Service Component Architecture Approach to Security, Transactions and Policy

Mike Rowley—BEA
Chris Sharp, Mike Edwards—IBM

SCA Collaboration Team
www.osoa.org

TS-3765

Goal of This Talk

Learn how to assemble Service-Oriented business solutions with Security and Transaction capabilities using Service Component Architecture

Agenda

Overview of SCA

SCA Approach to Security, Transactions

- Policy and Profiles
- Attaching Profiles to Components and Interactions
- Security Policies
- Transaction Policies

The SCA Collaboration

- Evolution of the Specification

The Apache “Tuscany” Project

The Eclipse SOA Tools Project

SCA in a Nutshell

- SCA models the “A” in SOA—
  for systems composed of reusable services
  - Model for service-based system
    - Service construction
    - Service assembly
    - Deployment
  - Heterogeneity—supports components from:
    - Multiple languages
    - Multiple container technologies
    - Multiple service access methods
- 0.9 Level specification published in November 2005

SCA—High-Level Points

- Unified declarative model describing service assemblies
    - Dependency resolution and configuration
    - Declarative policies for infrastructure services
        - Security, transactions, reliable messaging
- Business-level model for implementing services
    - Service components with service interfaces
    - No technical APIs like JDBC™ software, Java CA, Java Message Service (JMS API),…
- Binding model for multiple access methods
    - WSDL, SOAP over HTTP
    - But also: JMS API/messaging, Java Remote Method Invocation/IIOP…
    - Java-based interfaces are good, as are WSDL portTypes

Service Assembly Model

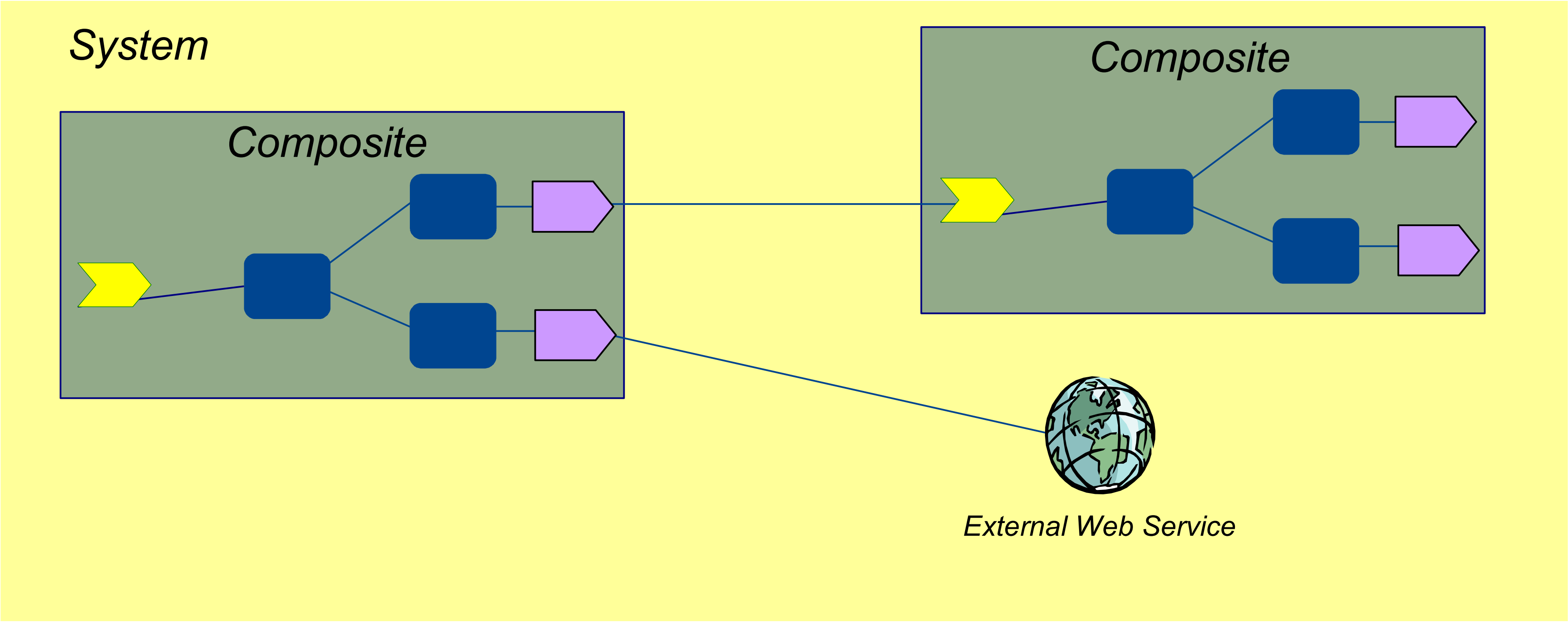
- Unified, language-independent way to expose implementations as services
  - Java technology, BPEL, PHP, other languages (including .NET)
- Technology-independent modelling and composition of service networks
  - Service dependencies
  - Resolution through wiring
- Facilities for dynamic service configuration
  - Properties/Protocols/Qualities of Service
  - Profiles
- Design time and deployment time configuration

Assembly Model Concepts

- Design Time Assembly
  - Composite
  - Implementation
  - Component
  - Service
  - Reference
  - Wire
- Deployment Time Assembly
  - System
  - Subsystem and composites

Service Assembly Model

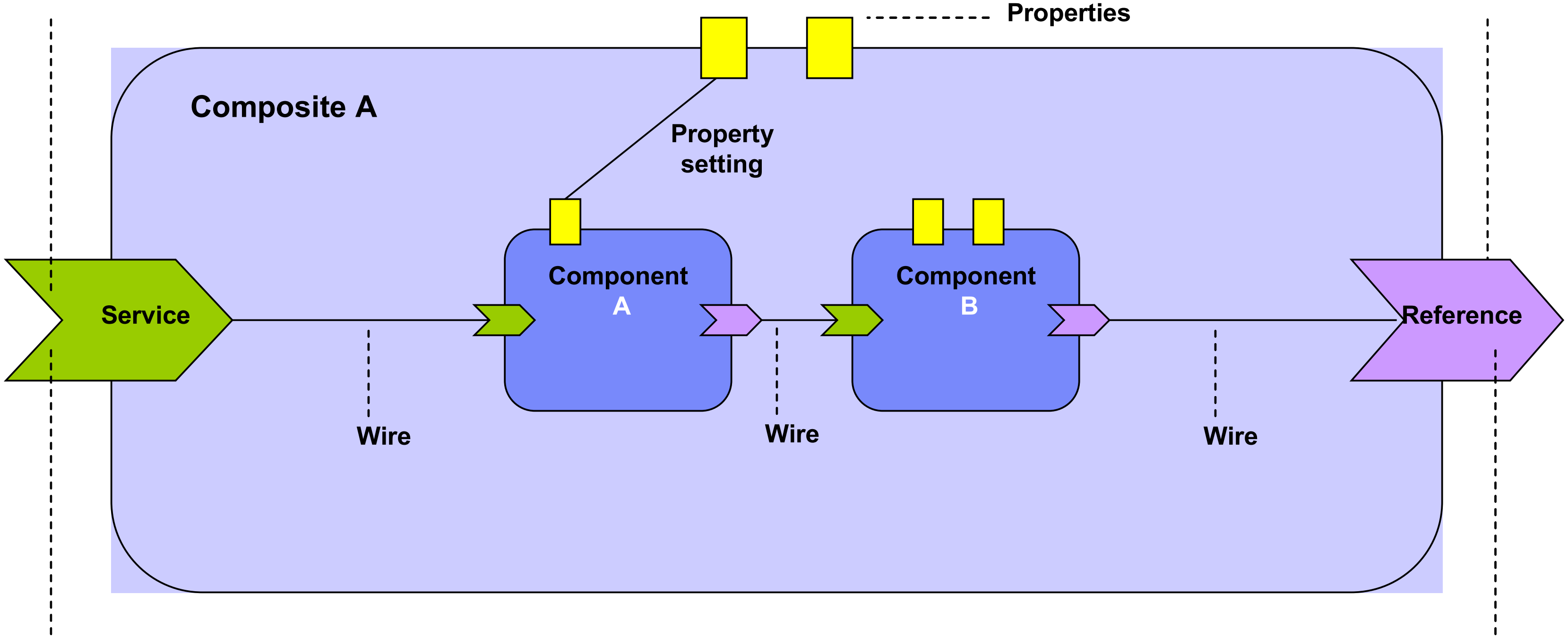
- Model for assembling tightly coupled code (Composites)
- Model for assembling loosely coupled services (Systems)



Composite

Service
- Java interface
- WSDL PortType

Reference
- Java interface
- WSDL PortType



Binding
Web Service
SCA
JCA
JMS
SLSB
…

Binding
Web Service
SCA
JCA
JMS
SLSB
…

Composite

- Assembly of services developed and deployed together
- Contains
  - Service implementations organized as components
  - Required services as references
  - Public services
  - Wires connect components, services, and references
- Used as implementation of components at next higher layer

Implementations

- Basic elements of business function
- Support for different implementation technologies
  - e.g., Java technology, BPEL, C++, PHP, … implementation type extensibility
  - A composite is also an implementation
- Provides business function via one or more services
- Uses other services through service references
- Service and references typed by interfaces
  - Remotable, bi-directional, companion callback interface
- Scoped
  - Runtime managed state and routing

Component

- Configured instance of implementation within a Composite
  - More than one component can use same implementation
- Provides and consumes services
- Sets properties
- Sets service references by wiring them to services
  - Wiring to services provided by other components or by references of the composite

Local and Remotable Services

Local Service

- Typed with local interface, e.g., Java-based interface with no @Remotable annotation
- Not addressable outside boundaries of module
- Fine-grained, tightly coupled interfaces
- Parameters and return values by-reference

Remotable Service

- Typed with remotable interface, e.g. , Java-based interface that has @Remotable annotation, or WSDL portType
- May be addressed outside scope of module if configured
- Coarse-grained, loosely coupled interfaces
- Parameters and return values by-value

Reference

Reference
StockQuote
Service

- Represent services that are external to the composite
  - Accessed by components within the composite like any other component service
- Use bindings to describe the access mechanism to the external service
  - e.g., Web service, stateless session EJB specification, JMS API, Java CA,…
  - Binding type extensibility
  - Overridable (no, may, must)
- Can define required characteristics through Policy

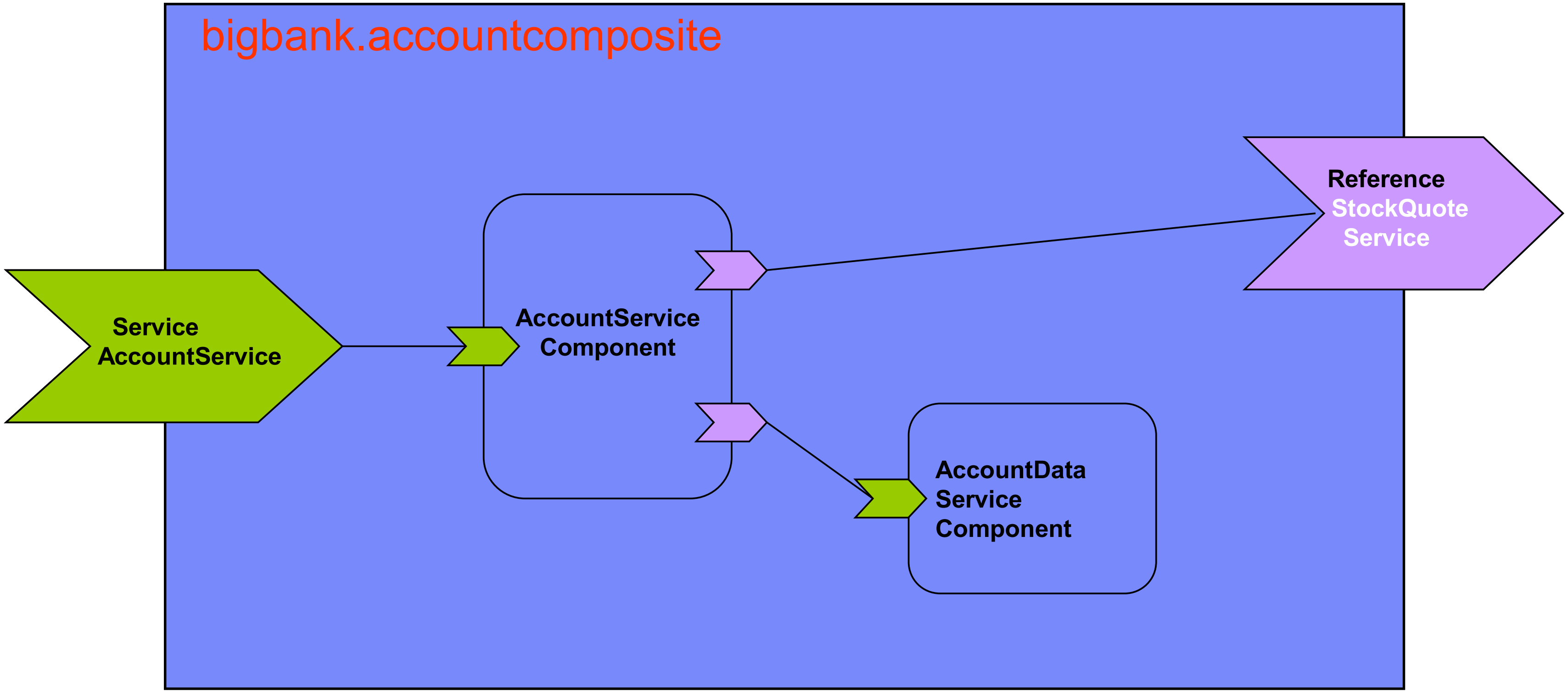
Service

- Used to publish services provided by composite, for use by external clients of the composite
  - Can be service provided by component or a reference
- Use bindings to describe the access mechanism that external clients have to use
  - Web service, stateless session EJB specification, JMS API…
  - Binding type extensibility
  - Always overridable
- Can define provided characteristics through Policy

SCA Interaction Model

- Synchronous and Asynchronous service relationships
- Scoping and Lifecycle
- Conversational services
- Asynchronous support
    - “Non-blocking” invocation
    - Asynchronous client to synchronous service
    - Callbacks

Example


bigbank.accountcomposite
Reference
StockQuote
Service
Service
AccountService
AccountService
Component
AccountData
Service
Component

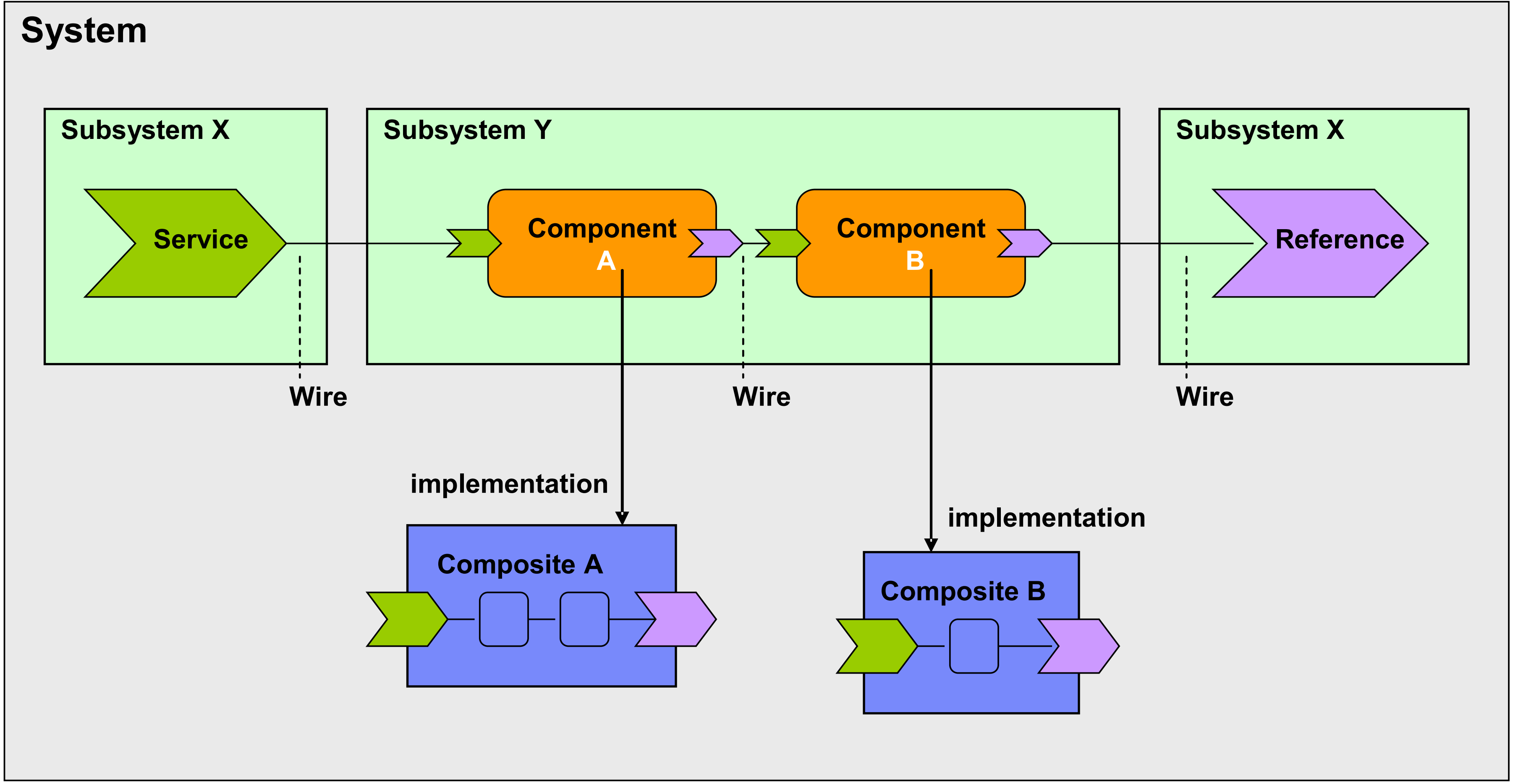
sca file for bigbank.accountcomposite

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://www.osoa.org/xmlns/sca/0.9"
           name="bigbank.accountcomposite" >

   <service name="AccountService">
      <interface.java interface="services.account.AccountService"/>
      <binding.ws port="http://www.bigbank.com/AccountService#
          wsdl.endpoint(AccountService/AccountServiceSOAP)"/>
      <reference>AccountServiceComponent</reference>
   </service>

   <component name="AccountServiceComponent">
      <implementation.java class="services.account.AccountServiceImpl"/>
      <property name=“currency”>EURO</property>
      <reference name="accountDataService" target="AccountDataServiceComponent"/>
      <reference name="stockQuoteService" target="StockQuoteService"/>
   </component>

   <component name="AccountDataServiceComponent">
      <implementation.java class="services.accountdata.AccountDataServiceImpl"/>
   </component>

   <reference name="StockQuoteService">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws port="http://www.quickstockquote.com/StockQuoteService#
          wsdl.endpoint(StockQuoteService/StockQuoteServiceSOAP)"/>
   </reference>
</composite>
```

Assembly Model Concepts

- Design Time Assembly
  - Composite
  - Implementation
  - Component
  - Service
  - Reference
  - Wire
- Deployment Time Assembly
  - System
  - Subsystem and composites

Subsystem Composites



System and Subsystem

- Composites deployed, configured into SCA system
  - SCA runtime—potentially distributed
- Composites, services, references, wires
  - Configured using SCA subsystems which are composites
  - May be partial rather than complete
- Subsystems make deployment simpler
  - Individual subsystems created, deployed independently
  - May contain only wires, components or externally provided services or references

Agenda

Overview of SCA

SCA Approach to Security, Transactions

- Policy and Profiles
- Attaching Profiles to Components and Interactions
- Security Policies
- Transaction Policies

The SCA Collaboration

- Evolution of the Specification

The Apache “Tuscany” Project

The Eclipse SOA Tools Project

Policies and Infrastructure Capabilities

- Infrastructure has many configurable capabilities
  - Security: Authentication and Authorization
  - Security: Privacy, Encryption, Non-Repudiation
  - Transactions, Reliable messaging, etc.
  - Complex sets of configurations across multiple domains of concern
- SCA abstracts out complexity with a declarative model
  - No implementation code impact
  - Simplify usage via declarative policy hints
  - Profiles bundle multiple capabilities—hide complexity
  - Simple to apply, modify

Policies, Profiles, and Quality of Service

- Framework consists of four main elements:
  - SCA policy hints
    - Represent a single abstract QoS intent
  - SCA profiles
    - Aggregates a set of abstract, cross-domain, QoS intents to represent an overall QoS
  - SCA policy sets
    - Represent a collection of concrete policies to realize an abstract QoS intent
  - WS-Policy
    - The syntax for concrete policies

Policy Hints

- Simple abstractions to provide the SCA developer with coarse-grained control
- Modelled as SimpleType enumerations
- e.g.,

```
<simpleType name="messageProtectionPolicy">
    <restriction base="string">
          <enumeration value="integrity" >
          <enumeration value="confidentiality">
    </restriction>
</simpleType>
```

```
<simpleType name=“authenticationPolicy">
    <restriction base="string">
          <enumeration value=“basic" >
          <enumeration value=“cert">
    </restriction>
</simpleType>
```

Profiles

- SCA defines core Profiles to represent common sets of QoS requirements
  - BasicWebServices
  - BasicSecurity
  - ReliableMessaging
- These aggregate a set of core Policy Hints with specific default values
  - Authentication
  - MessageProtection
  - ReliableDelivery
  - …

Profile Elements

- Profiles act as convenient short hand for combinations of policy hints

```
<element name=“Profile.BasicSecurity" type="sca:BasicSecurityProfile" substitutionGroup="sca:profile"/>
<complexType name=“BasicSecurityProfile">
    <complexContent>
      <extension base="sca:Profile">
        <attribute name="messageProtection" type="sca:messageProtectionPolicy“ default=“integrity” />
        <attribute name="authentication" type="sca:authenticationPolicy“ default=“basic” />
      </extension>
    </complexContent>
</complexType>
```

```
<element name=“Profile.RAMP" type="sca:BasicSecurityProfile" substitutionGroup="sca:profile"/>
<complexType name=“RAMP">
    <complexContent>
      <extension base="sca:BasicSecurityProfile">
        <attribute name=“reliable" type="sca:reliableMessagingPolicy“ default=“true” />
      </extension>
    </complexContent>
</complexType>
```

Using Profiles and Policy Hints

- Developer can just use a Profile in their assembly and pick up defaults

```
<sca:service>
   <sca:interface … />
   <sca:binding.WS />

   <sca:Profile.RAMP />
</sca:service>
```

- Or use the policy hints explicitly to add QoS intents and override defaults

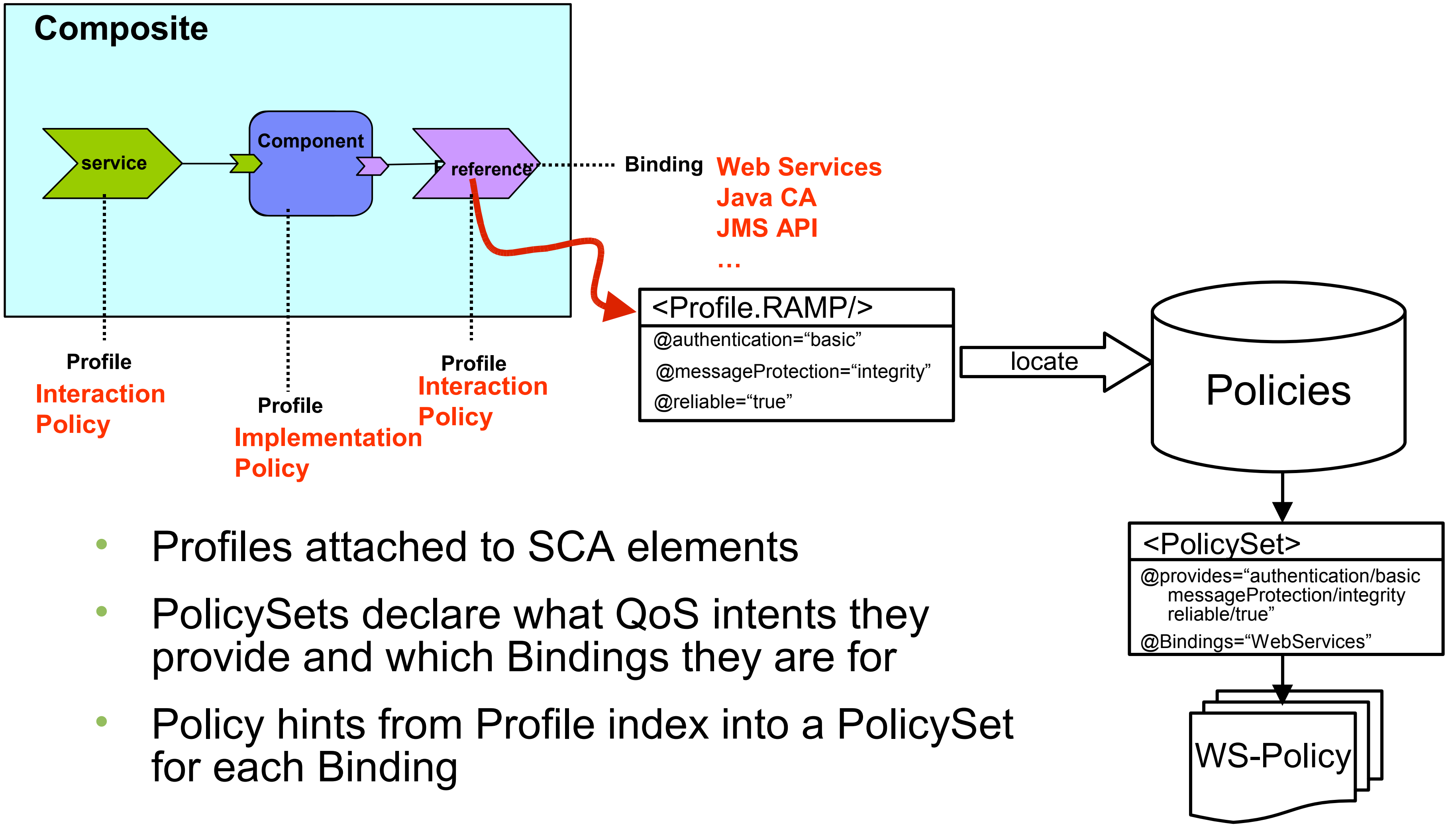
```
<sca:service>
   <sca:interface … />
   <sca:binding.WS />

   <sca:Profile.BasicSecurity authentication=“cert” reliable=“true” />
</sca:service>
```

- Or reference a PolicySet directly by its QName

PolicySet

- Provide concrete policies and attachment points
- Provides a mapping to abstract policy hints
- Declare support for one or more binding types
- Can be nested to build complex policy groupings

Attaching Profiles and Mapping to PolicySets



- Profiles attached to SCA elements
- PolicySets declare what QoS intents they provide and which Bindings they are for
- Policy hints from Profile index into a PolicySet for each Binding

PolicySet Example

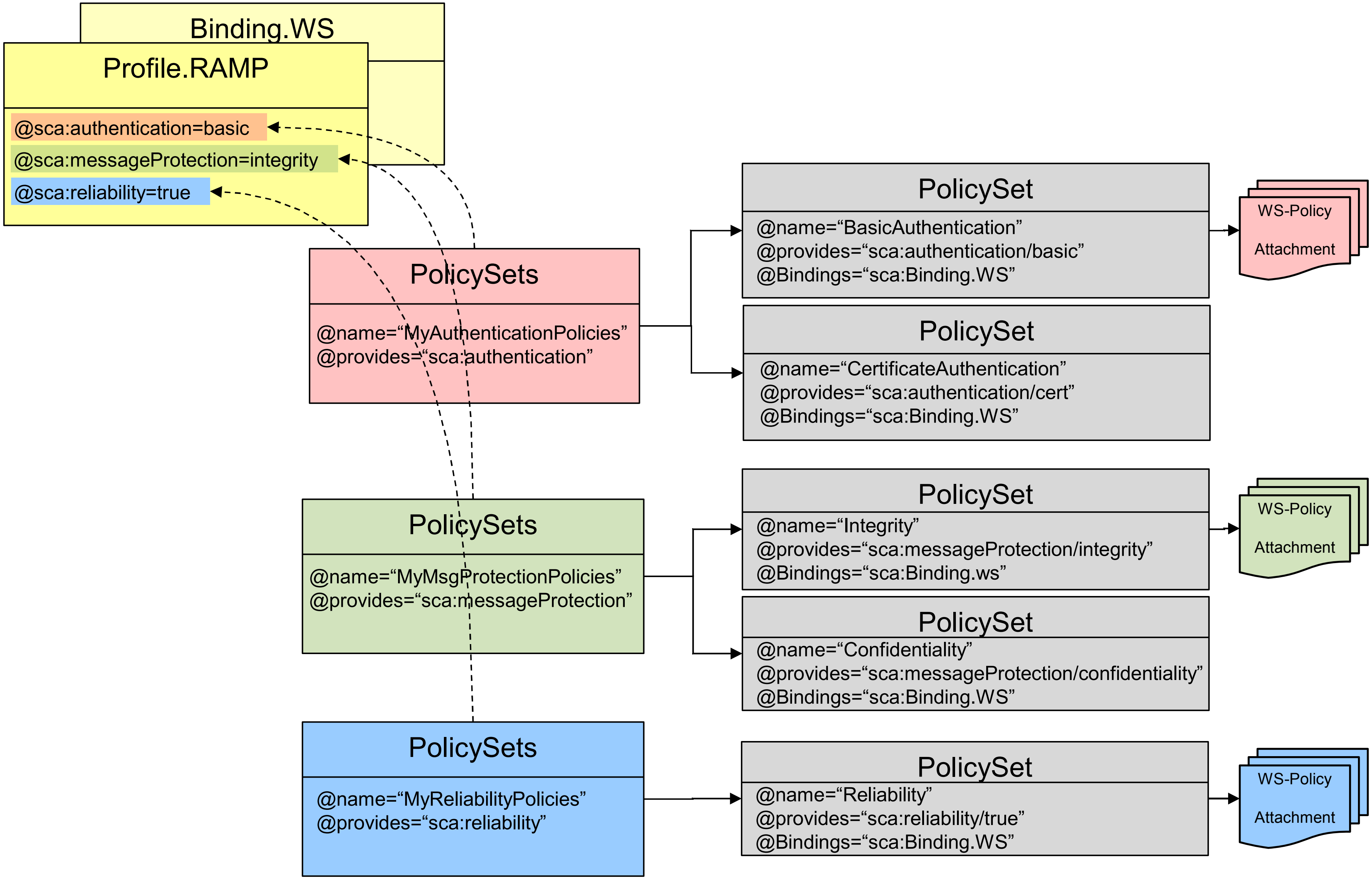
```
<sca:policySet name="BasicAuthSecurity"
   provides="sca:authentication/basic"
   bindings="sca:Binding.WS">
      <wsp:PolicyAttachment>
         <wsp:AppliesTo>
            <sca:ResourcePattern>
               *#wsdl.port(*/*)
            </sca:ResourcePattern>
         </wsp:AppliesTo>
         <wsp:PolicyReference
             URI="policies/security/2006/2/UsernameTokenPolicy.xml"/>
      </wsp:PolicyAttachment>
</sca:policySet>
```

Defines abstract policy hints which are implemented

Specifies binding to which PolicySet applies

Specifies location where Policy is applied

Reference to concrete Policy



Binding.WS

Profile.RAMP

@sca:authentication=basic

@sca:messageProtection=integrity

@sca:reliability=true

PolicySets

@name="MyAuthenticationPolicies"
@provides="sca:authentication"

PolicySets

@name="MyMsgProtectionPolicies"
@provides="sca:messageProtection"

PolicySets

@name="MyReliabilityPolicies"
@provides="sca:reliability"

PolicySet

@name="BasicAuthentication"
@provides="sca:authentication/basic"
@Bindings="sca:Binding.WS"

PolicySet

@name="CertificateAuthentication"
@provides="sca:authentication/cert"
@Bindings="sca:Binding.WS"

PolicySet

@name="Integrity"
@provides="sca:messageProtection/integrity"
@Bindings="sca:Binding.ws"

PolicySet

@name="Confidentiality"
@provides="sca:messageProtection/confidentiality"
@Bindings="sca:Binding.WS"

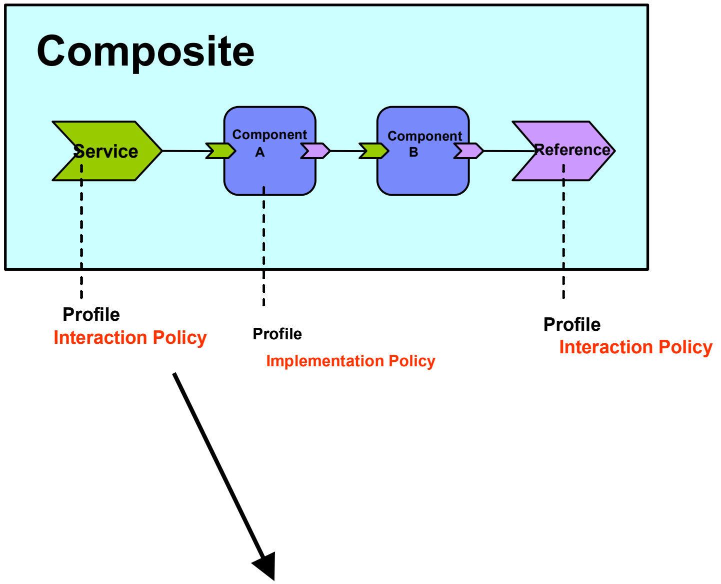
PolicySet

@name="Reliability"
@provides="sca:reliability/true"
@Bindings="sca:Binding.WS"

Interaction and Implementation Policies

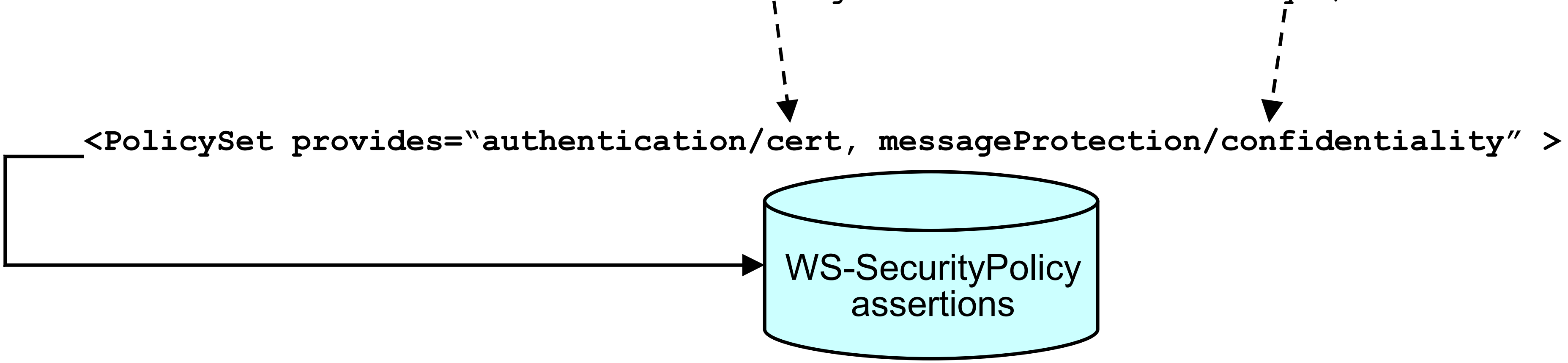
- Interaction policies affect the contract between a service requestor and a service provider
  - Things that affect the interaction between them, such as message contexts, wire formats, etc.
- Implementation policies affect the contract between a component and its container
  - Things that affect how the container should manage the component environment, such as transaction monitoring, access control, etc.

Security



- Interaction policy
  - @messageProtection=[confidentiality|integrity|both]
  - @authentication=[basic|cert|kerberos]
- Implementation policy
  - @AllowedRoles=[…]
  - @RunAs=[…]

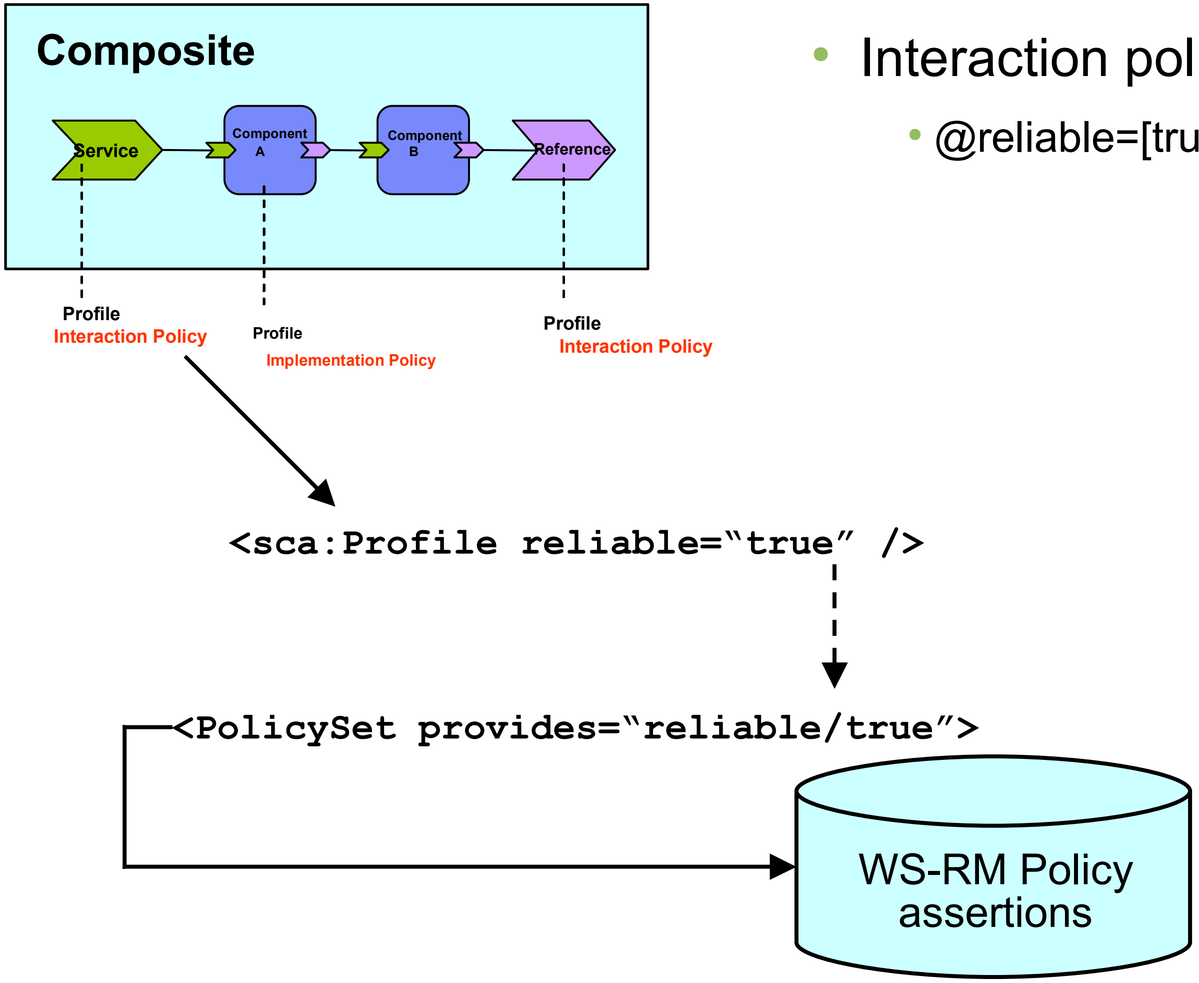
```
<sca:Profile authentication="cert" messageProtection="confidentiality" />
```



```
<PolicySet provides="authentication/cert, messageProtection/confidentiality" >
```

Reliability

- Interaction policy
  - @reliable=[true|false]



Composite

Profile
Interaction Policy

Profile
Implementation Policy

Profile
Interaction Policy

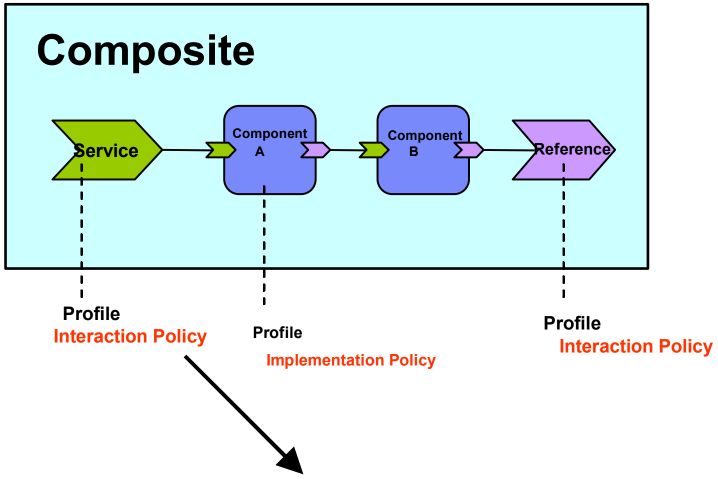
```
<sca:Profile reliable="true" />
```

```
<PolicySet provides="reliable/true">
```

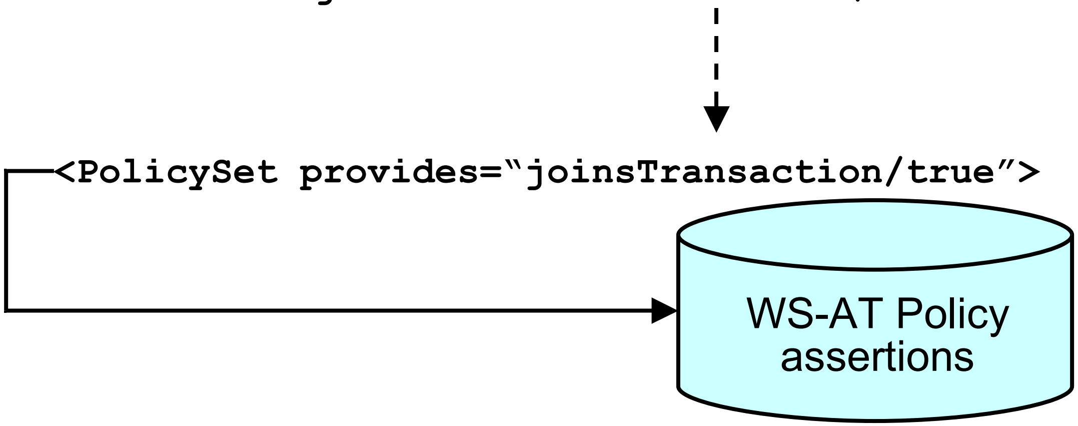
WS-RM Policy assertions

Transactionality

- Compensation model by default
- Interaction policy
  - @joinsTransaction=[true|false]
- Implementation policy
  - @transaction=[global|local|any]
  - @suspendTransaction=[true|false]



```
<sca:Profile joinsTransaction=“true” />
```



```
<PolicySet provides=“joinsTransaction/true”>
```

Agenda

- Overview of SCA
- SCA Approach to Security, Transactions
  - Policy and Profiles
  - Attaching Profiles to Components and Interactions
  - Security Policies
  - Transaction Policies
- **The SCA Collaboration**
  - Evolution of the Specification
- The Apache “Tuscany” Project
- The Eclipse SOA Tools Project

The SOA Collaboration

- SCA spec being evolved by group of collaborators
  - BEA, IBM, Interface21, IONA, Oracle, SAP, Sybase
  - Working towards 1.0 spec publication
    - Eventual submission to standards body
  - Comment and feedback welcome
  - Public website for specification soon
- Major work areas
  - Recursive SCA model
  - Policies and Bindings
  - SCA client and implementation model, including layering for Spring, EJB 3.0 specification, Celtix
  - BPEL implementation type

Agenda

Overview of SCA

SCA Approach to Security, Transactions

- Policy and Profiles
- Attaching Profiles to Components and Interactions
- Security Policies
- Transaction Policies

The SCA Collaboration

- Evolution of the Specification

The Apache “Tuscany” Project

The Eclipse SOA Tools Project

Apache “Tuscany” Project

- Aims to provide SOA programming runtime based on SCA and SDO
  - Currently has “incubator” status
- Java technology and C++ implementations today
  - Java technology implementation runs with Apache Tomcat + Axis
  - Aim to support more capable runtimes in future
    - e.g., Geronimo
  - C++ works with Apache Axis C++
  - Limited protocol support now—aim to expand
- Join the project!

Eclipse SOA Tools Project

- Aims to provide Eclipse-based tooling for SOA applications and systems
  - Based on SCA as model for solutions built using SOA
  - Target range of systems including SCA runtimes such as Tuscany

Useful Information

Contacts

- sharpc@uk.ibm.com
- mike_edwards@uk.ibm.com
- mrowley@bea.com

SCA, SDO specs and related material

- http://www.ibm.com/developerworks/webservices/library/specification/ws-sca/
- http://www.ibm.com/developerworks/webservices/library/specification/ws-sdo/

Apache “Tuscany” project

- http://incubator.apache.org/tuscany

Eclipse STP project

- http://www.eclipse.org/stp/

Summary

- SCA models systems built using a Service-Oriented Architecture
- Supports Service Implementation, Service Assembly
- Open to many kinds of service implementation
- Open to many types of service access
- Declarative policy and profile approach to application of Security and Transaction

Q&A

Service Component Architecture Approach to Security, Transactions and Policy

Mike Rowley—BEA
Chris Sharp, Mike Edwards—IBM

SCA Collaboration Team
www.osoa.org

TS-3765