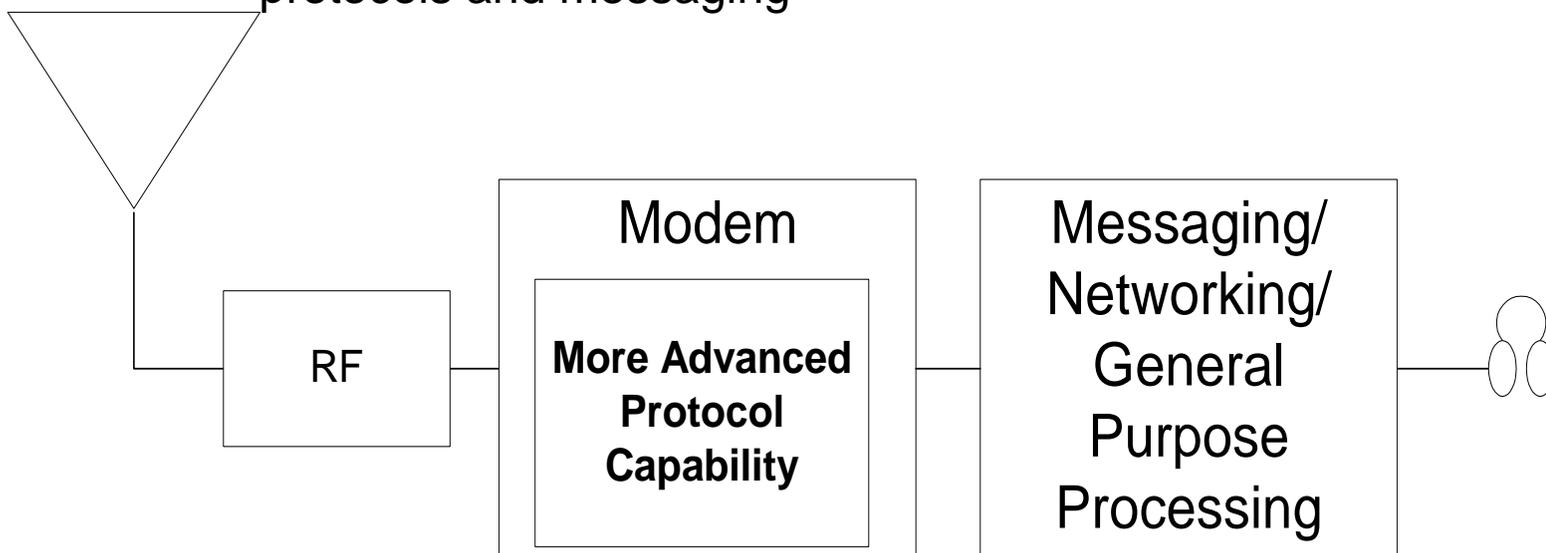
- Radio After Advent of Digital Processing
  - Modem protocol capability increasing
  - Increased reprogrammability



- Reprogrammable Digital Processing
- Complexity of Protocols Bounded by Available Processing

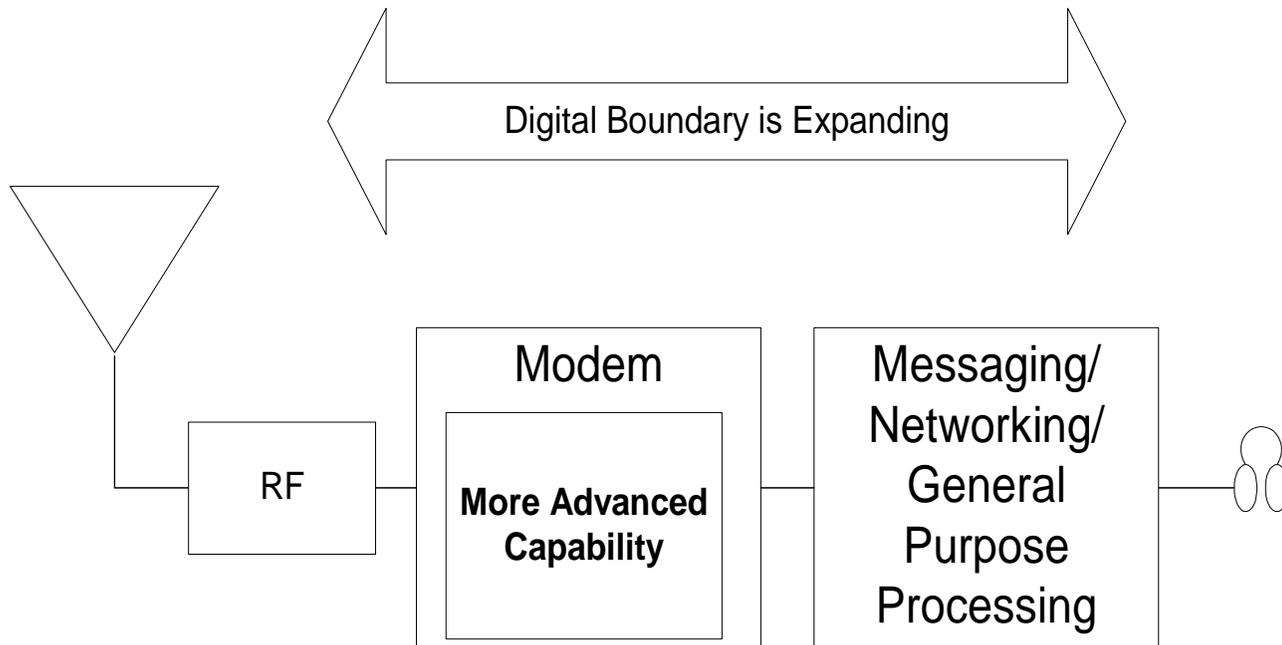
# Evolution to Digital Radios (cont'd)

- Radio With Increased Digital Processing
  - Modem protocol complexity increasing
  - Emergence of general purpose processors performing high level functions
  - Expanded digital signal processing hardware increased reprogrammability
    - protocols and messaging



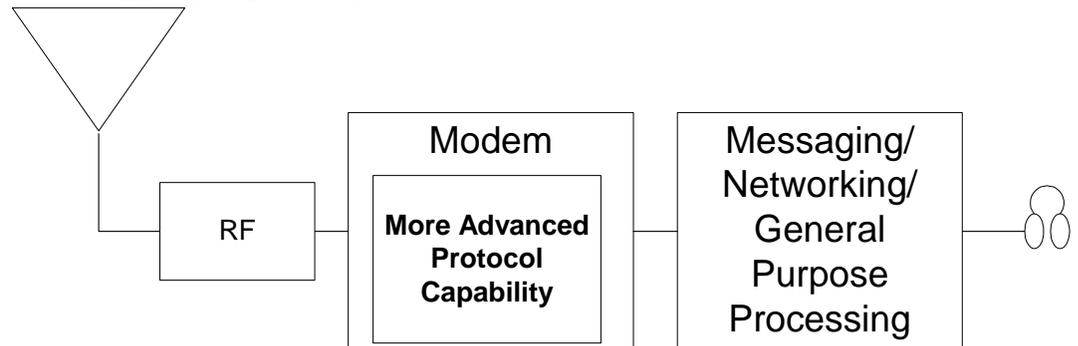
# Evolution to Digital Radios (cont'd)

- Growth of Processing Capability is Constantly Increasing
- Analog/Digital Boundary is Moving Toward the Antenna



# Evolution to Digital Radios (cont'd)

- Explosion in Reprogrammability of Devices Led to Many Solutions to Waveform Implementation
  - Many available devices
  - Many interfaces invented
  - Many service definition solutions
    - operating systems
    - message formats and techniques
  - Many languages
- Low Latency Devices Support Multi-Processing Implementation of Waveforms



# Evolution to Digital Radios (cont'd)

---

- Reuse Comes to the Forefront as a Most Important Requirement
- SCA Defines Waveform Implementation Standards
  - Standardization of interfaces
    - APIs
    - Devices
  - Standardization of service definitions
    - Operating systems
    - Message formats and techniques
      - CORBA
    - Core Framework
    - Waveform Services
    - Security Services

Open Architecture Definitions Form  
the Foundation for Reuse

# Radio Digitization

---

- Through the Advancement of Hardware and Software Technologies, Radios have Evolved From Stovepipe Implementations to Open Architecture, Multi-Function Communication Systems

# JTRS Step 1 & 2A

---

- Raytheon Formed and led the MSRC Consortium
  - Step 1 - Architecture Definition (Based on previous Raytheon programs)
  - Step 2A - Architecture Development & Validation
- Raytheon's Role
  - Raytheon led the Consortium Executive Council, the Management Council, the Architecture IPT, and the Software IPT
  - Initiated commercial acceptance of the SCA by promoting it at the SDR Forum where it was adopted in September 1999
  - Develop and prototype the Core Framework
  - Developed the Validation Demonstration Model (VDM) Prototype
    - Hosted all 2A waveforms & Core Framework
    - Delivered 2 VDMs to JPO, additional VDMs at Raytheon

# Step 2A Prototypes

---

- 4 Hardware Prototypes
- Description:
  - 2-4 Channels both Narrow and Wide Band, 2 MHz - 2 GHz
  - Waveforms Implemented - VHF, UHF, HQ I/II, DAMA/DASA, HF-ALE, SINCGARS ASIP/INC, WDW, VRC-99
- COTS Processors: Pentium, PowerPC
- Operating Systems: VxWorks, LynxOS
- Security: In-process
- Interoperability Demonstrations
  - Prototypes to Legacy Systems
  - Prototype to Prototype
- Porting
  - VHF FM to three other prototypes
  - HF ALE to one other prototype



# JTRS ORD Coverage

## Demonstrated in Raytheon VDM

---

- Basic Radio Functionality in an Open Systems Architecture
- Receive & Transmit over Frequency Range with Power Control.
- LOS Waveform Audio and/or Data Communications
  - UHF AM
  - Have Quick I&II
  - VHF-FM
  - VHF FM-Public Service
  - VHF AM/ATC (8-1/3 kHz tuning)
- DASA MIL-STD-188-181 Data Communications
- DAMA MIL-STD-188-183 Data Communications
- Crossbanding Between VHF-FM and HQ
- Multi-Channel Operation (1 to 4 Channels)
- Unencrypted GPS
- Dynamic load and configure/reconfigure operation using Parsed Domain and Device Manager XML Files for Component Connections

# Other Step 2A Prototypes

---

- **ITT**
  - PC-104 form factor (not integrated), VxWorks, X86
  - Demonstrated Partial SINCGARS SIP and WDW
  - Demonstrated IP Routing (Crossbanding)
- **Rockwell**
  - 6u VME form factor, LynxOS, PPC
  - Demonstrated HF-ALE and VHF-FM
  - Black Side Radio with External Crypto
- **BAE**
  - 6u VME form factor, VxWorks, PPC
  - Demonstrated VHF-FM only, VRC-99
- **Ported HF-ALE Waveform Software from Rockwell to Raytheon Prototype with over 90% Code Reuse**



# Step 2A Validation Results

---

- Multiple vendors can implement hardware and software compliant with the SCA
- SCA compliant software can operate in SCA compliant prototypes independently developed by other vendors.
- The architecture supports the stresses induced by selected waveform sets operating on multiple prototypes with diverse resources.
- The SCA can incorporate new technology.
- The architecture supports over the air programming/reprogramming.
- The architecture enables porting of waveforms between hardware implementations.
- Crossbanding can be achieved between applicable waveforms using SCA compliant applications.



# JTRS Step 2A Extension

---

- Submitting SCA to commercial standards organization Object Management Group (OMG)
- Updated, validated and delivered the CF to comply with SCA v2.0
- Developing Cryptographic Algorithms for Cornfield
- Generated “SCA 102” briefing for Government and Industry
- Performed EPLRS Study for mapping EPLRS waveform to SCA architecture
- Performed Security Requirements Analysis study
- Coordinated the closure of 106 change proposals to the SCA
- Supported Q&A Forums, JTRS Industry Day and various conferences to promote SCA to Government and industry
- Maintain and develop SCA
  - Create/Maintain the CP database
  - Facilitate the TAG / SCA change process
  - Prepare and submit the SCA and SRD documents



# Step 2B Activities

---

- Goal: 3rd Party Validation of SCA
  - 7 Agreements Awarded
- Scalability & Form Factor SCA Impacts
  - Man-pack
  - Hand-held - Tactical/Commercial
- Complex Waveform SCA Impacts
  - Link-16 S/W Implementation Risk Reduction
  - Modeling and Analysis, and Prototyping
- Core Framework Implementations
  - Government Open Source Core Framework
  - Commercial Based Conversion
- Platform Integration Study
- 3rd Party Waveform Development
  - License-Free SINCGARS ASIP Waveform from ATC with plans to Port SIP/INC to Raytheon VDM



# JTRS Successes

---

- DoD mandate of JTRS compliance for any new procurements
- Software Communications Architecture v2.2 released in support of Cluster procurements
- Gaining International Acceptance
  - JPO Initiating cooperative radio programs with Japan, U.K. and others
- Commercial Acceptance
  - OMG Interest
    - Raytheon, Mitre, and Mercury Computers co-chair the OMG Radio DSIG
  - SDR Forum adopted SCA 1.0 which Raytheon submitted in Sept 1999

# What the SCA Attempts To Do



# Criteria for the SCA

---

- Based on and To Be Accepted as Open, Commercial Standard
- Supports a Family of Radios
  - Interoperable,
  - Programmable (including Over-the-Air),
  - Scaleable (handheld to fixed-station),
  - Affordable
- Maximizes Independence of Software from Hardware
  - Application and device portability & reuse
  - Rapid technology insertion over time
- Extendible to New Waveforms and/or Hardware Components
- Incorporates Embedded, Programmable INFOSEC
- Supports Requirements in JTRS ORD
  - Operator reconfigurable
  - Multiple legacy and new waveforms (33)
  - Simultaneous multichannel operation (up to 10)



# How "low" does it go?

---

- There are many valid definitions of "architecture"
- The SCA is an Implementation-Independent Framework for development of JTRS software radios
  - Comprised of interface and behavioral specifications, general rules, waveform APIs and security requirements necessary to meet the criteria previously established
  - SCA requirements limited to those necessary to meet the criteria without restricting innovation or domain-specific requirements
- The SCA is not an "implementation architecture"
  - That will result from marriage of the SCA, the technical specification, and other procurement requirements



# The Resulting SCA

---

- Open, Distributed-Component Architecture Specification
- Separates Applications from the Operating Environment
- Segments Application Functionality
- Defines Common Interfaces for Managing & Deploying SW Components
- Defines Common Services & APIs to Support Device and Application Portability

# The SCA



# What the SCA Consists Of

---

- SW Architecture
- HW Architecture
- API Supplement
- Security Supplement

# SW Architecture Overview

# Software Architecture

---

- SCA Operating Environment (OE)
- Core Framework (CF) CORBA Interfaces
- Log Interface
- Event Service
- Naming Service
- Domain Profile

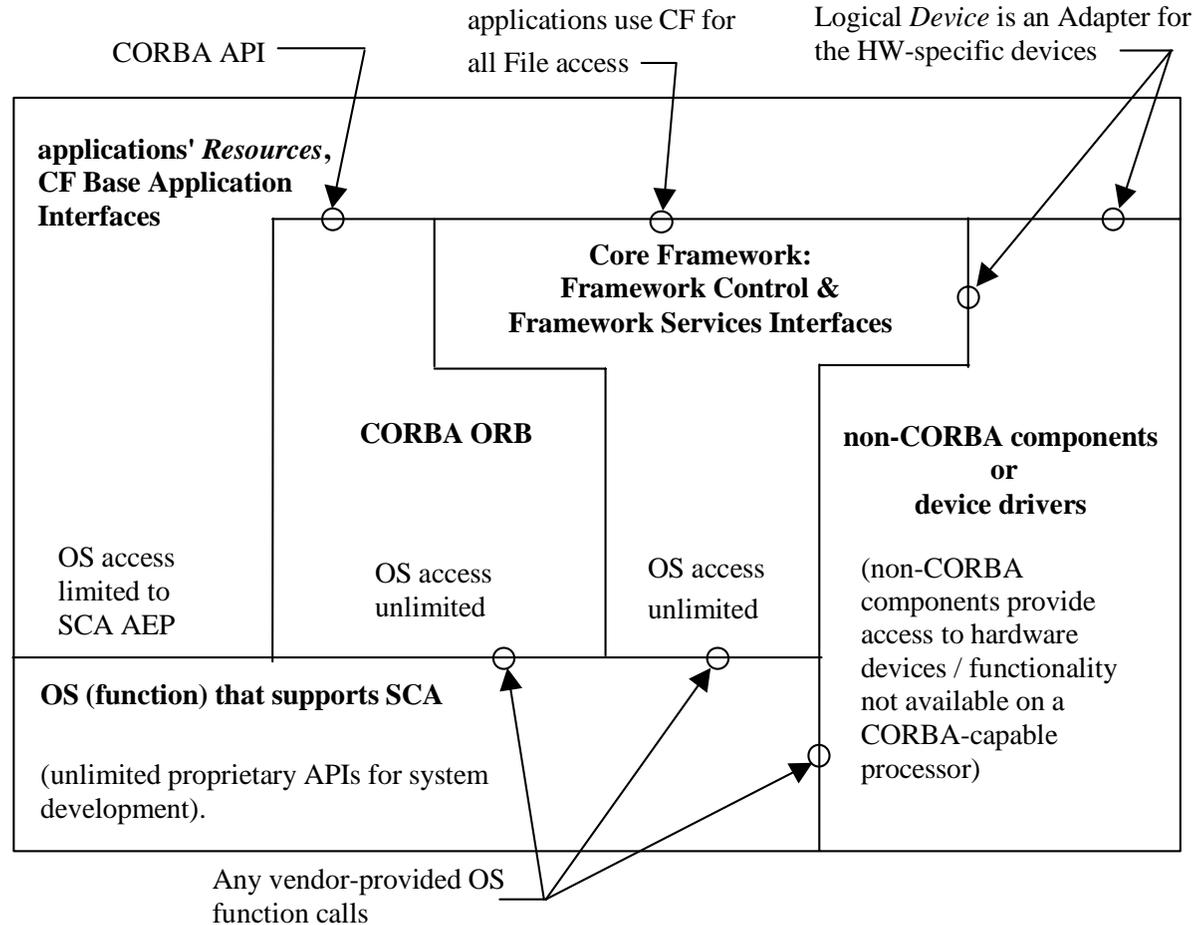


# SCA Operational Environment

---

- The JTRS Operating Environment (OE) defined in the SCA consists of three major parts:
  - A Real Time Operating System (RTOS)
  - A Real Time Object Request Broker (ORB)
  - The SCA Core Framework (CF)

# SCA Operating Environment Overview





# Real Time Operating System

---

- RTOS Must Support SCA Application Environment Profile (AEP)
  - The SCA AEP is a subset of the POSIX.13 Real-time Controller System Profile (PSE52)
  - Can be fully POSIX Profile 52 (or greater) compliant
- Applications shall be limited to using the RTOS services that are designated as mandatory in the SCA AEP

# SCA CORBA Usage

---

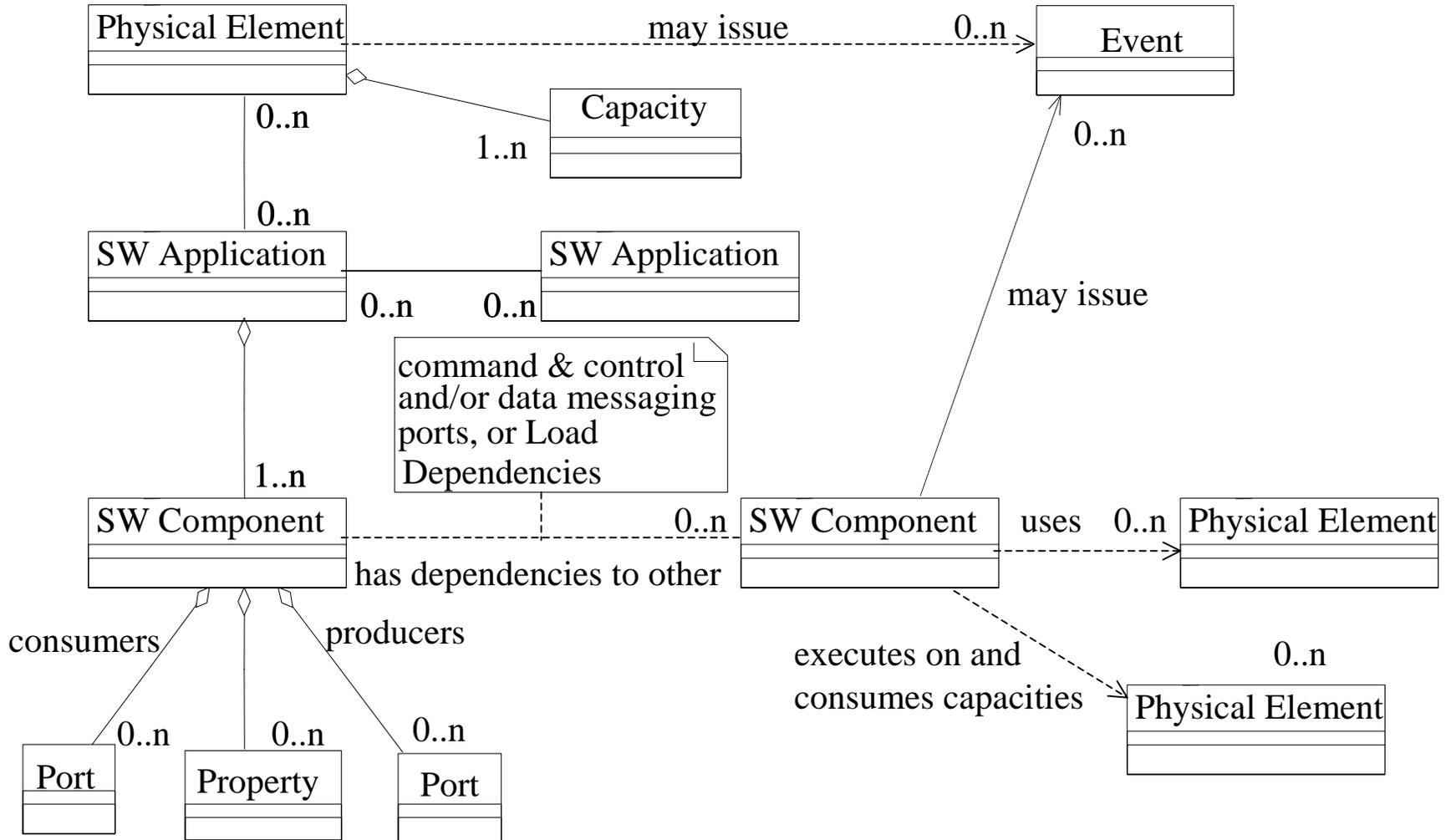
- No extensions and/or services above and beyond Minimum CORBA can be used except as specified in the SCA.
- Extensions currently allowed:
  - Naming Service
    - SCA- defined Lightweight Naming Service
  - Event Service
    - OMG Event Service (Push Model) with SCA-defined event types

# SCA Domain Profile

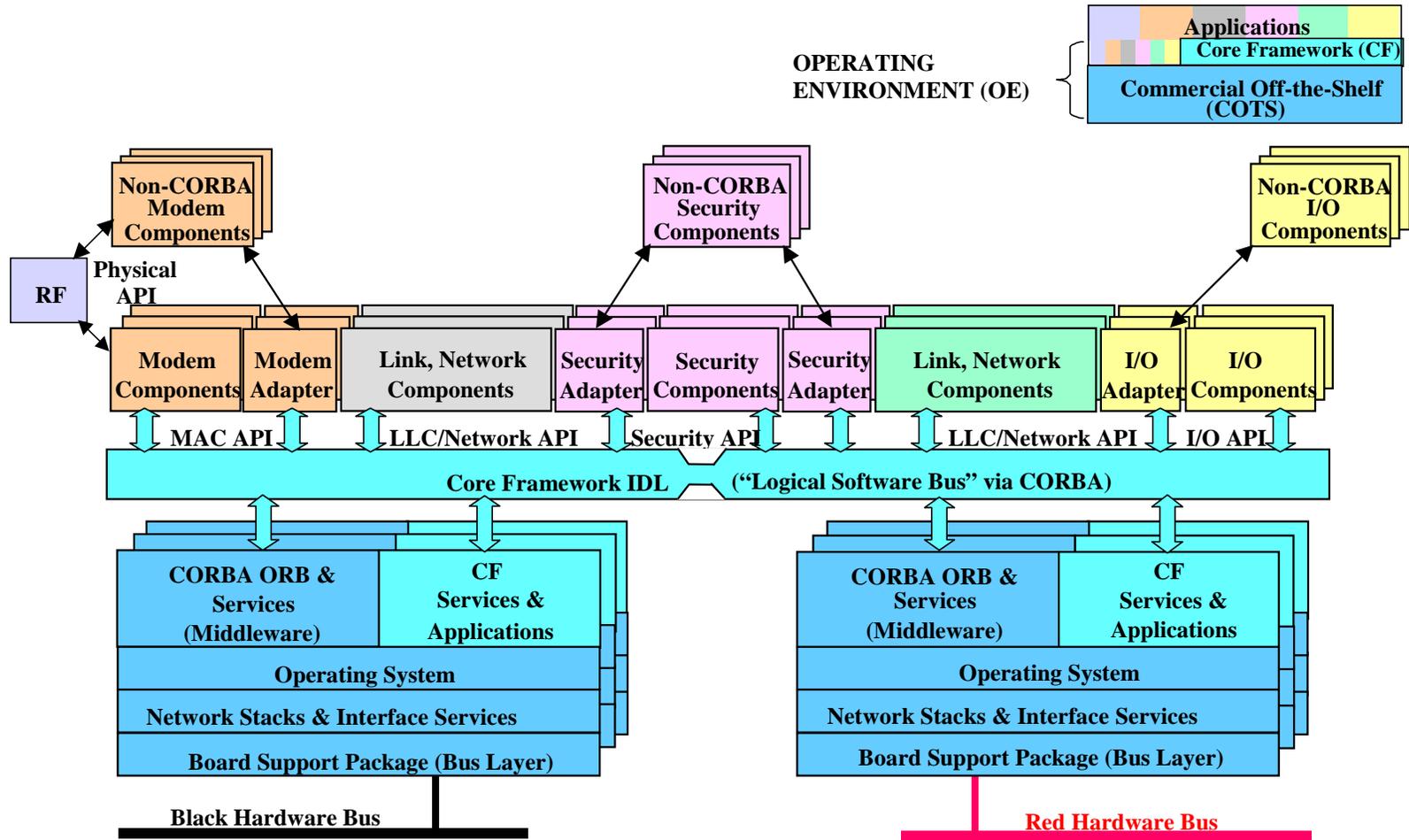
---

- A Domain Profile is a set of XML files that describe the hardware devices and software components of a system, their properties, and their interconnections
- Based on OMG CORBA Components Specification
  - eXtensible Markup Language (XML) format
  - Based upon the OMG's draft CORBA Components specification (orbos/99-07-01)
  - Customized from the OMG CORBA Component Specification
    - To better address Software Radio Needs.
    - Changes to be presented to OMG
  - Describes specific characteristics of software components or hardware devices
  - Describes interfaces, functional capabilities, logical location, inter-dependencies, and other pertinent parameters.
    - Description of applications, startup requirements of devices, etc.

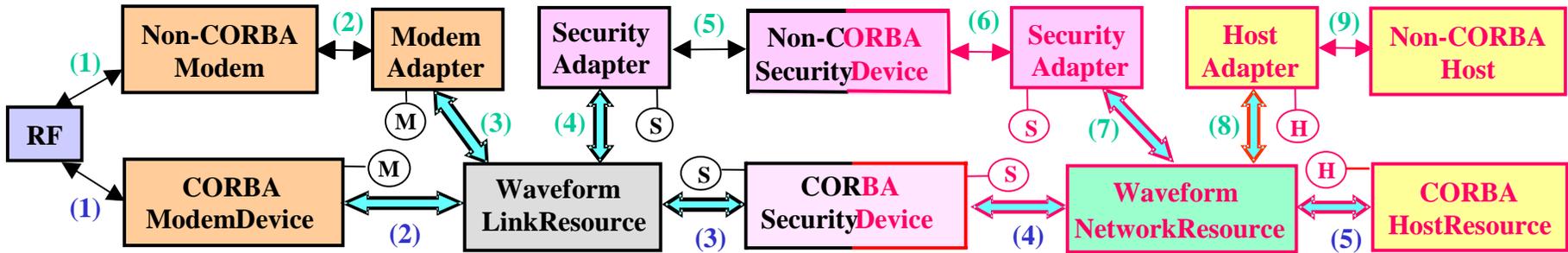
# SCA Problem Domain



# SCA Software Structure



# Example Message Reception Flow With & Without Adapters



## Message Reception Path (with Adapters)

- (1) RF Interface to Modem
- (2) non-CORBA Modem Interface
- (3) CORBA Interface to Waveform Link (M)
- (4) CORBA Interface to Security Adapter (S)
- (5) Black-side non-CORBA Security Interface
- (6) Red-side non-CORBA Security Interface
- (7) CORBA Interface to Waveform Network (S)
- (8) CORBA Interface to Host Adapter (H)
- (9) non-CORBA Host Interface

## Message Reception Path (without Adapters)

- (1) RF Interface to Modem
- (2) CORBA Interface to Waveform Link (M)
- (3) CORBA Interface to Security (S)
- (4) CORBA Interface to Waveform Network (S)
- (5) CORBA Interface to Host (H)

Note: The design goal of a CORBA gateway “Adapter” is to define the CORBA side of the gateway such that the eventual replacement of the non-CORBA device and its Adapter does not change the Core Framework CORBA interface.

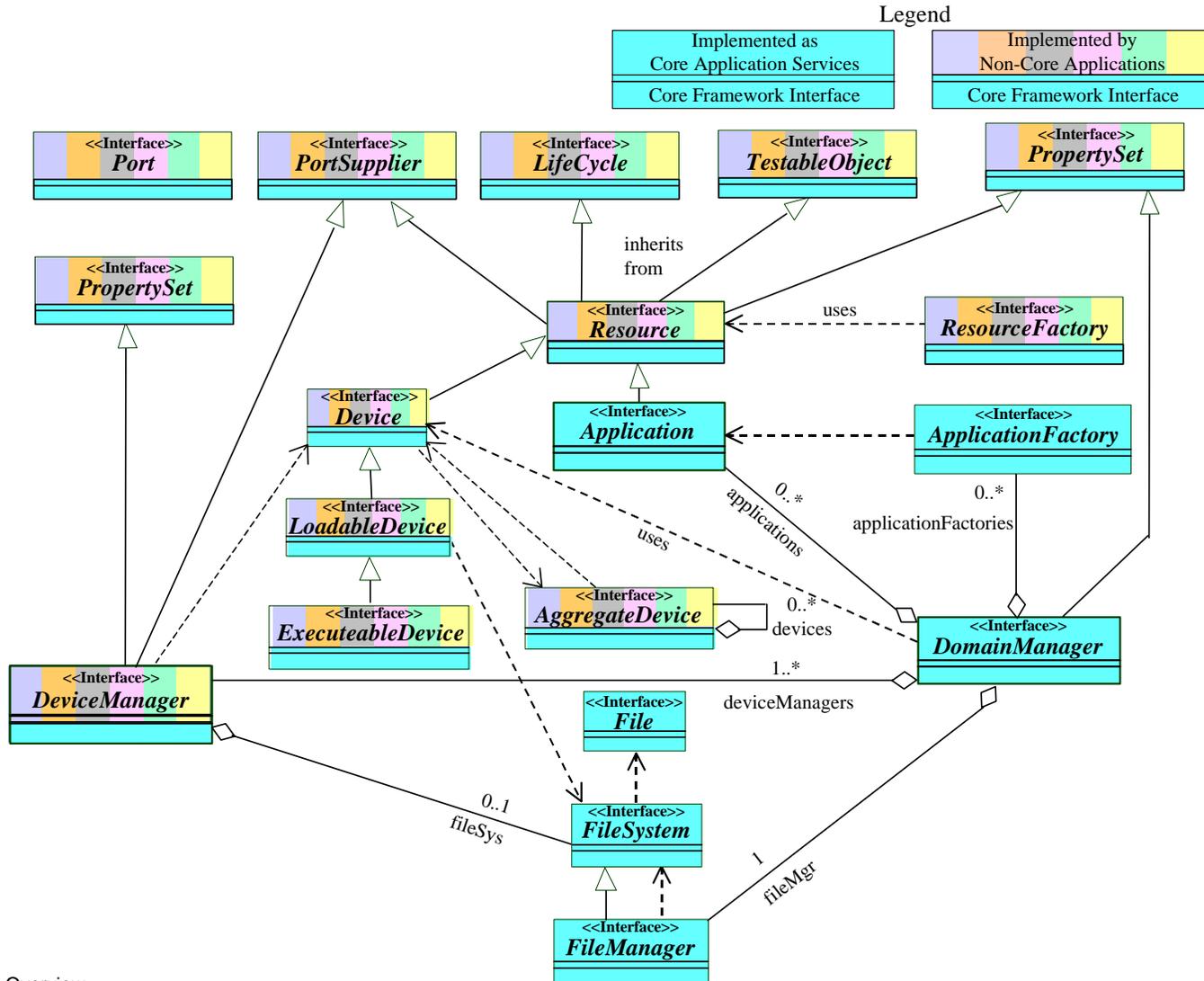
# SCA Core Framework

# SCA Core Framework Definition

---

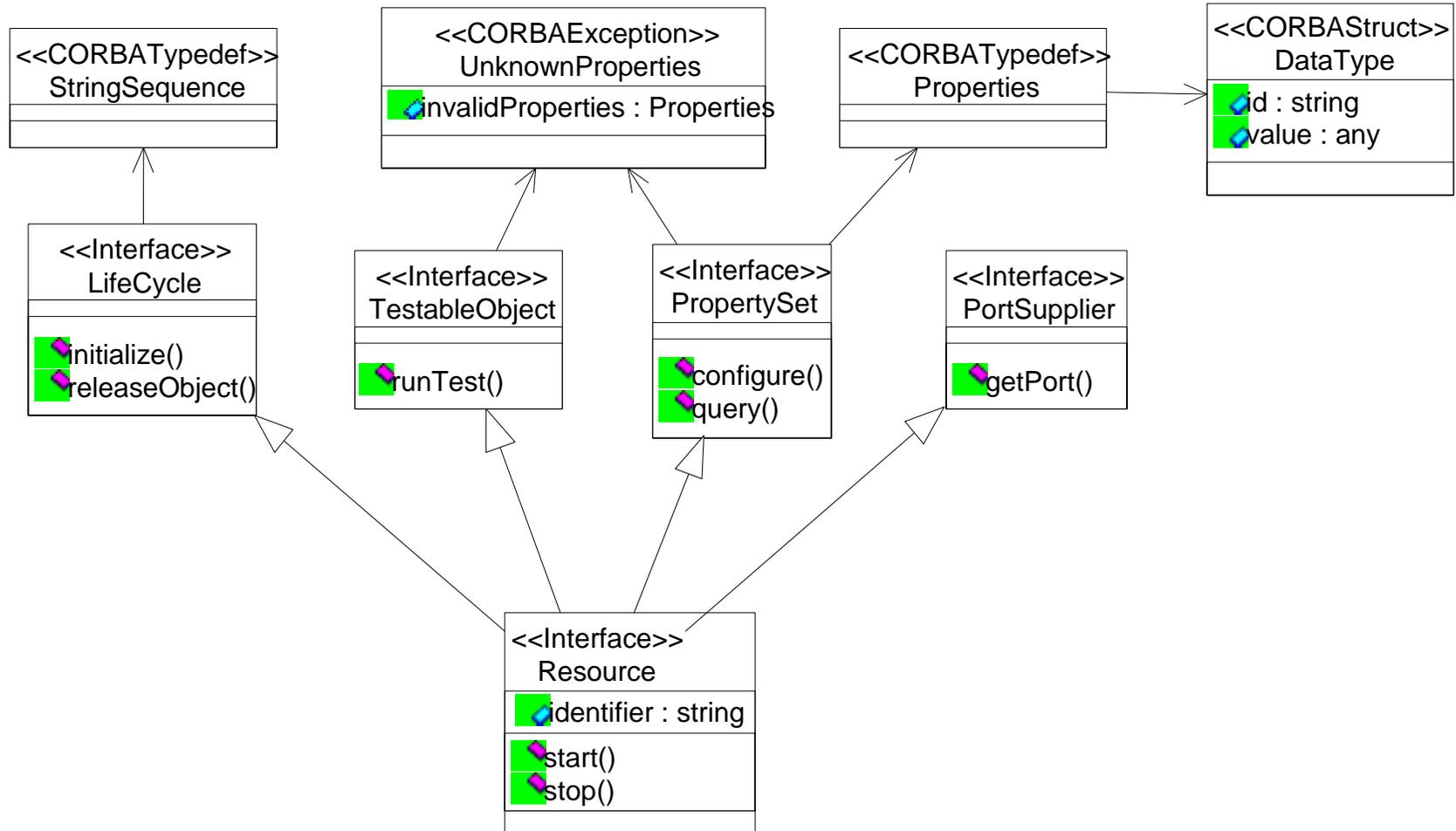
- The Core Framework (CF) is the essential, “core” set of open software Interfaces and Profiles that provide for the deployment, management, interconnection, and intercommunication of software application components in embedded communication systems.
- CF Interfaces (defined in IDL) consist of:
  - **Base Application Interfaces** (*Port, LifeCycle, TestableObject, PropertySet, PortSupplier, ResourceFactory, and Resource*) that provide a common set of interfaces for exchanging information between software application components.
  - **Framework Control** that provide interfaces for the control and management of hardware assets and applications, and domain (system).
    - **Device Interfaces** (*Device, LoadableDevice, ExecutableDevice, AggregateDevice*)
    - **Device Management Interfaces** (*DeviceManager*).
    - **Domain Management Interfaces** (*Application, ApplicationFactory, DomainManager*)
  - **File Service Interfaces** (*File, FileSystem, FileManager*) that provide interfaces for distributed file access services.
- A **Domain Profile** (defined in XML DTD) consists of a set of files that describe the individual components of a software application, their interconnection, and their properties. The properties of embedded hardware devices are also described in the **Domain Profile**.

# Core Framework IDL Relationships



# SCA CF Base Application Interfaces

# Resource Interfaces



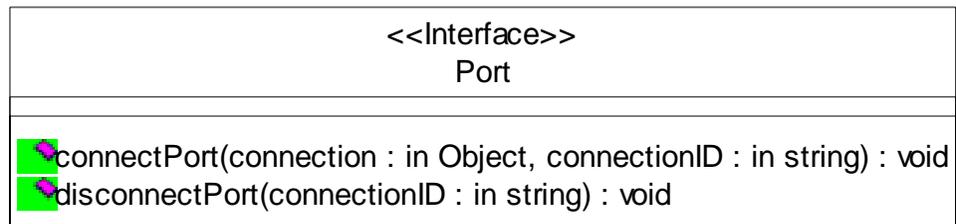
# Resource Interfaces

---

- Inherits from the following other base application interfaces:
  - *Lifecycle* - used to initialize or release the *Resource*
  - *TestableObject* - used to test *Resource* (i.e. BIT)
  - *PropertySet* - provides operations to configure and query *Resource* properties.
  - *PortSupplier* - provides an operation to get a port object reference.
- *Resource* provides additional behavior to:
  - Start / stop processing

# Port Interface

---

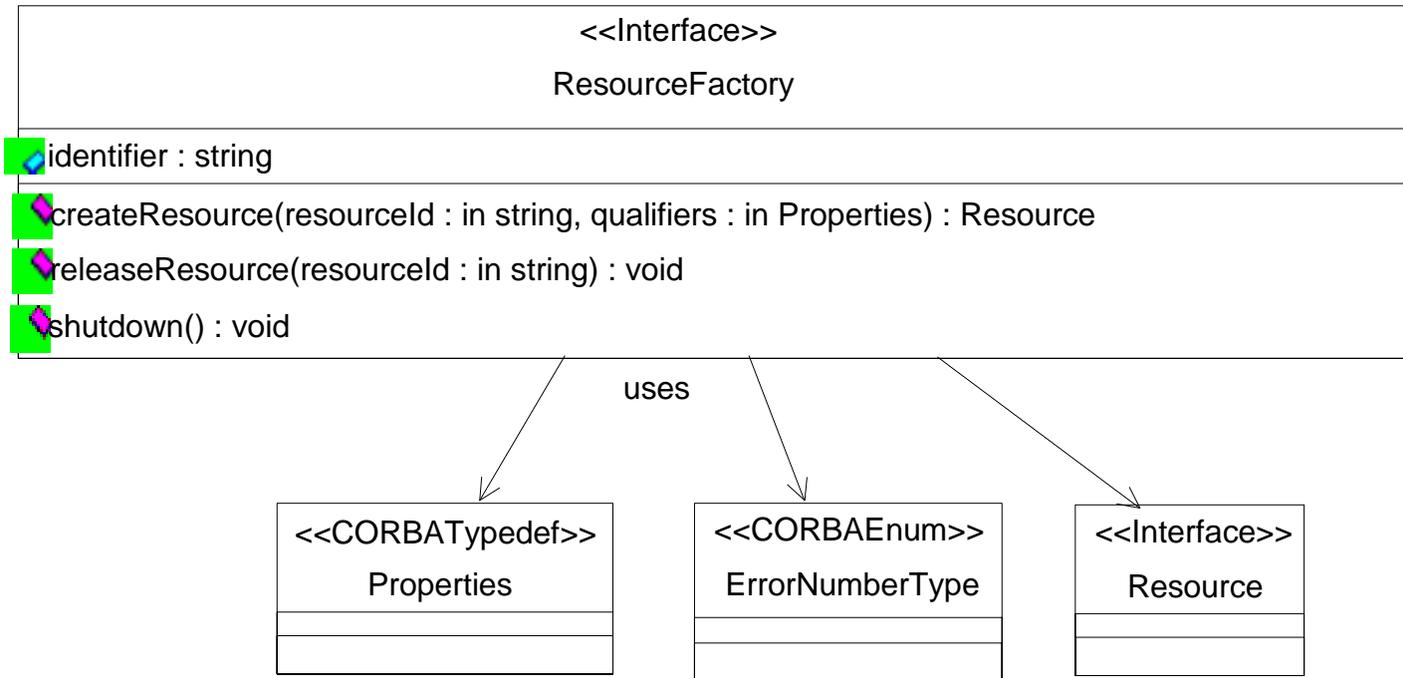


# Port Interface

---

- Used to Connect *Resource* Components
  - Via the connectPort / disconnectPort operations
- *Resource* Components have a set of
  - Uses port(s)
    - A port that uses some set of services provided by a provider component
    - All uses ports implement the *Port* interface
  - Provides port(s)
    - A port that provides some set of services at an interface
    - When that interface is one defined in the API Supplement, the component implements the behavior defined by the appropriate API
  - Port Types
    - Command and Control, Data, Status
    - Basic push types defined in the SCA

# ResourceFactory Interface



# *ResourceFactory* Interface

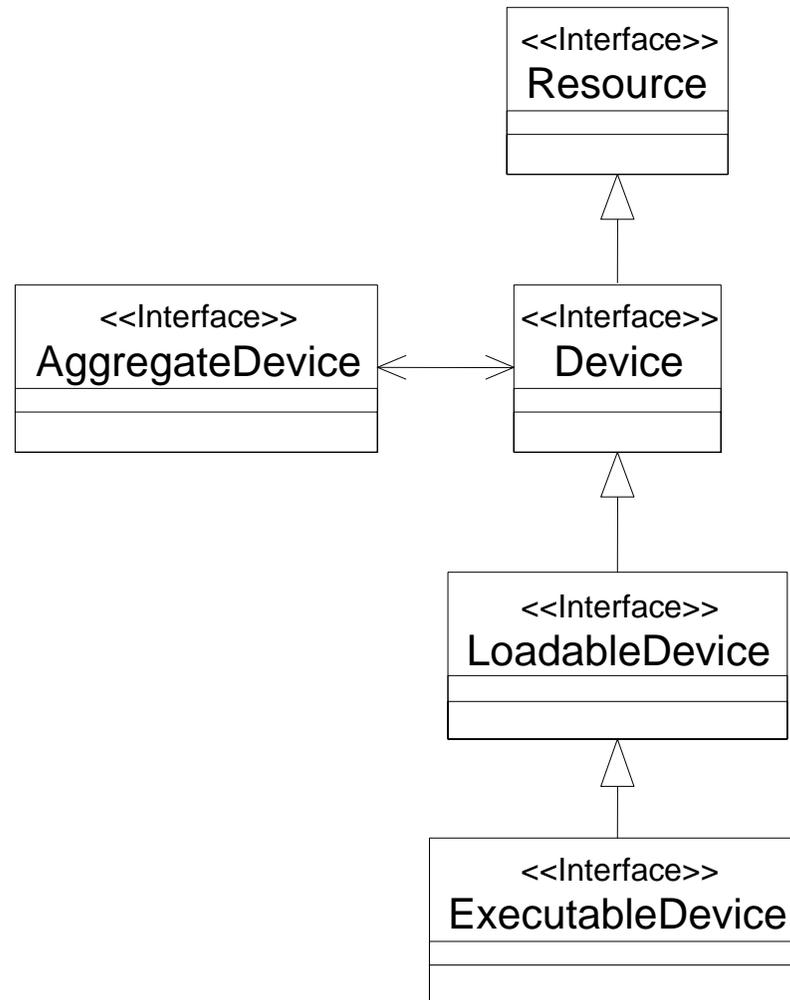
---

- Used to create/tear down Resource(s)
  - Behavior can be homogeneous or heterogeneous (Resources created do not have to be same type) depending on qualifiers in create call
- Modeled after the Factory Design Pattern.
- Provides industry standard mechanism of obtaining a Resource without knowing its identity.
- Optional Interface

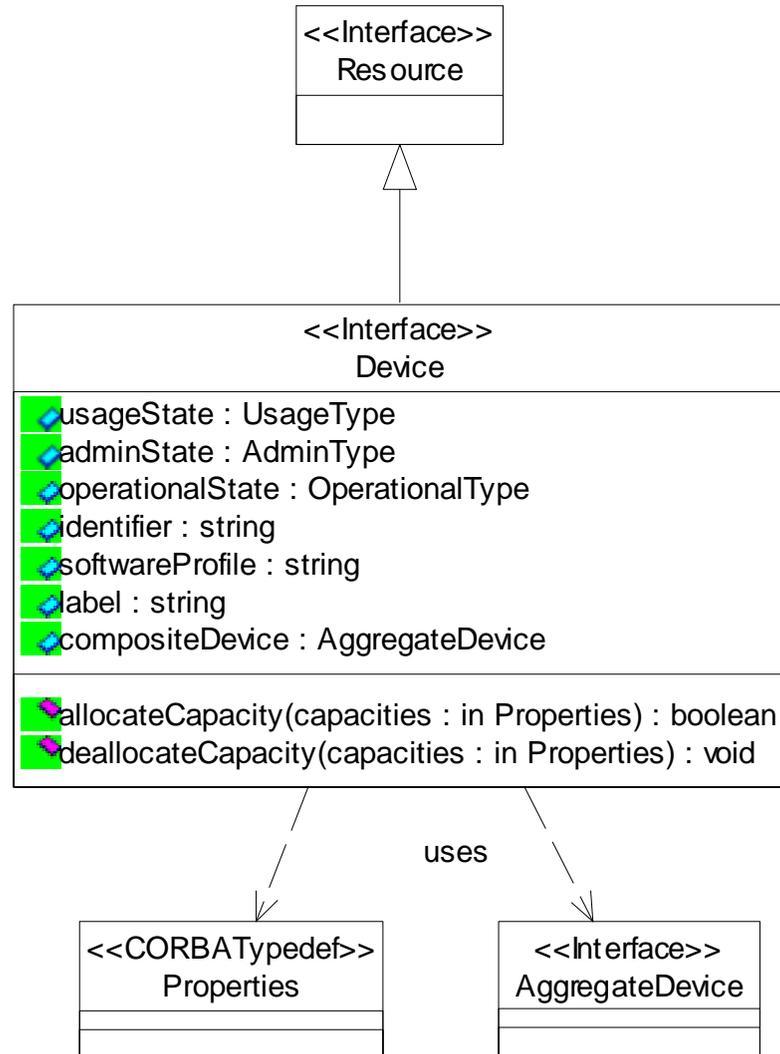
# SCA CF

## Device Interfaces

# Device Interfaces



# Device Interface

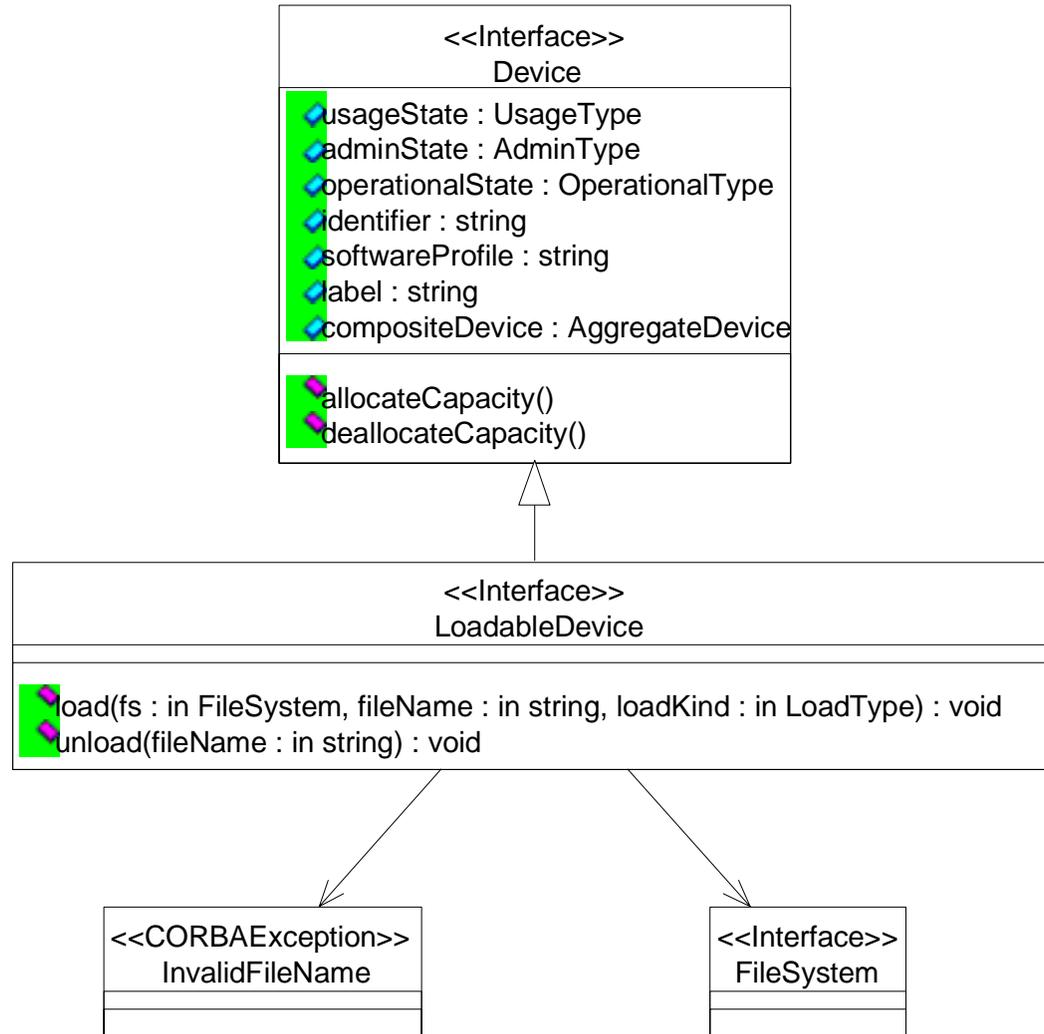


# Device Interface

---

- Defines a “logical Device” in the system
  - An abstraction of a H/W device
  - Typically one logical *Device* per H/W device
    - Can be aggregation of multiple child *Devices*
    - For example: A physical modem device represented by a logical modem *Device* with TDMA modem *Device*, CDMA modem *Device*, FM modem *Device*, etc. child *Devices*
- Provides state management interfaces based on X.731
- Defines the capacity model for the device

# LoadableDevice Interface

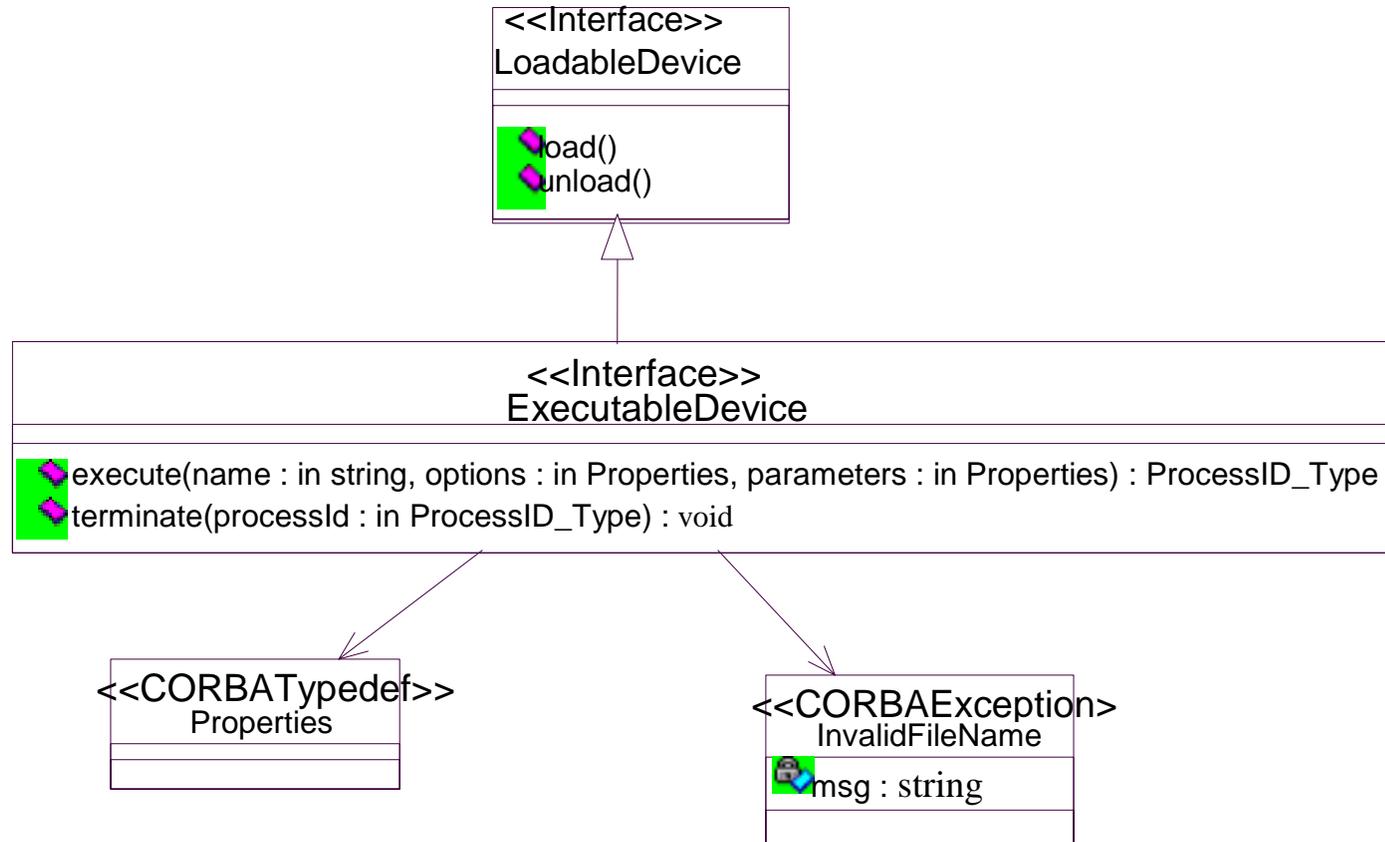


# *LoadableDevice* Interface

---

- This interface extends the *Device* interface by adding software loading and unloading behavior to a *Device*.
- Types of load that can be performed are
  - Kernel Module
  - Driver - Device Driver
  - Shared - Dynamic Linking
  - Executable - Main POSIX process
- Example Loadable Devices: Modem, FPGA, etc.

# ExecutableDevice Interface

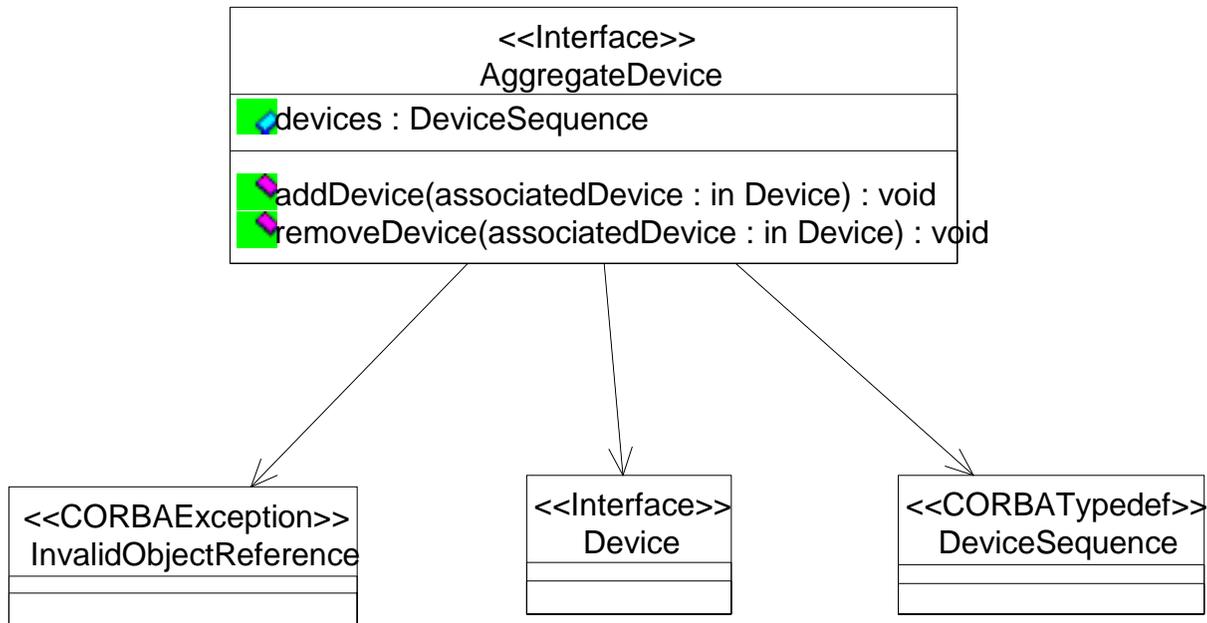


# ExecutableDevice Interface

---

- This interface extends the *LoadableDevice* interface by adding execute and terminate behavior to a Device.
- Execute's name parameter, may be the name of a function (thread) or a file (process) depending on the device implementation (OS capability).
- Execute's options parameter, has two options: `STACK_SIZE_ID` and `PRIORITY_ID`
- Execute's parameters parameter are user defined parameters passed to the executable image as “argv” parameters
- Example Executable Devices: General Purpose Processor (Intel, PowerPC), Modem that can be commanded to execute code

# AggregateDevice Interface



# AggregateDevice Interface

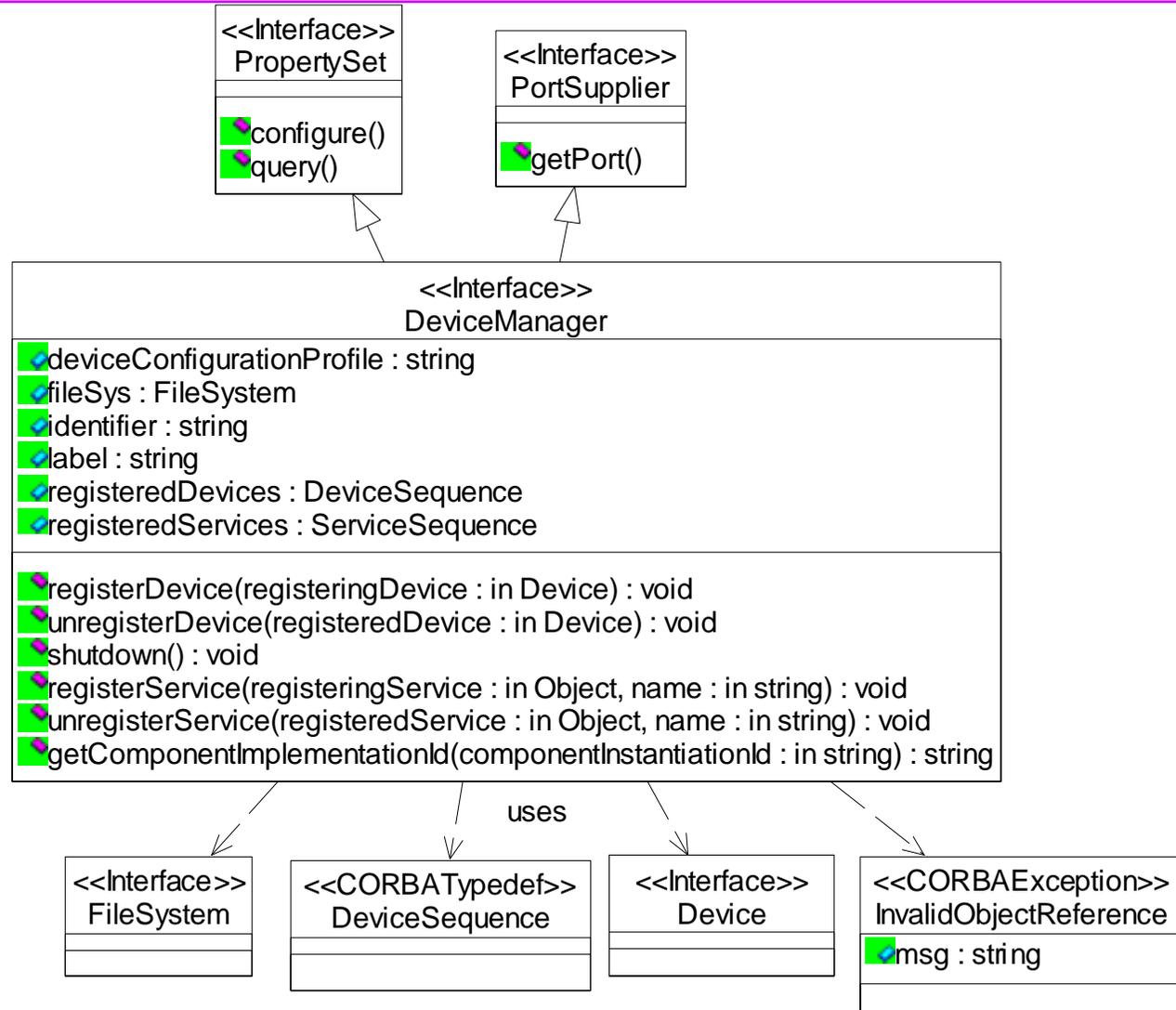
---

- An aggregated logical Device adds itself to a composite Device.
- An aggregate Device uses this interface to add or remove itself from the composite Device when it is being created or torn-down.
- The devices attribute contains a list of Devices that have been added to this Device through the `addDevice( )` operation and have not yet been removed through the `removeDevice( )` operation.
- Example Devices: INFOSEC Device with “n” crypto channel Devices or I/O Device with “n” ports

# SCA CF

## Device Management Interface

# DeviceManager Interface



# DeviceManager Interface

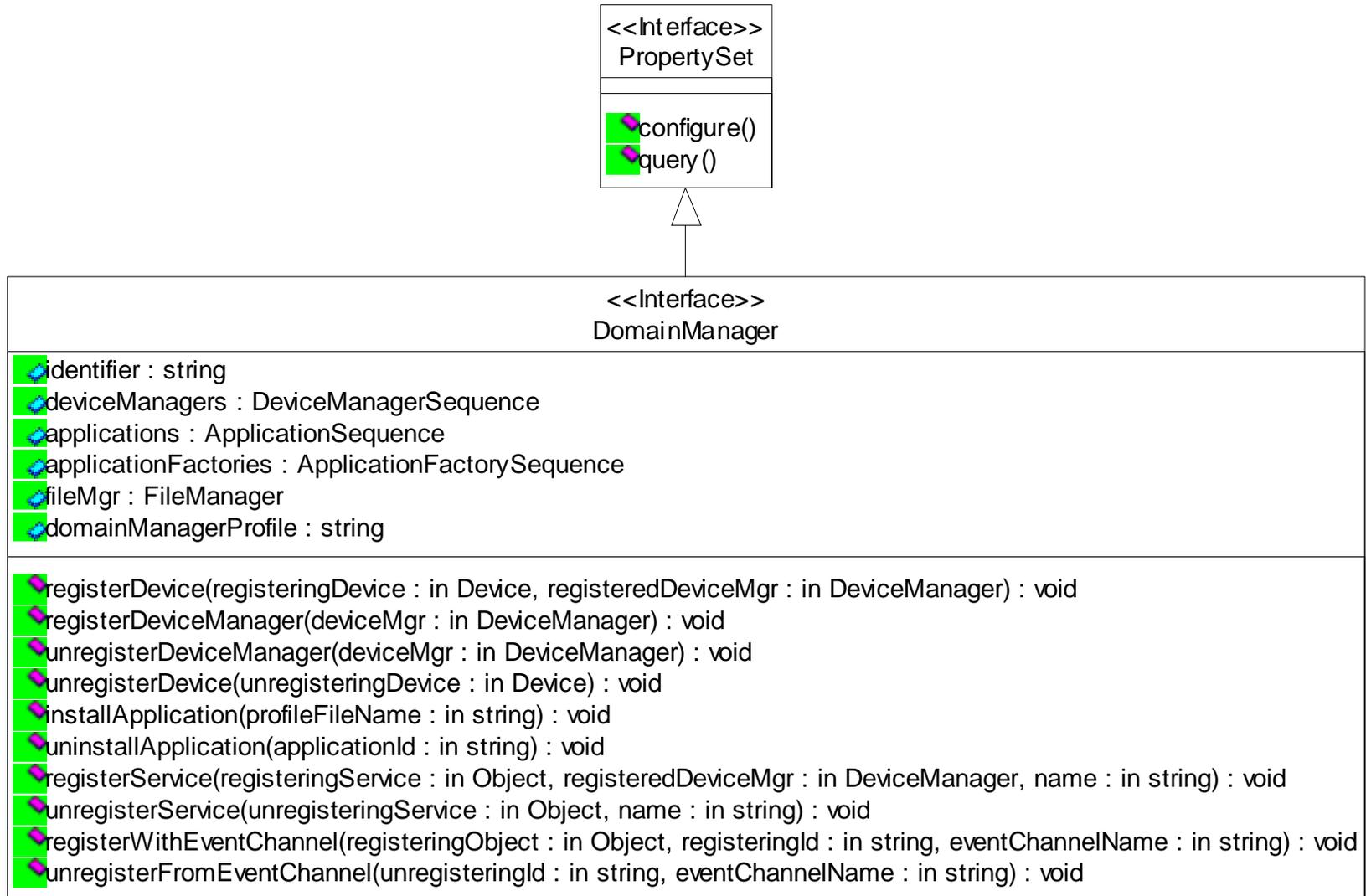
---

- The *DeviceManager* interface is used to manage a set of logical *Devices* and services on a node.
- Typically represents a CORBA capable “board” in a system
- Creates *FileSystem* object
- UnRegisters/Registers itself and its *Devices* and services with the *DomainManager*
- Uses its Device Configuration Descriptor (DCD) profile for determining
  - How to obtain object reference of *DomainManager*
    - Naming Service
  - *Device* and *Services* components to be deployed
  - File System names, etc.

# SCA CF

## Domain Management Interfaces

# DomainManager Interface

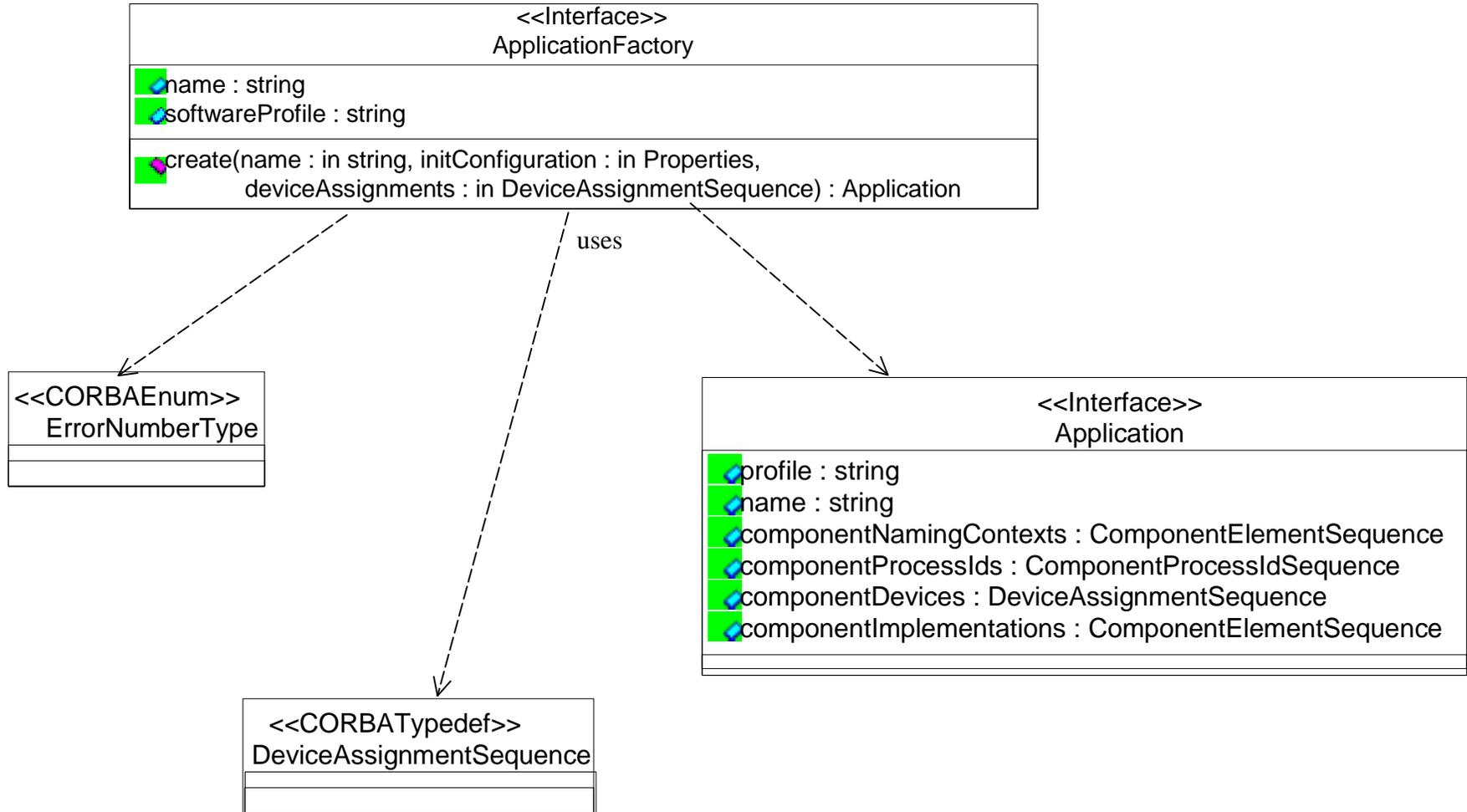


# DomainManager Interface

---

- *DomainManager* provides interfaces for:
  - Registration (register / unregister) of:
    - *DeviceManager(s)*, *Device(s)*, *Application(s)*, and *Services*
  - Access to:
    - Registered *DeviceManager(s)*
      - **Registered Devices and Services**
    - Installed and Executing Applications
    - The Radio's *File System*
  - HCI to:
    - Configure the domain
    - Get the domain's capabilities (*Devices* and *Applications*)
    - Initiate maintenance functions

# ApplicationFactory Interface

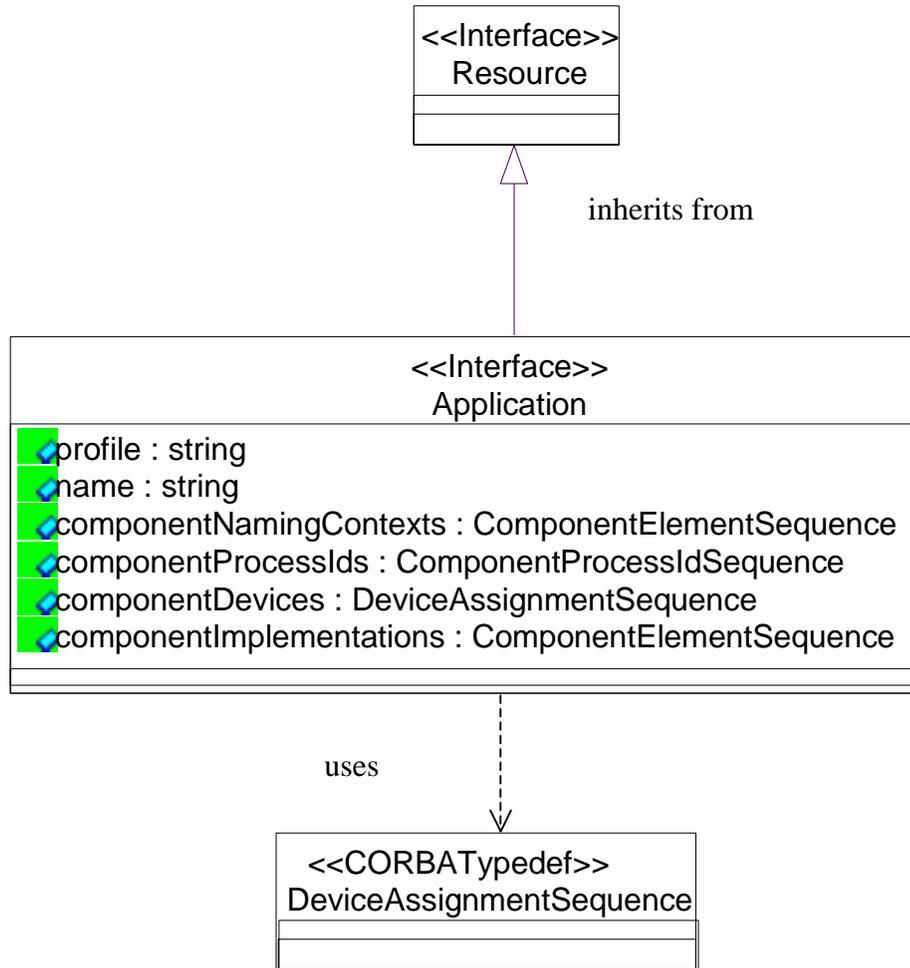


# *ApplicationFactory* Interface

---

- Part of Domain Management
  - Implemented by CF supplier, not waveform developers
- *ApplicationFactory*
  - Used to create *Application* (waveform) instances
  - Based on Domain Profile:
    - Allocates software (*Resources*) to hardware (*Devices*)
      - Allocates capacities against *Devices*
    - Connects *Resources* that make up an *Application*
    - Performs initial configuration

# Application Interface



# *Application* Interface

---

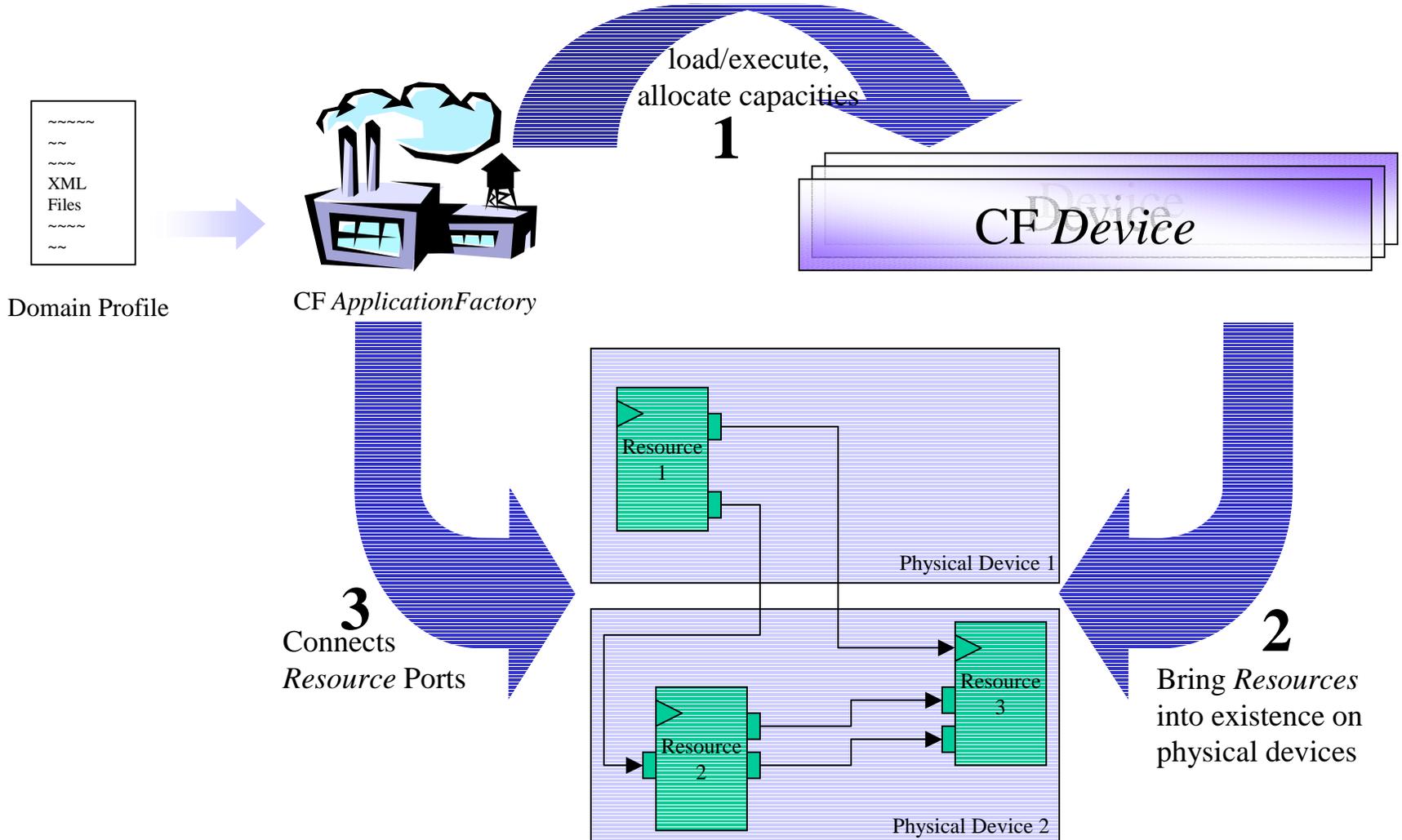
- Part of Domain Management
  - Implemented by CF supplier, not waveform developers
- *Application*
  - Container for *Resources* that make up an application
  - Provides the interface for instantiated applications:
    - Control, configuration, status, tear-down
    - Returns capacities to *Devices* on tear-down

# Application Instantiation

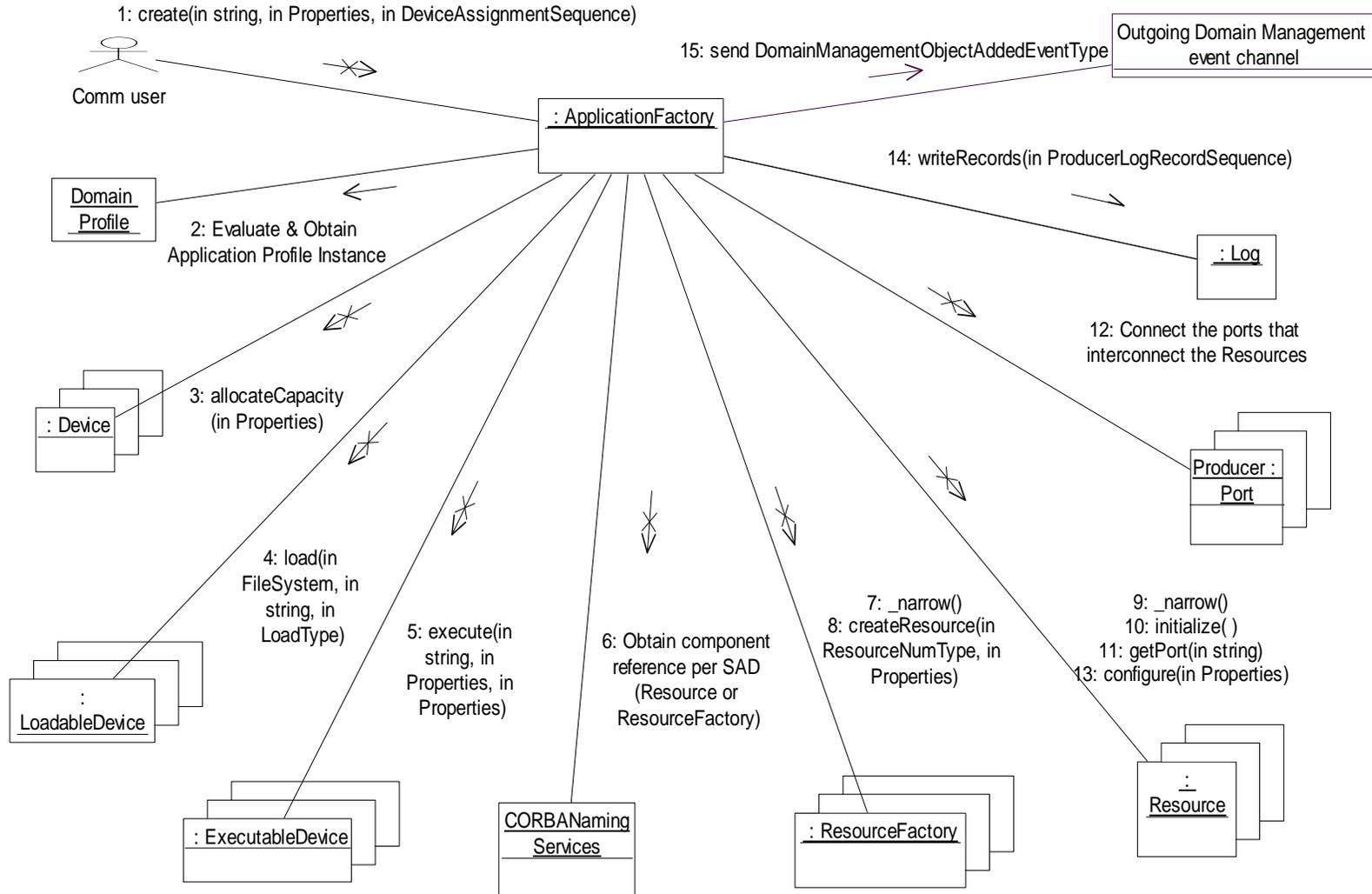
---

- User Interface (UI) asks for all *ApplicationFactory(s)*
  - Application Factory is chosen
  - UI issues create() on *ApplicationFactory*
- *ApplicationFactory* determines applicable *Device(s)* on which to load application code defined in Domain Profile
  - AllocateCapacity/load/execute are called on *Device(s)*
    - brings *Resource(s)* into existence
- *Resource(s)* bring *Port(s)* into existence
- *ApplicationFactory* connects the *Port(s)*
- *Resources* are then configured, initialized, and started
- CF Application is returned – providing Proxy interface to Assembly Controller of created *Resources*.

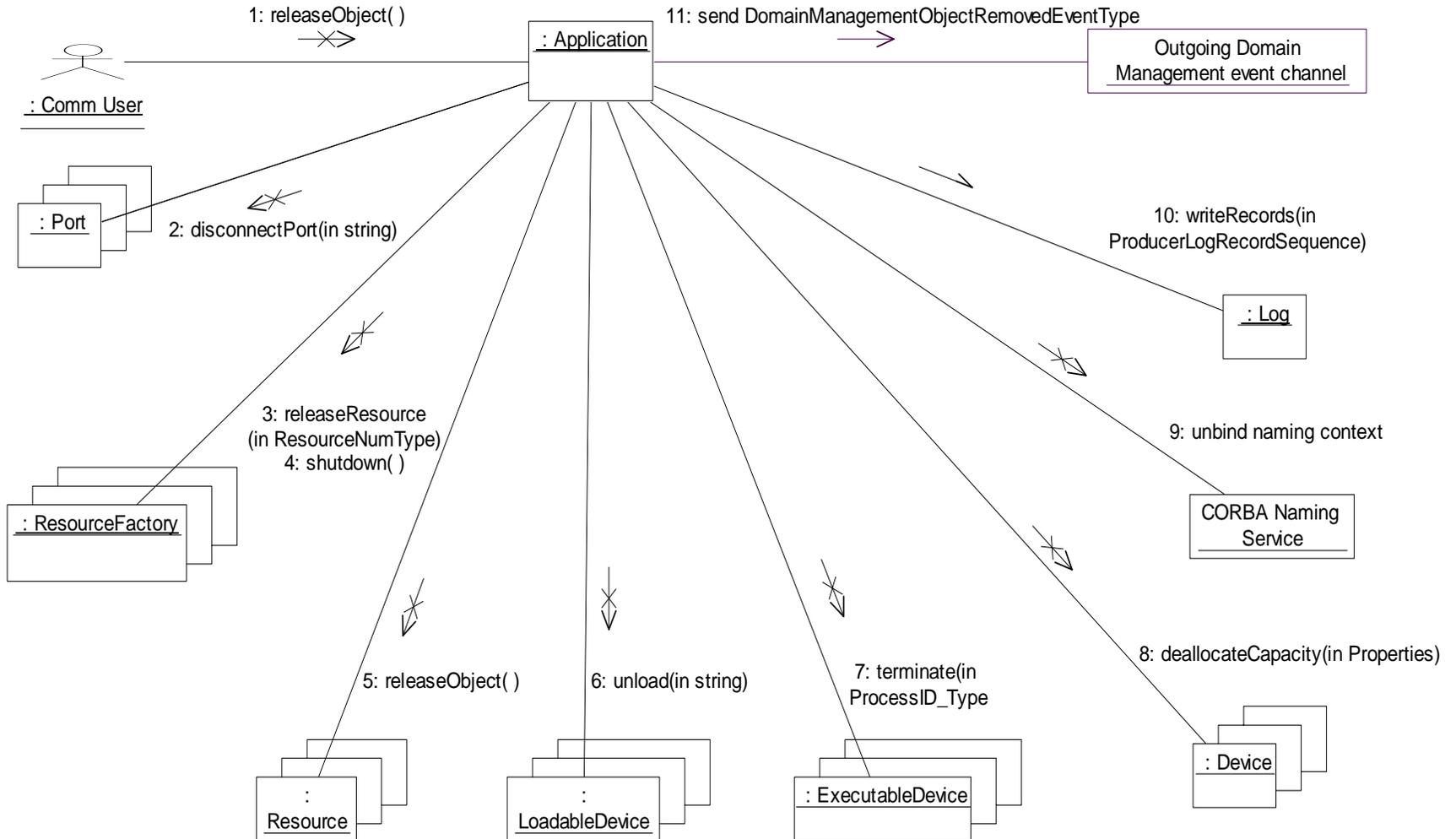
# Application Instantiation/Deployment



# Application Instantiation Example



# Application Tear-Down Example



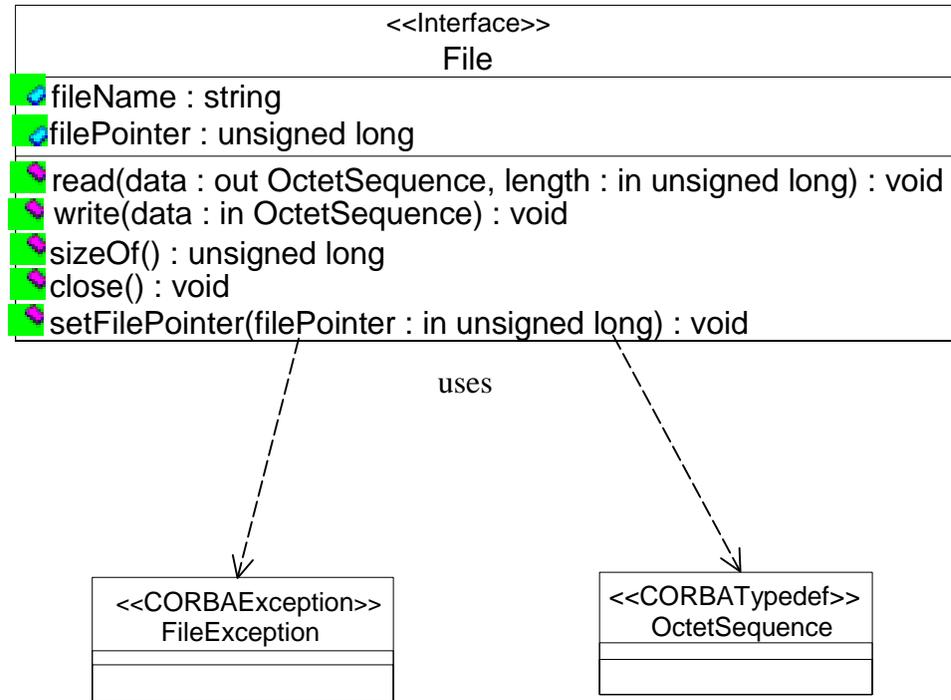
# SCA CF File Service Interfaces

# *File, FileSystem & FileManager Interfaces*

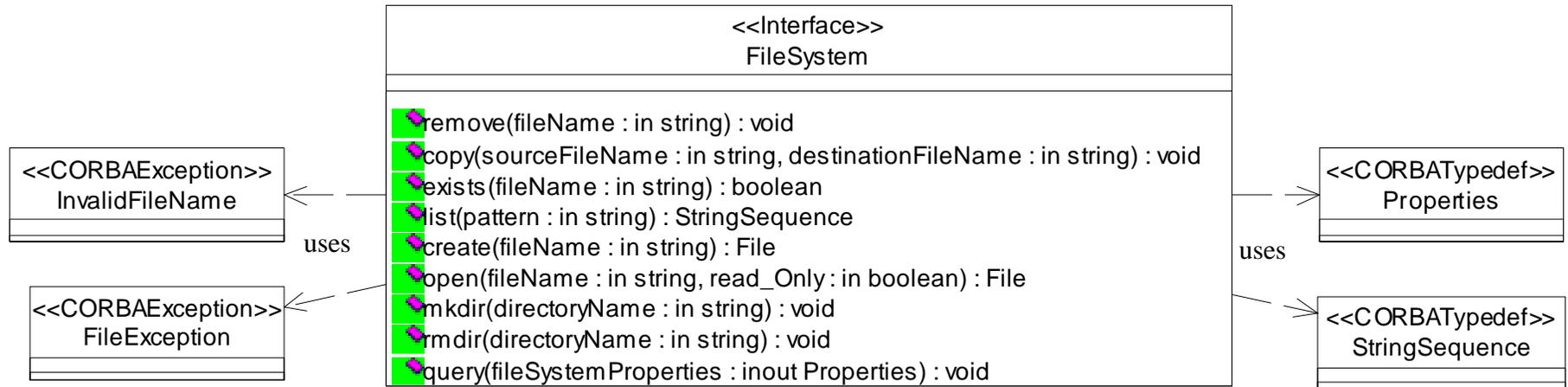
---

- *FileManager, FileSystem, and File* provide
  - Access to Files, FileSystems
  - Clients use these interfaces for all file access.
- *FileManager*
  - Manages multiple distributed *FileSystems*
  - Looks like a *FileSystem* to the client
- *FileSystem*
  - Enable remote access to physical file systems
  - Allows creation, deletion, copying, etc. of files
- *File*
  - Provides access to files within the radio
  - Allows access across processor boundaries (distributed *FileSystems*)

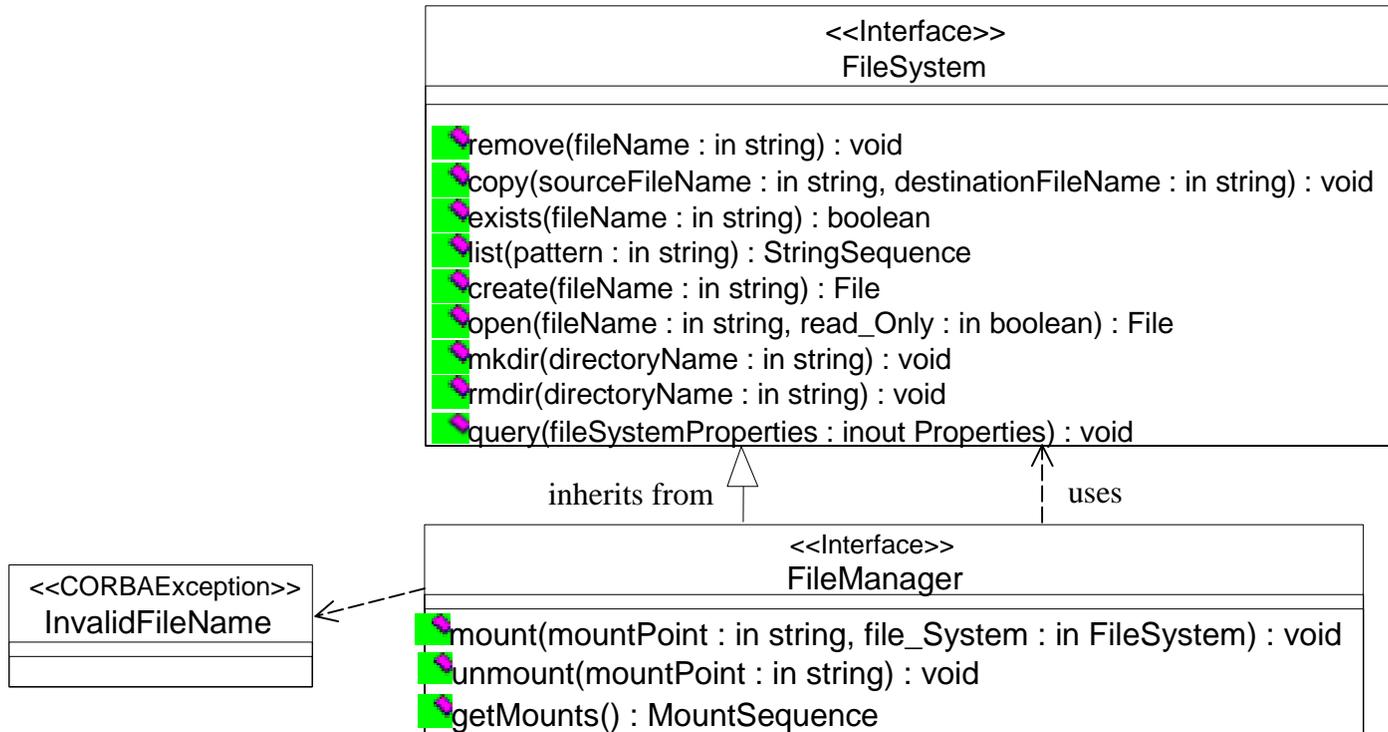
# File Interface



# FileSystem Interface



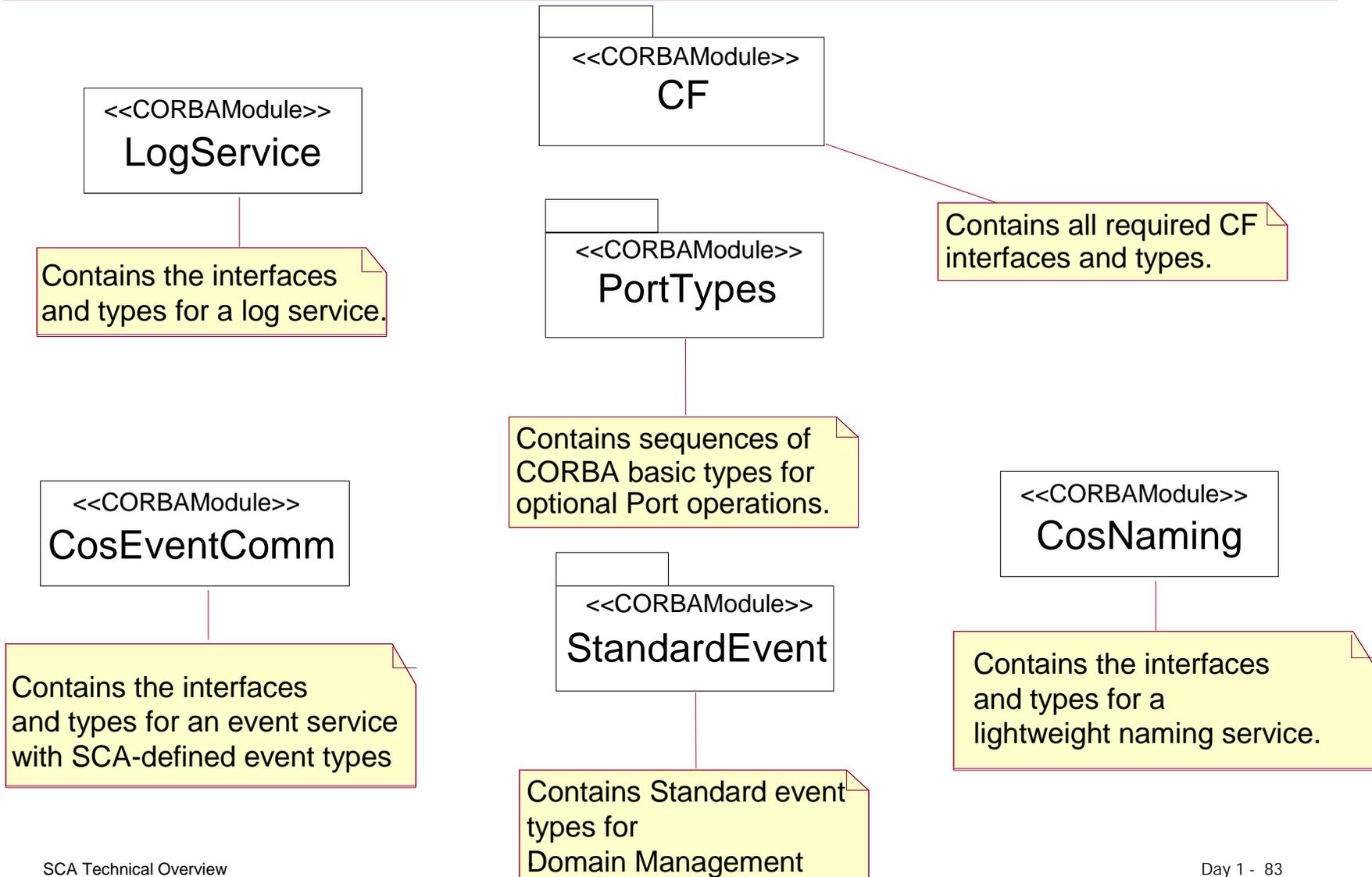
# FileManager Interface



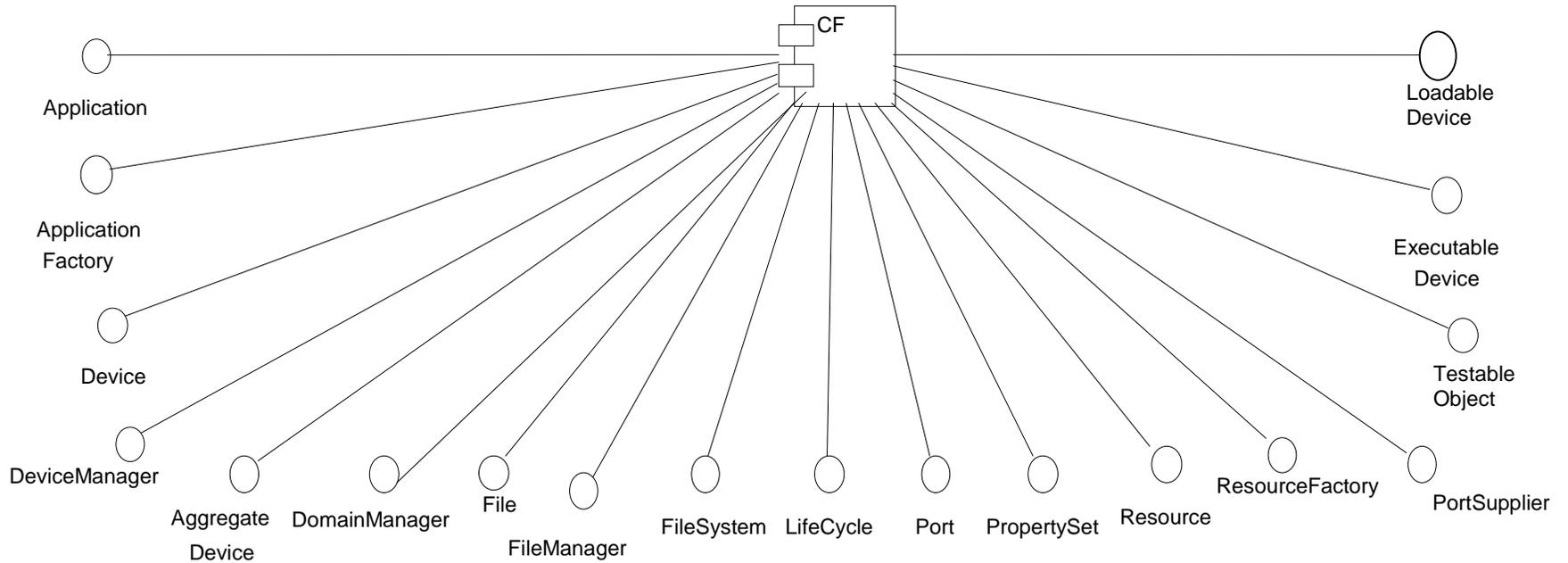
# SCA IDL Modules

# Core Framework

## IDL Module Relationships



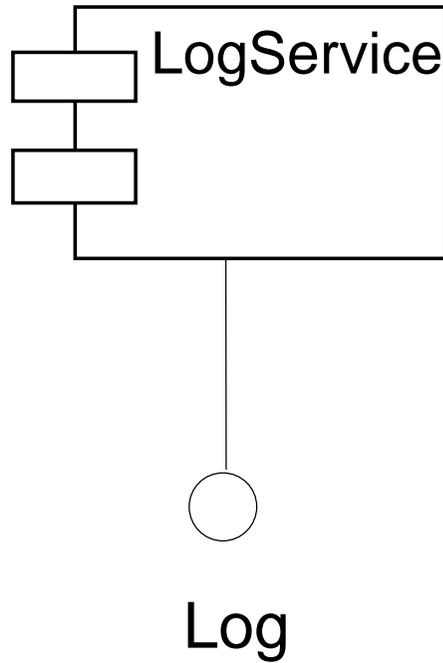
# CF Module Interfaces



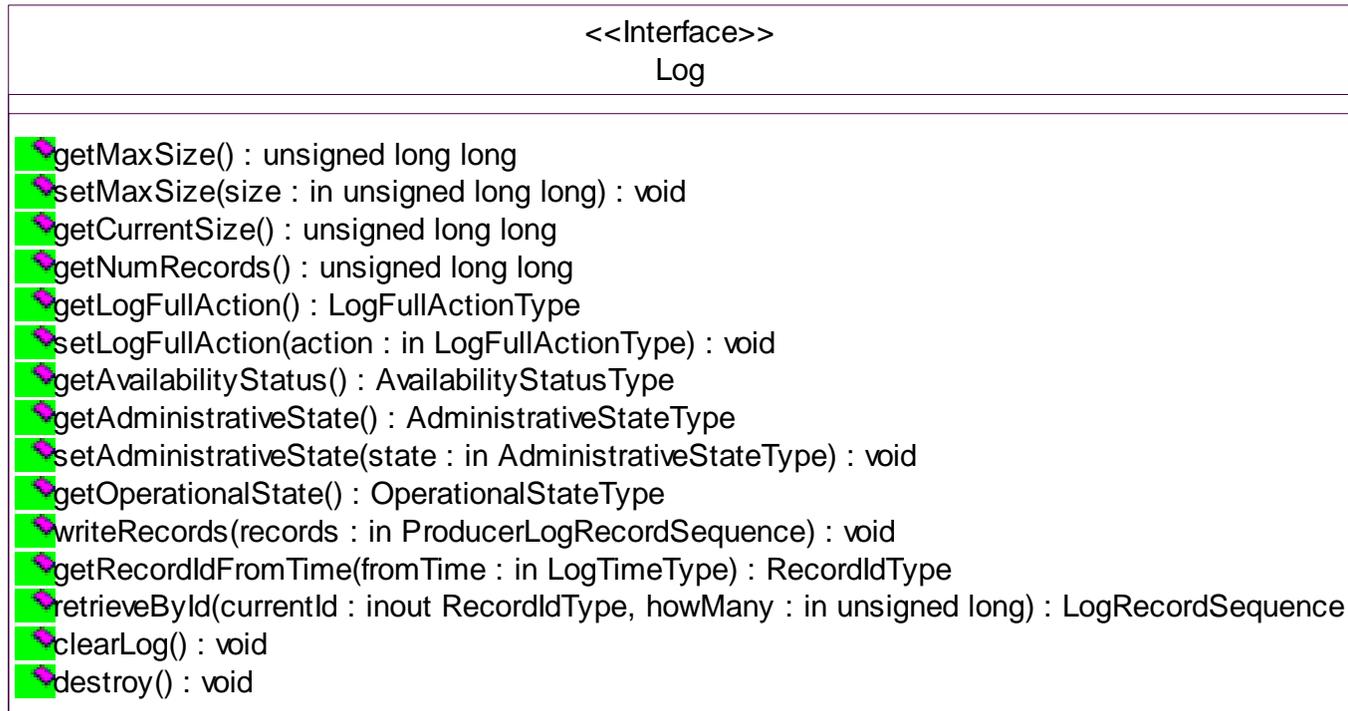
# SCA Log Interface

# LogService Module

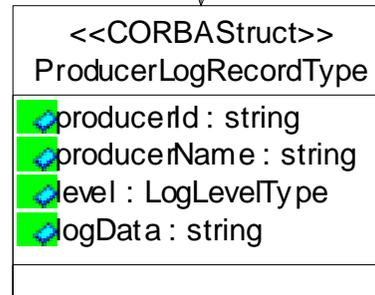
---



# Log Interface



uses



# Log Interface

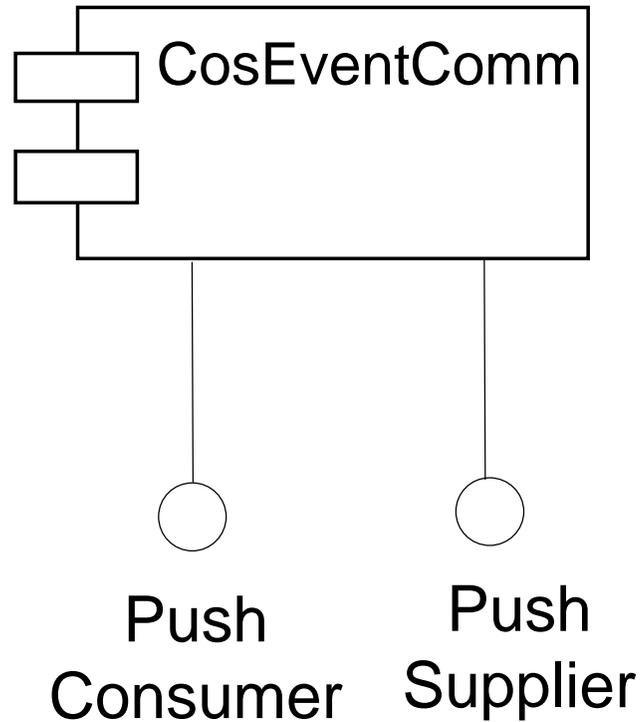
---

- Capabilities
  - Can write a set of producer log records at a time
  - Can control the size of the *Log*
  - The *Log* time stamps each log record received
  - Can clear a *Log*
  - Can control Log full condition (halt, wrap)
  - Can retrieve a set of log records at one time.
  - The *Log* has states (operational and administrative)

# SCA Event Service Interface

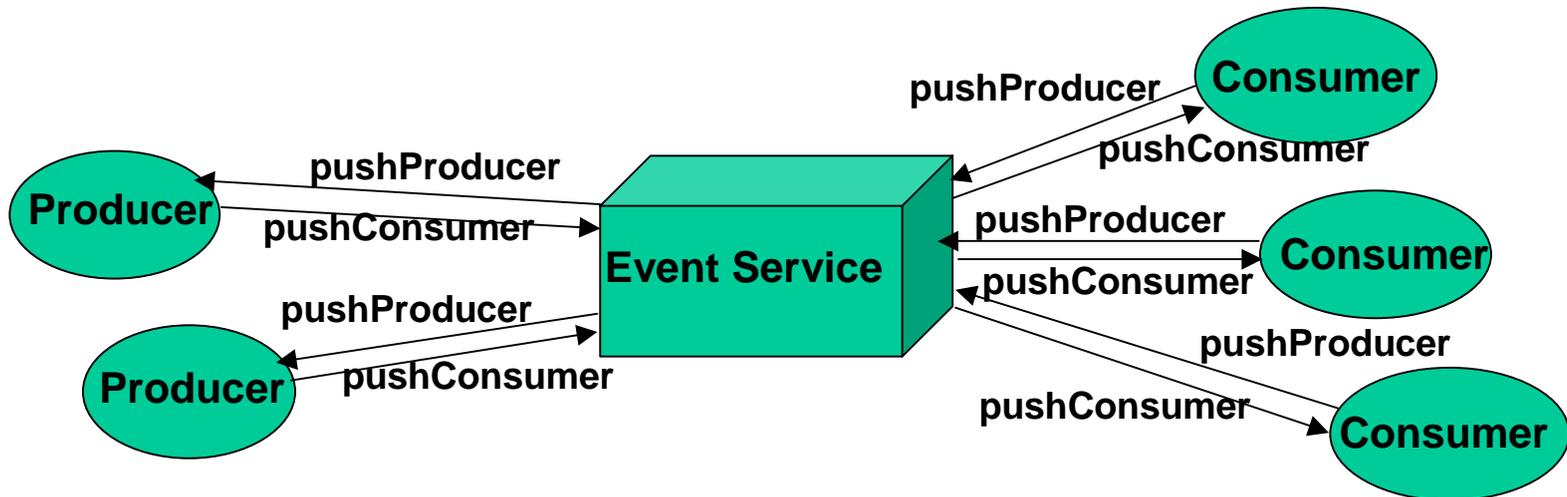
# Event Service Module

---



# Event Service Concepts

- Based upon OMG Push Model



# Event Service

---

- Event Service decouples the communication between consumer and producer objects, where consumer components are unaware of producer components, and vice versa.
- Consumer components process event data that are produced by producer components.
- Based upon the Push Model approach where producers push events to consumer, as described in OMG Document formal/01-03-01: Event Service, v1.1.
- Includes capability to create event channels.
  - An event channel is both a consumer and a producer of events.
  - An event channel permits multiple suppliers to communicate with multiple consumers asynchronously.

# SCA Standard Event Interface

# Standard Event Module

---



# Standard Events

---

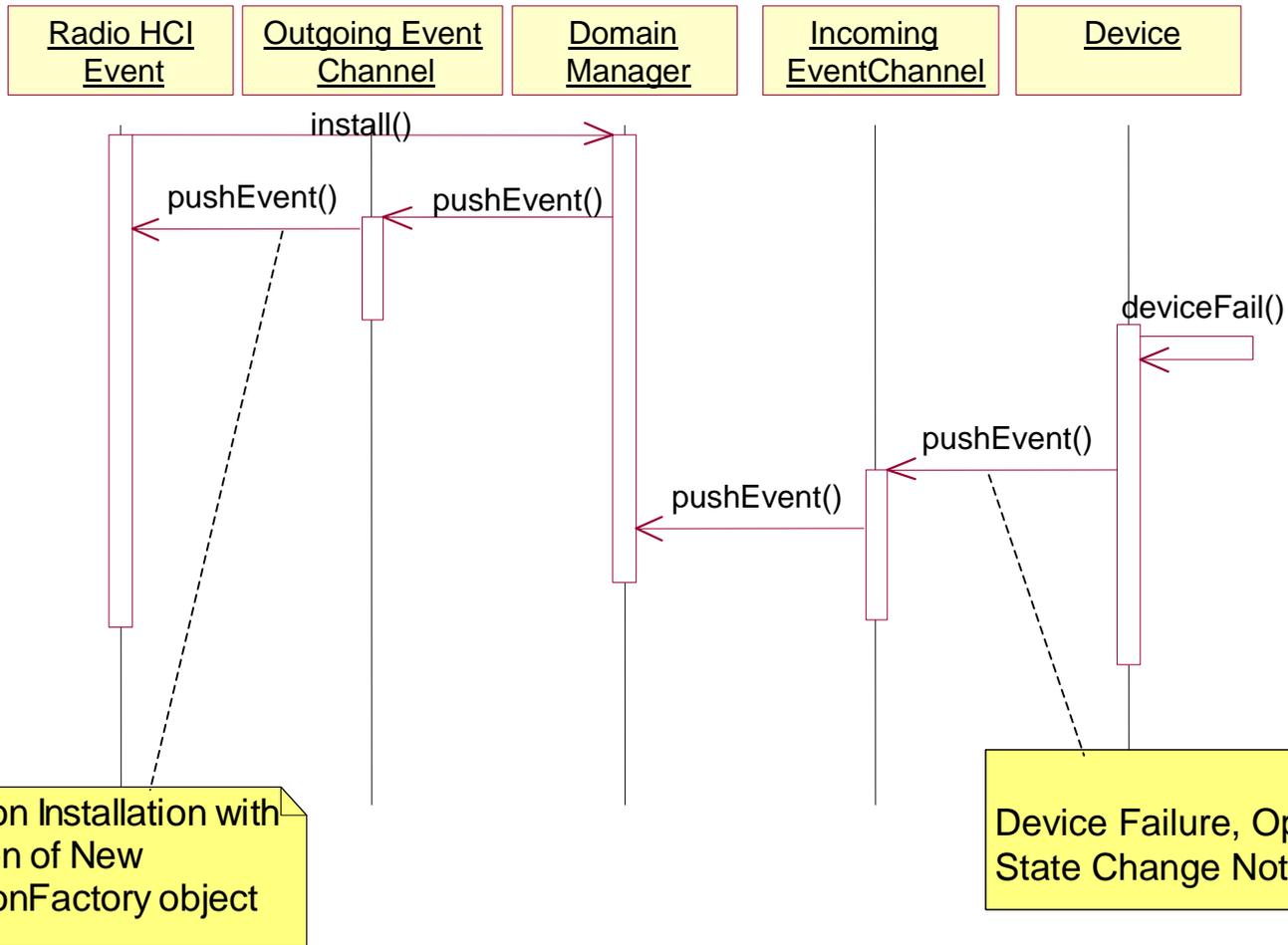
- Event Channels
  - Standard Event Channels
    - Incoming Domain Management (IDM) Channel
      - Defined as the standard event channel for incoming events to be processed by the Domain Management function.
      - Used by Components within Domain to generate events that are consumed by Domain Management functions.
    - Outgoing Domain Management (ODM) Channel
      - Defined as standard event channel for Domain Management outgoing events.
      - Used by clients to receive events generated from Domain Management functions.
  - User-Defined Channels
    - WF can specify creation and usage of event channels

# Standard Event Types

---

- **IDM Channel**
  - **StateChangeEvent** - Structure that indicates that the state of the event source has changed.
    - **StateChangeCategoryType** - Category of the state change that occurred.
    - **StateChangeType** - Enumeration that identifies the specific states of the event source before and after the state change.
- **ODM Channel**
  - **DomainManagementObjectAddedEvent** - Structure that indicates that the event source has been added to the domain.
    - **DomainManagementObjectRemovedEvent** - Structure that indicates that the event source has been removed from the domain.
    - **SourceCategoryType** - Enumeration that identifies the type of Object that has been added/removed from the domain.

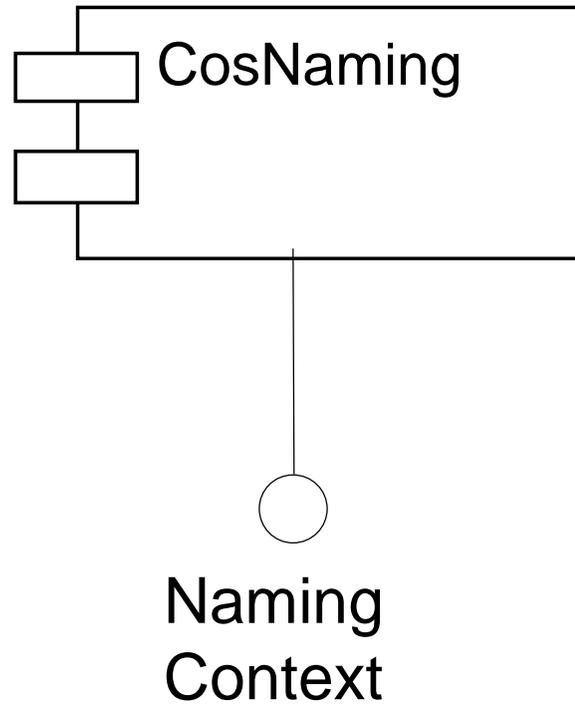
# Standard Event Interface



# SCA Naming Service Interface

# Naming Service Module

---



# SCA Naming Service

---

- Naming Service
  - Works like a white pages directory.
  - Objects register with the Naming Service giving the name and their object reference (or handle).
  - Client may specify name to Naming Service in order to obtain a object reference of the registered object.
  - Interoperable Naming Service desired - specified by the OMG Document formal/00-11/01:Interoperable Naming Service Specification.
    - SCA defines minimum Naming Service functionality

# SCA Domain Profile



# Domain Profile Purpose

---

- JTRS Requirements
  - Portability
    - Across Platforms and Platform Classes
  - Interoperability
- Domain Profile
  - Used to deploy Applications, logical Devices, and Services into a SCA compliant system
  - Based on the CORBA Components Model
  - Extended for purposes of supporting SCA Requirements for Device Management.

# Domain Profile Files

---

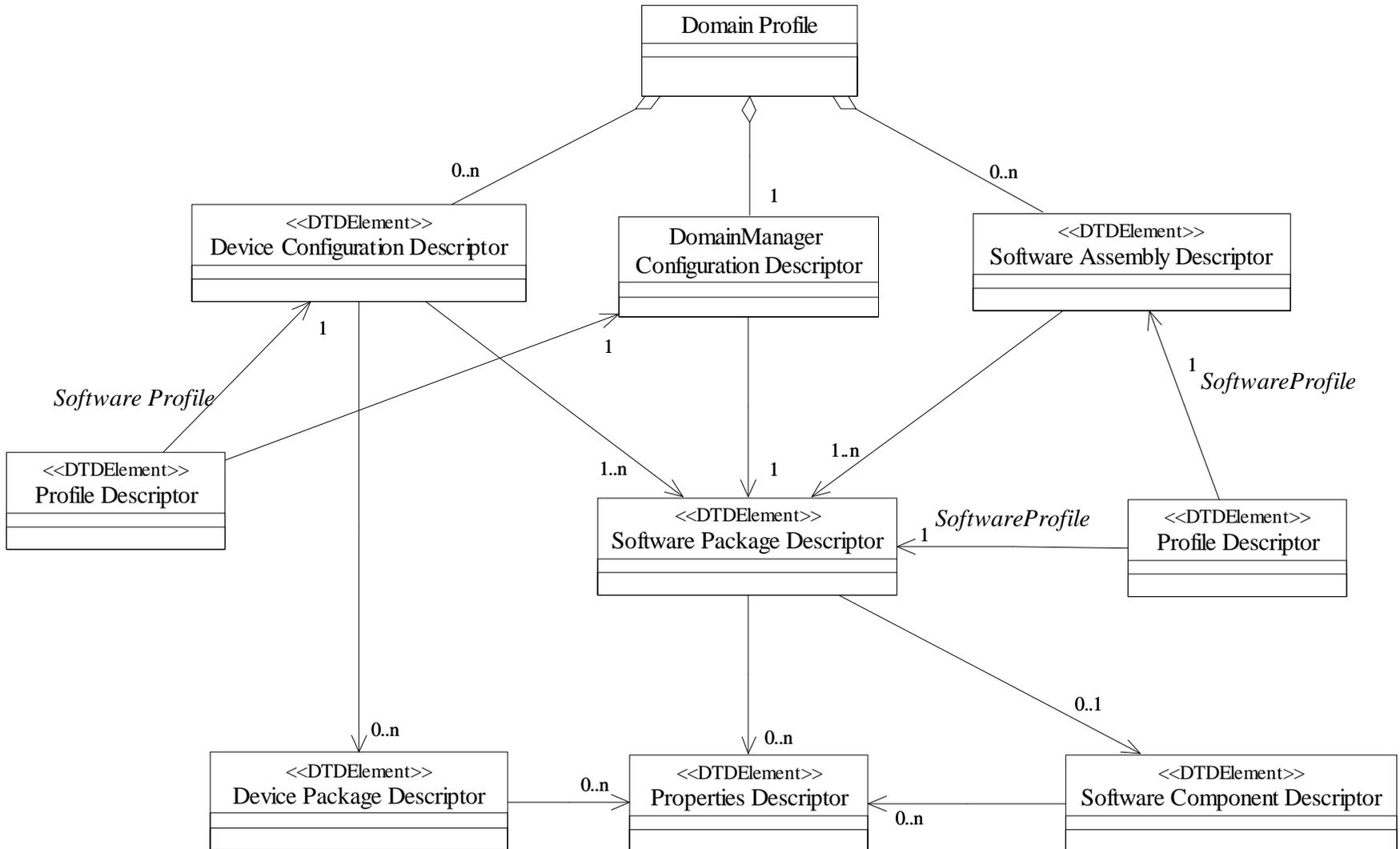
- Made up of 8 file types
  - Software Package Descriptor (SPD)
    - Describes a component (CORBA and non-CORBA) implementations
  - Property File (PRF)
    - Describes properties for a component.
  - Software Component Descriptor (SCD)
    - Describes a CORBA component characteristics
  - Software Assembly Descriptor (SAD)
    - Describes an application's deployment characteristics
  - Device Configuration Descriptor (DCD)
    - Describes configuration characteristics for a *DeviceManager*.

# Domain Profile Files, (cont'd)

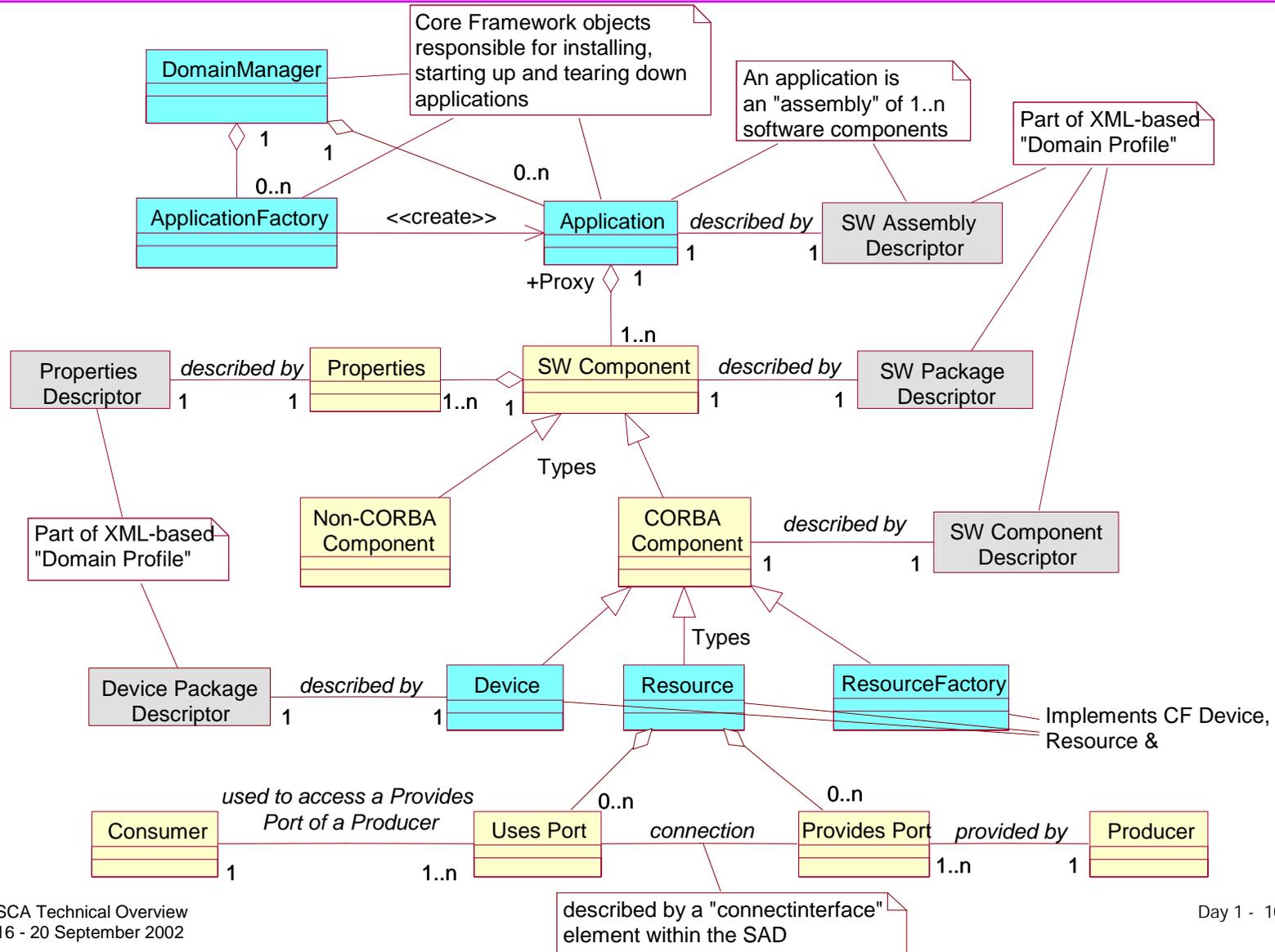
---

- Made up of 8 file types, cont'd
  - DomainManager Configuration Descriptor (DMD)
    - Describes configuration characteristics for a DomainManager.
  - Device Package Descriptor (DPD)
    - Identifies a class of hardware device and its characteristics
  - Profile Descriptor
    - Describes a type of file (SAD, SPD, DCD, DMD) along with the file name.
- Domain Profile files are based upon eXtensible Markup Language (XML) Document Type Definitions (DTDs)
  - A DTD defines the meaning and compliant syntax for a XML file
  - Each XML File contains
    - XML declaration “?xml” - Specifies the XML version and whether the document is standalone.
    - Document Type “!DOCTYPE” - specifies the DTD for the XML

# Domain Profile XML DTD Relationships



# SCA Components to XML Relationships



# HW Architecture Overview

# Hardware Architecture

---

- HW described in Object Oriented terms
  - consistent with SW Architecture description
  - reinforces concept that application functions can be implemented in either HW or SW
- HW not specified to level that SW is
  - domain constraints will drive HW implementation choices
- General rules apply to provide for HW device portability where possible

# Hardware Rule Set

---

- Each HW device provided with Domain Profile files
- HW Critical Interfaces defined in an ICD
  - available without restriction
  - critical interfaces defined a replaceable device boundaries
- HW Critical Interfaces in accordance with commercial / government standards
  - exception allowed for performance but open documentation of non-standard interfaces required

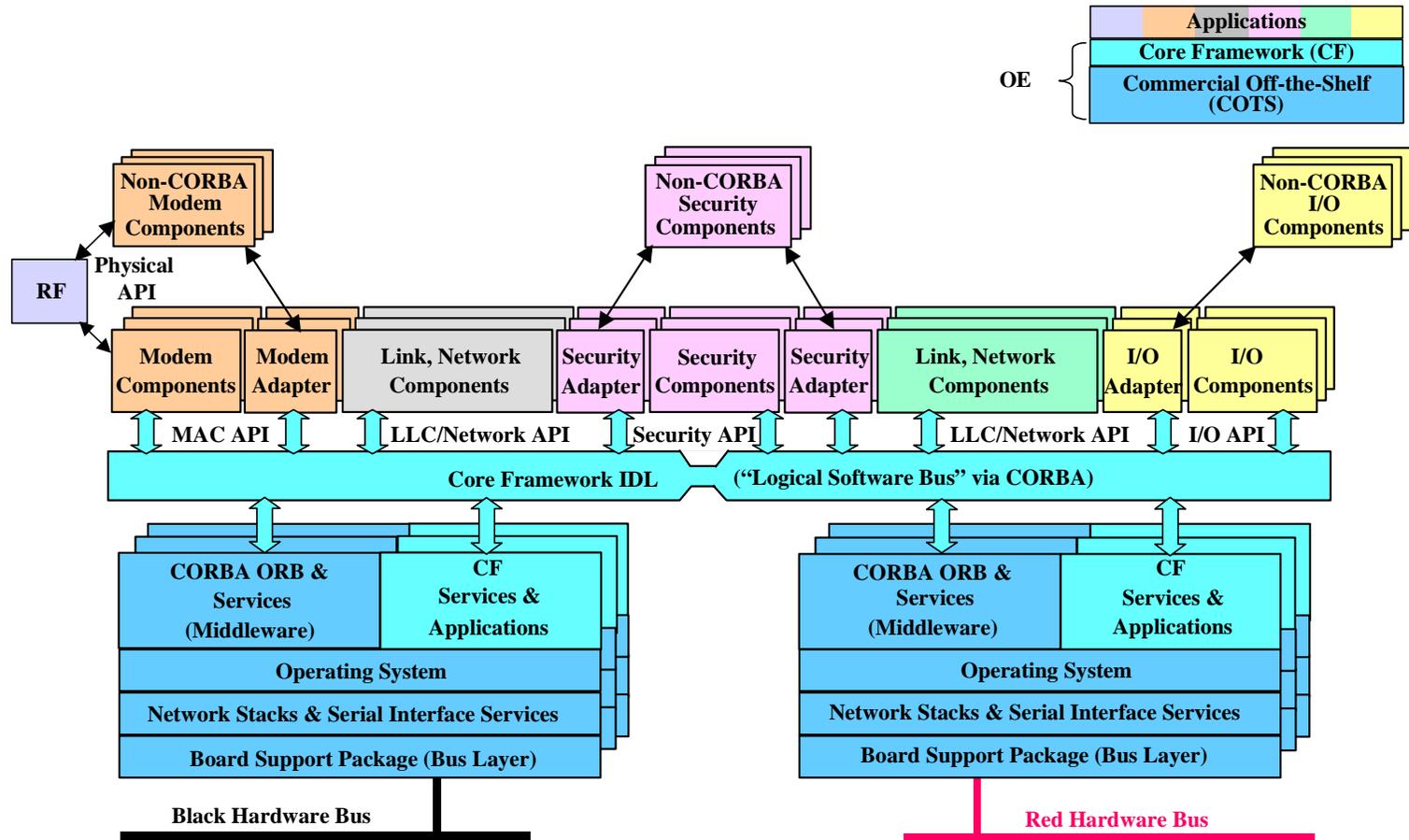
# Intro to API Supplement

# Why APIs are Defined in SCA

---

- Standardized APIs are essential for portability of applications and interchangeability of devices.
- APIs guarantee Service Provider and User can communicate regardless of OE or programming language.

# Application Program Interfaces



*APIs are Located at Logical Boundaries  
Common to Waveforms*

# SCA API Decisions

---

- Why waveform-specific?
  - desire is single API at an interface
  - the range and variety of services at the various interfaces, most notably the MAC and Physical, make a common API for all waveform applications large and burdensome for resource constrained implementations
  - Building Blocks have been defined to provide as much commonality across waveform APIs as possible
- Goal is a standard API set for each waveform

# SCA APIs

---

- I/O
  - provides a common audio/data interface at a domain component containing voice and/or data processing
- Security (in Security Supplement)
- Network
  - provides a component level interface used for waveform network behavior
- Logical Link Control
  - component level interface used by waveform applications requiring link layer behavior (Data Link Service conforming to the Open Systems Interconnect (OSI) model for networking systems)

# SCA APIs (cont'd)

---

- Medium Access Control (MAC)
  - analogous to and fully supports services of the Medium Access Control sub-layer of the OSI Link Layer model
  - provided for waveform applications that have medium access control behavior (e.g. transmit/receive time slot control in TDMA, error correction coding control, etc.)
- Physical (Real-time & non-Real-time)
  - Real-time provides for translation from bits/symbols to RF and RF to bits/symbols for waveform transmission and reception.
  - non-Real-time provides for initialization and configuration

# Intro to Security Supplement



# SCA Security Supplement

---

- Defines security requirements for JTRS
- Identifies behavioral impacts of these requirements
- Defines an API set to support the requirements
- Uses Common Criteria Evaluation Assurance Levels (EAL) used as a basis for requirements definition
- Written for SECRET System-High (EAL-3) operation but does not prohibit higher levels of assurance

# Security Requirements

---

- Requirements are broken down into the following elements
  - Cryptographic Subsystem (CS/S)
  - INFOSEC Boundary
  - Equipment Level Boundary
  - JTRS Security Policy

# How the SCA Fits Into a Procurement / System Design

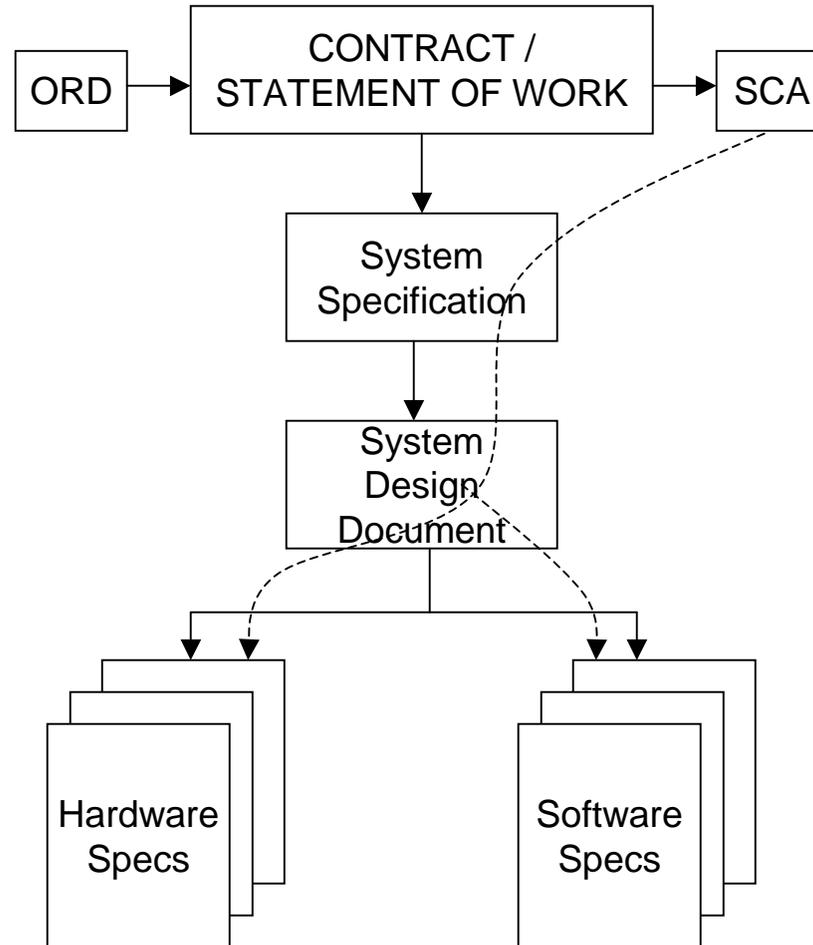


# SCA Procurement Impacts

---

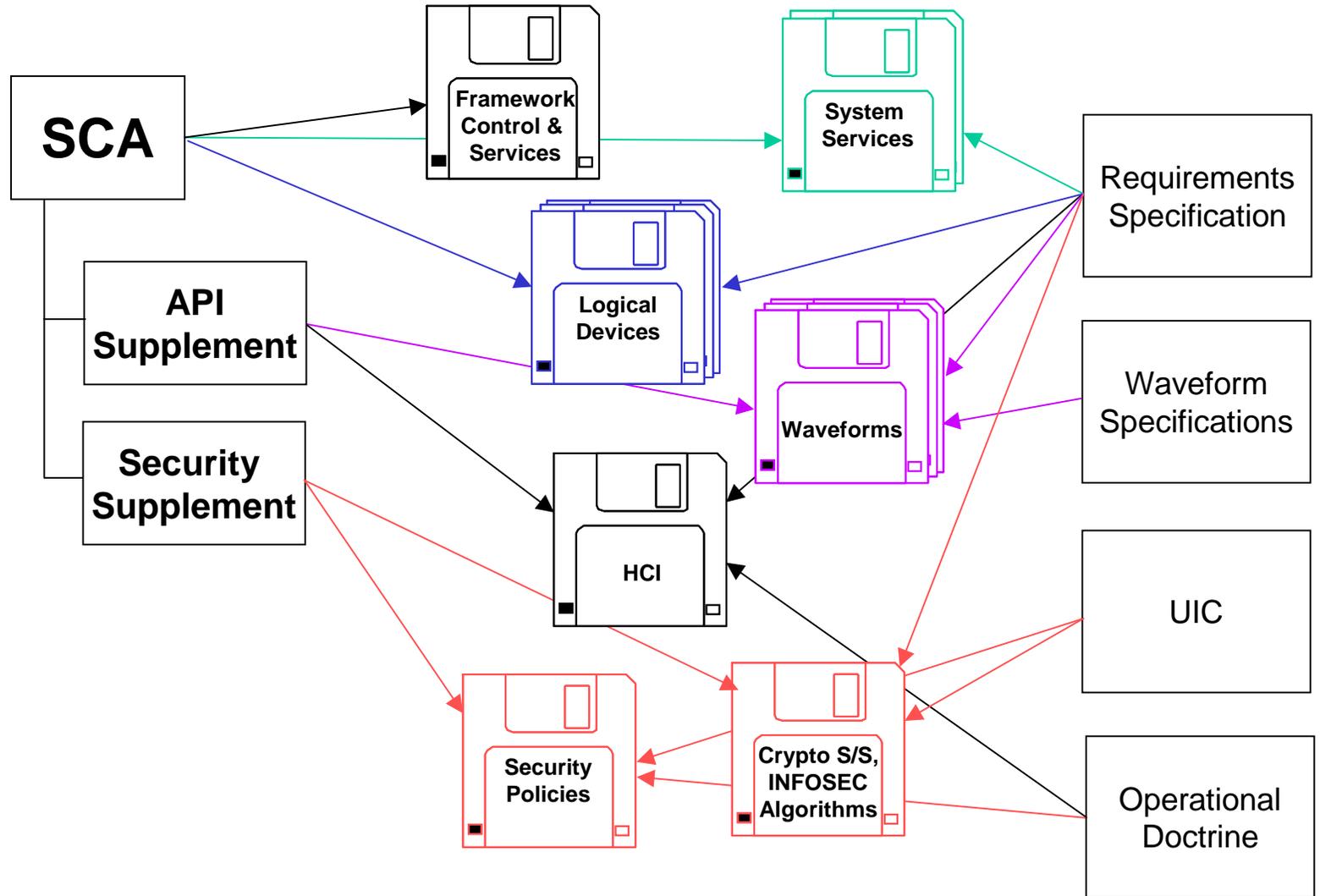
- Contracts for SCA-based products may contain additional implementation guidance / requirements than traditional procurements
  - Separate hardware and software suppliers
  - Standardized hardware guidance
  - Buyer-furnished software applications
  - Adherence to specific APIs
- Compliance to SCA requirements must be verified
  - Compliance certification using approved methods, tools

# SCA and System Design



**SCA Impacts System Design at Several Levels**

# SCA and SW Applications



# System Design Decisions

---

- Operating System
  - acquired commercially by JTRS supplier
- CORBA Middleware
  - acquired commercially by JTRS supplier
- Framework Control and Service Applications
  - developed by JTRS supplier
  - acquired commercially by JTRS supplier
  - provided by JTRS JPO (future)

**JTRS Supplier  
Determines Source of Operating Environment**

- Software Applications
  - developed by JTRS supplier
  - provided by JTRS JPO
  - provided by procuring authority
- Hardware Implementation
  - developed by JTRS supplier
  - identified by procuring authority

**JTRS Procurement Authority  
Determines Source of Software Applications**

# SCA Summary

---

- SCA has impact on traditional radio communications procurements, requiring additional program planning and direction to suppliers
- SCA becomes an important source of requirements throughout the system design process



# SCA Documentation

---

- SCA Specification
  - Currently released - version 2.2
  - API Supplement
    - Currently released - version 1.0
    - Errata was published with SCA v2.1
  - Security Supplement
    - Currently released - version 1.0
    - Errata was published with SCA v2.1



# SCA Documentation

---

- Support and Rationale Document
  - No requirements; provides background rationale for decisions that drove the SCA
  - Currently released - version 2.2
- SCA Developers Guide
  - Currently released - version 1.1
- APIs for I/O and HF-ALE, SINCGARS, Have Quick, Line of Sight waveforms

# Open Discussion