# Software Radio Architecture (SRA) 2.0 Technical Overview


Jerry Bickle
Raytheon
(219) 429-6280
Gerald_L_Bickle@raytheon.com

Kent Bruner
ITT
(219) 451-6072
klbruner@itt.com

# Content

- SRA Specification Background
- Operating Environment Overview
- Core Framework CORBA Interfaces
- XML Profiles & Examples


# What is the SRA Specification?

- The SRA is the commercial adaptation of the Software Communications Architecture (SCA) specification sponsored by the Joint Tactical Radio System (JTRS) program under contract with members of the Modular Software-programmable Radio Consortium (MSRC)
- SRA adaptation is an ongoing process within the Software Defined Radio Forum (SDRF)
- Focus of the SRA:
  - Specifies a common framework to build-up, configure, connect and tear-down distributed, embedded Radio Applications
  - Specifies Software Interfaces to support the installation and use of distributed Applications to support flexible, re-programmable communication capabilities.
- Accessible at:
  - SRA: http://www.sdrforum.org/
  - SCA: http://www.jtrs.sarda.army.mil

# SRA Outline

- SRA Specification
  - 1. Introduction
  - 2. Overview
  - 3. Software Architecture Definition
  - App. A. Glossary
  - App. B. SRA Application Environment Profile
  - App. C. Core Framework IDL
  - App. D. Domain Profile
- SRA Supplements
  - Hardware Architecture Supplement
  - Application Programming Interface (API) Supplement

- A Military Security Supplement is included with the SCA Specification


# Who Has an Interest in the SRA Specification?

- U.S. Government Participants - DOD Joint Services, Mitre, FAA
- Industry Participants
  - MSRC Members
    - Raytheon, BAE Systems, Rockwell Collins, ITT Aerospace/Communications
  - Communication System Developers
    - Boeing, Harris, Motorola, Racal
  - Software Developers
    - Assurance Technology, Exigent, Object Computing Inc.
- Software Defined Radio (SDR) Forum Members
- Embedded CORBA and OS Vendors
- OMG

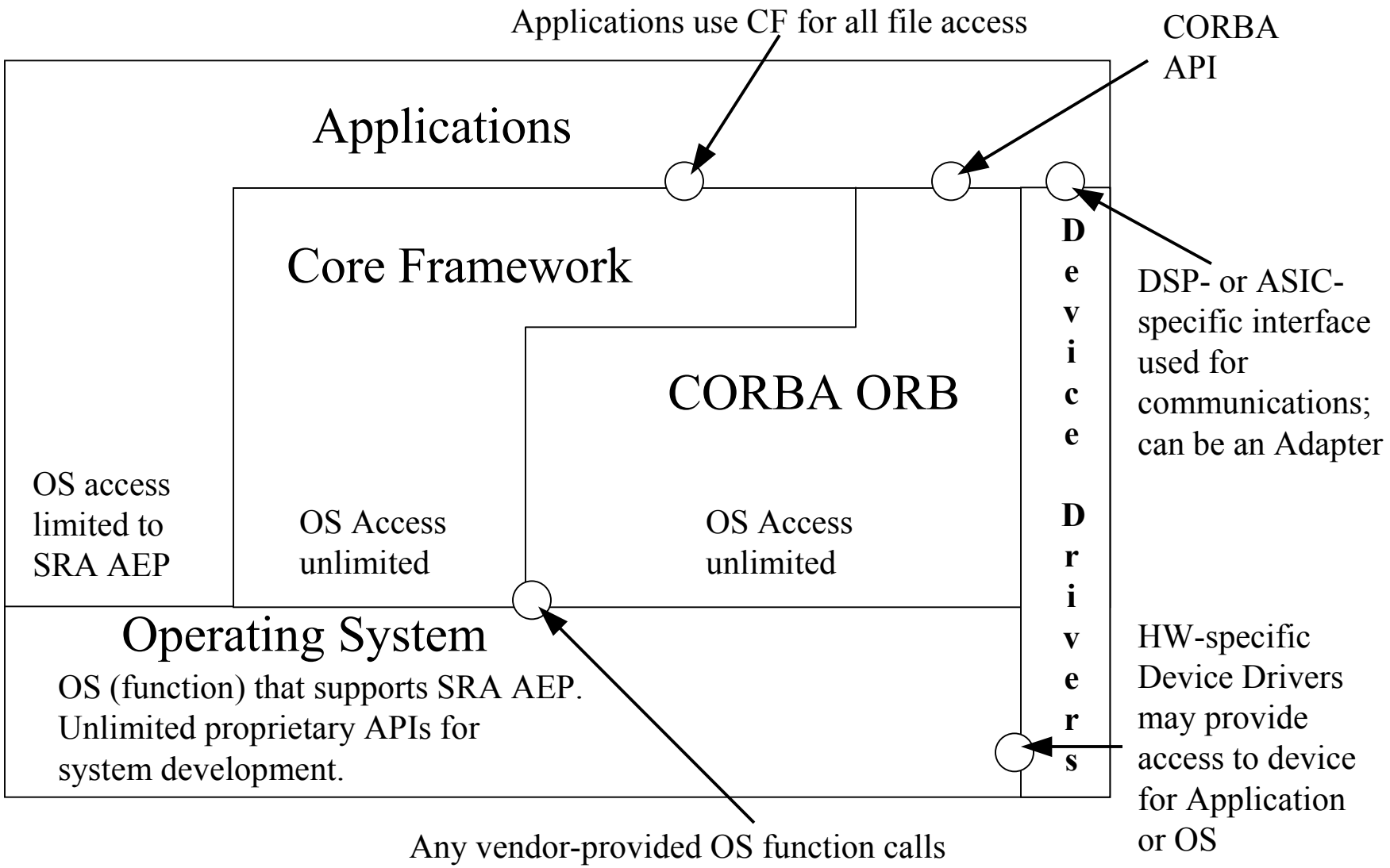
# SCA / SRA
# Development and Validation

- Develop Architecture Definition (Step 1) √
- Specify and Validate Architecture Specification (Step 2)
  - Step 2A - MSRC Develops the SCA and Validates with Prototypes
    - SCA 1.0 released on 17 May 2000 √
    - SCA Rationale Document submitted 30 June 2000 √
    - Security and Networking API Supplements Started √
    - Validation Plan submitted on 7 July 2000 √
    - SCA 2.0 Release in Dec. 2000 with Security and API Supplements √
  - Step 2B - Others Develop Prototypes to Validate the SCA
- Maintain and Evolve the Architecture into SRA (Step 3)
  - Commercial Acceptance
  - Industry Forums (SDR Forum) and Interested Standards Groups
- Procure Systems and Communication Services (Step 3)
  - All DOD Communication Systems Required to be SCA-Compliant

# SRA Operational Environment

- The JTRS Operational Environment (OE) called out in the SRA consists of three major parts.
  - A Real Time Operating System (RTOS)
  - A Real Time Object Request Broker (ORB)
  - The SRA Core Framework (CF)

# SRA Operational Environment Overview




# Real Time Operating System

- RTOS Must Support SRA Application Environment Profile (AEP)
- The SRA 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 SRA AEP

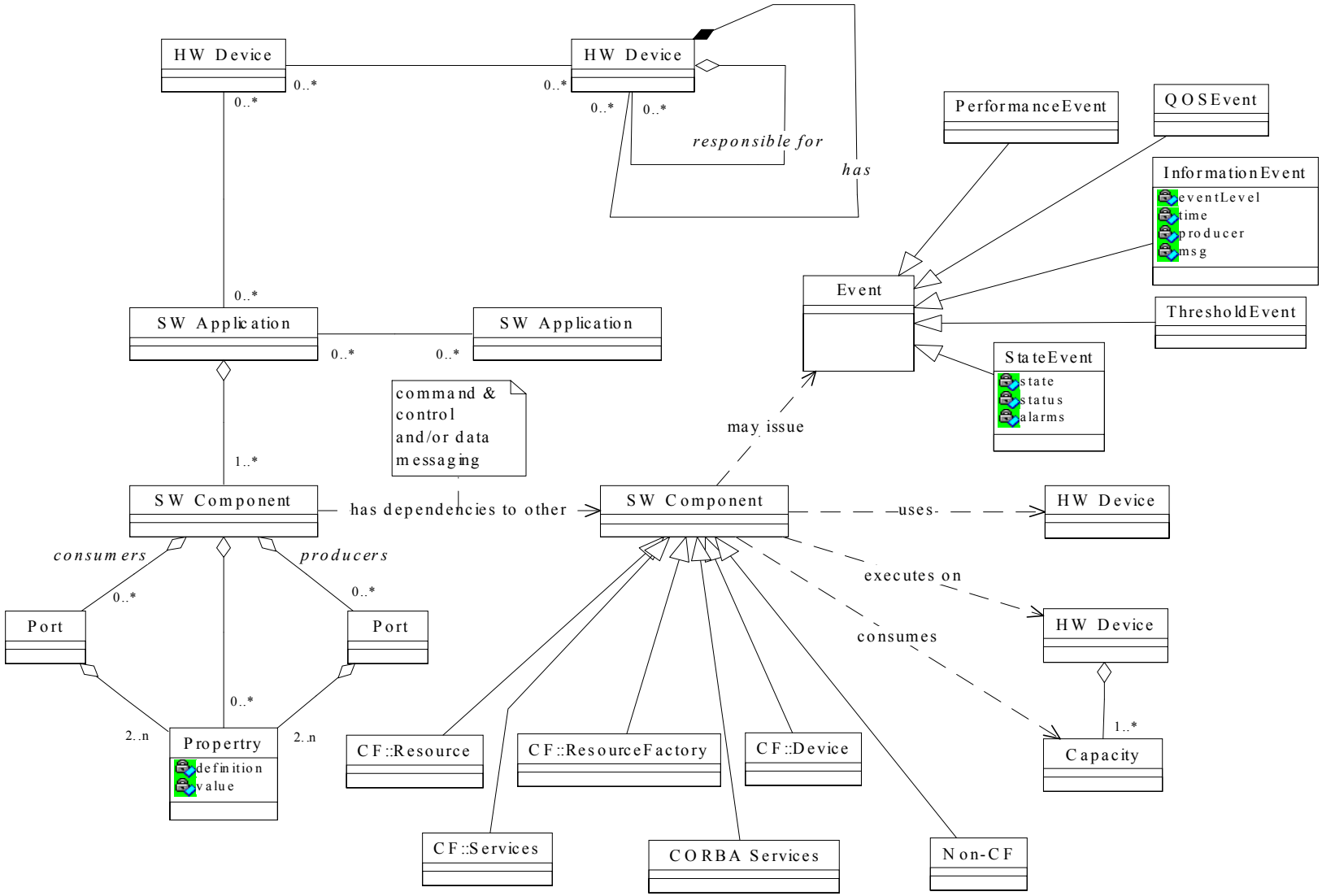
# SRA CORBA Usage

- No extensions and/or services above and beyond Minimum CORBA can be used except as specified in the SRA.
- Desired extensions listed as optional, pending commercial availability
  - Interoperable Naming Service - specified by the OMG Document orbos/98-10-11, October 19, 1998.
    - Naming service works like a white pages directory. Objects register with the Naming Service giving the name and their object reference (or handle). The CF Domain Manager can then go look these names up in the Naming Service in order to obtain a object reference to them.
    - Software components can use a stringified Interoperable Object References (IORs) in their Application Profile (if no Naming Service).
  - Real-Time CORBA extension as specified by the OMG Document orbos/98-12-05, December 21, 1998.
  - CORBA Messaging as specified by the OMG Document orbos/98-05-05, May 18, 1998
    - Provides the policy framework used by the Real-Time CORBA extension.

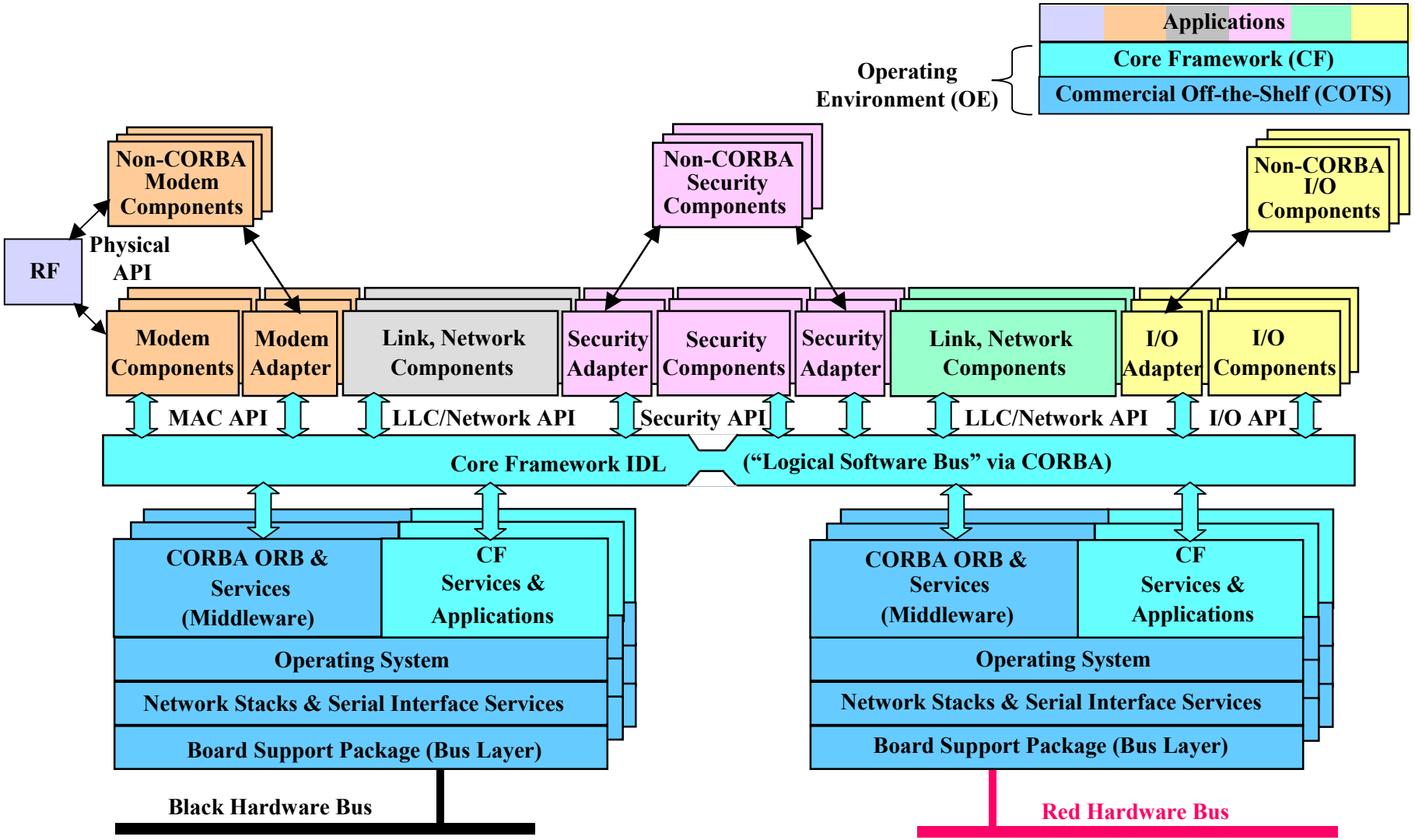

# SRA Domain Profile

- A Domain Profile is a set of XML files that describe the hardware 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 Component Specification
    - To better address Software Radio Needs.
    - Changes to be presented to OMG
  - Describes specific characteristics of software or device components
  - Description of interfaces, functional capabilities, logical location, inter-dependencies, and other pertinent parameters.

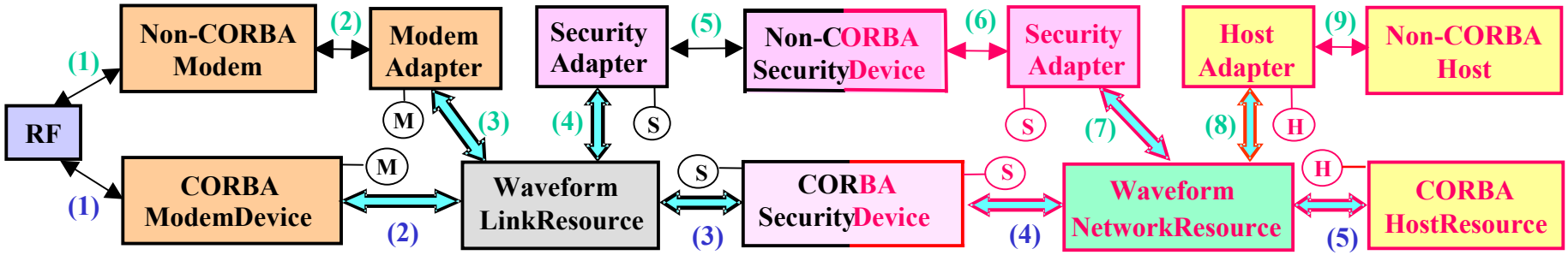
# SRA Problem Domain



# SRA 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)**

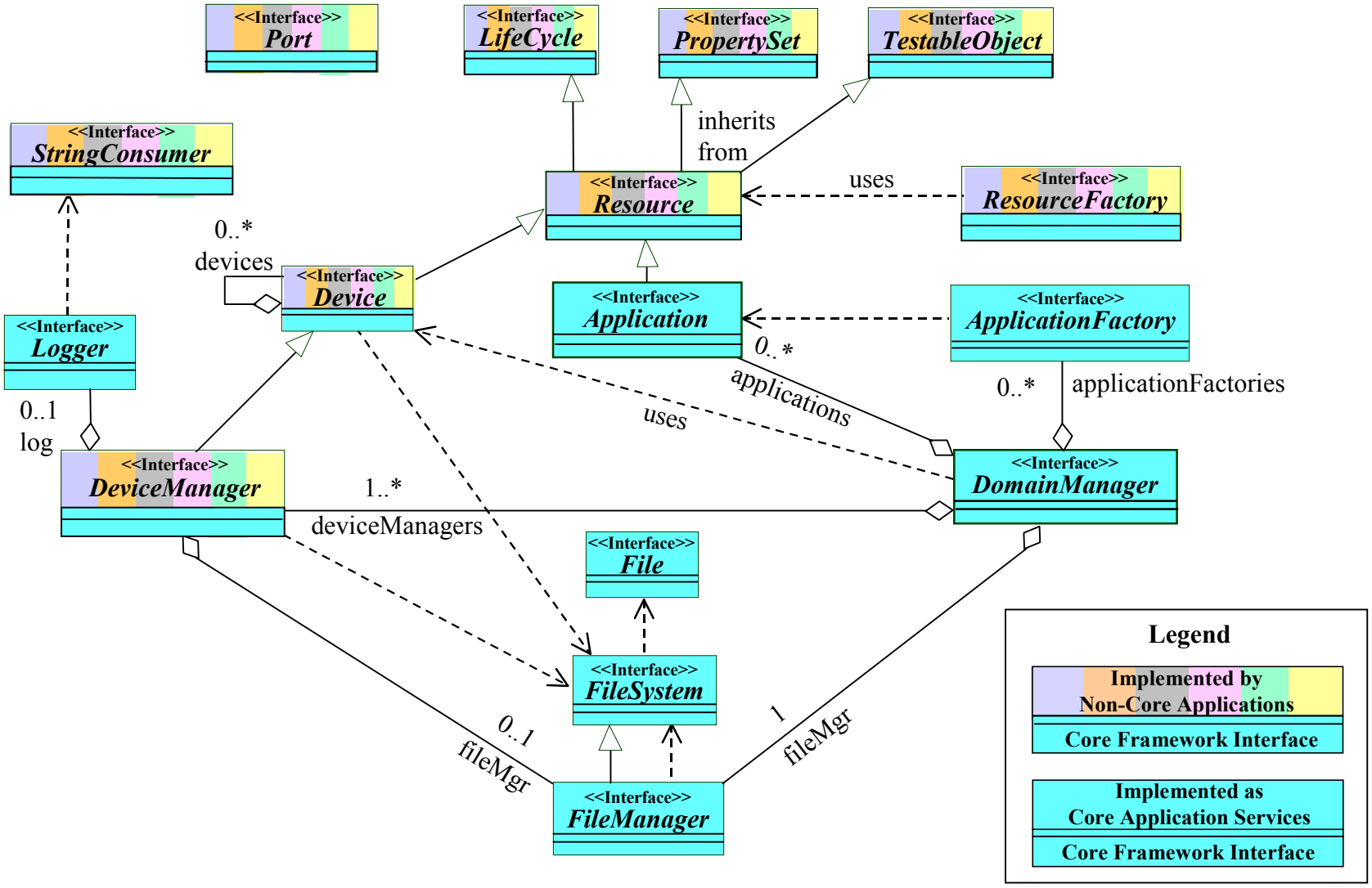
(1) RF Interface to Modem
(2) CORBA Interface to Waveform Link (M)
(3) CORBA Interface to Security (S)
(4) CORBA Interface to Waveform Network (S)
(5) CORBA Interface to Host (H)

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.

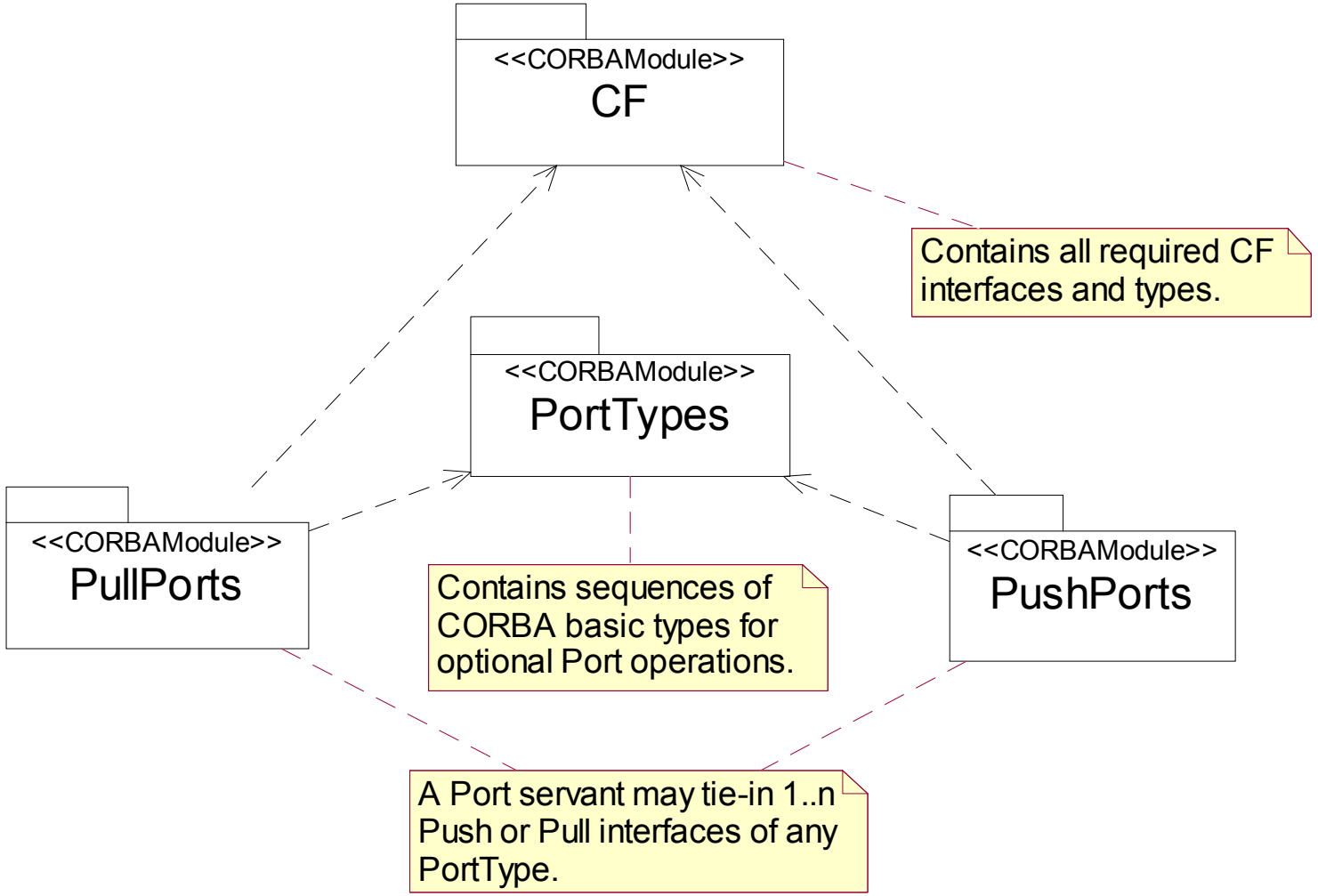
# SRA Core Framework Definition

- 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 Component Interfaces** (***Port, LifeCycle, TestableObject, PropertySet, ResourceFactory,*** and ***Resource***) that provide a common set of interfaces for exchanging information between software application components.
  - **Framework Control Interfaces** (***Application, ApplicationFactory, DomainManager, Device,*** and ***DeviceManager)*** that provide interfaces for the start-up, control, and tear-down of software application components, and the allocation and control of hardware assets.
  - **Framework Service Interfaces** (***File, FileSystem, FileManager, Logger, StringConsumer)*** that provide interfaces for distributed file access services and event logging services to software application components.
- 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 a **Domain Profile**.

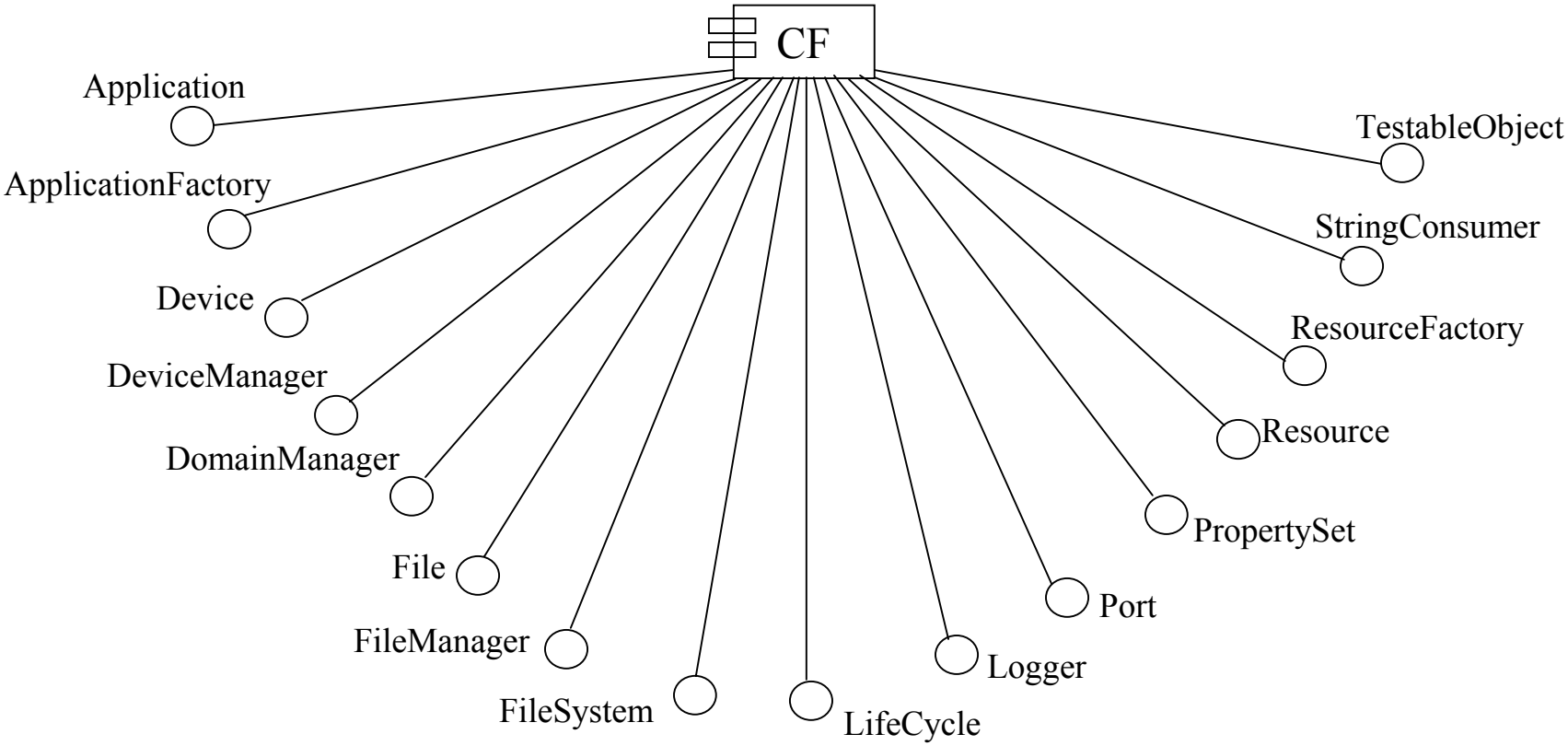
# Core Framework IDL Relationships



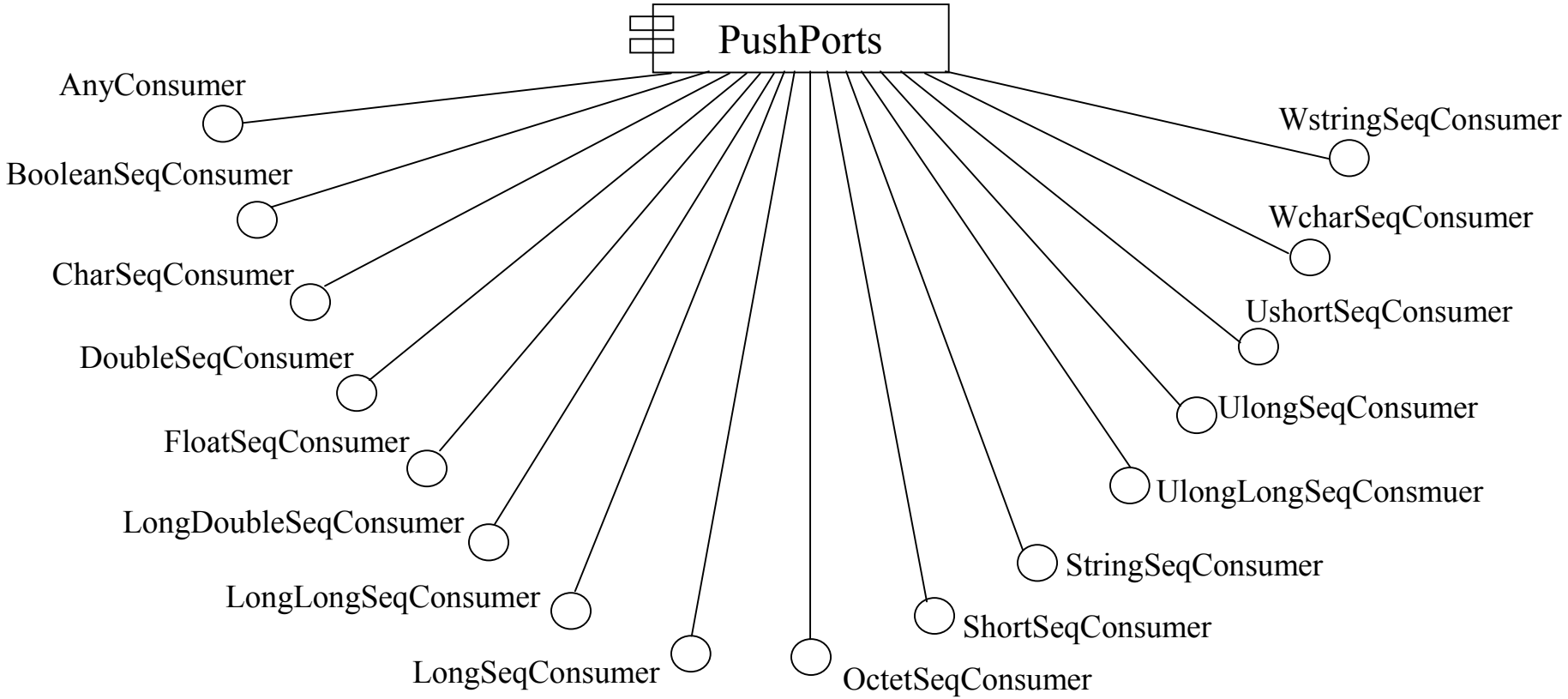
# Core Framework
# IDL Module Relationships



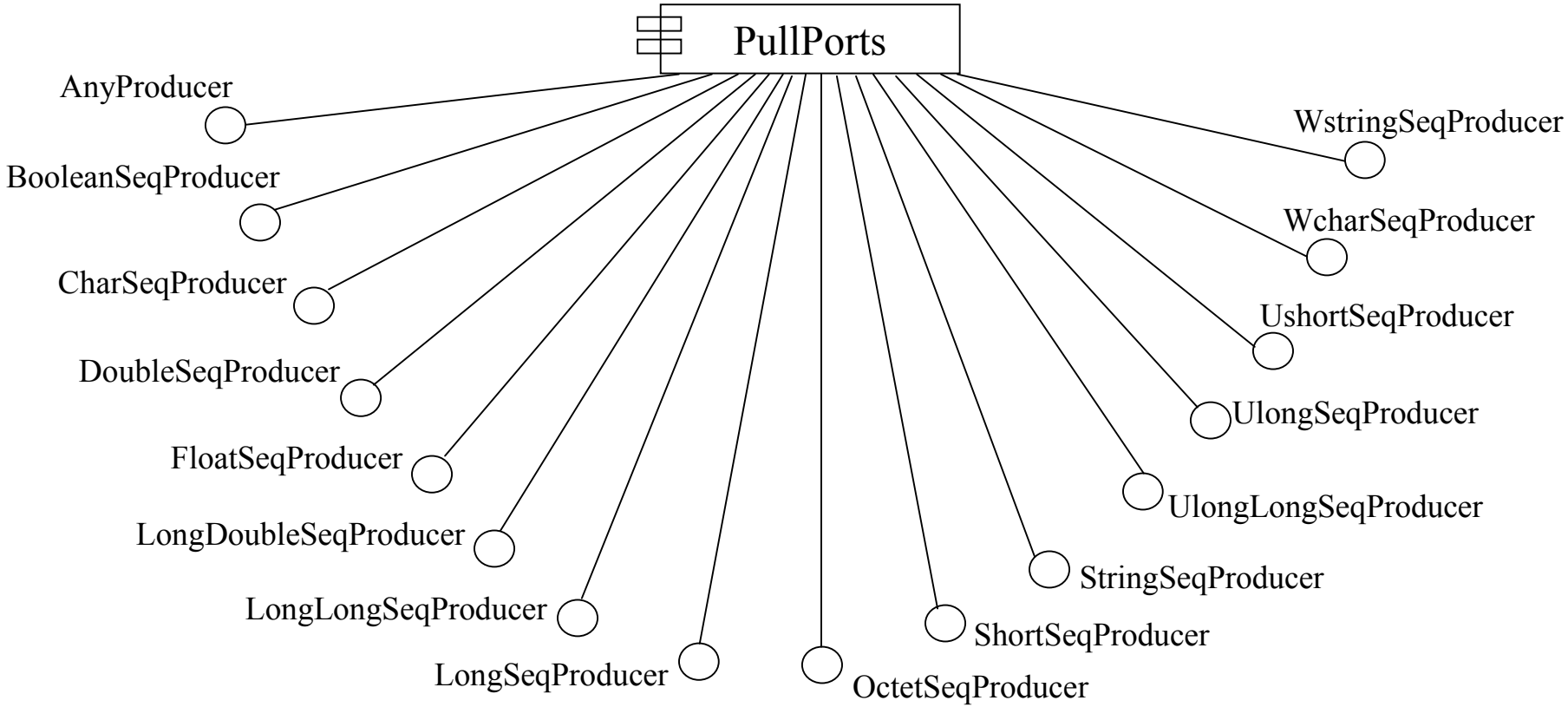
# CF Module Interfaces



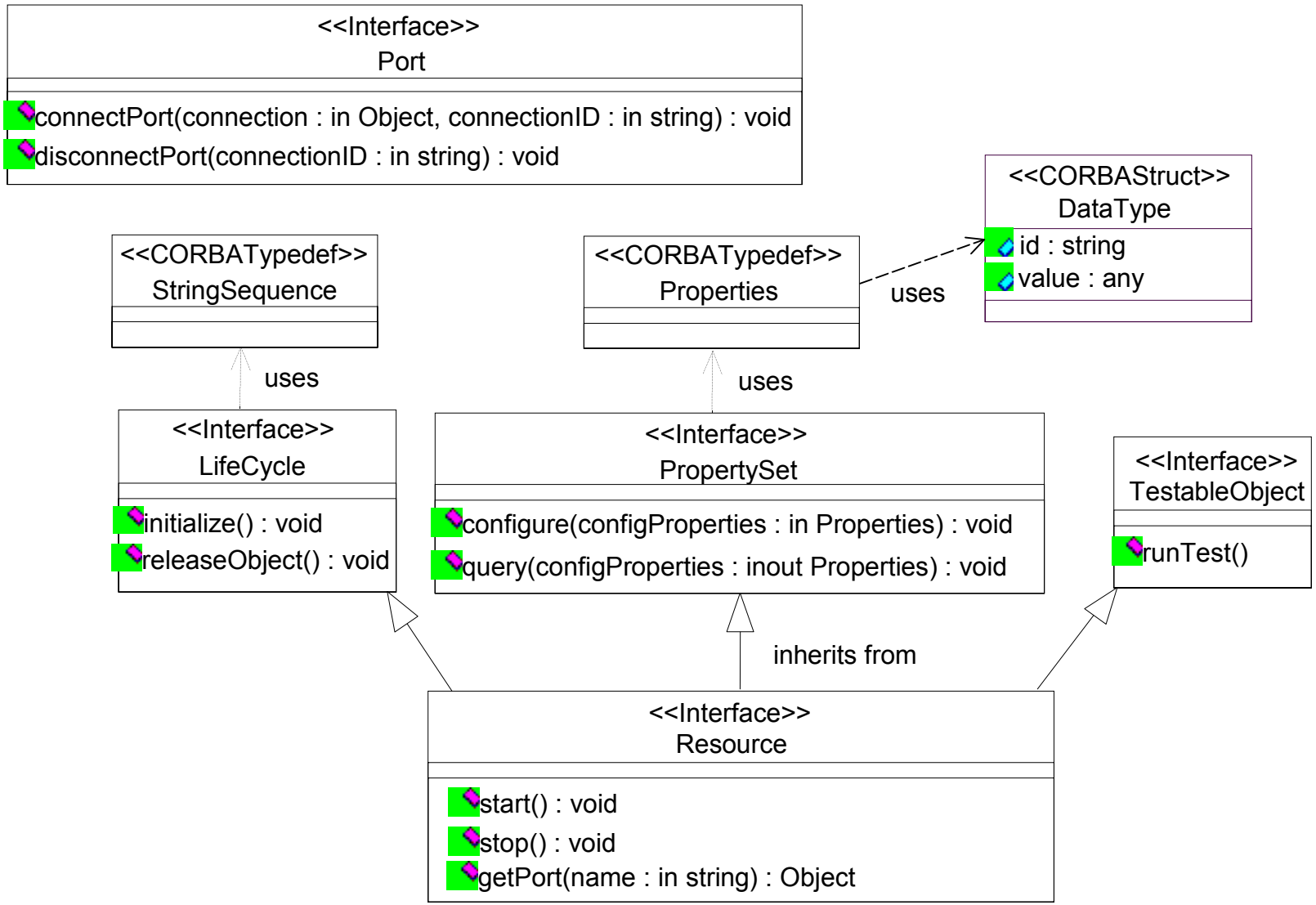
# PushPorts
# Module Interfaces



# PullPorts
# Module Interfaces



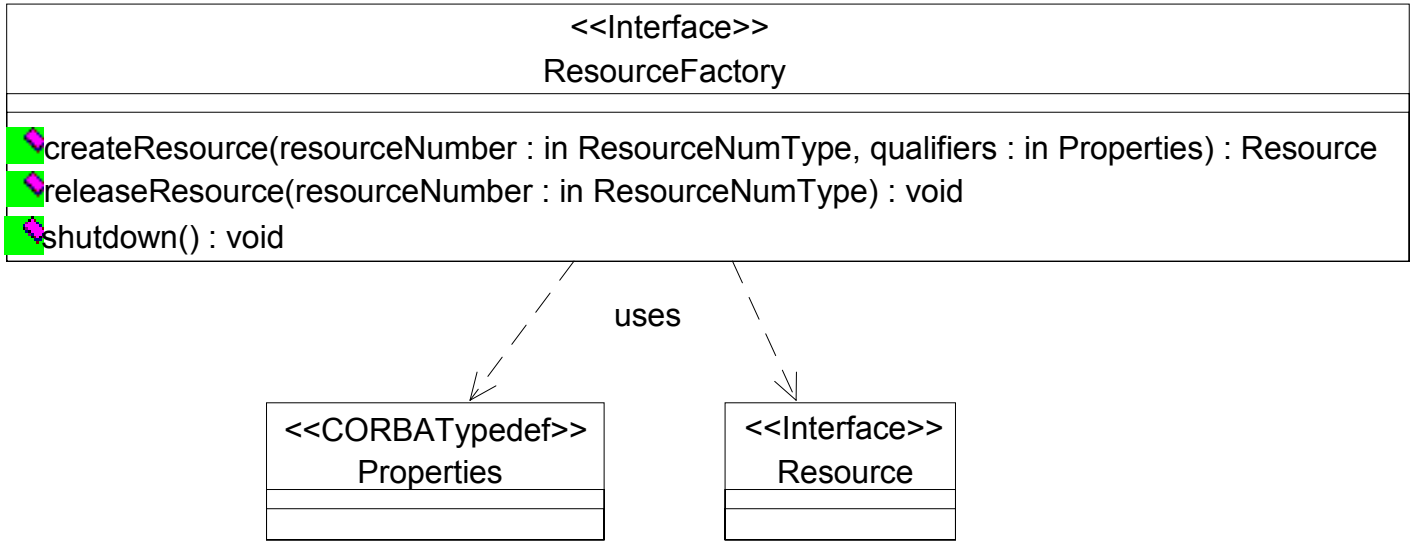
# Base Component Interfaces




# Resource Interface

- Inherits from the following other base application interfaces:
  - Lifecyle - used to initialize or release the Resource
  - TestableObject - used to test Resource (ie. BIT)
  - PropertySet - provides operations to access Resource properties.
- Uses the Port interface to Connect Components
  Via the connectPort/disconnetPort interfaces
- Resource provides interfaces to:
  start/stop processing
  get references to its Ports

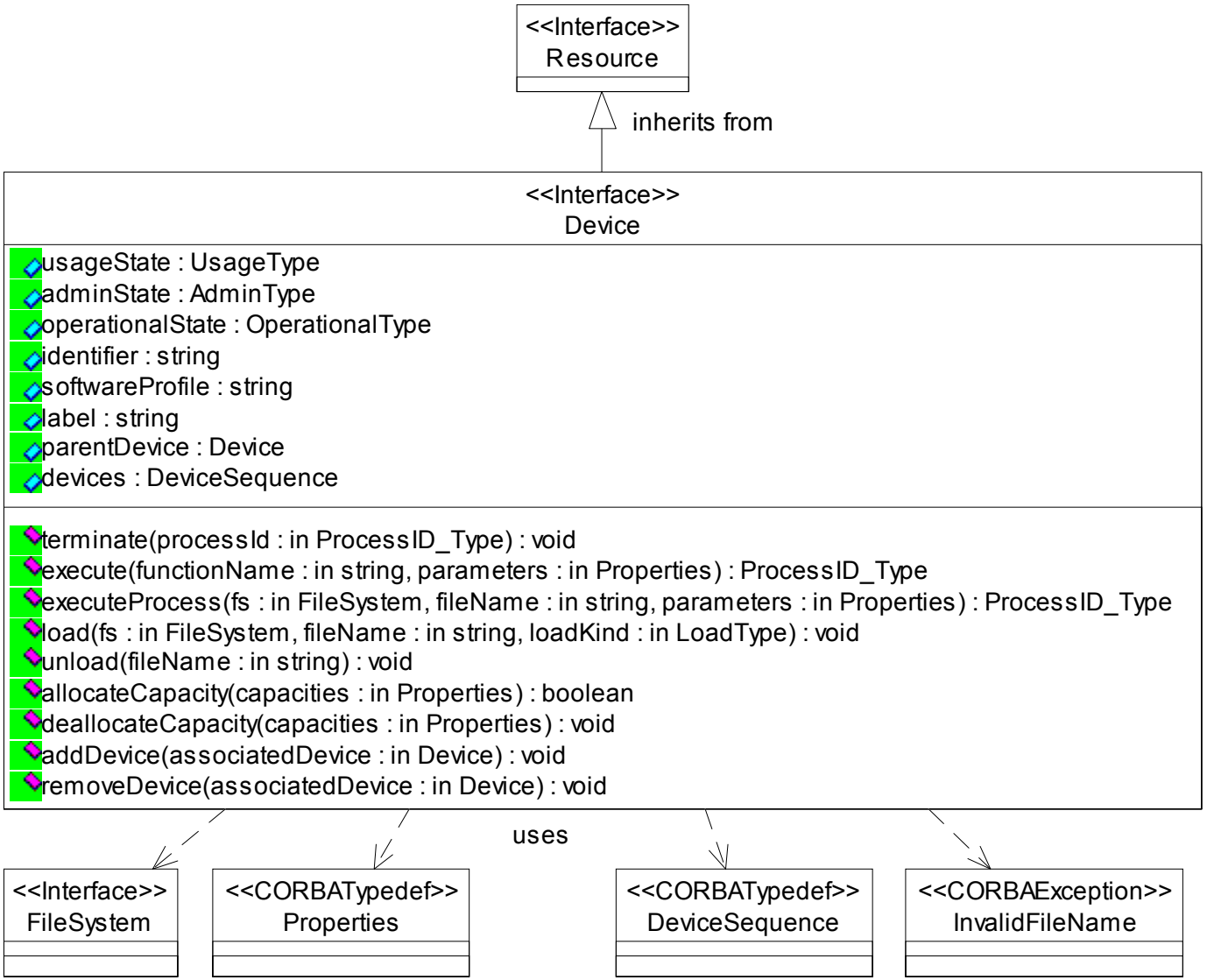
# ResourceFactory Interface




# ResourceFactory Interface

- Used to create/tear down a Resource.
- Modeled after the Factory Design Pattern.
- Provides industry standard mechanism of obtaining a Resource without knowing its identity.
- Optional Interface

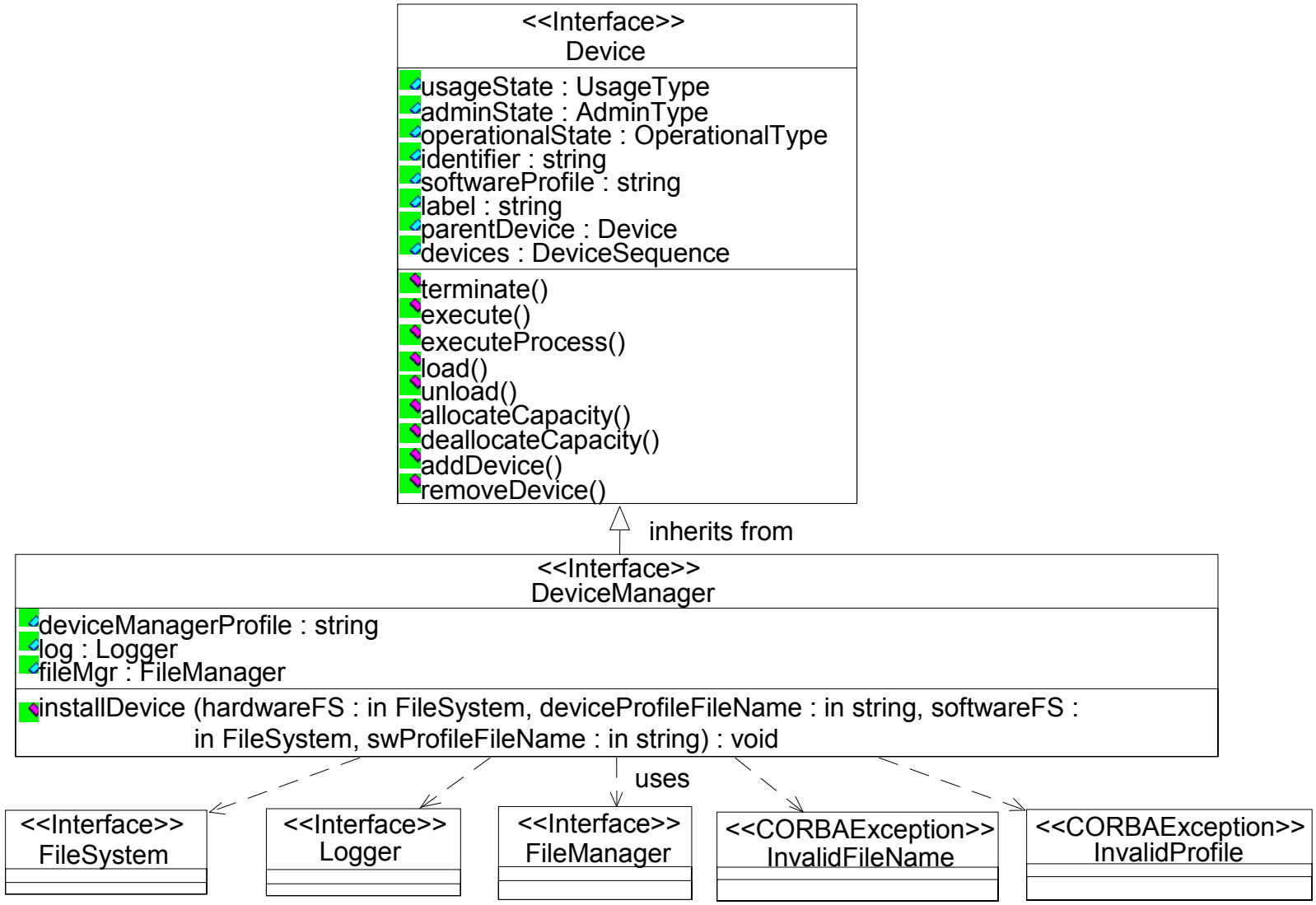
# Device Interface




# Device Interface

- Defines a “logical device” in the system
  - an abstraction of a H/W device
  - can be more than one logical device per H/W device
  - for example: A physical MODEM device could represent the following logical devices simultaneously:
    - TDMA MODEM, CDMA MODEM, FM MODEM, etc.
- Each use logger to display any start up information
- Provides state management interfaces based on X.731
- Defines the capacity model for the device.
- Responsible for the loading/unloading of software components.
- Provides an execute operations to start software components.

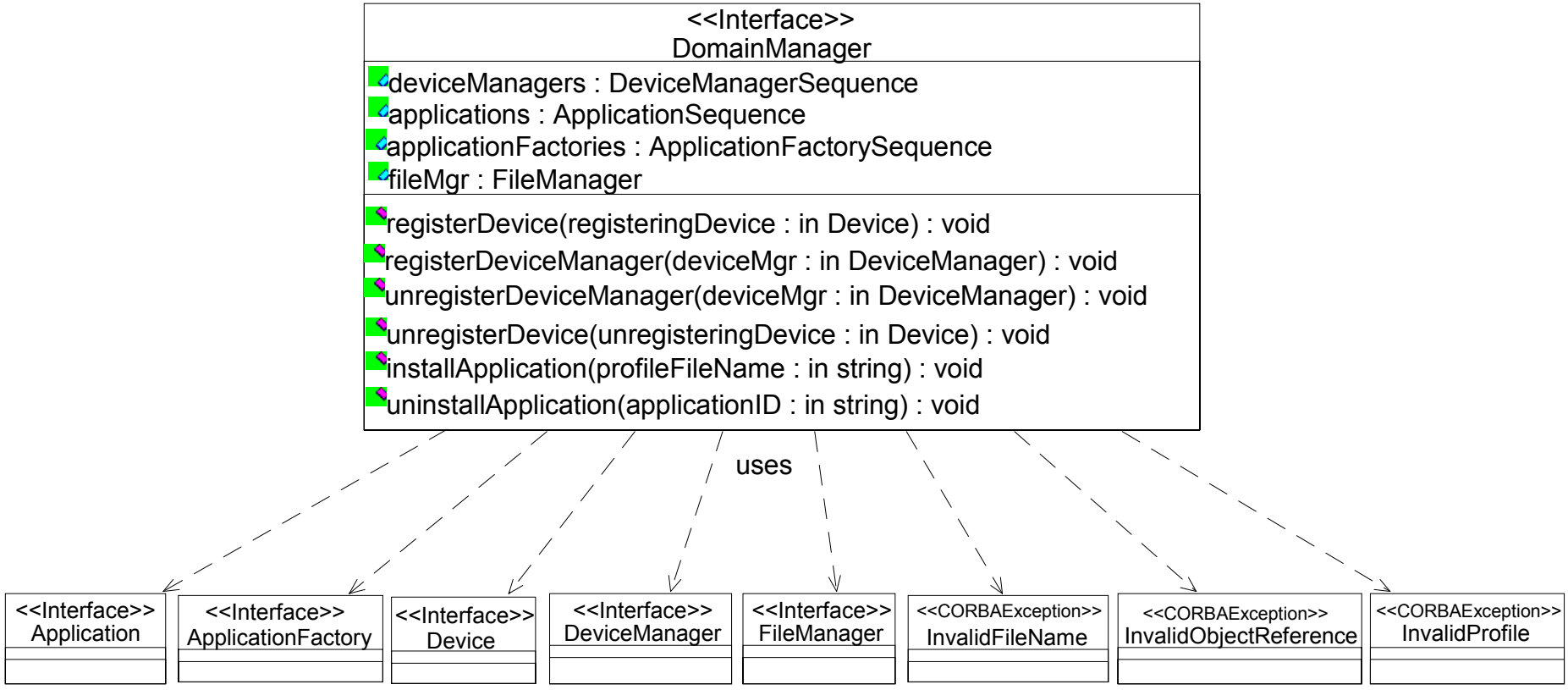
# DeviceManager Interface



# DeviceManager Interface

- A “container” for Devices
- Typically represents a CORBA capable “board” in a system
- Creates FileSystem and FileManager objects
- Registers itself with the DomainManager
- Provides interfaces for installing new devices
- Provides the Devices it is “managing” when queried
- Uses its profile for determining
  - How to become known to DomainManager
    - Naming Service
    - Stringified Interoperable Object Reference(IOR)
  - Creation of Logger component
  - Location of Device components.

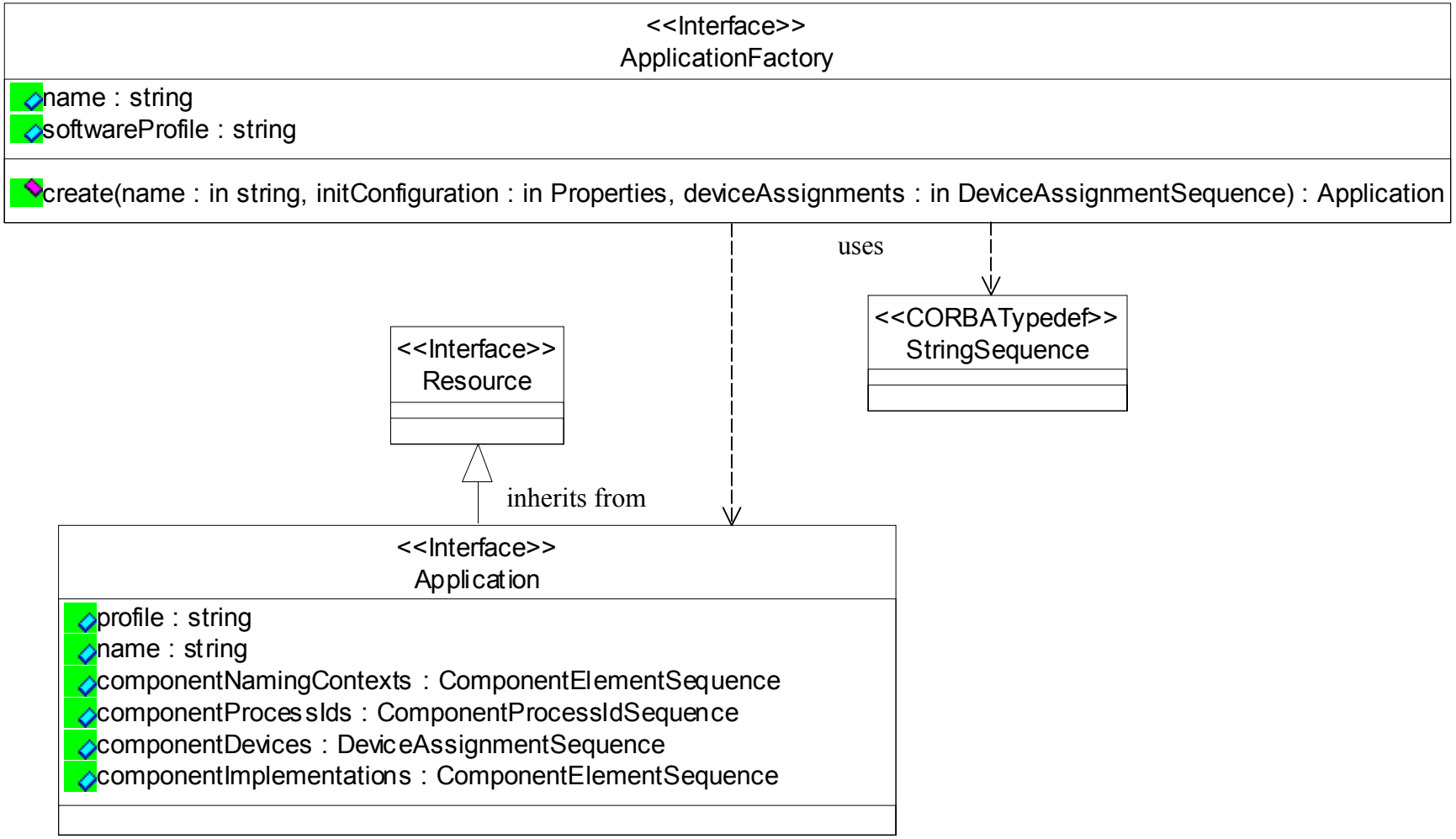
# DomainManager Interface



# DomainManager Interface

- Domain Manager provides interfaces for:
  - MMI's to:
    - configure the domain
    - get the domain's capabilities (Devices and Applications)
    - initiate maintenance functions
  - Registration (register/un-register) of:
    - DeviceManager(s), Device(s), Application(s)
  - Provides access to:
    - Registered DeviceManager(s)
    - Installed and Running Applications
    - The Radio's FileManager

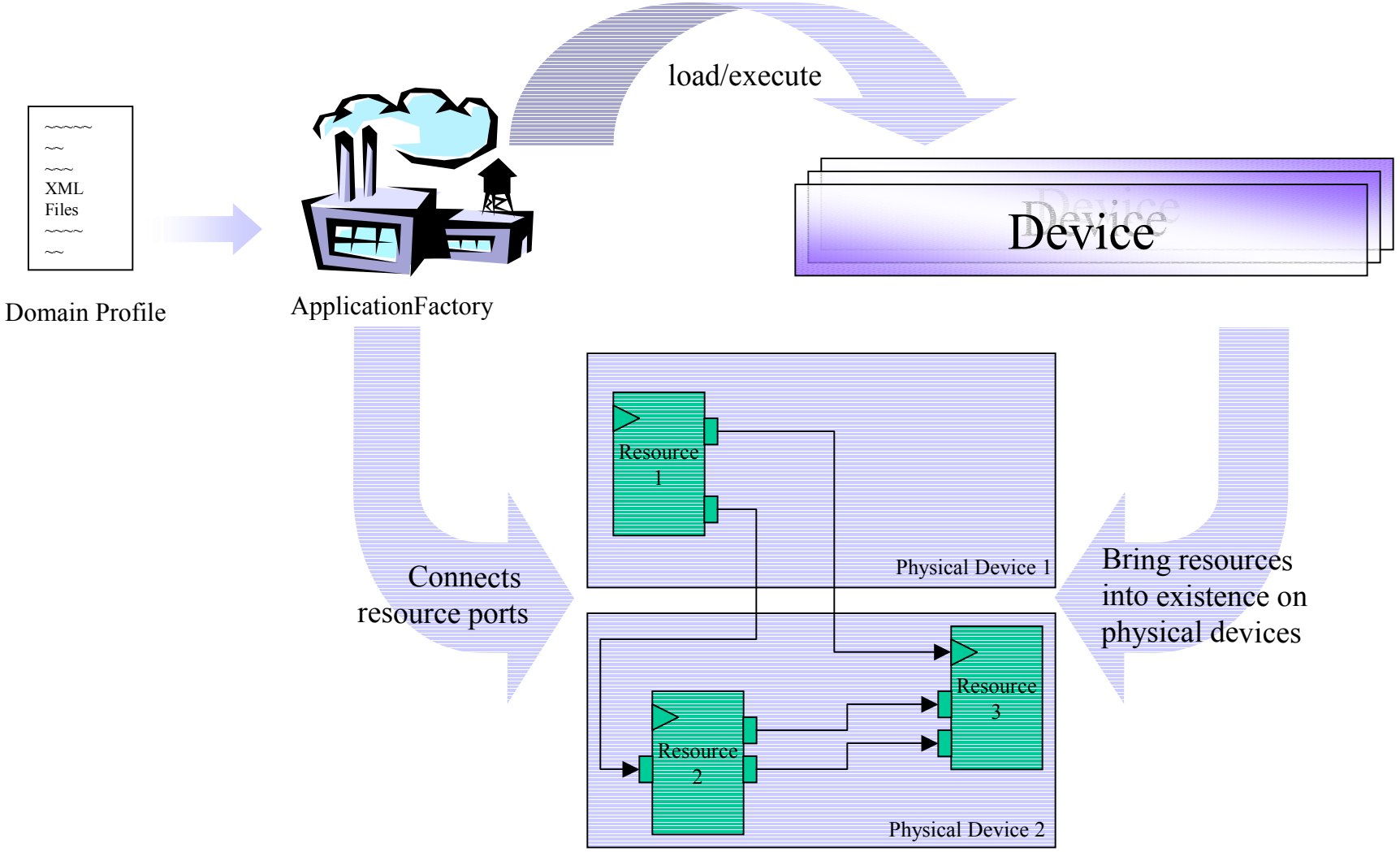
# ApplicationFactory & Application Interfaces



# ApplicationFactory & Application Interfaces

- Part of Domain Management
- Application Factory
  - used to create Application (waveform) instances
  - Based on Domain Profile:
    - allocates software (Resources) to hardware (Devices)
    - connects Resources that make up an Application
    - performs initial configuration
- Application
  - container for Resources that make up Application
  - provides the interface for instantiated Applications:
    - control, configuration, status, tear-down

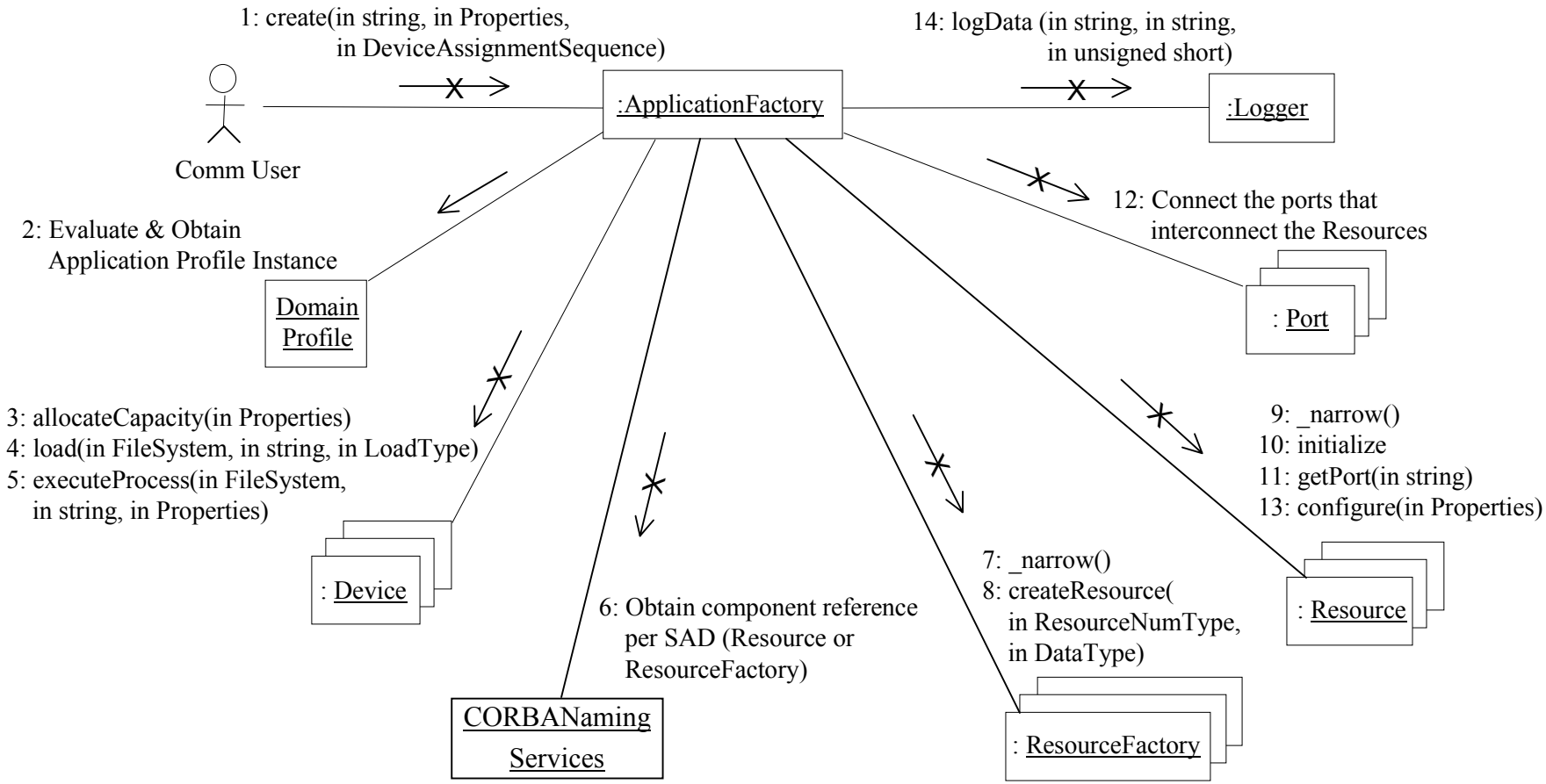
# Application Deployment



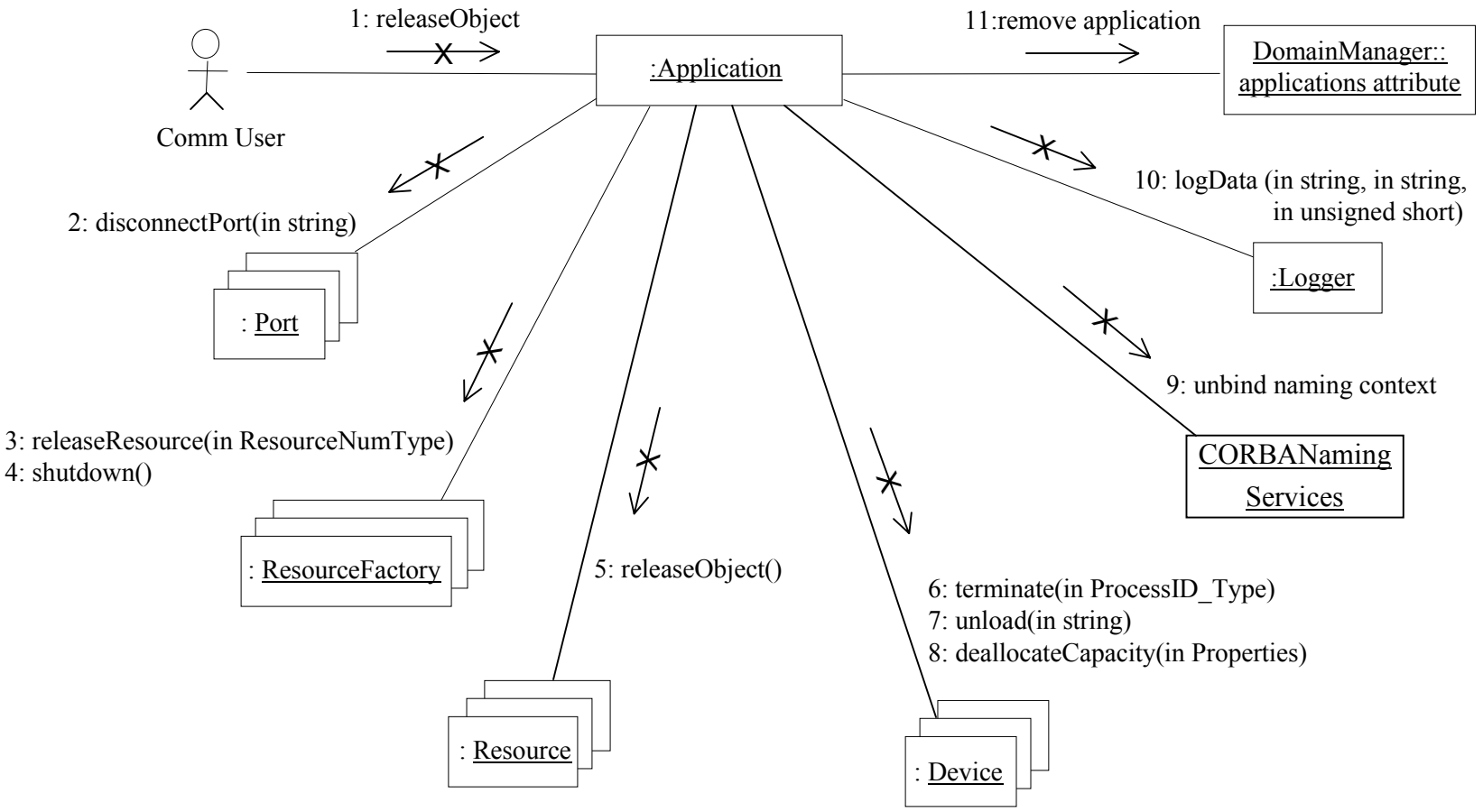
# Application Instantiation

- UI asks for all ApplicationFactory(s)
  - Application to instantiate is chosen
  - UI issues create() on ApplicationFactory
- ApplicationFactory determines applicable Device(s) on which to load application code defined in Domain Profile
  - Load/Execute are called on Device(s)
    - brings Resource(s) into existence
- Resource(s) bring Port(s) into existence
- ApplicationFactory connects the Port(s) to form Application
- Resources are then configured, initialized, and started

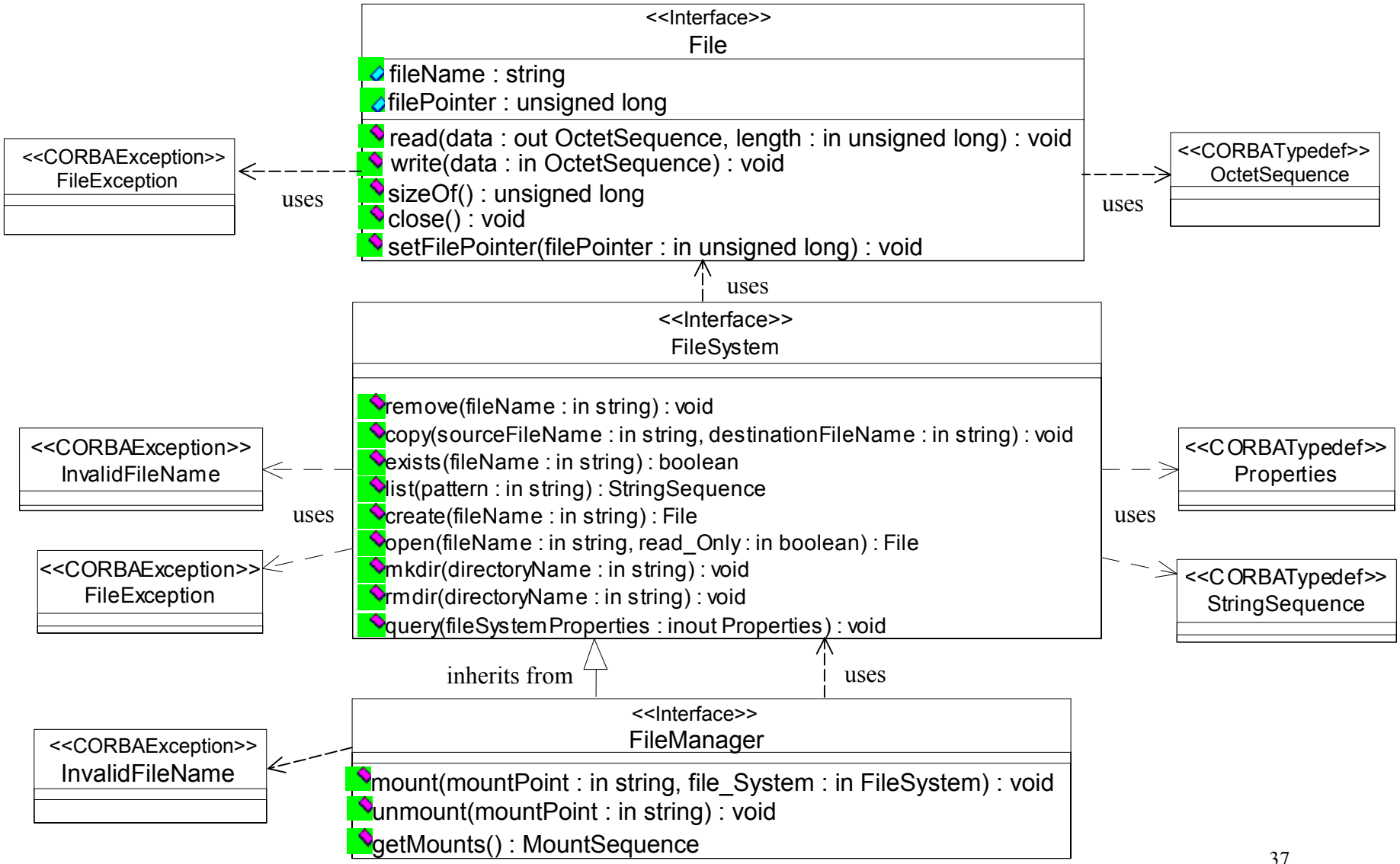
# Application Instantiation Example



# Application Tear-Down Example



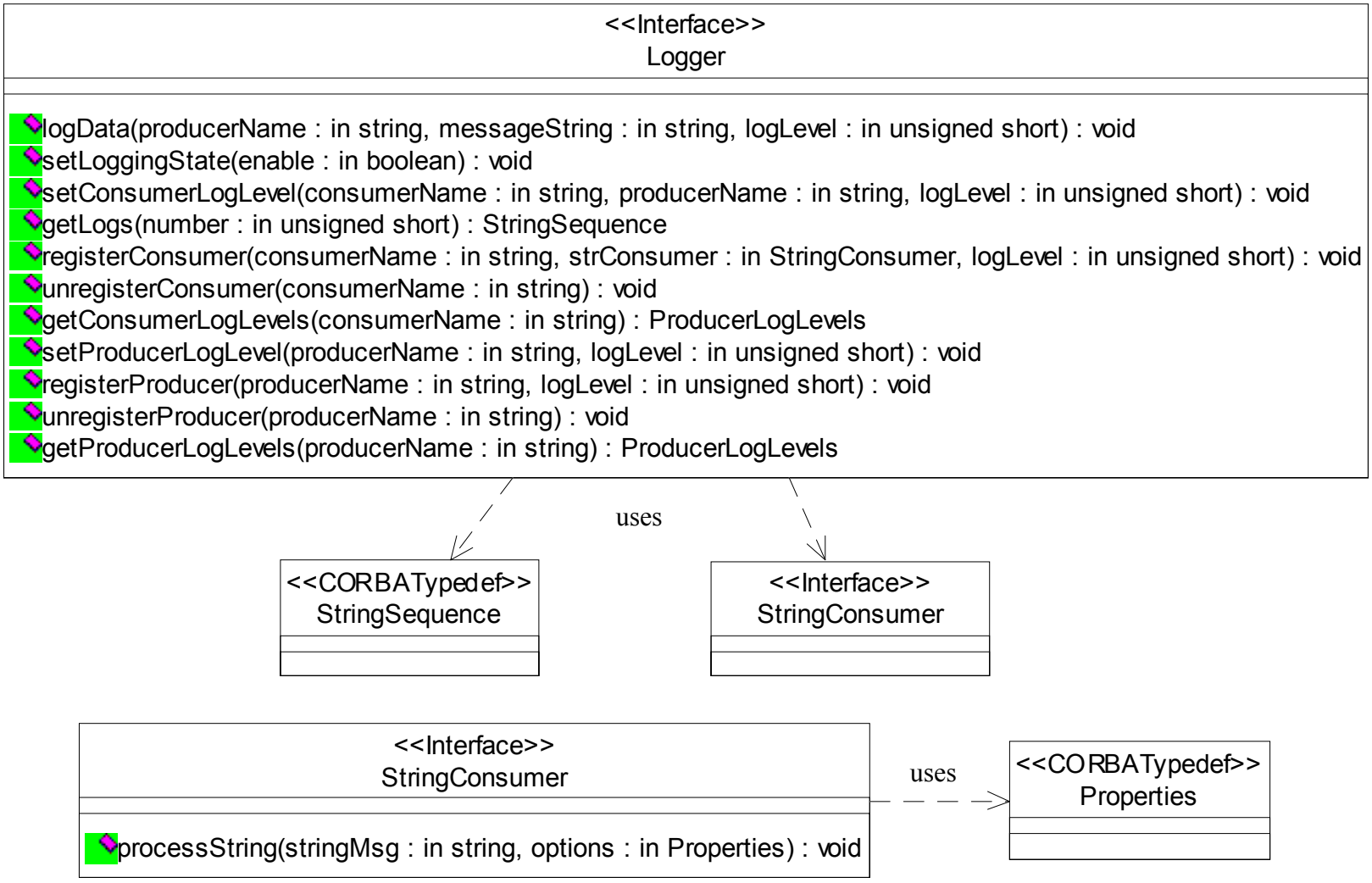
# File, FileSystem & FileManager Interfaces




# File, FileSystem & FileManager Interfaces

- FileManager, FileSystem, and File provide
  - Application Files installation or removal.
  - Loading and unloading application files on the various processors that *Devices* execute on.
- Applications 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)

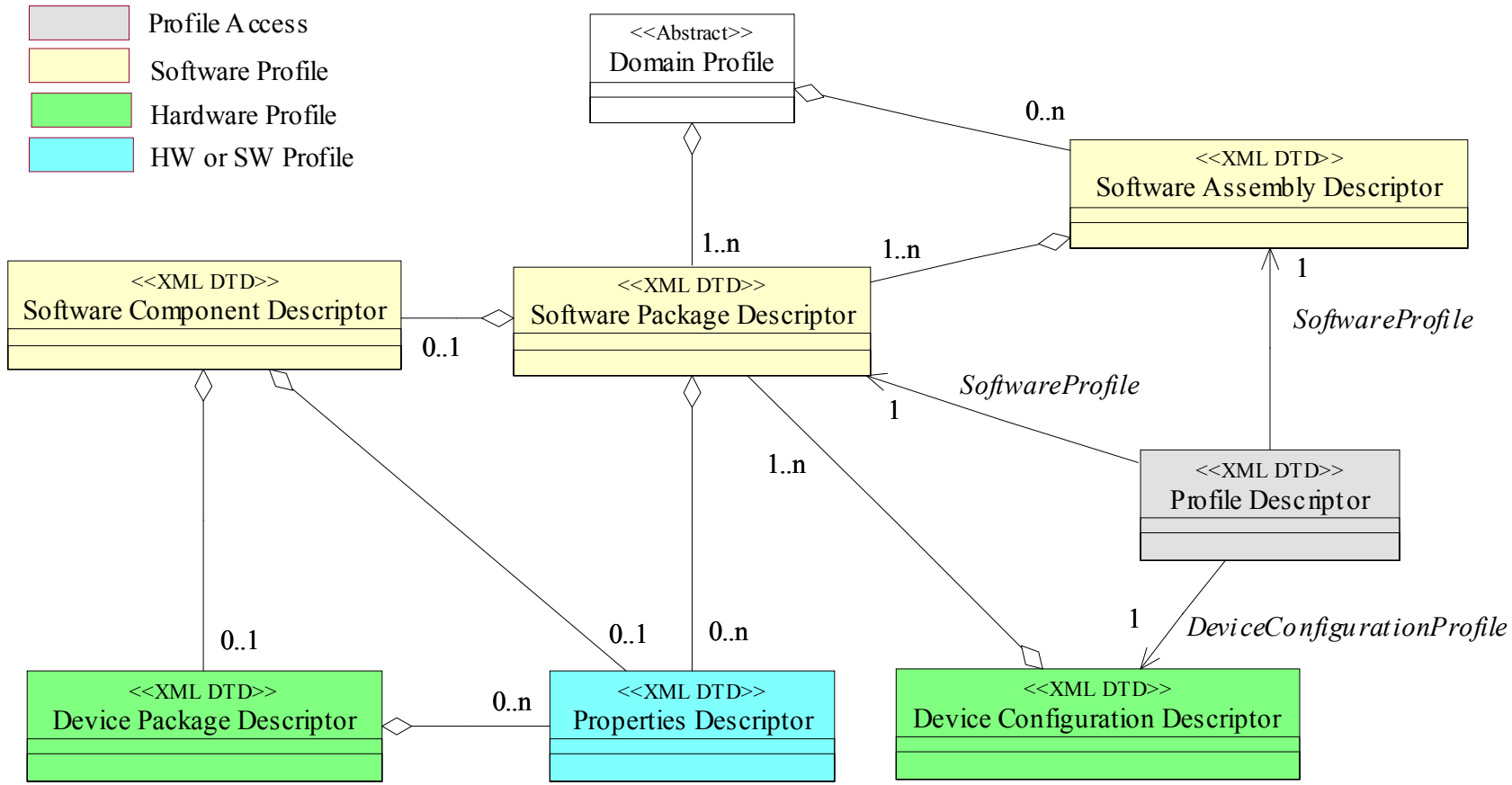
# Logger & StringConsumer Interfaces



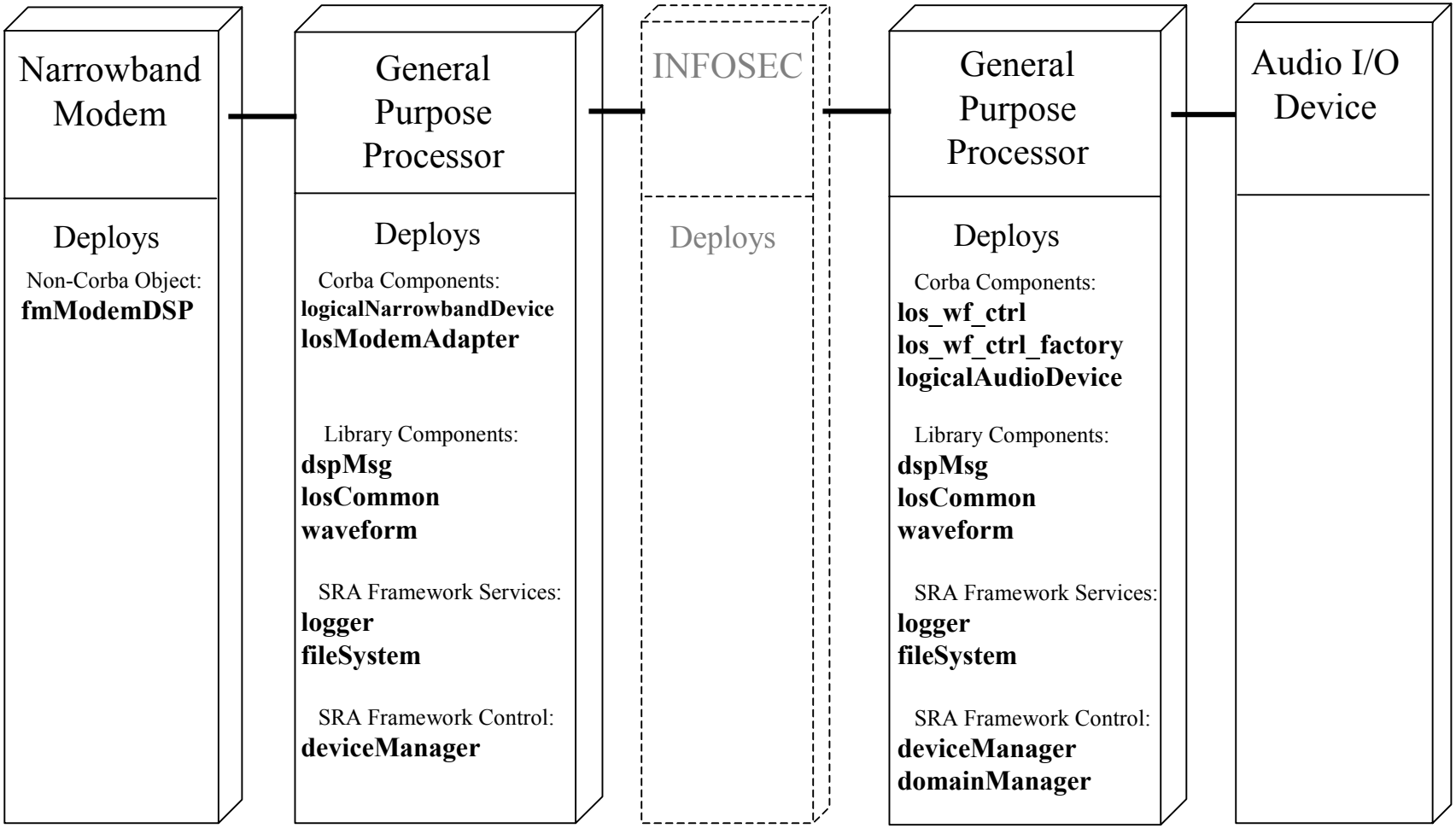
# XML-Based Domain Profile DTD

- Used to describe Devices and to deploy Application and Application components into an SRA compliant system.
- SRA specified DTD based on OMG CORBA Component DTD
- 7 XML DTD's:
  - **Profile Descriptor (accessed via a CF Interface "profile" attribute)**
    - Provides a filename reference to a SAD, SPD, or DCD instance.
  - **Software Assembly Descriptor (SAD)**
    - Describes the deployment characteristics and connectivity of components.
  - **Software Package Descriptor (SPD)**
    - Describes implementation(s) of a specific component.
  - **Software Component Descriptor (SCD)**
    - Describes a CORBA-capable software component and its interfaces
  - **Device Package Descriptor (DPD)**
    - Identifies a class of a device
  - **Device Configuration Descriptor (DCD)**
    - Identifies components to initially start on a device and how to find the DomainManager
  - **Properties Descriptor File (PRF)**
    - Describes properties applicable to a software package or a device package.

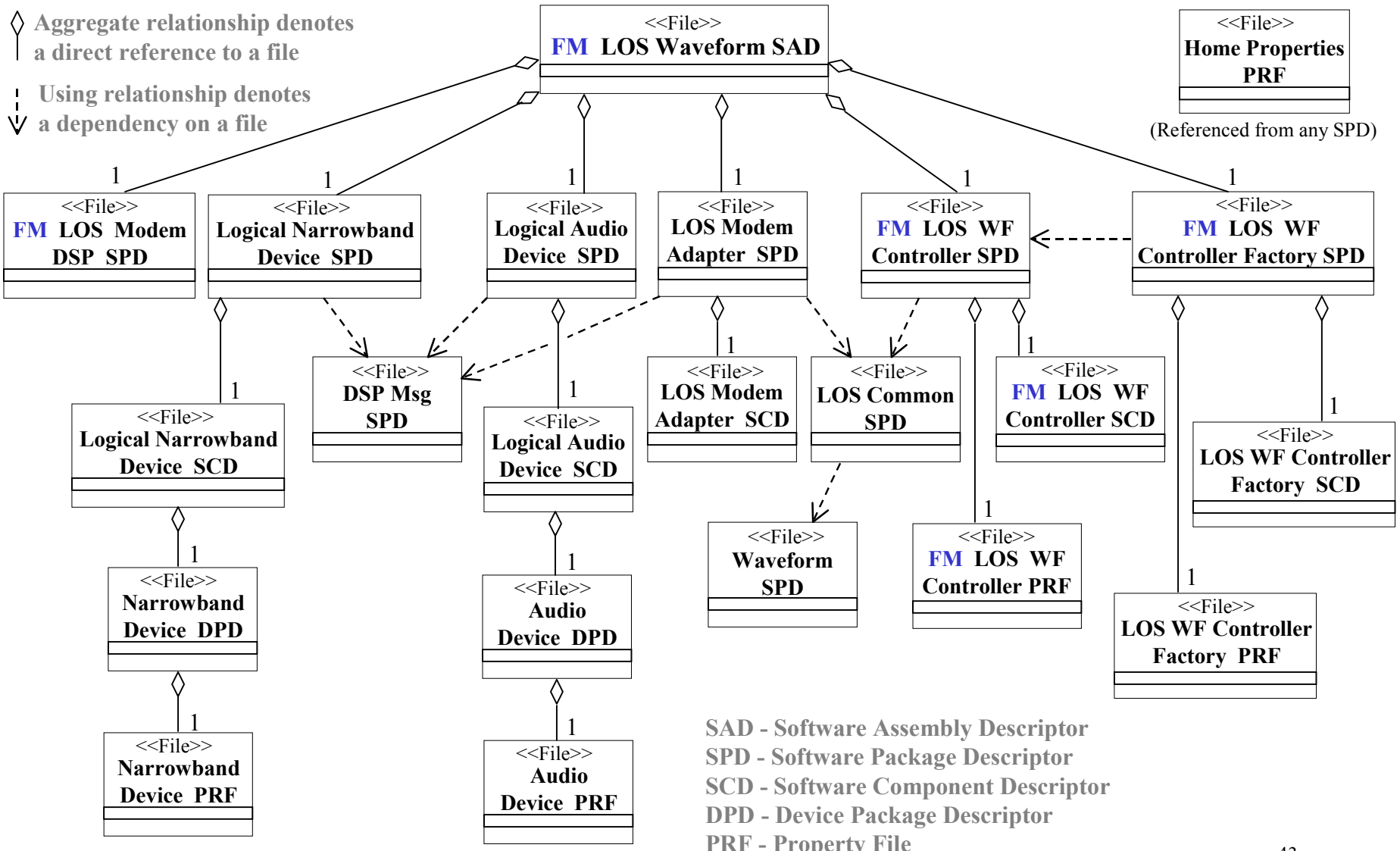
# Domain Profile XML DTD Relationships



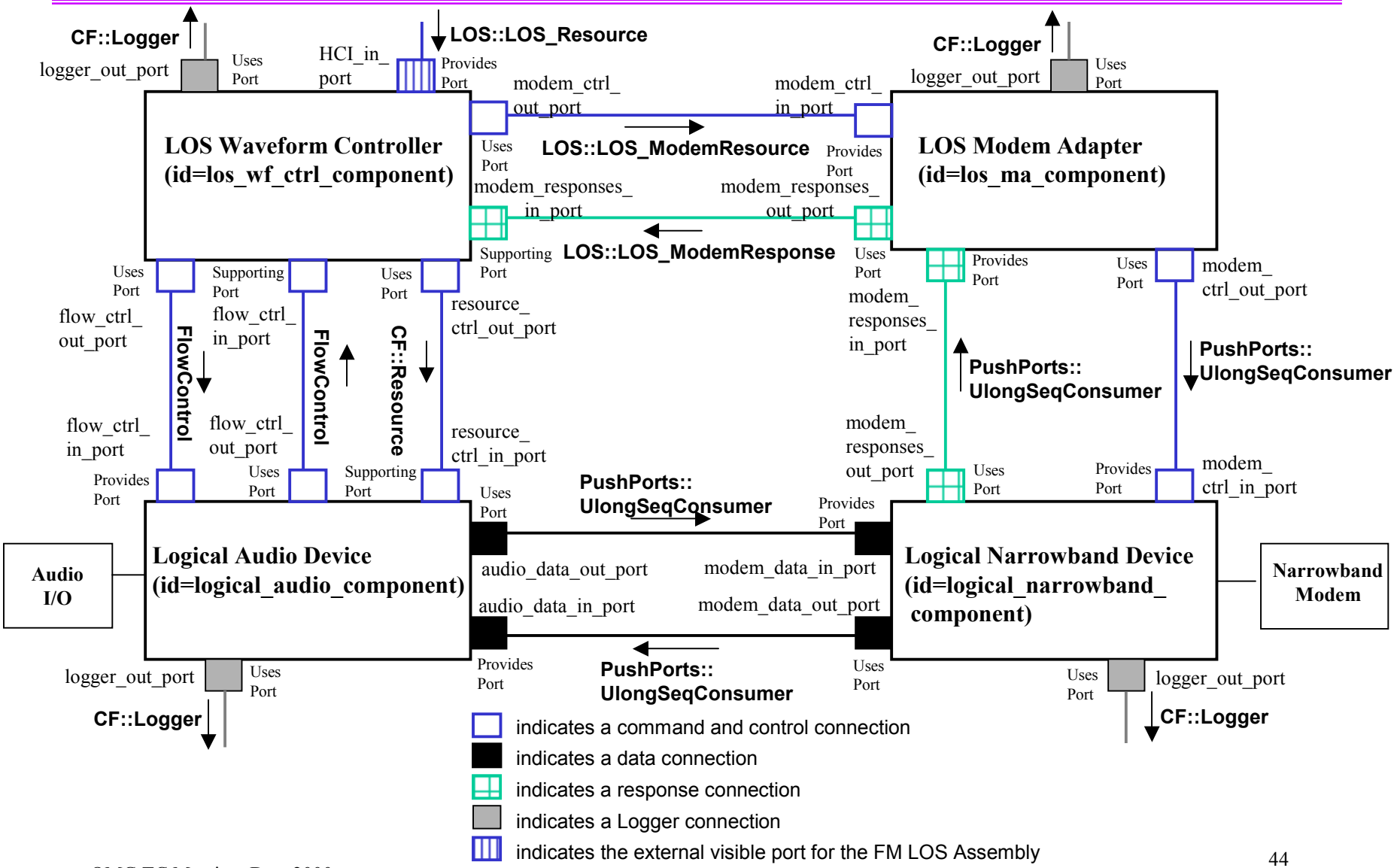
# Example FM Line-of-Sight Waveform Deployment Diagram



# Example Domain Profile - FM Line-of-Sight Waveform Assembly



# Example FM LOS Waveform CORBA Component Connections




# Summary

- SRA Specification is sorely needed for software-defined radios
  - Legacy architectures initially presented obstacles to SRA consensus
  - Proprietary radio solutions with non-reusable software are the norm
  - Commercial standards are evolving extremely fast
  - 3rd party development of radio applications and software components introduces new business models to radio manufactures
- Factors that limit radio software application portability
  - Resource constraints of embedded systems
  - Demand for high bandwidth rates, security, and QoS issues
  - High variation of radio domains - handheld, mobile, fixed site
  - Lack of mature OS and CORBA products for DSPs
  - Variation in embedded OS and CORBA implementations