



Joint Program Executive Office Joint Tactical Radio System

Lightweight Components



02 December 2010
JTRS SCA Working Group

JPEO JTRS



Task Overview

- **Objective**

- To have a flexible architecture that can accommodate various platforms requirements (mobile versus static, single channel versus multiple channels, single waveform versus multiple waveforms, small form factor, etc.) instead of one size fits all architecture

- **Benefits**

- The elimination of interfaces that are not needed by a component results in:
 - Assurance –increased assurance by removing operations that are not intended to be utilized when deployed
 - Footprint Size – reduced footprint size
 - Performance – increased performance
 - Development Time – reduced requirements, which leads to reduction in implementation, testing, and integration

- **Impact**

- SCA Test Software on CF components optional behavior
- To maintain existing CF Application implementation need to turn on optional interfaces during CF IDL compiler generation

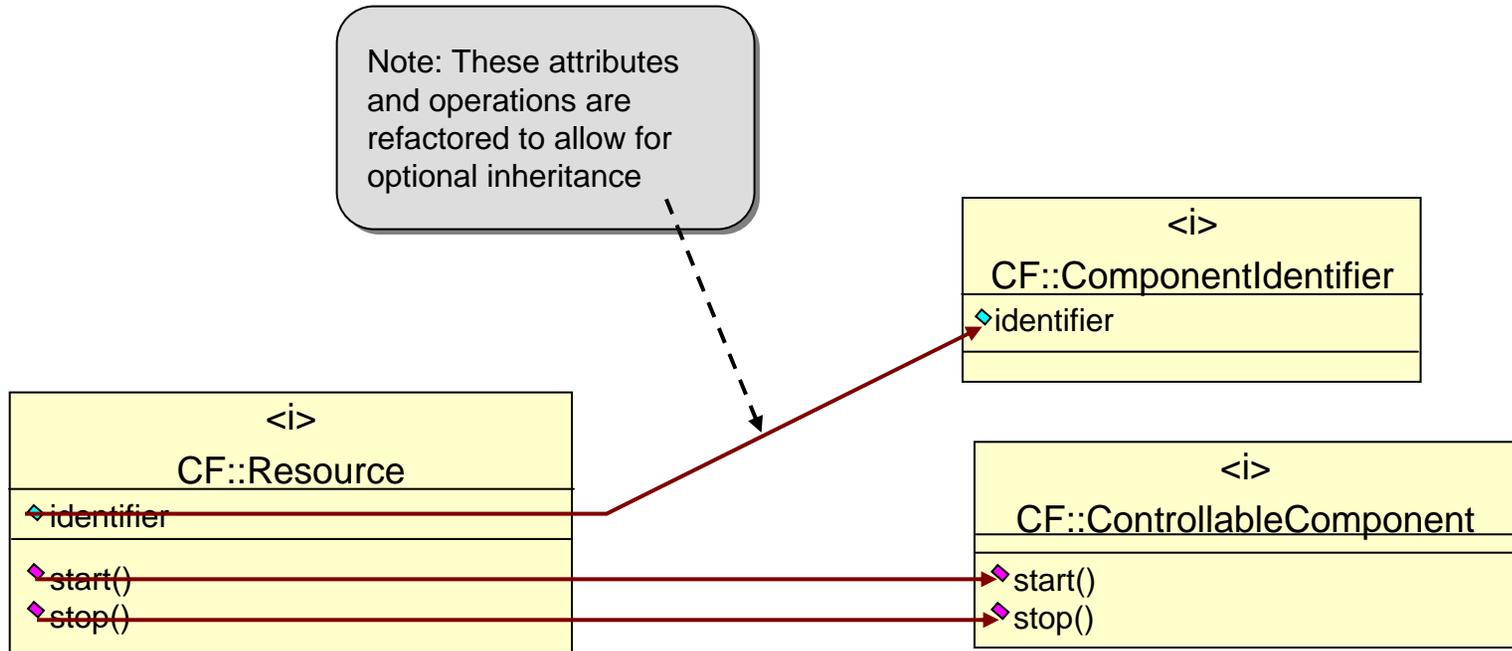


Solutions

- **CF Interfaces Refactoring**
- **Resource Interface Refactoring**
- **ResourceFactory Interface Refactoring**
- **Device Interface Refactoring**
- **LoadableDevice Interface Refactoring**
- **ExecutableDevice Interface Refactoring**
- **Interface Details**

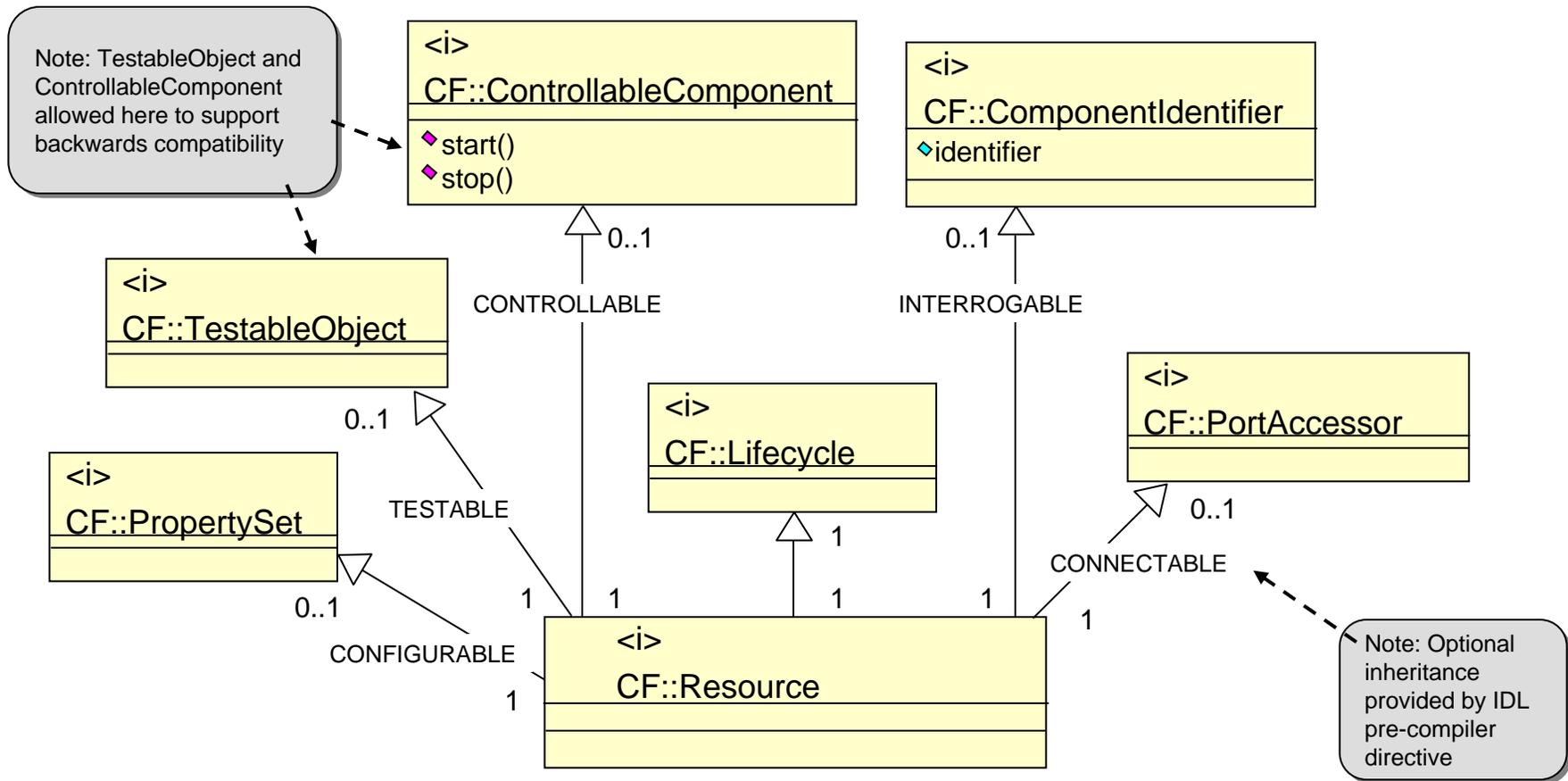


Resource Interface Refactoring



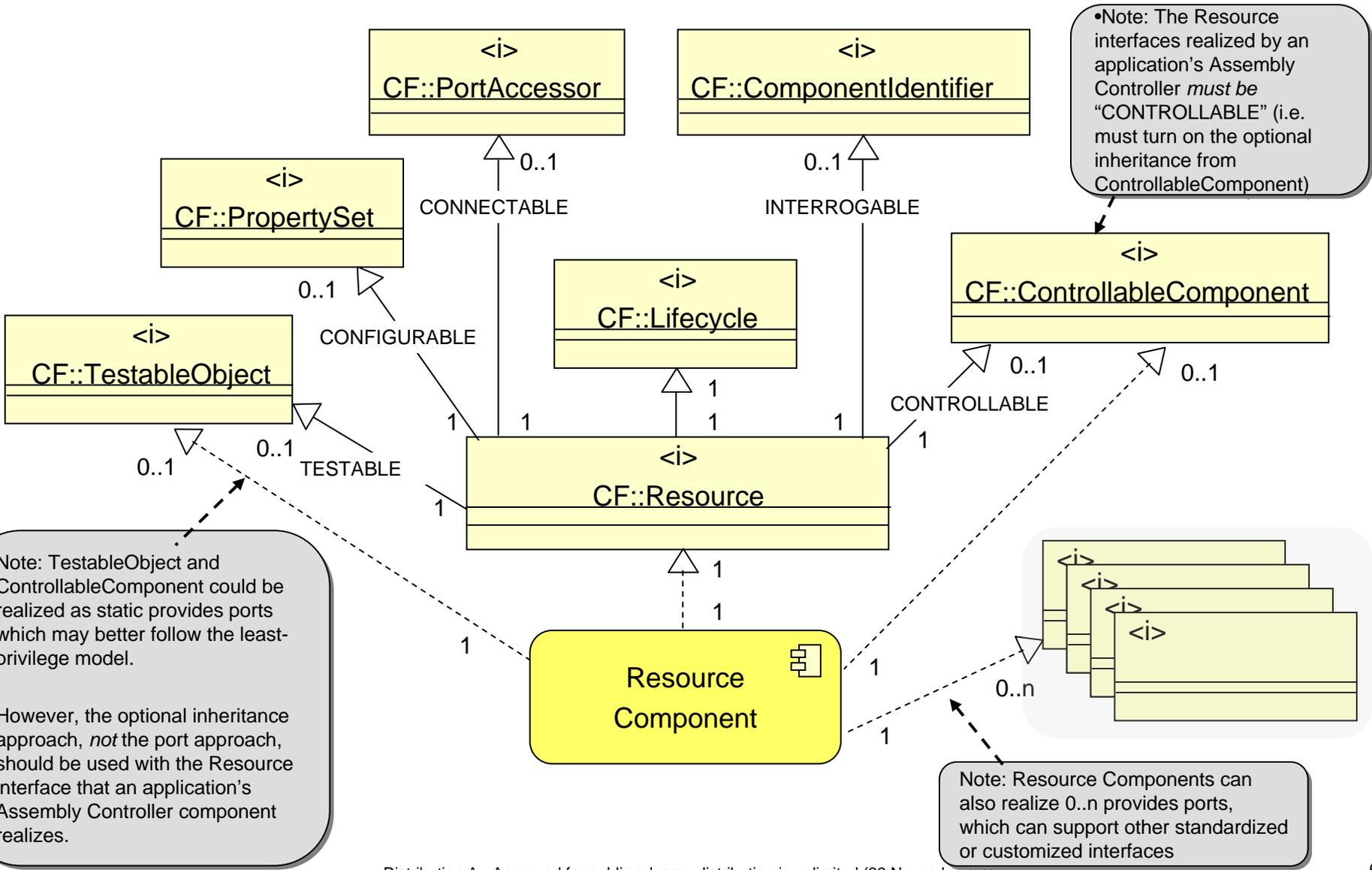


Resource Interface Refactoring, cont'd



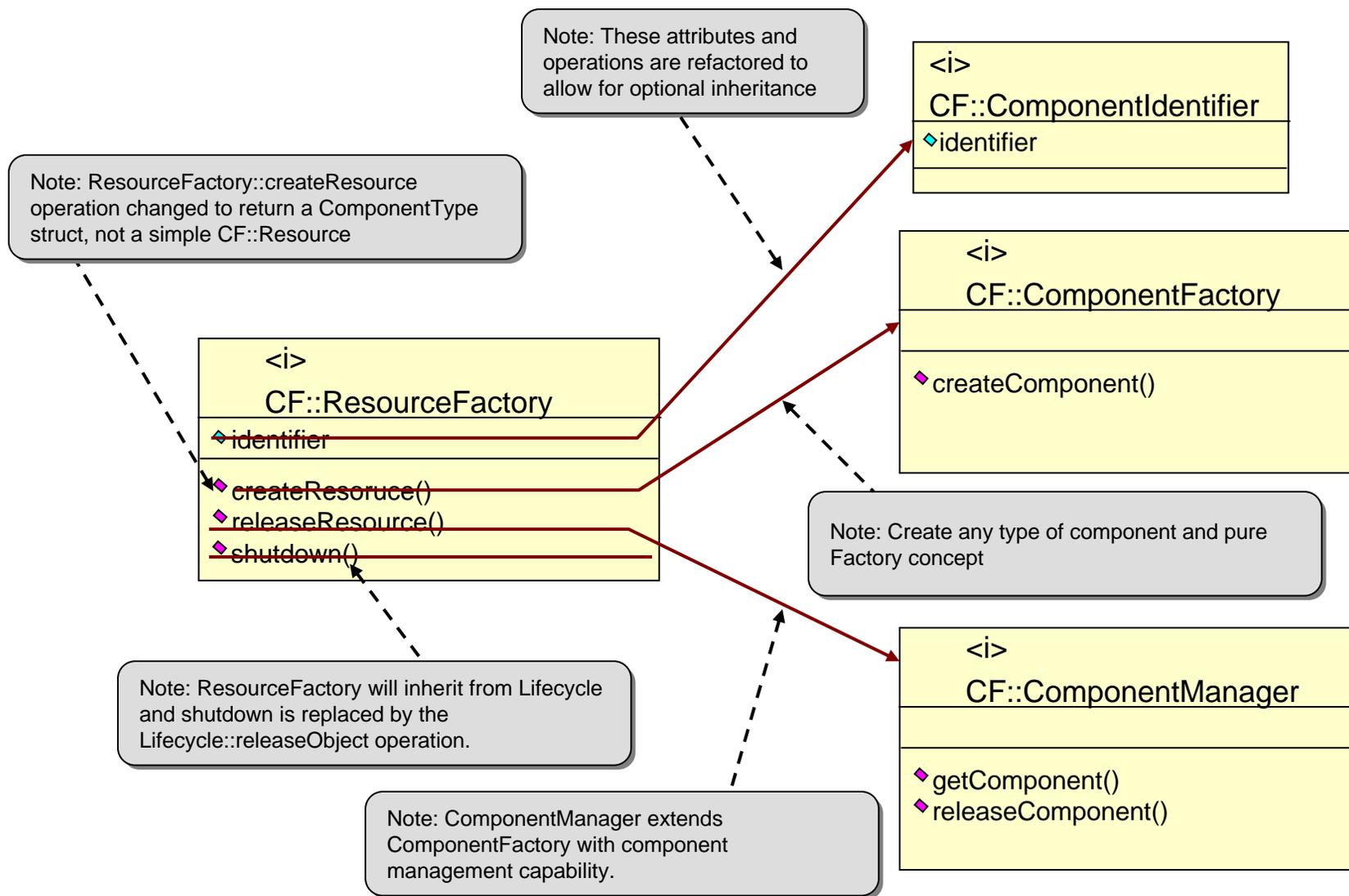


Resource Component



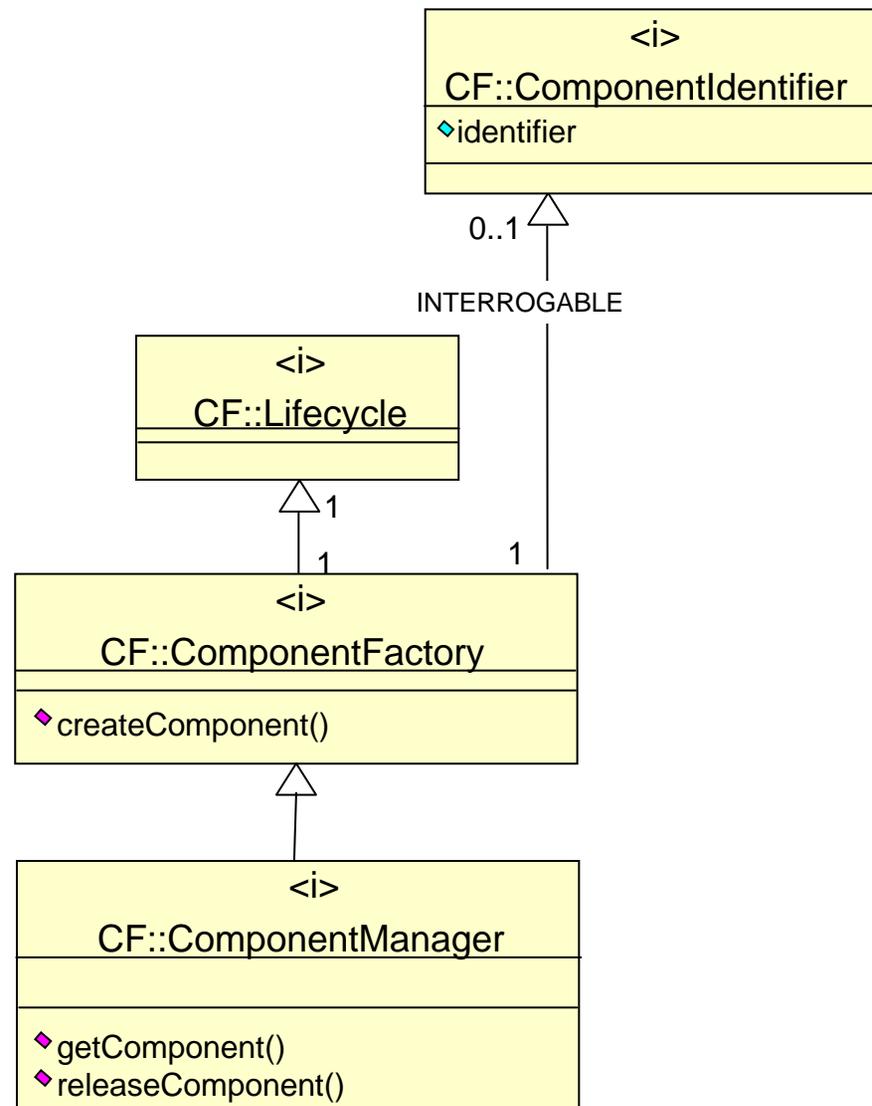


ResourceFactory Interface Refactoring



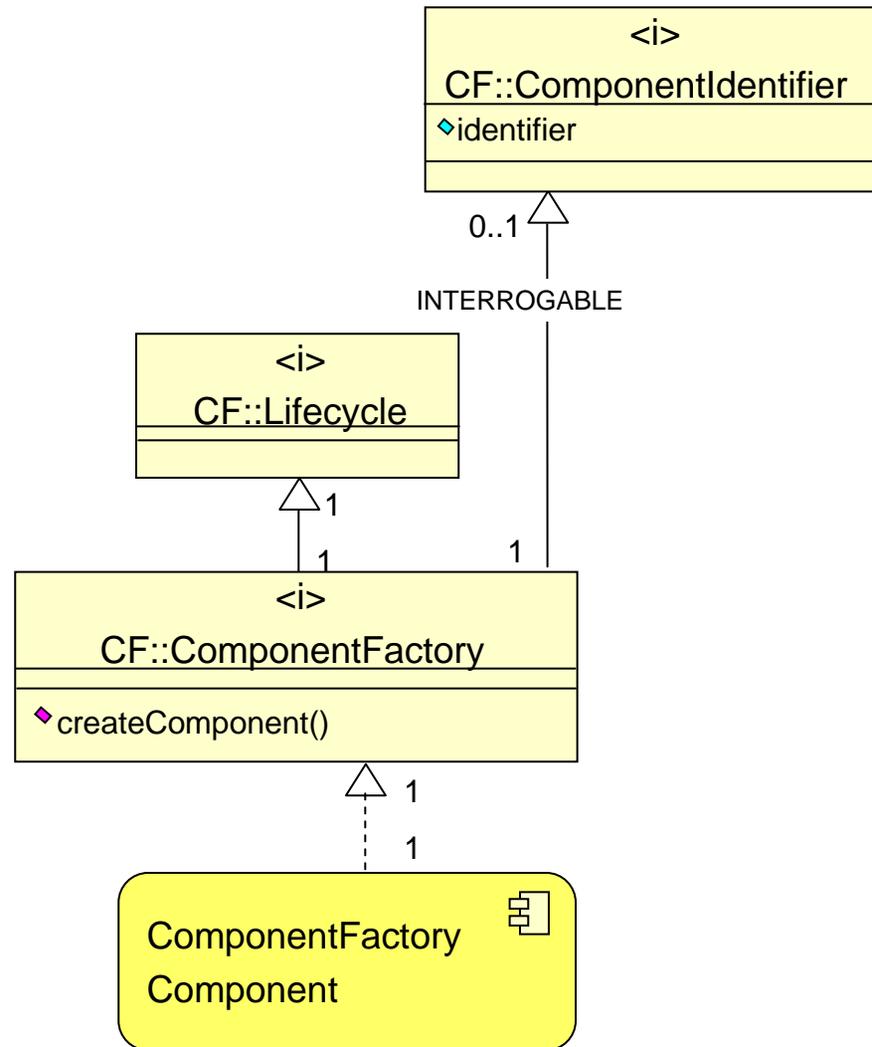


ResourceFactory Interface Refactoring, cont'd



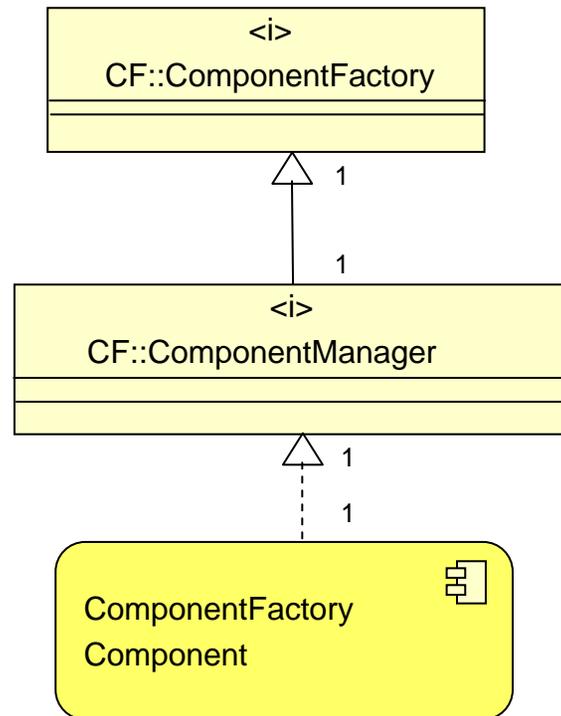


ComponentFactory Component



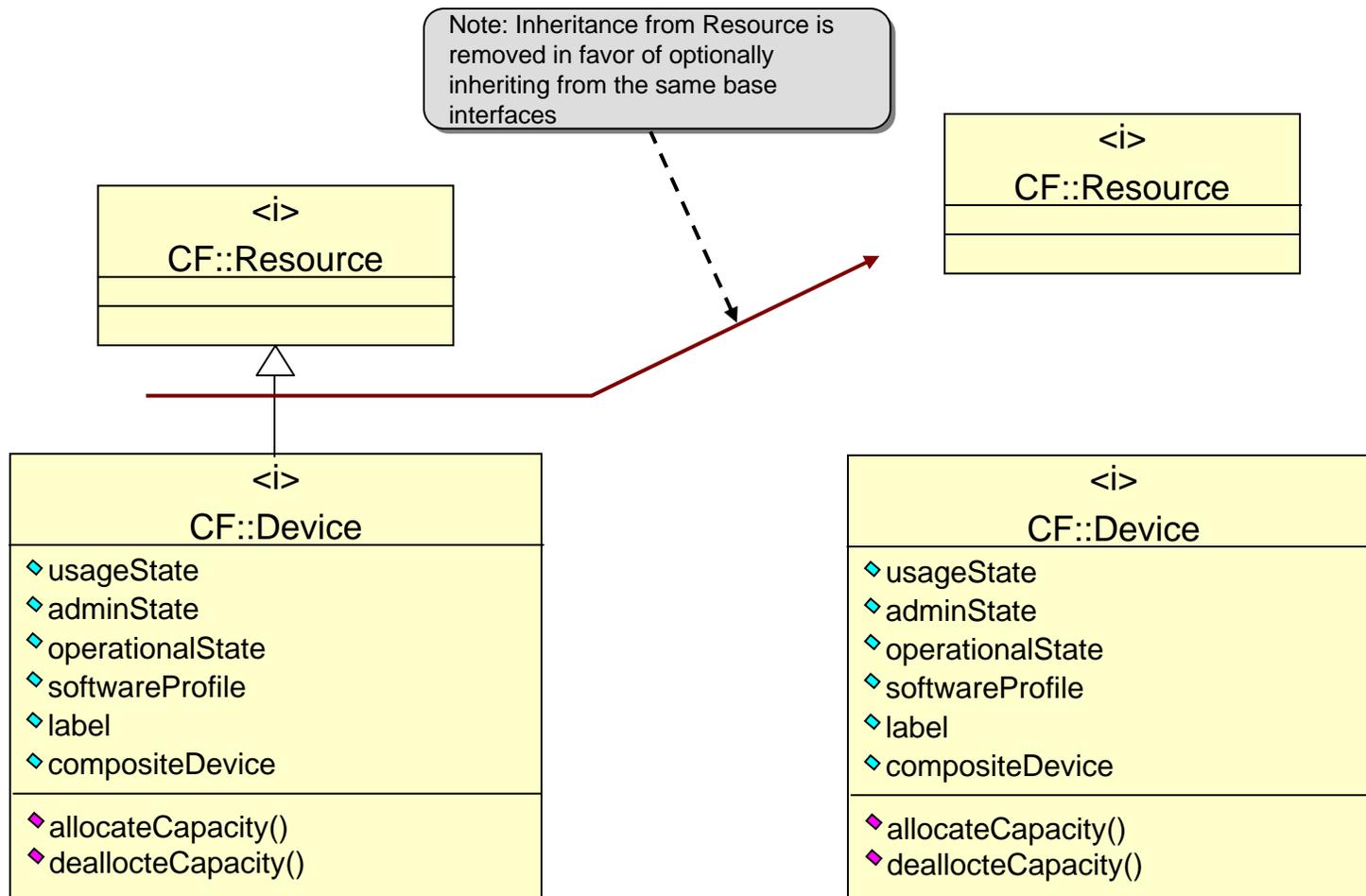


ComponentManager Component



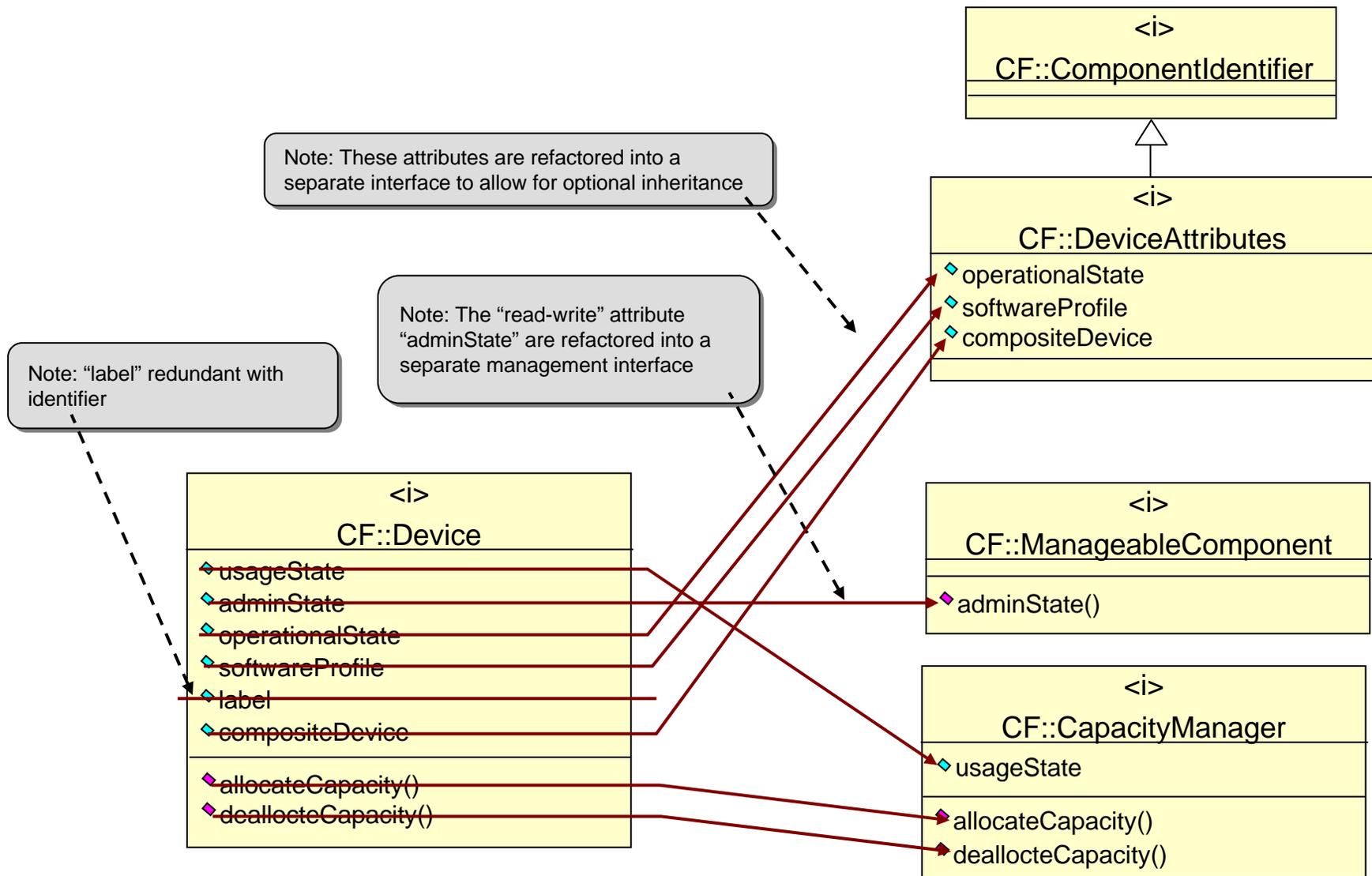


Device Interface Refactoring



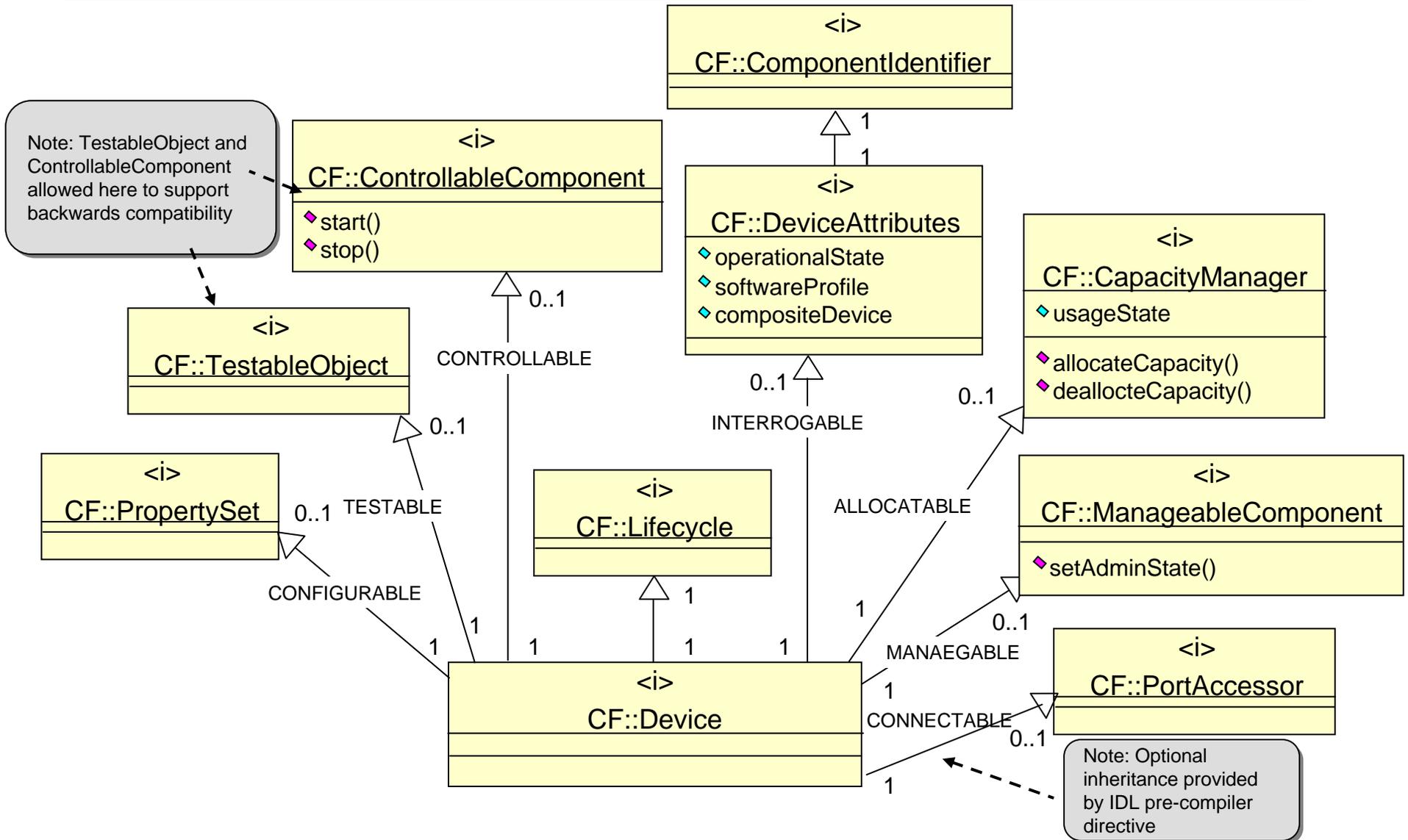


Device Interface Refactoring, cont'd



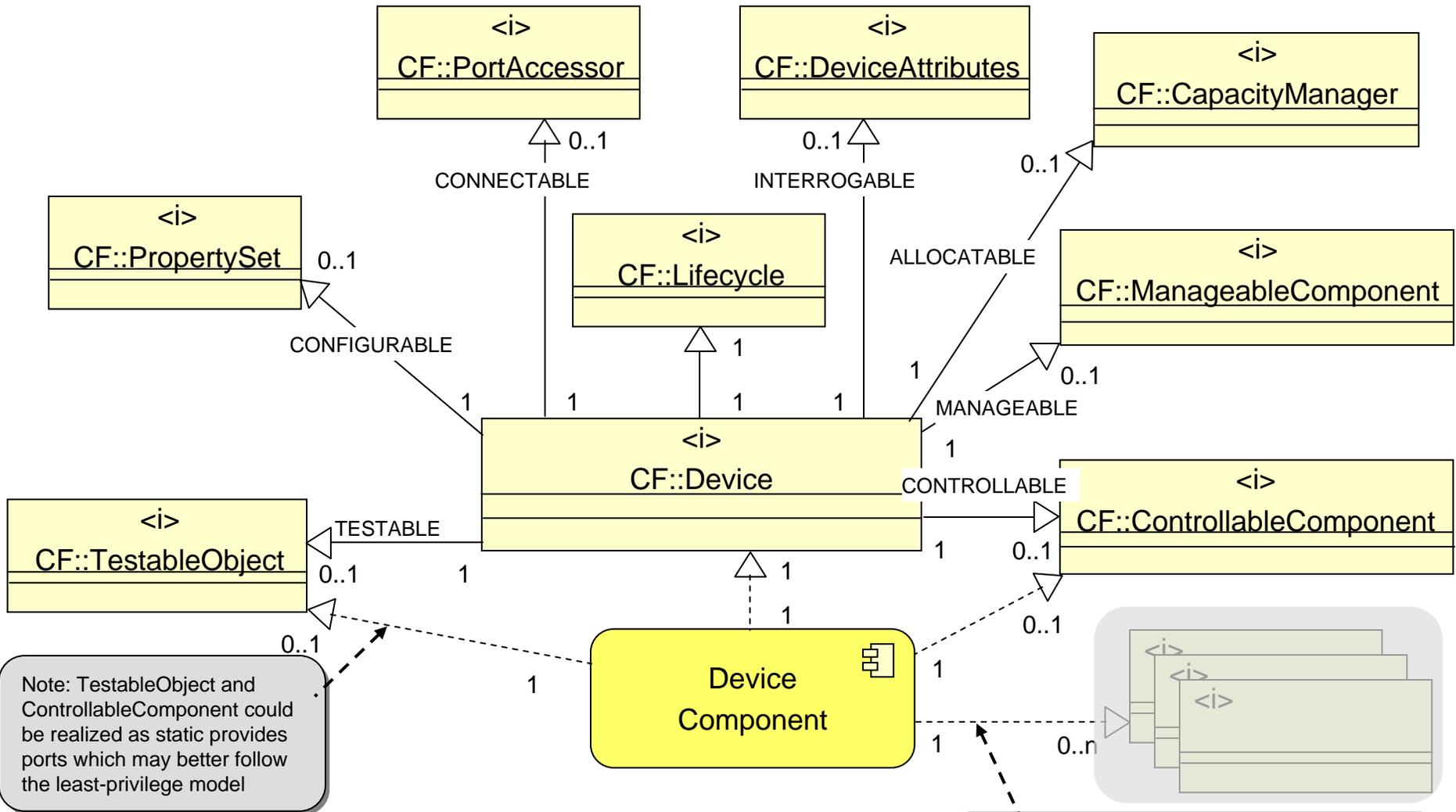


Device Interface Refactoring, cont'd





Device Component

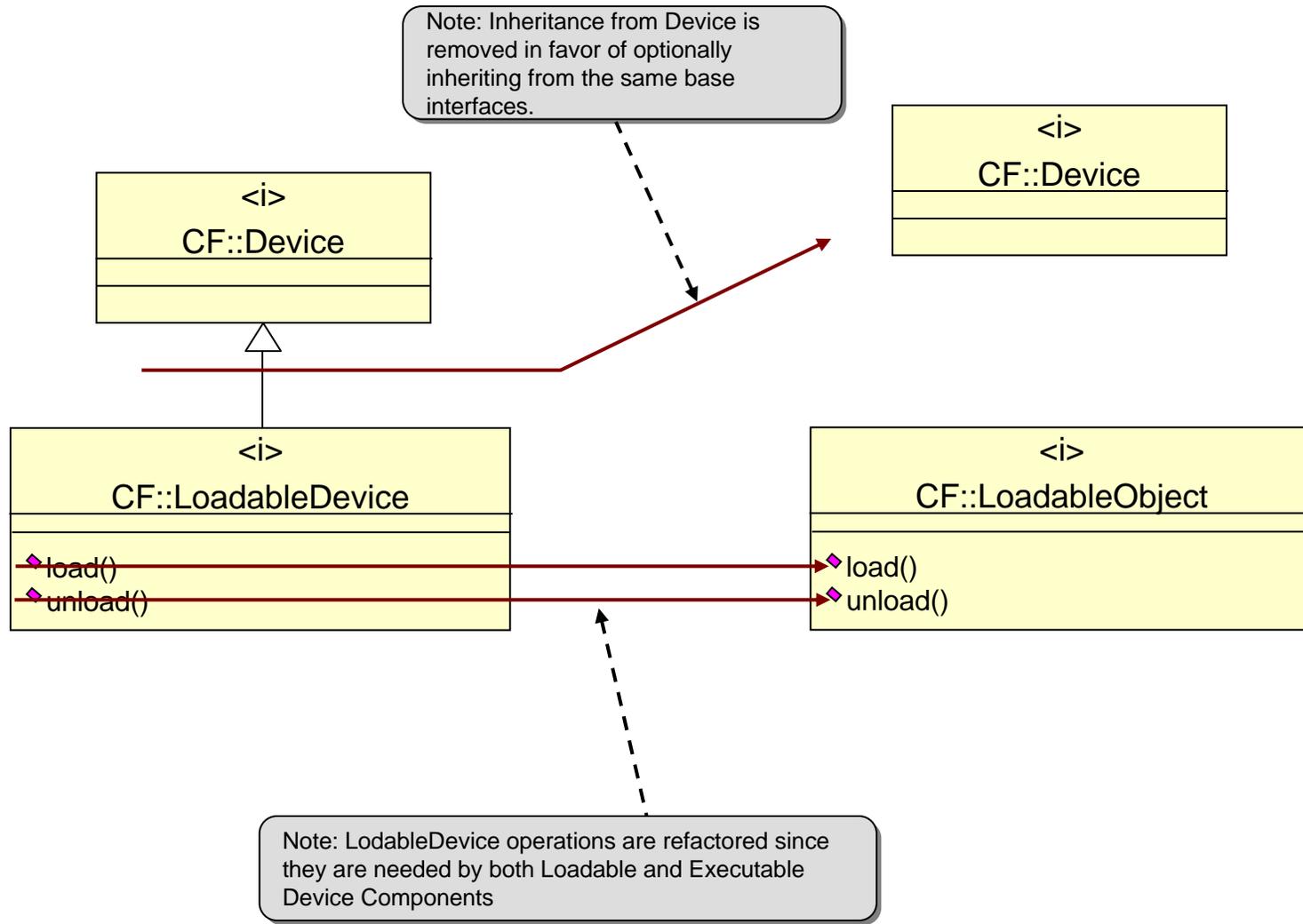


Note: TestableObject and ControllableComponent could be realized as static provides ports which may better follow the least-privilege model

Note: Device Components can also realize 0..n provides ports, which can support other standardized or customized interfaces

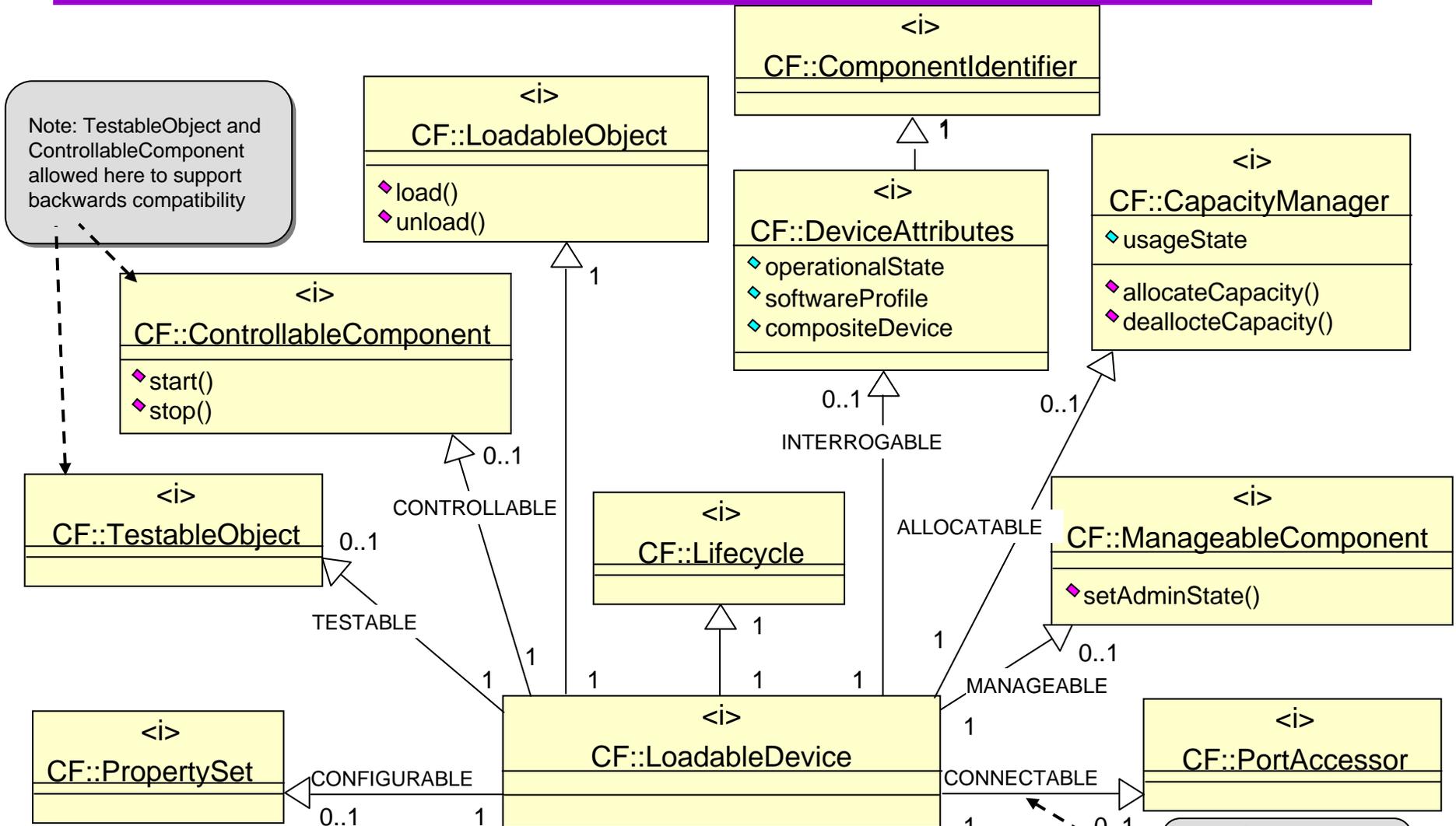


LoadableDevice Interface Refactoring, cont'd





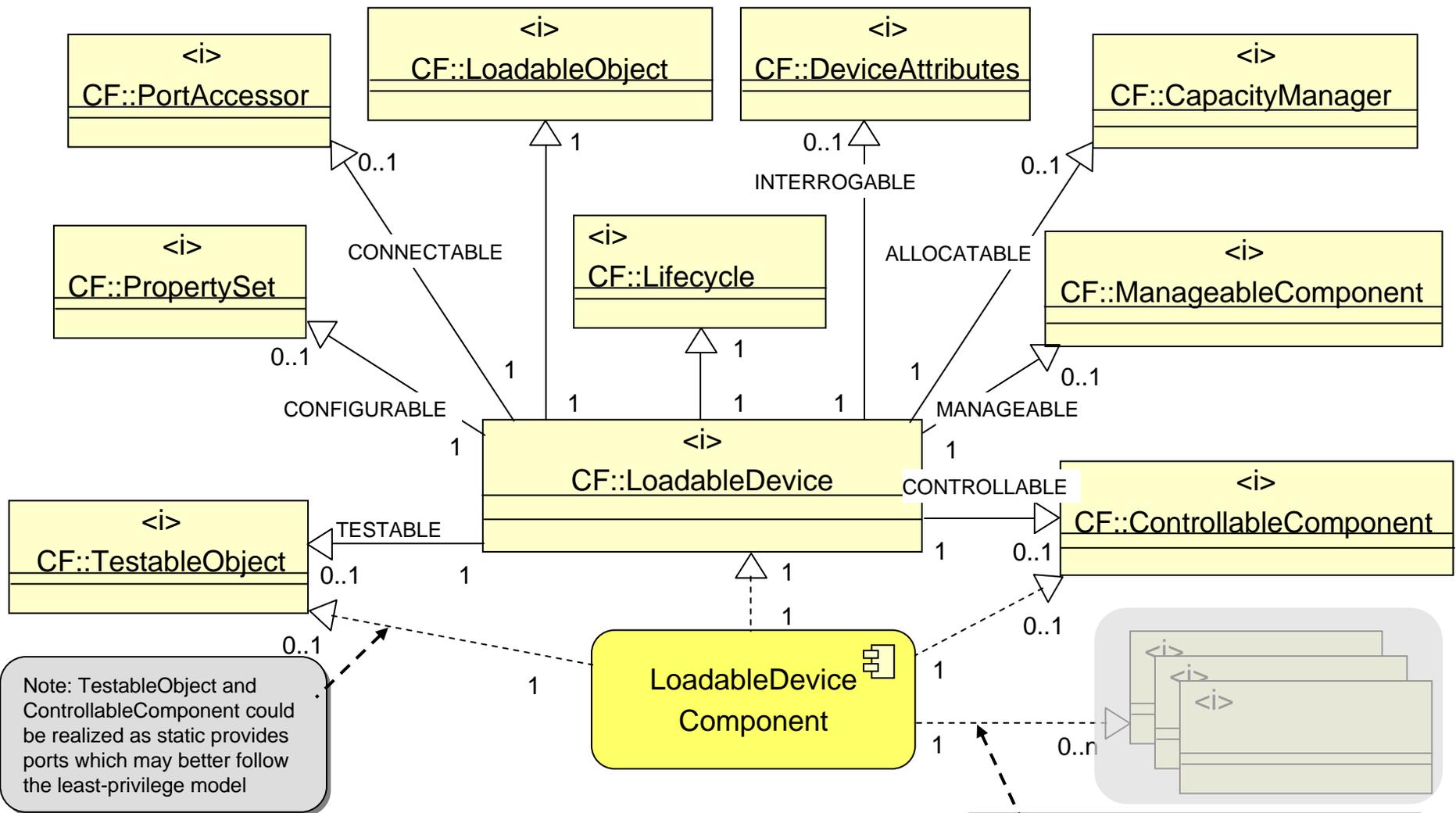
LoadableDevice Interface Refactoring, cont'd



Note: Optional inheritance provided by IDL pre-compiler directive.



LoadableDevice Component

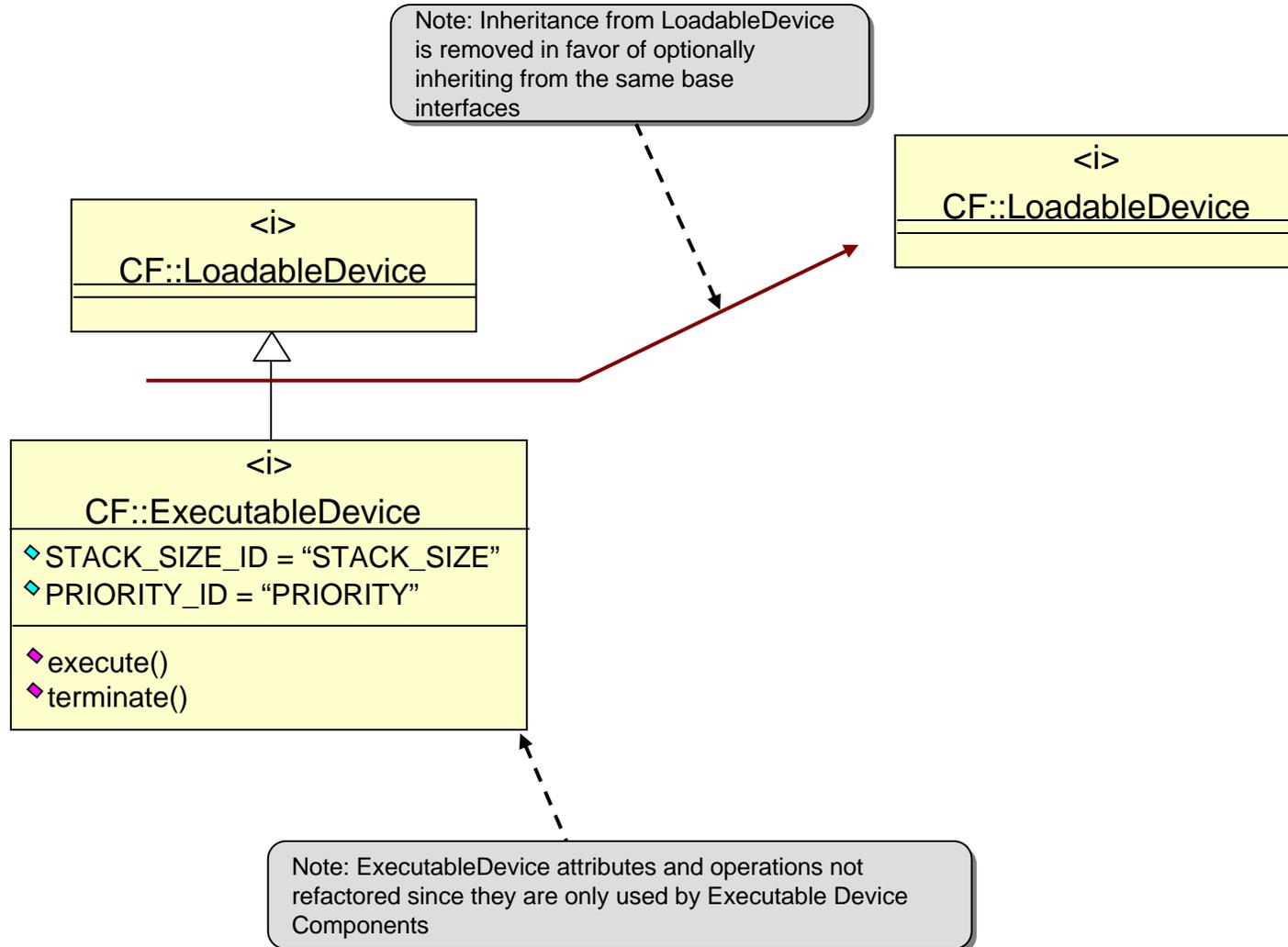


Note: TestableObject and ControllableComponent could be realized as static provides ports which may better follow the least-privilege model

Note: LoadableDevice Components can also realize 0..n provides ports, which can support other standardized or customized interfaces

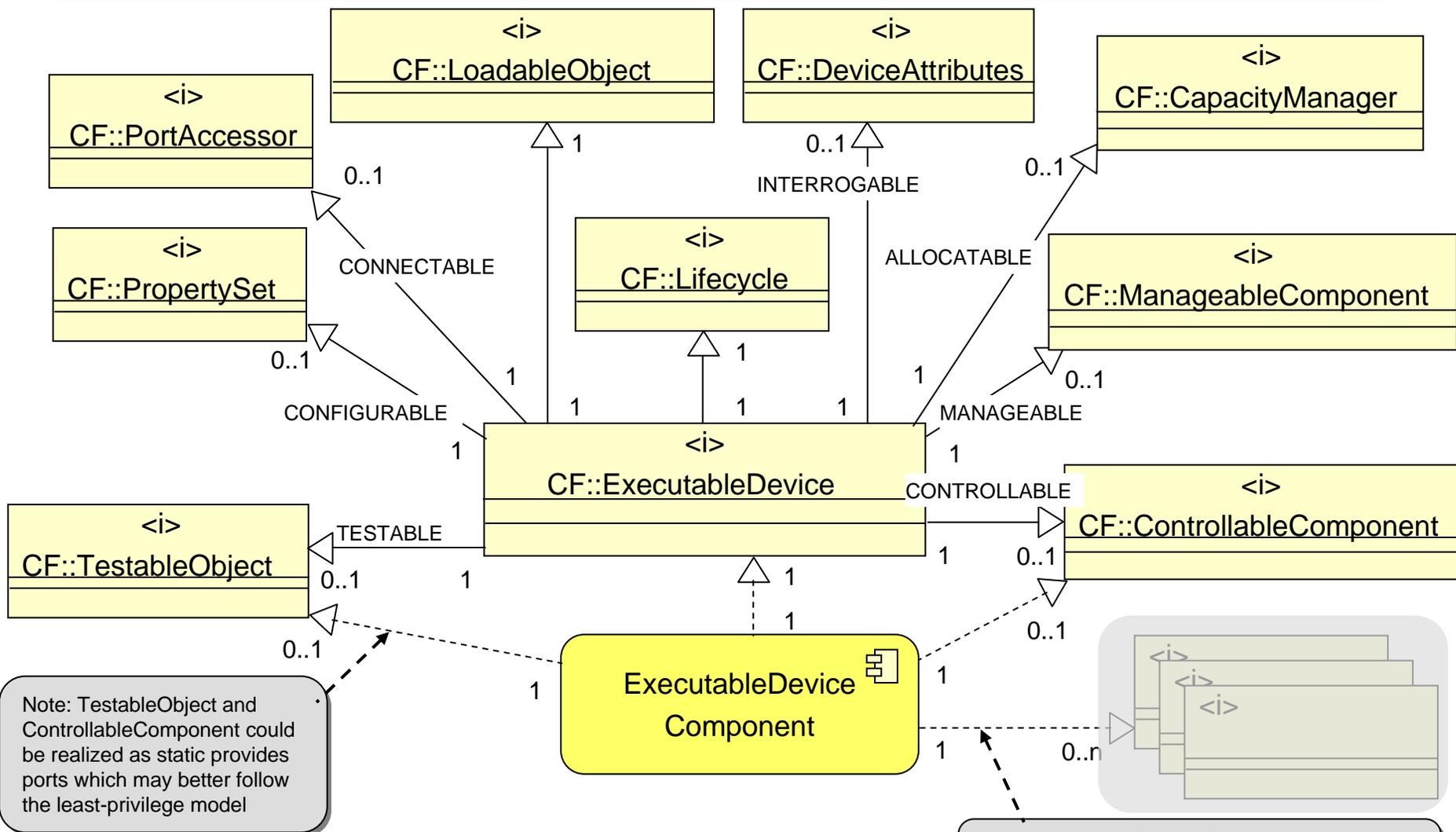


ExecutableDevice Interface Refactoring, cont'd





ExecutableDevice Component





Interface Details

```
interface Lifecycle {
    void initialize();
    void releaseObject();
};

interface PropertySet {
    void configure( in CF::Properties configProperties );
    void query( inout CF::Properties configProperties );
};

interface TestableObject{
    void runTest( in unsigned long testid,
                 inout CF::Properties testValues );
};

interface ControllableComponent {
    readonly attribute boolean started;
    void start();
    void stop();
};

interface ComponentIdentifier {
    readonly attribute string identifier;
};
```

•Note: Green text is to identify areas where, even when using the V222_COMPAT flag, there are a few other minor changes that go against 100% backwards compatibility



Interface Details, cont'd

•Note: V222_COMPAT flag can be defined to allow the translated IDL to be as compatible as possible with the v2.2.2 version

```
interface Resource : Lifecycle
#if defined(INTERROGABLE) || defined (V222_COMPAT) ,ComponentIdentifie #endif
#if defined(CONNECTABLE ) || defined (V222_COMPAT) ,PortAccessor #endif
#if defined(CONFIGURABLE ) || defined (V222_COMPAT) ,PropertySet #endif
#if defined(TESTABLE ) || defined (V222_COMPAT) ,TestableObject #endif
#if defined(CONTROLLABLE ) || defined (V222_COMPAT) ,ControllableComponent#endif
{};

interface ComponentFactory : Lifecycle
#if defined(INTERROGABLE) || defined (V222_COMPAT) ,ComponentIdentifier #endif
{
CF::ComponentType createComponent ( in string componentId,
in CF::Properties qualifiers )
};

interface ComponentManager : ComponentFactory
{
boolean releaseComponent ( in string componentId);
CF::ComponentType getComponent (in string componentId);
};
```

•Note: ComponentType is included here to be consistent with the Deployment Optimization Task.



Interface Details, cont'd

```
interface ManageableComponent {
    attribute CF::Device::AdminType adminState;
};
```

```
interface CapacityManager {
    readonly attribute CF::Device::UsageType usageState;
    boolean allocateCapacity( in CF::Properties capacities );
    void deallocateCapacity( in CF::Properties capacities );
};
```

```
interface DeviceAttributes : ComponentIdentifier {
    readonly attribute CF::Device::OperationalType operationalType;
    readonly attribute string softwareProfile;
    readonly attribute CF::AggregateDevice compositeDevice;
};
```

•Note: **Inheritance from Resource is broken** in favor of optionally inheriting from the same base interfaces.

```
interface Device : Lifecycle
#if defined( INTERROGATABLE ) || defined( V222_COMPAT ) ,DeviceAttributes      #endif
#if defined( CONNECTABLE ) || defined( V222_COMPAT ) ,PortAccessor          #endif
#if defined( CONFIGURABLE ) || defined( V222_COMPAT ) ,PropertySet          #endif
#if defined( TESTABLE ) || defined( V222_COMPAT ) ,TestableObject           #endif
#if defined( CONTROLLABLE ) || defined( V222_COMPAT ) ,ControllableComponent #endif
#if defined( MANAGEABLE ) || defined( V222_COMPAT ) ,ManageableComponent     #endif
#if defined( ALLOCATABLE ) || defined( V222_COMPAT ) ,CapacityManager         #endif
{};
```



Interface Details, cont'd

```
interface LoadableObject {  
    void load( in CF::FileSystem fs, in string fileName, in CF::LoadableDevice::LoadType loadKind );  
    void unload( in string fileName );  
};
```

•Note: **Inheritance from Device is broken** in favor of optionally inheriting from the same base interfaces.

```
interface LoadableDevice : Lifecycle, LoadableObject  
#if defined( INTERROGATABLE ) || defined( V222_COMPAT ) ,DeviceAttributes #endif  
#if defined( CONNECTABLE ) || defined( V222_COMPAT ) ,PortAccessor #endif  
#if defined( CONFIGURABLE ) || defined( V222_COMPAT ) ,PropertySet #endif  
#if defined( TESTABLE ) || defined( V222_COMPAT ) ,TestableObject #endif  
#if defined( CONTROLLABLE ) || defined( V222_COMPAT ) ,ControllableComponent #endif  
#if defined( MANAGEABLE ) || defined( V222_COMPAT ) ,ManageableComponent #endif  
#if defined( ALLOCATABLE ) || defined( V222_COMPAT ) ,CapacityManager #endif  
{};
```

•Note: **Inheritance from LoadableDevice is broken** in favor of optionally inheriting from the same base interfaces.

```
interface ExecutableDevice : Lifecycle, LoadableObject  
#if defined( INTERROGATABLE ) || defined( V222_COMPAT ) ,DeviceAttributes #endif  
#if defined( CONNECTABLE ) || defined( V222_COMPAT ) ,PortAccessor #endif  
#if defined( CONFIGURABLE ) || defined( V222_COMPAT ) ,PropertySet #endif  
#if defined( TESTABLE ) || defined( V222_COMPAT ) ,TestableObject #endif  
#if defined( CONTROLLABLE ) || defined( V222_COMPAT ) ,ControllableComponent #endif  
#if defined( MANAGEABLE ) || defined( V222_COMPAT ) ,ManageableComponent #endif  
#if defined( ALLOCATABLE ) || defined( V222_COMPAT ) ,CapacityManager #endif  
{  
    const string STACK_SIZE_ID = "STACK_SIZE"  
    const string PRIORITY_ID = "PRIORITY"  
    CF::ExecutableDevice::ProcessID_Type execute( in string name,  
                                                in CF::Properties options,  
                                                in CF::Properties parameters );  
    void terminate( in CF::ExecutableDevice::ProcessID_Type processId );  
};
```