Raytheon

Customer Success Is Our Mission

SCA Next Lessons Learned and Impact Analysis

Raytheon Network Centric Systems

Jerry Bickle

Dec 1, 2011

The Wireless Innovation Forum Conference on Communications Technologies and Software Defined Radio 2011

Agenda

- SCA Re-Factored Core Framework (CF) IDL Impacts
- SCA Component IDL Impacts
- CORBA Profiles Impacts
- SCA Domain Profiles Impacts
- CF Implementation Impacts

Raytheon

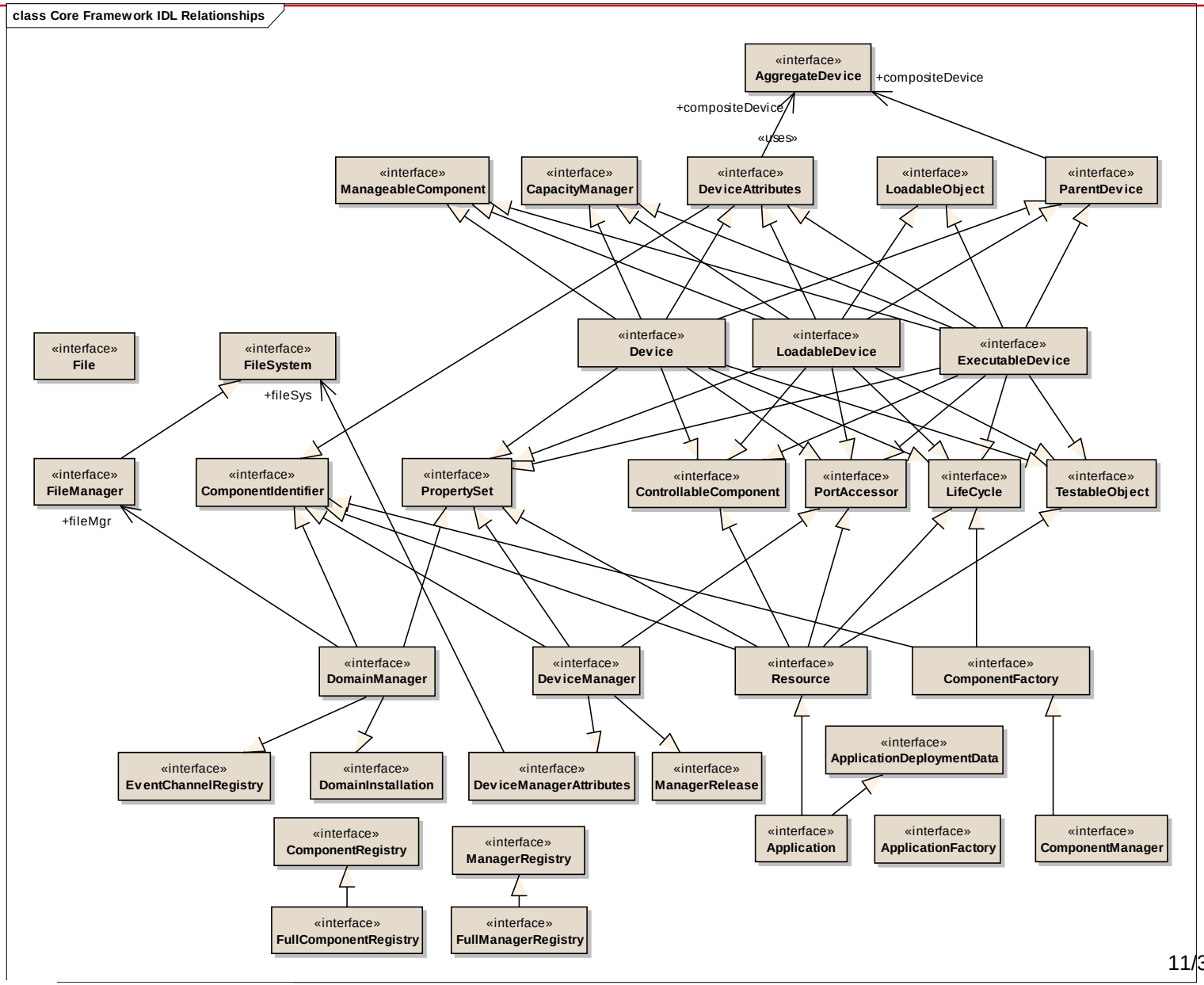
SCA Re-Factored CF IDL Impacts

- SCA Re-Factored CF IDL Interfaces
- SCA Affected Components
- SCA Servant Impact
- SCA Re-Factored Benefit

SCA Re-Factored CF IDL Interfaces

- Application Component Interfaces: Component Factory, Component Manager, Resource
- Base Component Interfaces: Common Types, Component Identifier, Controllable Component , Lifecycle, Port Accessor, Properties, Property Set,
- Device Component Interfaces: Aggregate Device, Capacity Manager, Device, Device Attributes, Executable Device, Loadable Device, Loadable Object, Manageable Component
- Device Management Interfaces: Component Registry, Device Manager, Device Manager Attributes, Full Component Registry,
- Domain Management Interfaces: Application, Application Deployment Data, Application Types, Application Factory, Component Registry, Domain Installation, Domain Manager, Event Channel Registry, Full Component Registry, Full Manager Registry, Manager Registry, Manager Release
- File Services Interfaces: File, File System, File Manager
- Platform Component Interfaces: Platform Types, Component Factory, Component Manager, Resource

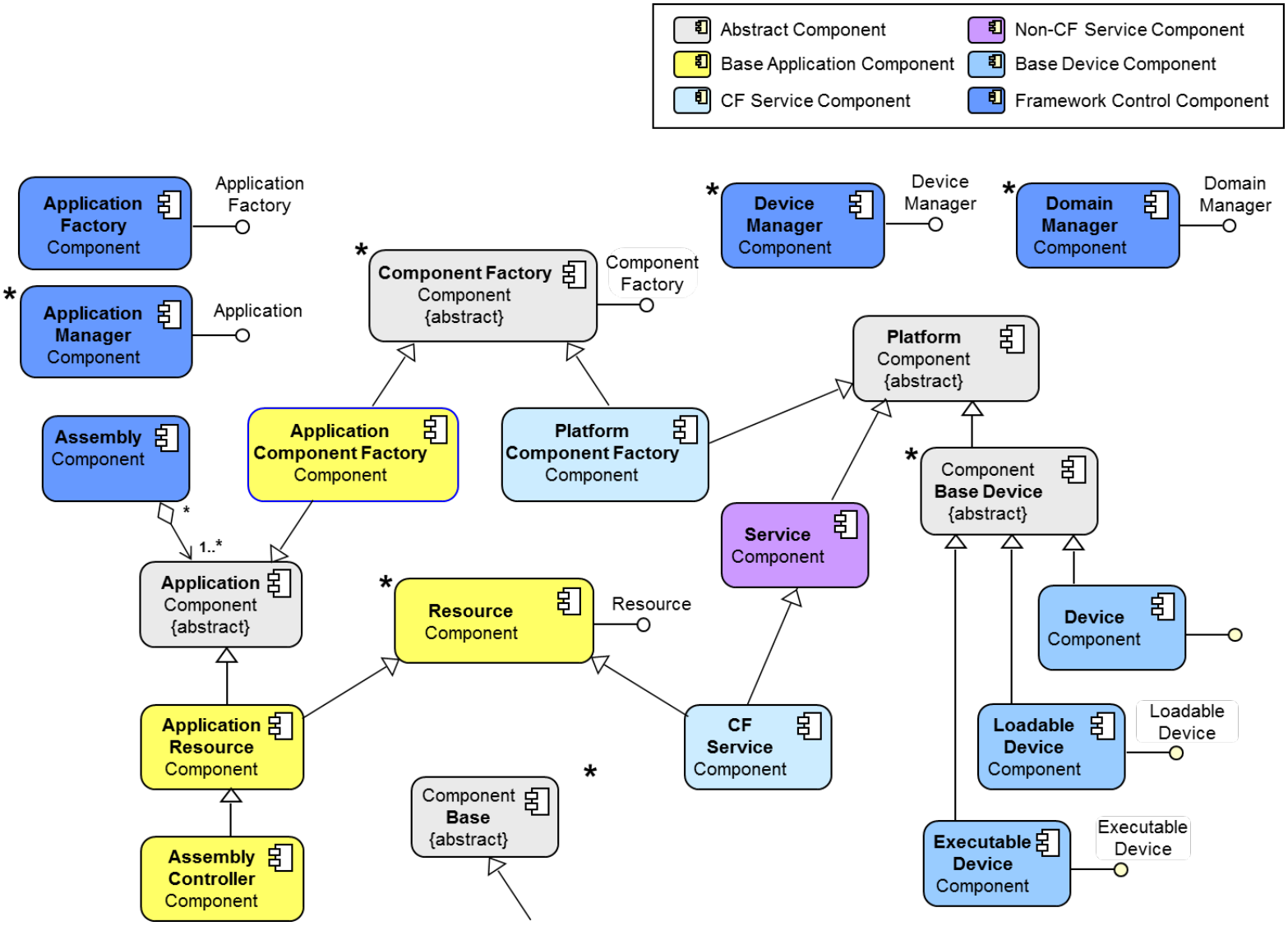
SCA Core Framework Interfaces



SCA Affected Components

- If one wants to take advantage of the Re-Factored CF then the impact is on the component ‘s servant class.
  - Application Manager
  - Application Factory
  - Application Components: Assembly Controller, Resource, Application Component Factory
  - Device Manager
  - Domain Manager
  - File Services: File, File System, File Manager.
  - Platform Components: CF Service (Resource), Device, Loadable Device, Executable Device, Platform Component Factory

SCA Component Hierarchy



SCA Servant Impact

Change Servant class from

Include “CFServer.idl”

To

Include “CFXXXServer.idl”

Where XXX is the CF interface name: Application, Resources, Device, Executable Device, Device Manger, Domain Manager, etc.

For Example: Include “CFDeviceServer.idl”

SCA Servant Impact (cont'd)

- To support backwards compatibility or to support re-factored versus non re-factored CF one may add a compile directive such as “SCA_REFACTORED“

```
#if defined(SCA_REFACTORED)
#include "CFDeviceServer.h"
#else
#include "CFServer.h"
#endif
```

SCA Servant Impact (cont'd)

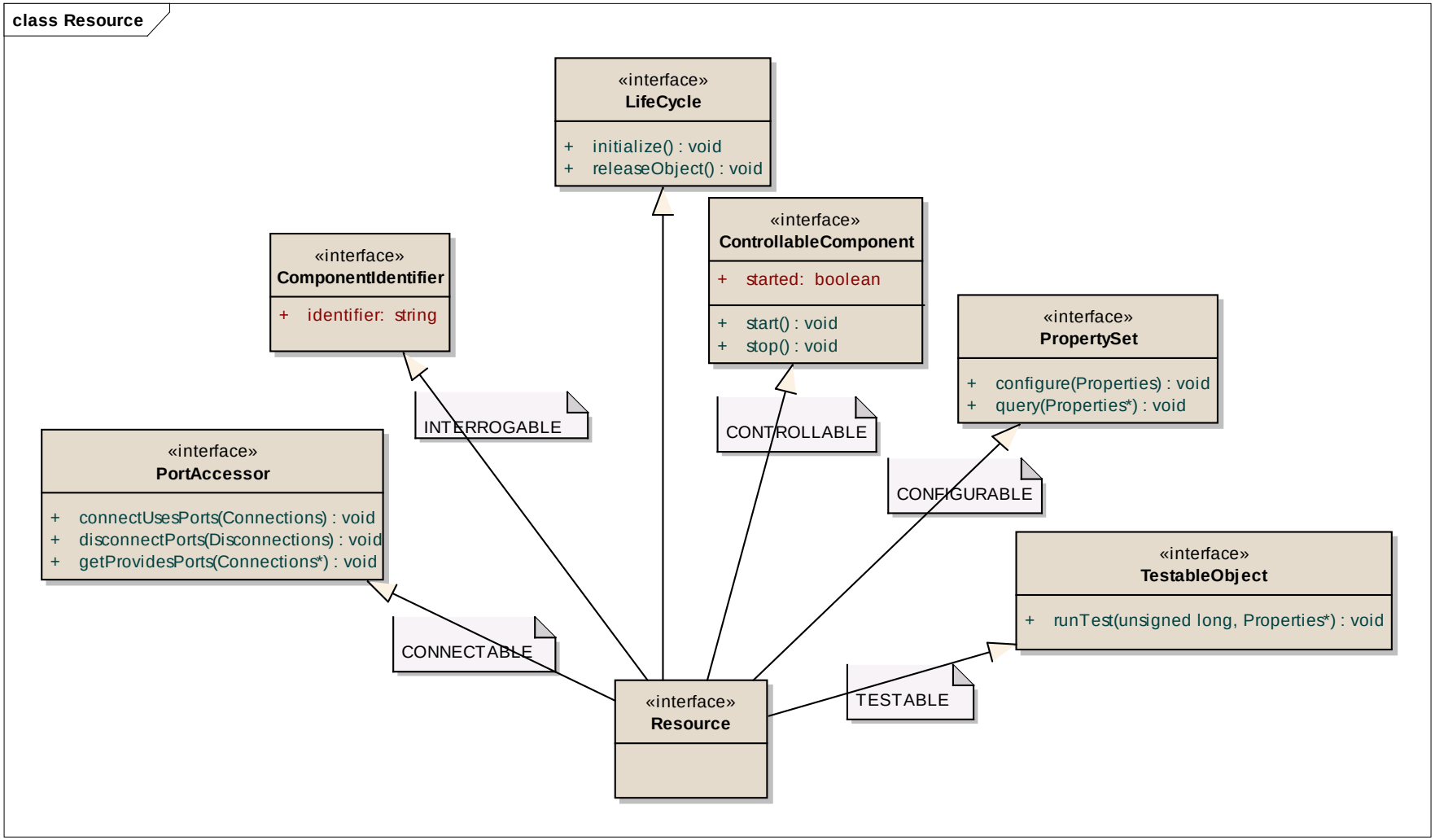
- If a component servant implementation is supporting the optional Component Un-Registration Units of Functionality (UOF) then an additional include is needed for the un-registration behavior as follows:

```
#if defined(SCA_REFACTORED)
#include "CFDeviceServer.h“
#include “CFFullComponentRegistryClient.h”
#else
#include "CFServer.h"
#endif
```

SCA Re-Factored Benefit

- Size benefit achieved by Operating Systems (OS) supporting separate process address space with static linking instead of a flat/single address space, allowing components to be in separate process address spaces.
  - Examples of OS process address space:
    - Green Hills Integrity
    - LYNUXWORKS LynxOS RTOS
    - Wind River VxWorks Real -Time Process (RTP)
    - Linux

Conditional Resource Interfaces



SCA Re-Factored Benefit (cont'd)

- Size Measurements based upon ARM processor and GNU 4.1.2 compiler
  - Full SCA Resource Size with uses ports
    - Minimum of 300 kilobytes in savings depending on component type
    - More for Resource type and less for an Executable Device type
- Better Assurance using a Least Privilege design pattern
- Direct support to the optional Units of Functionality (UOF)

Raytheon

SCA Component IDL Impacts

- Component Registration Impacts
- Resource Factory Impacts
- Component Ports Impacts

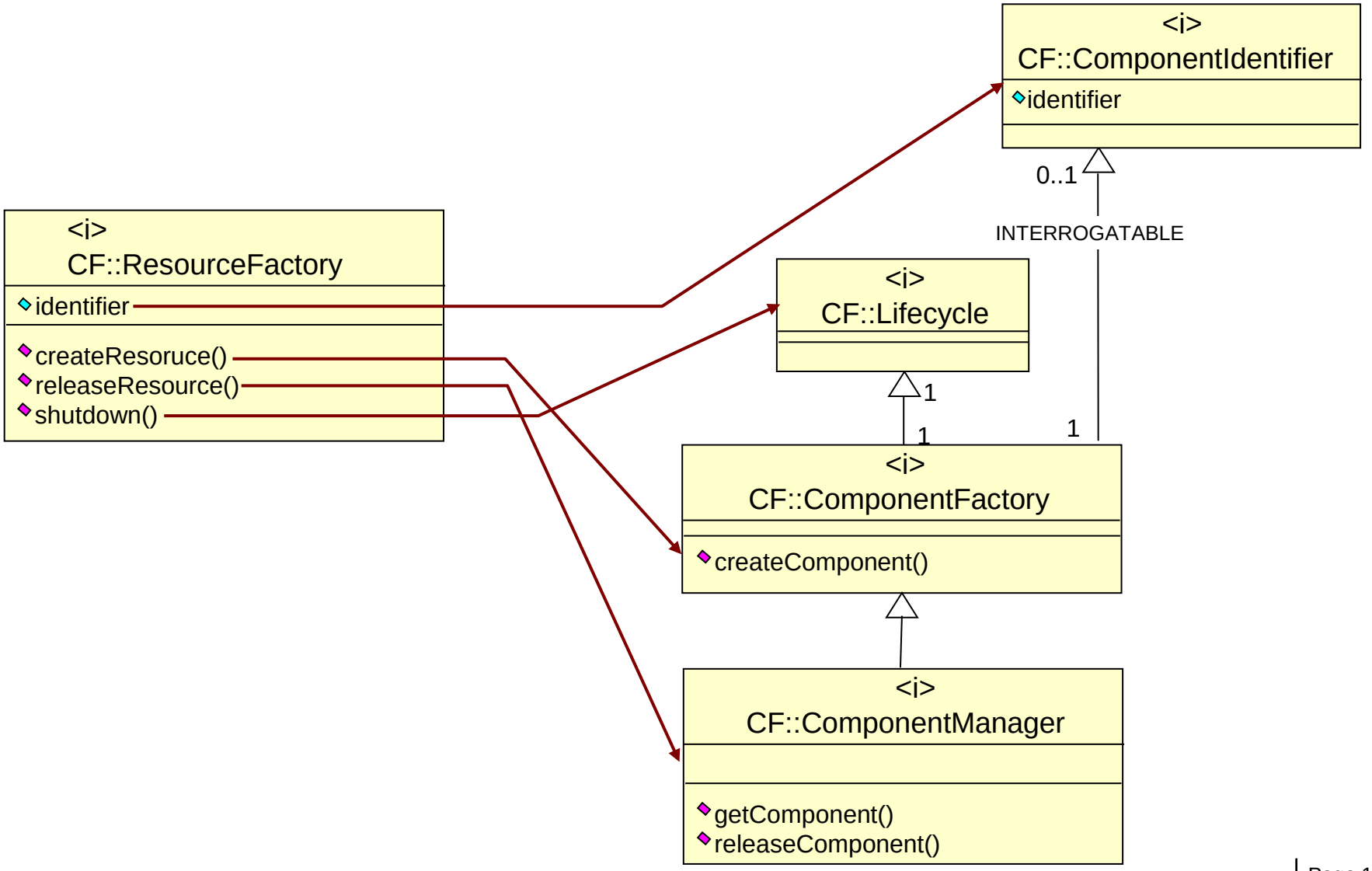
SCA Component Registration Impacts

- Component Registration impacts the component's main code
  - If using Re-Factored CF then "include CFComponentRegistryClient.h"
  - For Application Component or Platform Service Component
    - Replaces the Naming Service code
      - Replaces the Naming Service Context and Binding parameters with Component Registry parameter
      - Replaces the code that binds an object to a naming context with the Component Registry registerComponent operation.
  - For Platform Component
    - Replaces the Device Manager registration code
      - Replaces the Device Manager parameter with Component Registry parameter
      - Replaces the Device Manager register operations with the Component Registry registerComponent operation.

SCA Resource Factory Impacts

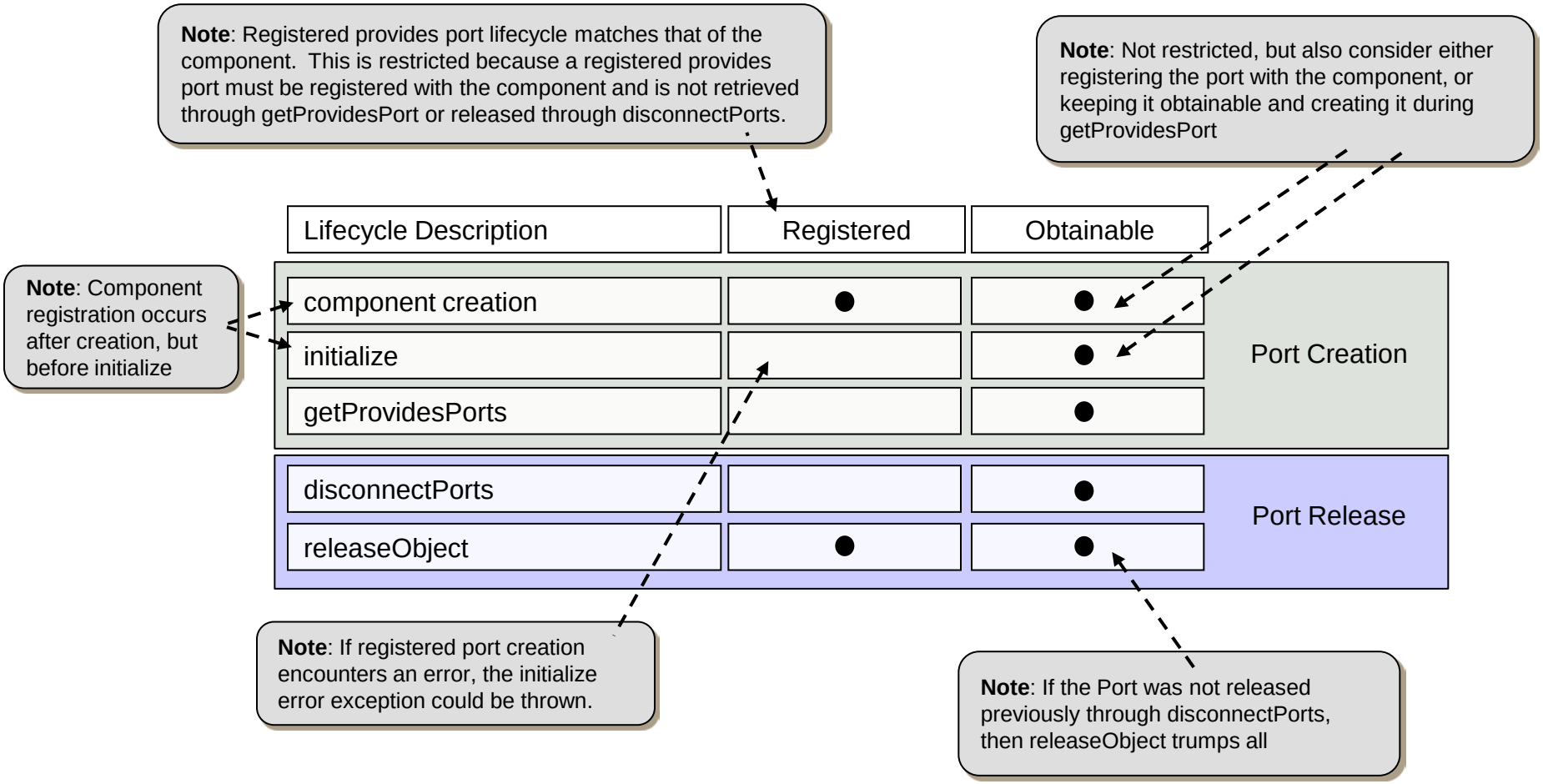
- Resource Factory replaced by 2 interfaces with similar behavior but different operations:
  - Component Factory
    - ComponentFactory inherits Lifecycle
      - Provides consistent behavior for component deployment.
        - » Initialize – new behavior
      - Lifecycle::releaseObject replaces shutdown operation that provides a consistent behavior for component teardown.
    - Create component operation returns a ComponentType instead of a Resource object reference.
  - Component Manager
    - Extends Component Factory with component release and getComponent capability.
- Component Factory now applies to both application and platform component deployment.

SCA Resource Factory Impacts (cont'd)



Component Ports Impacts

- No “Uses Port” CORBA Objects (CF::Port interface)
- Provides Ports Lifecycle



[1]Common Object Request Broker Architecture (CORBA) provides a seamless software bus between heterogeneous processing elements and physical interfaces

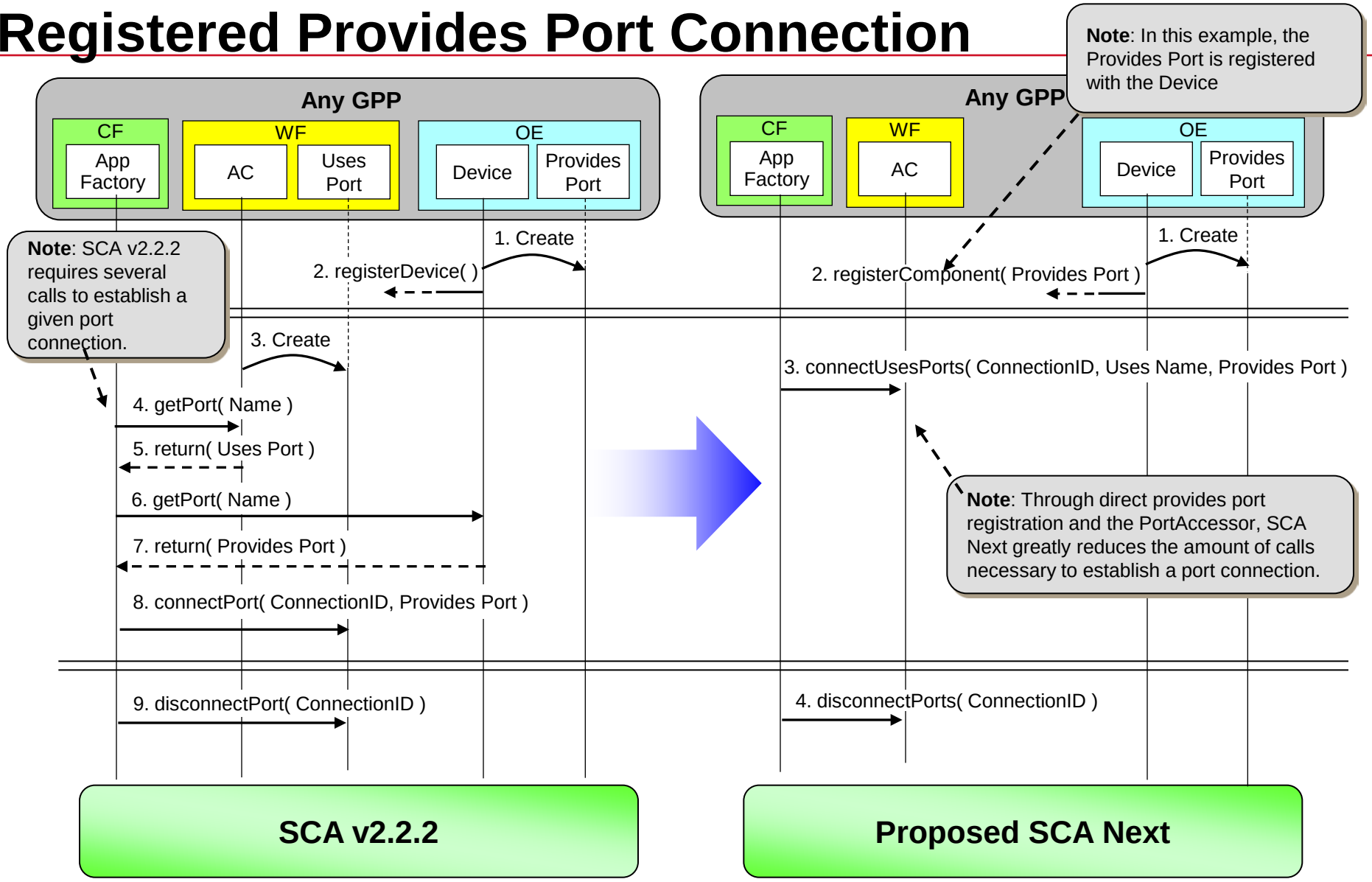
Component Ports Impacts (cont'd)

- To minimize impact on existing code one could register a component with no ports registration as is currently done for a SCA 2.2.2 implementation that is still valid for a SCA Next implementation
- To speed up deployment one can modify existing code to register ports upon component registration, which may result in modifying existing code as follows:
  - If ports are created and activated on Lifecycle::initialize operation then this code would have to be moved to component servant constructor behavior or a new operation that is called by the main code.
  - Need to add code to main to retrieve the ports from the component servant. One could use the PortAccessor::getProvidedPorts operation for this retrieval or a different servant operation.

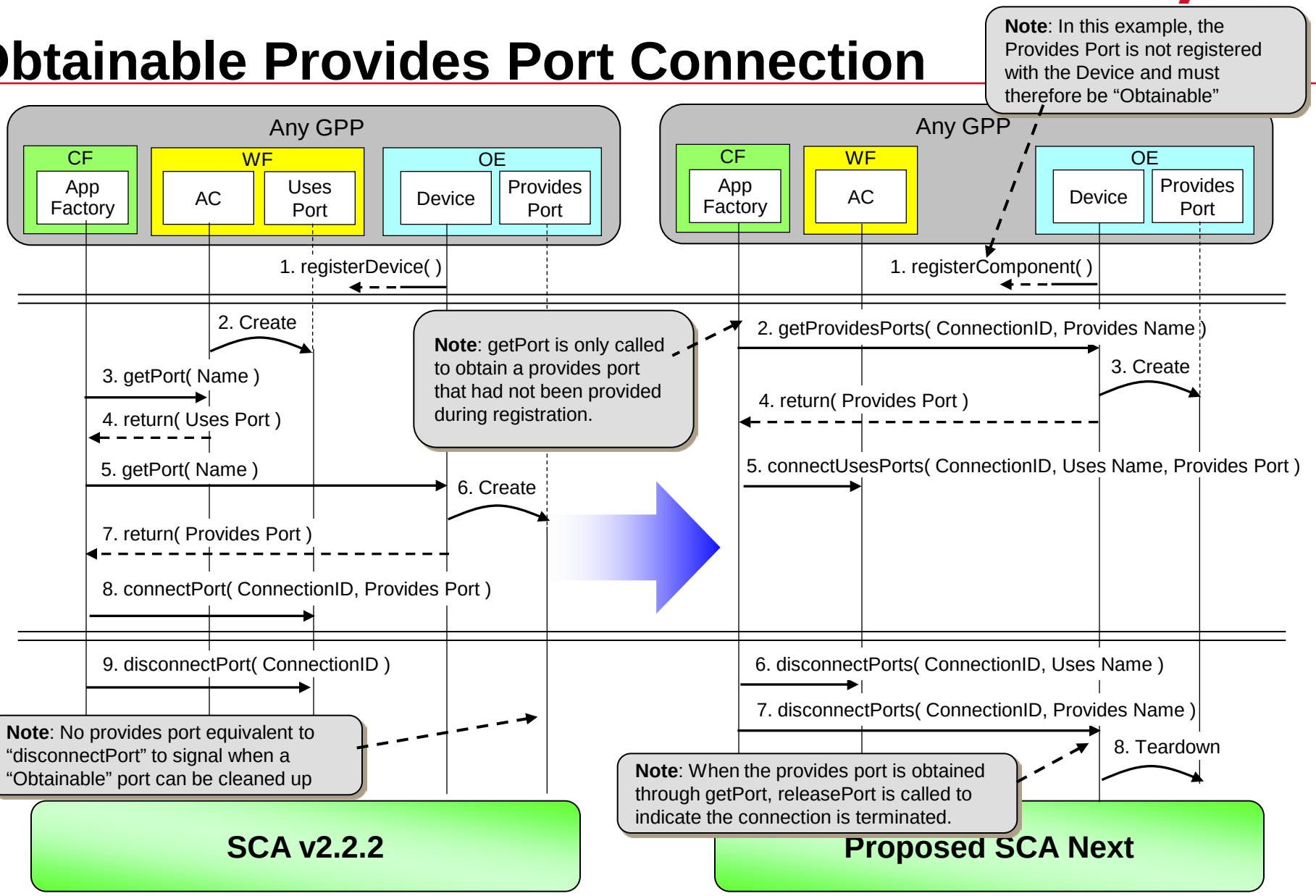
Component Ports Impacts (cont'd)

- To minimize impact, the same “uses” and “provides” port design patterns used for SCA 2.2.2 can still be used with the component servant’s Port Accessor operations.
- The Port Accessor operations can be a layer on top the existing port classes.
  - For example, the connect and disconnect operations can delegate to uses port classes
  - The getProvidedPorts operation mimics the SCA 2.2.2 PortSupplier::getPort operation without SCA 2.2.2 uses ports (CF::port)
- SCA 2.2.2 Uses Port changes from a CORBA object (CF::Port) to a regular cpp class.

Registered Provides Port Connection



Obtainable Provides Port Connection



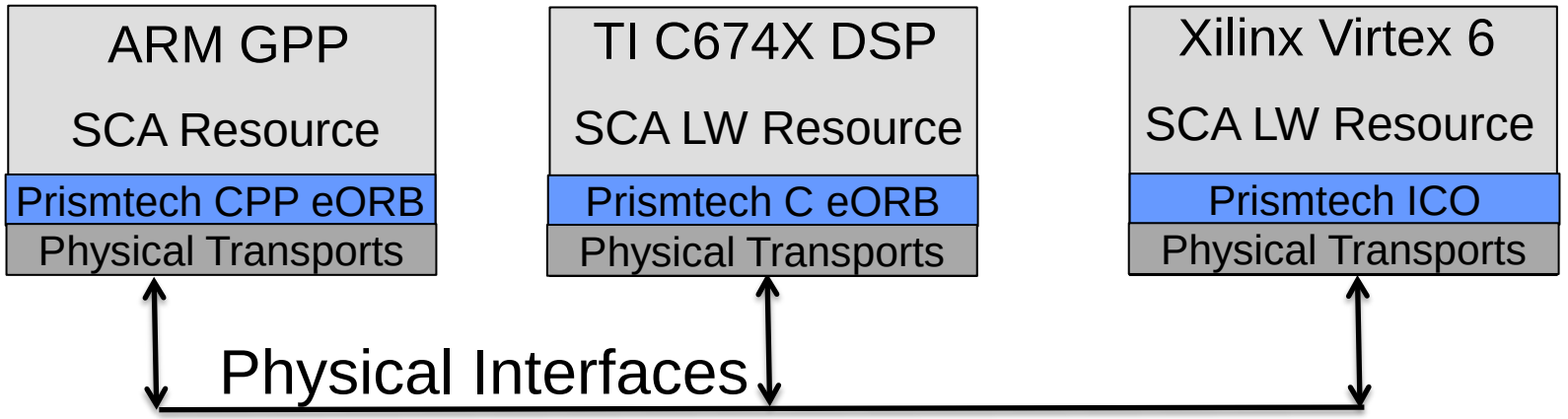
Raytheon

SCA CORBA Profiles Impacts

- CORBA Profiles Observations
- How does CORBA compare to MHAL?
- CORBA and MHAL Approaches Comparisons

CORBA Profiles Observations

- Evaluated on ARM GPP, TI C674x DSP, and Xilinx Virtex 6 FPGA processing elements using Prismtech CORBA ORB products
  - For DSP, a lightweight (LW) Resource component without Property Set and Testable Object interfaces was used, resulting in a larger code size and more memory usage than non-CORBA implementation
  - For FPGA, component with provides interfaces and uses port.



CORBA Profiles Observations (cont'd)

- The technology is viable beyond the GPP.
- Expect a learning curve for both software and hardware signal processing engineers.
  - CORBA and SCA Modeling tools can reduce development time by
    - Eliminating development time implementing internal message protocols for communication between processing elements
    - Eliminating development time implementing SCA Resource and Device infrastructure requirements
- The main impact is the transition of legacy code that used the MHAL approach to using CORBA approach
  - The cost and schedule to transition from MHAL to CORBA depends on how well the application defines an interface that abstracts away the MHAL transport mechanism
  - Cost and schedule is also driven by how close one can define a corresponding CORBA interface that matches an existing application interface to minimize code changes

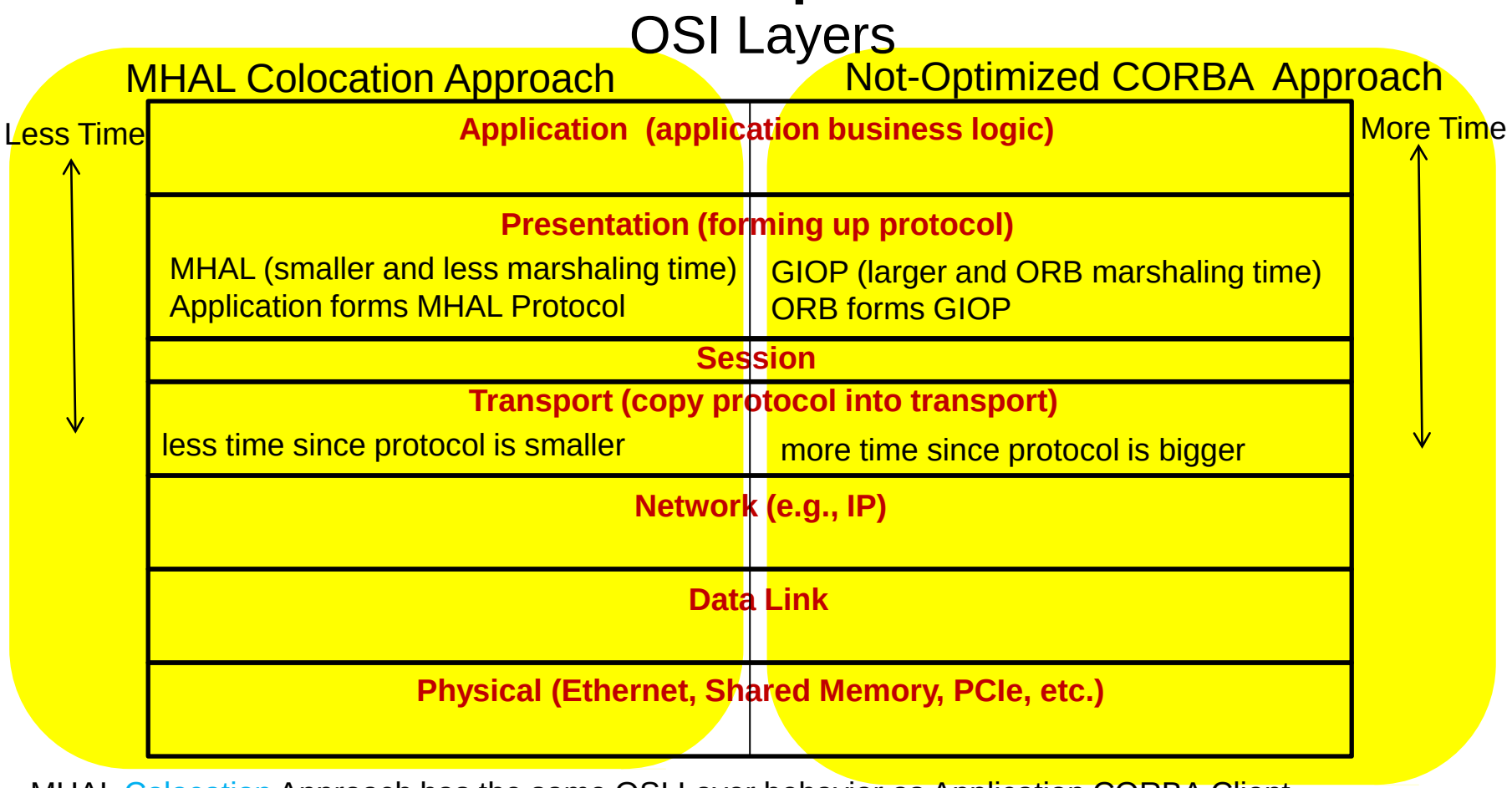
How does CORBA compare to MHAL?

- This depends on the approach taken for MHAL and for CORBA.
  - Modem Hardware Abstraction Layer (MHAL) API Approaches
    - MHAL Colocation – in same address space as application
    - MHAL non-Colocation – in separate address space from application
  - CORBA Approaches
    - Not-optimized CORBA – (i.e. GIOP)
    - Optimized CORBA – Industry optimized Inter-ORB Protocols (IOP) and minimal marshaling techniques offers savings over MHAL approaches

How does CORBA compare to MHAL? (cont'd)

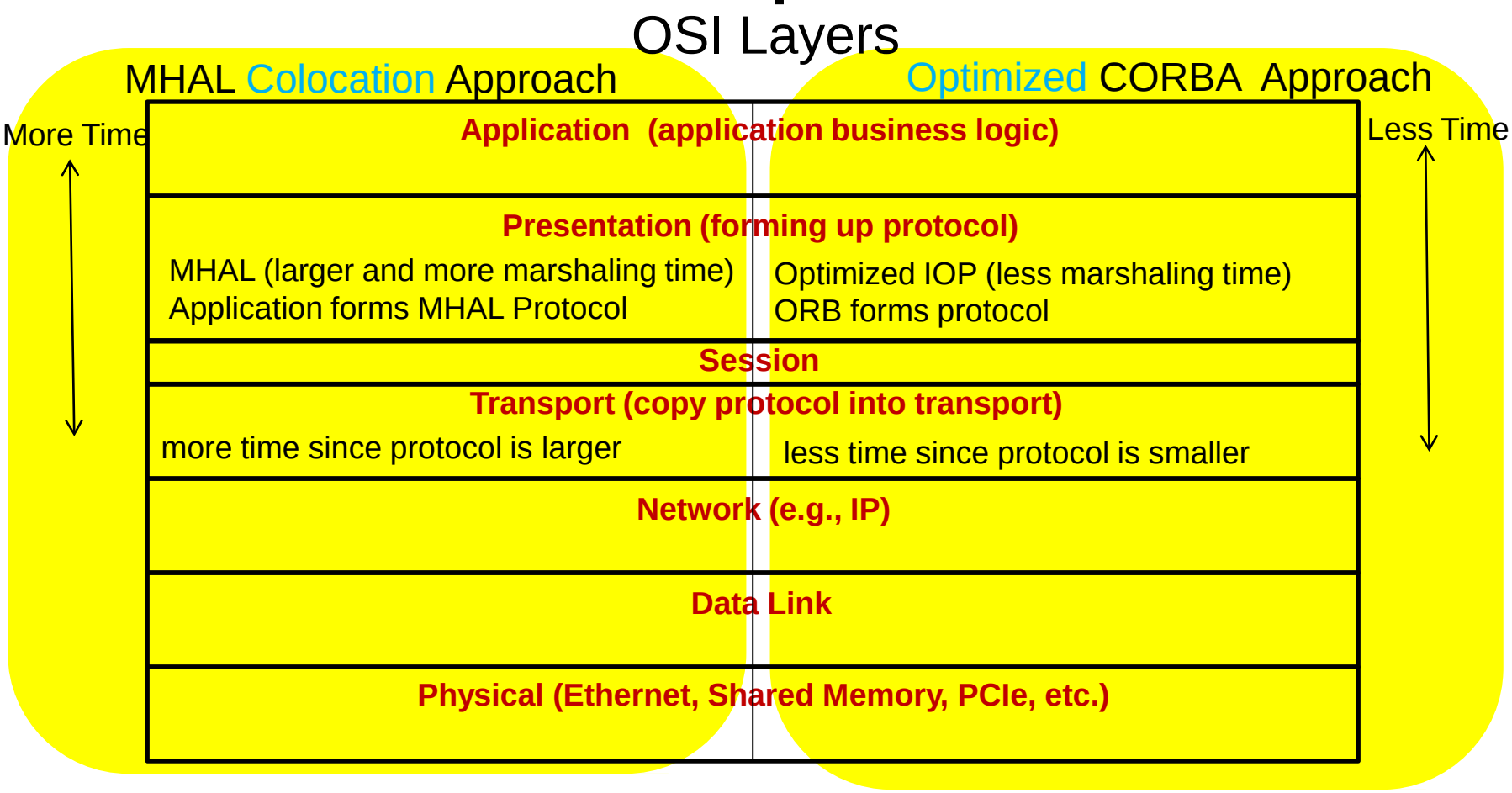
- Measured Parameters
  - Throughput Performance and Latency
    - For all approaches, the throughput performance and latency are determined by the Open Systems Interconnection (OSI) presentation and transport layers, along with the number of times the OSI layers are traversed
    - MHAL Colocation Approach has same OSI Layer behavior as Application CORBA Client Approaches as illustrated on next slides
    - MHAL non-Colocation Approach has more OSI layer behavior as illustrated on next slides
    - For throughput performance and latency comparisons, all approaches are based upon a non-flat (i.e. multiple addressing spaces/separate process address space) operating system addressing architecture and same transport, network, data link and physical interface layers.

How does CORBA compare to MHAL? – MHAL Colocation versus Not-Optimized CORBA



1. MHAL Colocation Approach has the same OSI Layer behavior as Application CORBA Client
2. If application for MHAL Colocation Approach is just a proxy middleman to move data for a CORBA-nonCORBA communication then CORBA Approach would be better since it would be similar to MHAL non-Colocation Approach
3. CORBA Approach can provide Quality of Service (QoS) unlike MHAL Approach (i.e. message priority)
4. CORBA Approach can provide tool support unlike MHAL Approach (i.e. interface and SCA code generation)

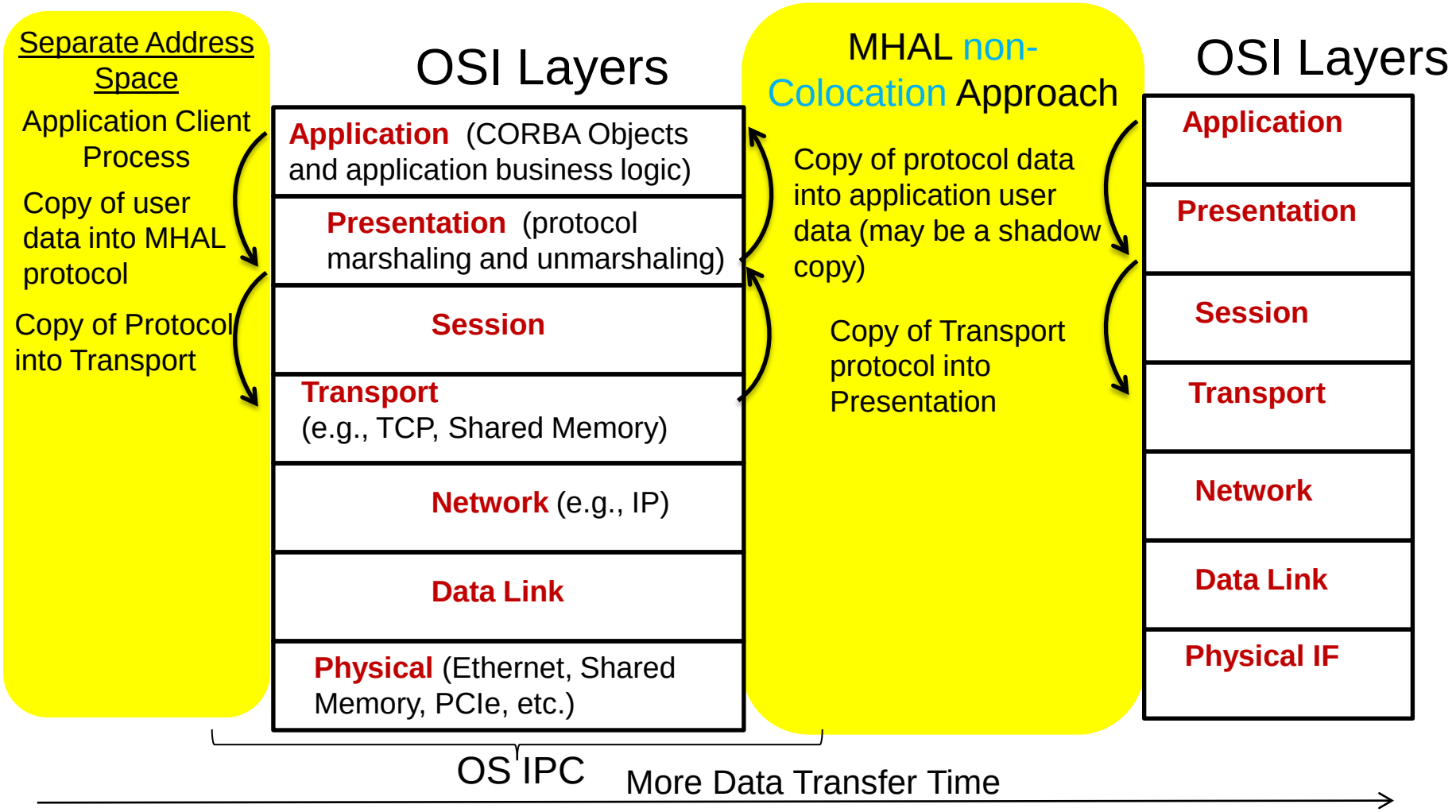
How does CORBA compare to MHAL? – MHAL Colocation versus Optimized CORBA



1. Same notes as MHAL Colocation Approach versus Not-optimized CORBA Approach

How does CORBA compare to MHAL? – MHAL non-Colocation Illustration

Raytheon



1. MHAL non-Colocation Approach requires more OSI layers
2. The amount of processing time varies depending on the OS Inter-process communication (IPC) mechanism being used between Application and MHAL Server

CORBA and MHAL Approaches Comparisons

- From Best to Worst for throughput performance and reduced latency using a high assurance model.
  - Optimized CORBA Approach
  - MHAL Colocation Approach
  - Not-optimized CORBA Approach
  - MHAL non-Colocation Approach

Domain Profiles Impacts

- The use of XML files within the radio is not explicitly required by an SCA implementation.
  - This was true even for SCA 2.2.2 and is still true for SCA Next.
- The minimum requirements for XML are the profile attributes for components:
  - Application
  - Application Factory
  - Device Manager
  - Domain Manager
  - Device, Executable Device and Loadable Device
- Profile attributes are optional for a Device and Device Manager components that corresponds to the optional UOF INTERROGABLE.

Domain Profiles Impacts (cont'd)

- Application (Software Assembly Descriptor)
  - Nested Applications – optional for CF implementation and number of nested levels supported is not stipulated
  - Domain Finder Extensions for connecting to an Application
  - Deployment Dependencies
  - Removed FindBy – related to the removal of Naming Service
- Software Component Descriptor (SCD)
  - Uses Port maximum number of connections and Component Type
- Software Package Descriptor
  - Added Lightweight AEPs
- Device Manager Configuration Descriptor (DCD)
  - Domain Finder service extensions is also applicable for DCD
  - Any type of Device's allocation property can be overridden in the DCD.
- Properties
  - Structure Sequence does not have to have initial values.
- ID – are not required to be UUID but required to be unique within domain.
- Profile Descriptor - removed

CF Implementation Impacts

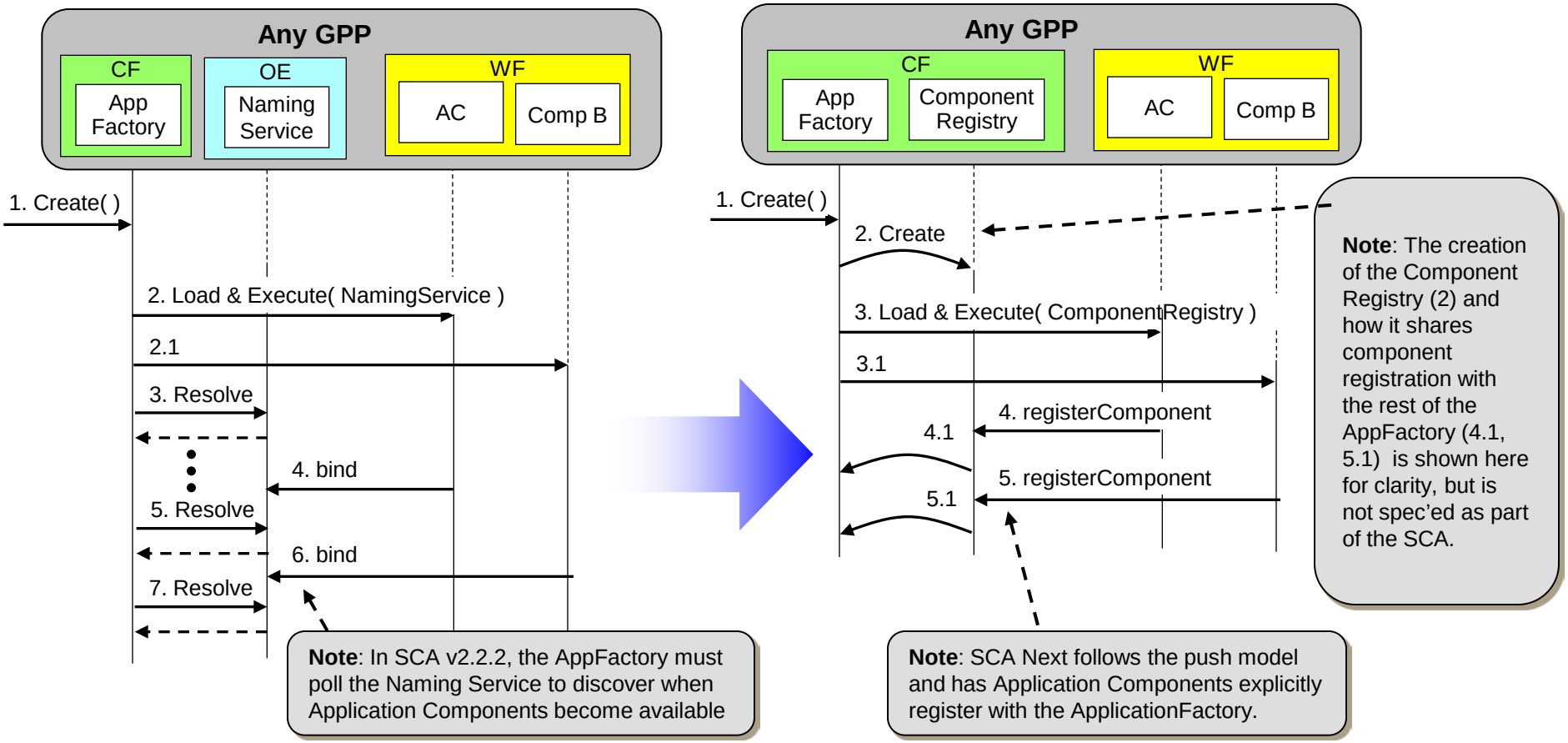
- Component Registration
- Manager Registration
- Application Factory and Application Manager Impacts
- Managers Optional UOFs

CF Implementation Impacts – Component Registration

Raytheon

- Application Factory Component/Application Manager
  - Removal of the Naming Service
  - Replaced by Component Registry CORBA object
    - Implementation specific on number of Component Registry objects.
- Device Manager Component
  - Registration and unregistration operations removed from Device Manager.
  - Replaced by a Component Registry CORBA object.

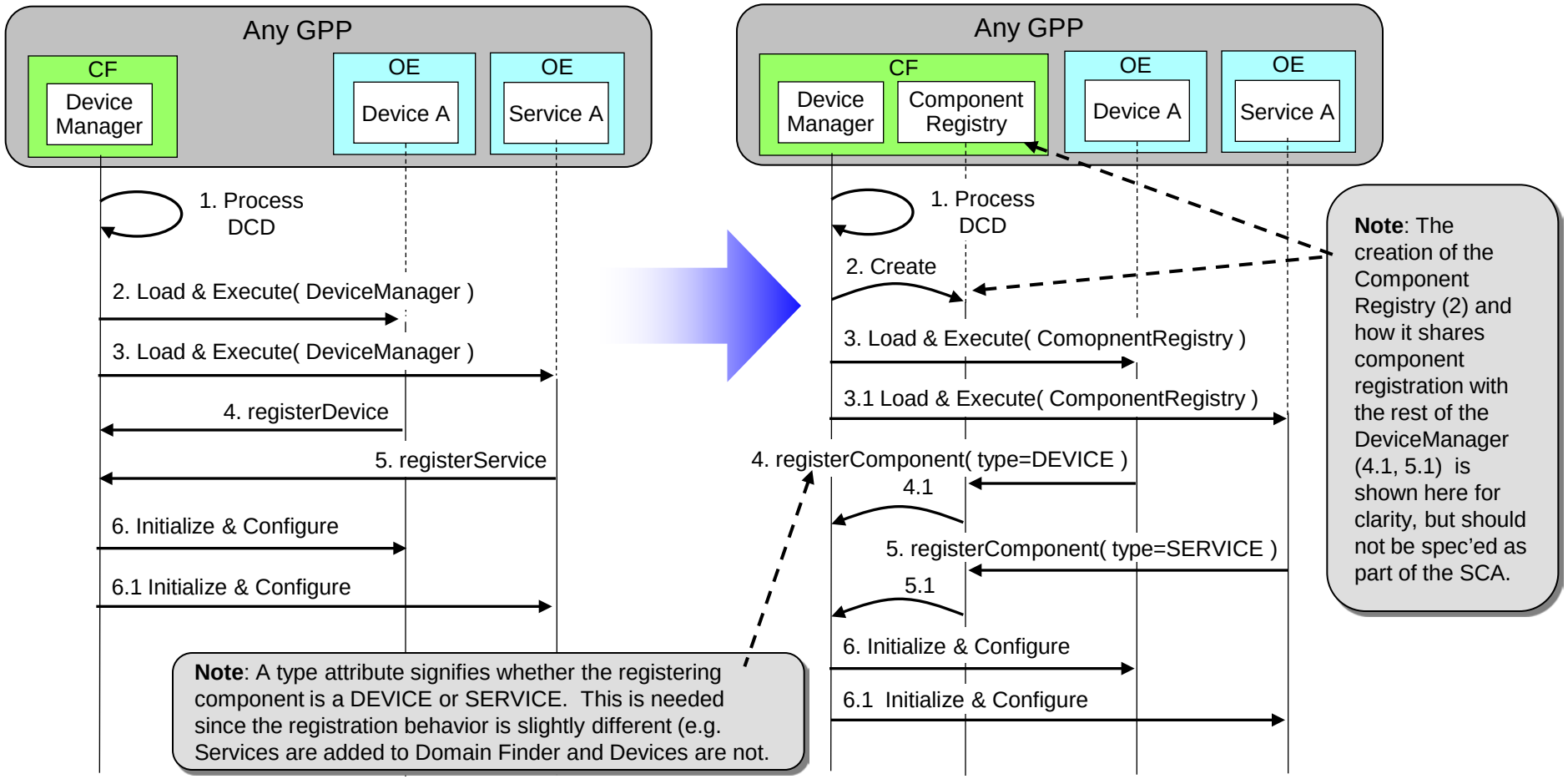
Application Factory Component Registration



In SCA v2.2.2, a Naming Service is used and shared with Application Components for the purposes of registration.

In SCA Next, a standalone Component Registry interface is shared with Application components to be used for registration.

Device Manager Component Registration



In SCA v2.2.2, the entire DeviceManager interface was provided to each OE component for the purposes of registration.

: In SCA Next, only the separate Component Registry interface is shared with OE components for the purposes of registration.

CF Implementation Impacts – Manager Registration

Raytheon

- Device Manager Registration
  - Obtaining Domain Manager Object Reference
    - Use of Naming Service not required
    - Domain Manager object reference can be:
      - CORBALOC string
      - IOR string
      - Object Reference can be in a file
  - Registration to Domain Manager is now done by Manager Registry interface

CF Implementation Impacts – Application Factory and Application Manager

Raytheon

- Optional Nested Application UOF
  - The maximum number of nested applications supported is not specified in SCA
  - Minimum of one when Nested Application UOF is implemented
- Deployment Dependencies
  - Overriding component deployment dependencies at the SAD and Application Factory create level.
    - Added deploymentDependencies parameter to Application Factory::create operation.

CF Implementation Impacts – Managers Optional UOFs

Raytheon

- Device Manager Optional UOFs
  - Management Releasable provides the device manager releasing capability
- Device and Domain Managers Optional UOFs
  - Management Registration provides the registry interfaces for registering components to domain and device manager components capability
  - Management Un-Registration provides the un-registry interfaces for unregistering components from domain and device manager components capability
- Domain Manager Optional UOFs
  - Application Installable provides capability for dynamic application installation and un-installation
  - Channel Extension provides the concepts of platform channels and deployment of applications onto platform channels capability
  - Event Channel provides event channels and the event service capability in the SCA OE

Summary

- The use of SCA Model Driven Development tools can off load the transition from SCA 2.2.2 to SCA Next for:
  - SCA Component Infrastructure and SCA requirements
    - Component Registration impacts
    - Component Ports impacts
    - Component Mains
    - Optional UOF
- CORBA is viable technology beyond GPP
- Use of Re-Factored SCA CF IDL can reduce your code size and time for development.