


SCA Approach to Policy and Bindings

Dr Mike Edwards
IBM Hursley
mike_edwards@uk.ibm.com



All statements regarding IBM's future plans, direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Such statements do not represent a commitment of future availability, content, performance or function of any products or features.

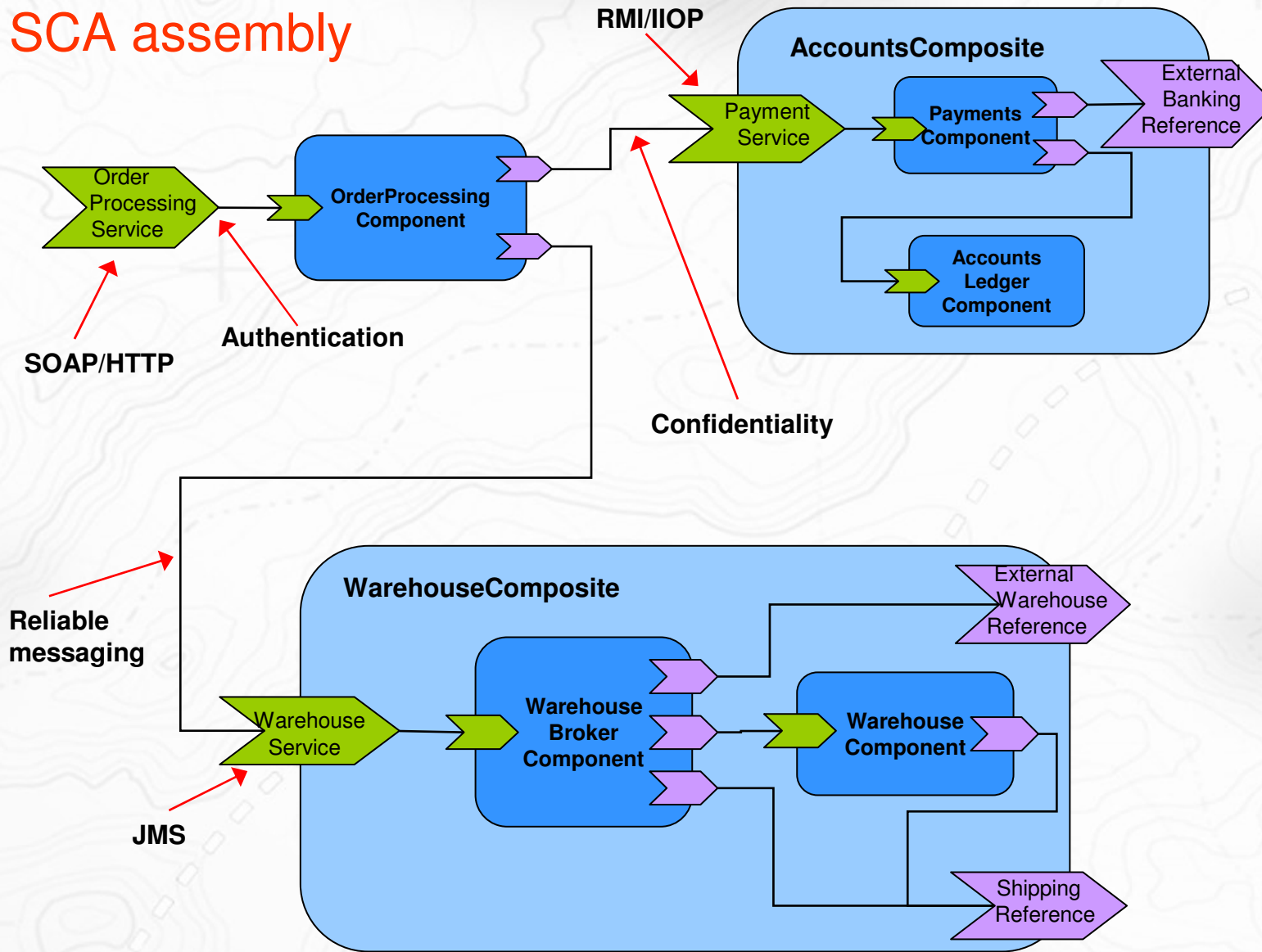
Agenda

- Bindings and Policy in the SCA Model
- Bindings in Detail
- Policy in Detail

Service Component Architecture (SCA): Simplified Programming Model for SOA

- **executable** model for:
- **building** service components
- **assembling** components into applications
- **deploying** to (distributed) runtime environments
 - Service components built from **new or existing code using SOA principles**
 - **vendor-neutral** – supported across the industry
 - **language-neutral** – components written using any language
 - **technology-neutral** – use any communication protocols and infrastructure to link components

SCA assembly



Key benefits of SCA

- **Loose Coupling**: components integrate without need to know how others are implemented
- **Flexibility**: components can easily be replaced by other components
- **Services** can be *easily* invoked either synchronously or asynchronously
- **Composition** of solutions: clearly described
- **Productivity**: easier to integrate components to form composite application
- **Heterogeneity**: multiple implementation languages, communication mechanisms
- **Declarative** application of infrastructure services
- **Simplification** for **all** developers, integrators and application deployers

SCA – Separation of Concerns

- SCA Components & Composites
 - define business logic
 - describe structure of business solution
- Components and composition separated from infrastructure
 - Communication mechanisms = **SCA Bindings**
 - Infrastructure facilities = **SCA Policy**
- SCA provides for **late binding** of Bindings and of Policies
 - **agility** and **flexibility**

Agenda

- SCA Bindings in Detail



SCA Bindings

- Specific to particular:
 - Access Method / Protocol / Transport
 - Serialization
 - Framework

- Apply to services and references
 - Typically added during deployment

- Extensible – more bindings can be added

- Currently defined bindings:
 - Web service binding
 - JMS binding
 - EJB Session Bean binding
 - “SCA Binding” (“default binding”)

Example Bindings

...

```
<service name="ExampleService" promote="ComponentA/serviceA">  
  <interface.java interface="com.foo.ServiceAInterface" />  
  <binding.ws port="ExampleService#wsdl.endpoint (ExampleService/ExampleServiceSOAP)" />  
</service>
```

← Web services

```
<service name="ExampleService" promote="ComponentA/serviceA">  
  <interface.java interface="com.foo.ServiceAInterface" />  
  <binding.jms>  
    <destination name="MyValueServiceQueue" create="always" />  
    <activationSpec name="MyValueServiceAS" create="always" />  
  </binding.jms>  
</service>
```

← JMS

```
<service name="ExampleService" promote="ComponentA/serviceA">  
  <interface.java interface="com.foo.ServiceAInterface" />  
  <binding.ejb uri="corbaname:rir:#ejb/EJBServiceAHome"  
    homeInterface="com.foo.EJBServiceAHome"  
    ejb-link-name="fooServiceEJB.jar#ServiceAComponent" />  
</service>
```

← EJB

...

The SCA default binding

- Explicit bindings only needed for services, references with connections outside the SCA domain
 - clients/providers are known to use specific protocols
- Within SCA domain, explicit bindings unnecessary
 - can use the SCA Binding (the default)
 - minimal configuration
 - allows SCA runtime to choose appropriate way to connect reference to service

Web Service Binding

- WSDL-based
- Supports WSDL 1.1 and WSDL 2.0
- Two ways to specify a WS binding
 - Reference an existing WSDL binding/service/endpoint/port element
 - Specify metadata to synthesize a SOAP-based WSDL binding

```
<binding.ws wsdlElement="xs:anyURI"?  
    wsdl:wsdlLocation="list of xs:anyURI"?>  
    <wsa:EndpointReference>...</wsa:EndpointReference>*</binding.ws>
```

Web Service Binding Examples

- Point to an existing WSDL document

```
<binding.ws  
  wsdlElement="http://www.stockquote.org/StockQuoteService#  
              wsdl.service (StockQuoteService) "  
  wsdl:wsdlLocation="http://www.stockquote.org/StockQuoteService  
                    http://www.stockquote.org/StockQuoteService.wsdl" />
```

- Synthesize WSDL

```
<binding.ws uri="http://www.sqs.com/StockQuoteService"/>
```

- Defaults to SOAP/HTTP binding
- Defaults to document/literal

JMS Binding

- Based on JMS API
- Allows JMS headers and user properties to be set on a per-operation basis
- Support for callbacks and conversations
- Default data binding and operation selection
- Uses `<binding.jms .../>`
- Example:

```
<binding.jms>  
  <destination name="StockQuoteServiceQueue" />  
  <connectionFactory name="StockQuoteServiceQCF" />  
  <resourceAdapter name="com.example.JMSRA" />  
</binding.jms>
```


EJB Session Bean Binding

- Support stateless and stateful session beans
- Stateful session bean implies ***conversational***
- Covers both exposure and consumption usecases
- Supports EJB 2.x and 3.0
- Uses <binding.ejb .../>
- Example

```
<binding.ejb uri="corbaname:rir:#ejb/JobBankServiceHome"  
  homeInterface="com.app.jobbank.JobBankServiceHome"  
  ejb-link-name="jobbankEJB.jar#JobBankComponent"/>
```

Agenda

- SCA Policies



Why Policy is Important for SCA

- Policy provides flexibility
 - a component can be used in different configurations with different QoS requirements
 - with different bindings on its services and references.

- Policy complexity tamed
 - by starting with simple, relatively abstract requirements
 - bound *later* to one or more concrete realizations
 - complexity encapsulated in concrete realizations

Policies and Infrastructure Capabilities

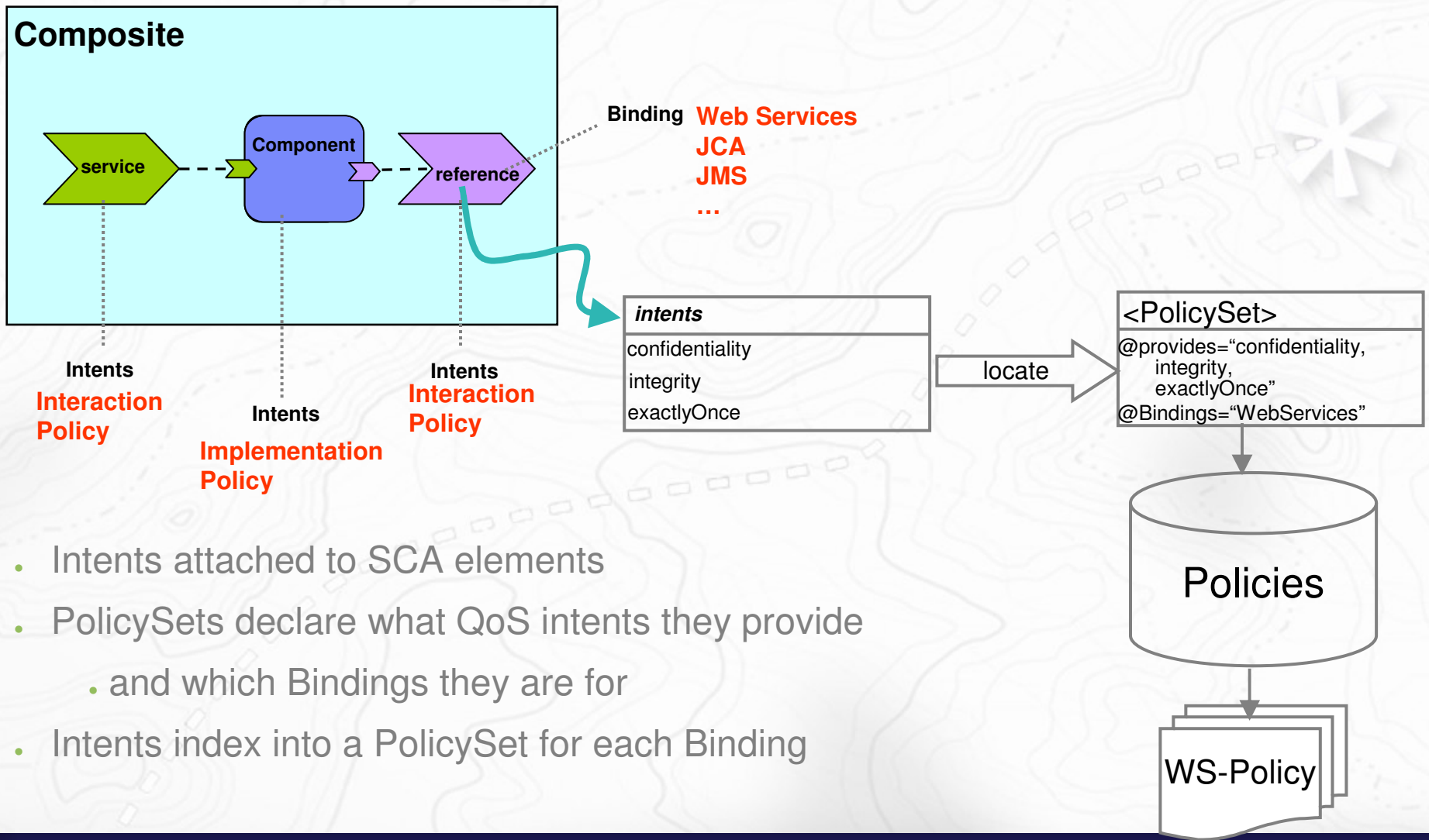
- **Infrastructure** has many configurable capabilities
 - Security: Authentication and Authorization
 - Security: Privacy, Encryption, Non-Repudiation
 - Transactions, Reliable messaging, etc.
 - Complex sets of configurations across multiple domains of concern

- SCA abstracts out complexity with a **declarative model**
 - no implementation code impact
 - simplify usage via declarative **policy intents**
 - simple to apply, modify
 - complex details held in **PolicySets**

Policies, Profiles and Quality of Service

- Framework consists of:
 - SCA policy *intent*
 - each represents single *abstract* QoS requirement - eg. *integrity*
 - may be *qualified* – eg. *integrity.message*
 - effectively *constrain* binding/policy set combinations
 - SCA *policy sets*
 - collection of *concrete* policies to realize abstract QoS intent
 - apply to specific binding types or implementation types
 - binding / implementation may provide policies intrinsically
 - WS-Policy
 - syntax for concrete policies in policy sets
 - other forms of syntax possible...

Attaching Profiles and mapping to PolicySets



- Intents attached to SCA elements
- PolicySets declare what QoS intents they provide
 - and which Bindings they are for
- Intents index into a PolicySet for each Binding

Interaction and Implementation Policies

- **Interaction policies** affect contract between service requestor and service provider
 - things that affect interaction between them, such as message contexts, wire formats, etc.
 - eg. authentication, confidentiality, integrity
 - eg. atLeastOnce, ordered
- **Implementation policies** affect contract between component and its container
 - things that affect how container should manage component environment, such as transaction monitoring, access control, etc.
 - eg managedTransaction.global

SCA Intents for Reliable Messaging

- **atLeastOnce:**
message sent by client is always delivered
- **atMostOnce:**
message sent by client is delivered at most once
- **exactlyOnce:**
message sent by client is delivered exactly once
Combination of atLeastOnce and atMostOnce
- **ordered:**
messages are delivered in order they were sent by client

SCA Intents for Security

- **authentication:**
requirement that client must authenticate itself in order to use an SCA service.
- **confidentiality:**
requirement that message contents are accessible only to those authorized to have access (typically service client and service provider)
 - common approach is to encrypt the message; other methods are possible
- **integrity:**
requirement that message contents have not been tampered with and altered between sender and receiver
 - common approach is to digitally sign the message; other methods are possible

SCA Intents for Security - qualifiers

Each of the three basic security intents can be qualified by either “message”, or transport.

message:

indicates that the facility is provided at the message level

transport:

indicates that the facility provided by the transport, say, SSL

For example: confidentiality.message conveys requirement that confidentiality be provided at message level.

Associating Policies with SCA Components

- Intents and/or policySets can be associated with any SCA component
- At deployment time intents are mapped into Policies contained in policySets
- Examples attaching intents:

Confidentiality applied to any use of the service

```
<service name "AccountService" promote="AccountServiceComponent"
  requires="sca:confidentiality">
  <interface.java interface "services.account.AccountService"/>
  <binding.ws port "http://www.bigbank.com/AccountService#
    wsdl.endpoint(AccountService/AccountServiceSOAP)"/>
</service>
<reference name "StockQuoteService"
  promote="AccountServiceComponent/ stockQuoteService">
  <interface.java interface "services.stockquote.StockQuoteService"/>
  <binding.ws port "http://www.quickstockquote.com/StockQuoteService#
    wsdl.endpoint(StockQuoteService/StockQuoteServiceSOAP) "#
    requires="sca:confidentiality"/>
</reference>
```

Confidentiality applied to Web service binding

Policy Sets

- At deployment time, intents are mapped to concrete WS-Policies and WS-Policy Attachments via Policy Sets
- A Policy Set has the following structure:

```
<policySet name="xs:QName"
  provides="list of xs:QNames"
  appliesTo="XPath expression">

  <policySetReference name="xs:QName"/>*
  <intentMap/>*
  <wsp:PolicyAttachment>*
  <wsp:Policy>
  <wsp:PolicyReference>
  <xs:any>*
</policySet>
```


Intent Maps

- intentMaps map qualified intents to concrete policies
- Each <qualifier/> element associates a qualified intent name with one or more policy assertions (could be wsp:PolicyAttachment)
 - Each PolicyAttachment element contains a policy expression and a policy subject (what the policy applies to)
- All policies in an intentMap are from a single policy domain
- policySets aggregate intentMaps to create intent-to-policy mappings for multiple domains

Intent Maps

- intentMaps associate intent names with PolicyAttachments: WS-Policy expression plus policy subject

```
<intentMap provides="sca:confidentiality" default="transport">
  <qualifier name="transport">
    <wsp:PolicyAttachment>
      <!-- policy expression and policy subject for
           "transport" alternative -->
      ...
    </wsp:PolicyAttachment>
    <wsp:PolicyAttachment>
      ...
    </wsp:PolicyAttachment>
  </qualifier>
  <qualifier name="message">
    <wsp:PolicyAttachment>
      <!-- policy expression and policy subject for
           "message" alternative" -->
      ...
    </wsp:PolicyAttachment>
  </qualifier>
</intentMap>
```

Policy Sets

- Policy Sets can also contain Policies or References to Policies directly, without intent maps.

```
<policySet name="sca:userNameTokenHashPassword"
  provides="sca:authentication"
  appliesTo="sca:binding.ws">
  <wsp:Policy>
    <sp:SupportingToken>
      <wsp:Policy>
        <sp:UserNameToken>
          <wsp:Policy>
            <sp:HashPassword>
            </wsp:Policy>
          </sp:UserNameToken>
        </wsp:Policy>
      </sp:SupportingToken>
    </wsp:Policy>
  </policySet>
```

Associating Policies with SCA Components

- Intents and/or policySets can be associated with any SCA component.
- At deployment time intents are mapped into Policies contained in policySets
- For example, attaching intents to service or reference definition:

```
<service> or <reference>...  
  <binding.binding-type requires="sca:confidentiality"  
  </binding.binding-type>...  
</service> or </reference>
```

Associating Policy Sets with Bindings

- Use binding with an explicitly specified PolicySet
- Default alternatives can be overridden

```
<sca:service> or <sca:reference>...  
  <sca:binding.binding-type policySets="sns:enterprisePolicy"  
  </sca:binding.binding-type>...  
</sca:service> or </sca:reference>
```

- Overriding default intent alternatives in the PolicySet

```
<sca:reference name="RentalCarService">  
  <sca:interface/>  
  <sca:binding.WS policySet="sns:BasicSecurity"  
    requires="sca:authentication.certificateAuthentication  
             sns:messageProtection.protectBodyAndHeader" />  
  </sca:binding.WS>  
</sca:reference>
```


Intents Provided by Bindings

- Some binding types may satisfy intents by virtue of their implementation technology. For example, an SSL binding would natively support confidentiality.
- Binding instances which are created by configuring a bindingType may be able to provide some intents by virtue of its configuration.
- When a binding type is defined in SCA, these properties are declared as values of the @alwaysProvides and @mayProvide attributes.
- Proprietary implementations on binding types may support different intents.

```
<bindingType type="xs:QName"  
    alwaysProvides="list of intent QNames"?  
    mayProvide = "list of intent QNames"?/>
```

Recursive Composition

- Intents CANNOT be overridden higher up in recursive composition
 - Intents can be further qualified (i.e. constrained)
- Intent set for SCDL element derived from the element and its ancestor elements
 - See example, both qualified intents MUST be satisfied by the binding/policySets attached to reference “bar”
- PolicySets can be overridden

```
<composite requires="confidentiality.transport">  
  <service name="foo" />  
  <reference name="bar"  
    requires="confidentiality.message"/>  
</composite>
```

Mapping Intents to Policy Sets

- We start with a component with abstract QoS requirements
- We want to deploy this with other components in a composite
- So, we must find bindings and/or policySets that satisfy the required intents. This is as follows:
 - Expand out all profile intents
 - Calculate the required intents set
 - Remove intents directly satisfied by the binding or implementation
 - Calculate the explicitly specified policySets
 - Remove intents satisfied by these policySets
 - Find the smallest collection of available policySets that satisfy remaining intents

Implementation Policies

- Intents and PolicySets can be associated with implementations

```
<sca:component name="myComponent">
  <sca:implementation.* policySets="list of policySet xs:QNames"
    requires="list of intent xs:QNames">
    ...
    <sca:operation name="xs:string" service="xs:string"?
      policySets="list of policySet xs:QNames"?
      requires = "list of intent xs:QNames"?/>*
    ...
  </sca:implementation>
  ...
</sca:component>
```

- Example of non WS-Policy policySet

```
<policySet provides="sns:logging.trace"
  appliesTo="sca:implementation.bpel">
  <acme:processLogging level="3"/>
</policySet>
```

Security Implementation Policies: Policy Assertions

- **Authorization** controls who can access the protected SCA resources.
- **Security role** is a concept that represents a set of access control constraints on SCA resources.

This is defined as:

```
<allow roles="list of role NCNames">  
<permitAll/>  
<denyAll/>
```

- Security Identity declares the security identity under which an operation will be executed. This is defined as

```
<runAs role="NCName">
```


Transaction Implementation Policy Intents

- ***managedTransaction*** intent specifies that component must run within a managed transaction
- ***managedTransaction.local*** – the component must run within local transaction containment
- ***managedTransaction.global*** – component runs within a global transaction
- ***noManagedTransaction*** – component ***must not*** run within a managed transaction

```
<sca:component name="myComponent">  
  <sca:implementation.java class="foo.bar.implClass"  
    requires="managedTransaction.global" >  
  <sca:service name="mainService" requires="propagatesTransaction"/>  
  <sca:reference name="someReference" requires="suspendsTransaction"/>  
</sca:component>
```

Transacted One-Way Messages

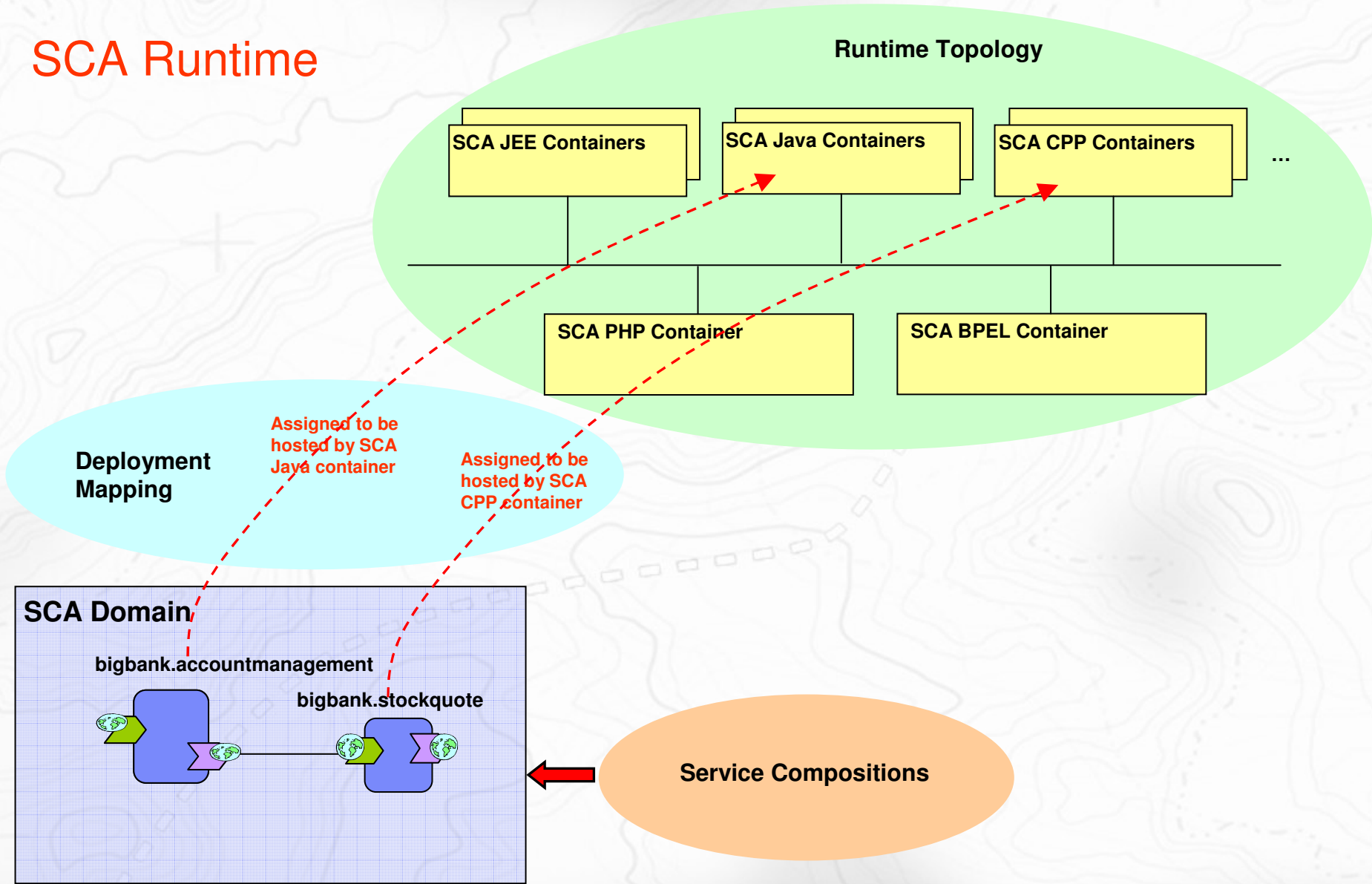
- ***transactedOneWay*** intent used with `managedTransaction.global` to indicate that one-way message is committed (sent) with completion of the transaction
- ***immediateOneWay*** – intent used to indicate that one-way message is sent immediately and is not transacted

- ***propagatesTransaction*** intent on service to indicate need to receive transaction, on reference to indicate that client transaction is propagated to target service
- ***suspendsTransaction*** – intent used to indicate that transaction context is not accepted on service or sent on a reference

Packaging and Deployment: Domains

- Composites deployed, configured into **SCA Domain**
 - Defines the boundary of visibility for SCA
 - Typically an area of functionality controlled by single organization/division
 - E.g.: accounts
- Configuration represented by virtual composite
 - potentially **distributed** across a network of **nodes**
 - contains **components, services, references, wires**
 - configured using **composites**
- Composites make deployment simpler
 - individual composites created, deployed independently
 - may contain only wires or components or externally provided services or references
- Abstract services provided for management of the domain

SCA Runtime



Packaging and Deployment: Contributions

- Contributions hold artifacts available for use in the Domain
- Package containing artifacts necessary for SCA
 - SCA defined artifacts
 - E.g.: composites, constrainingType, etc
 - Non-SCA defined artifacts
 - E.g.: WSDL, XML schema, Java classes, object code etc
- Packaging must be hierarchical
- Metadata included in the “META-INF” directory

```
<contribution xmlns=http://www.ooa.org/xmlns/sca/1.0>  
  <deployable composite="xs:QName"/>*  
  <import namespace="xs:String" location="xs:AnyURI"?/>*  
  <export namespace="xs:String"/>*  
</contribution>
```

- Interoperable packaging format: ZIP
- Other formats possible:
 - filesystem directory, EAR, JAR, OSGi bundle

Summary

- SCA provides excellent facilities for handling of late-bound communication technologies
- SCA has a simple, flexible declarative mechanism to handle complex QoS and Policy requirements



Useful links...

- OASIS Open CSA
<http://www.oasis-openca.org/>
- OASIS SCA Technical Committees
<http://www.oasis-openca.org/committees>
- Open SOA Collaboration
<http://osoa.org/display/Main/Home>
- V1 level of SCA specifications
<http://osoa.org/display/Main/Service+Component+Architecture+Specifications>
- Useful papers and interesting SCA information:
<http://osoa.org/display/Main/SCA+Resources>
- OASIS Webinar downloads:
<http://www.oasis-openca.org/resources>



Questions and Answers

© IBM Corporation 2007. All Rights Reserved.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM trademarks, see www.ibm.com/legal/copytrade.shtml
AIX, CICS, CICSplex, DB2, DB2 Universal Database, i5/OS, IBM, the IBM logo, IMS, iSeries, Lotus, OMEGAMON, OS/390, Parallel Sysplex, pureXML, Rational, RCAF, Redbooks, Sametime, System i, System i5, System z, Tivoli, WebSphere, and z/OS.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.