Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

Building Asynchronous Services with SCA

Mike Edwards
IBM
Hursley Park

Agenda

- **Service Component Architecture**
- Why Asynchronous Services?
- Facilities for Async Services
- SCA support for service interactions
- Example Async Service & Client
- Events and Async Services compared

What SCA *is*

- ***executable*** model for assembling services
  - composites provide language to compose and configure service components
  - handles service dependencies
- Simplified ***component programming model*** for implementing services
  - BPEL processes, Java™ POJOs, EJBs, COBOL, PHP scripts, C++ apps, JavaScript & AJAX, XSLT…
- Late binding of ***policy*** and ***communication*** methods, with ***distributed deployment*** model

Why SCA makes life simpler

- One way to look at SCA is that it takes details of
  - access methods and endpoints
  - implementations and configuration
  - policy such as encryption, authentication
- …and moves them into middleware layer
- Application developers write business logic: code that actually builds value
  - details of using services handled by SCA
  - late binding: as details change, applications (and developers who wrote them) aren't affected
  - ***"no plumbing in the code"***

Why SCA makes life simpler

- SCA gives developers **single programming model for using services** for all aspects of service lifecycle:
  - ***Construction***
  - ***Assembly***
  - ***Deployment***

- As your SOA gets more complex, developers have to learn more and more interfaces
  - In Java™ alone: EJBs, RMI, JCA, JAX-WS, JMS...

Agenda

- Service Component Architecture
- **Why Asynchronous Services?**
- Facilities for Async Services
- SCA support for service interactions
- Example Async Service & Client
- Events and Async Services compared

Asynchronous interactions

- ***Typical*** in business processes
- …not everything happens immediately
- …a single request can result in a sequence of responses over time
- …a client probably does not want to wait while all steps are completed

Synchronous Processing

Price of IBM stock?

IBM = $121.50

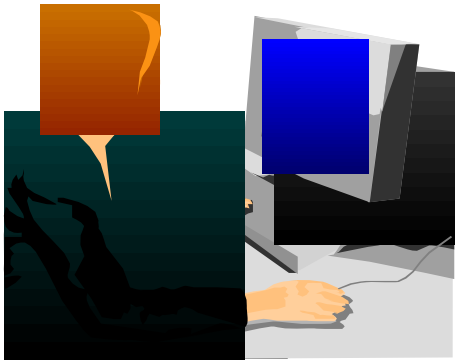
Price of AT&T stock?

AT&T = $38.72

Client

Provider

Asynchronous Processing



Order Books X, Y, Z

Acknowledge, OrderID = 3824

OrderID = 3824 Expect dispatch = 12/06/2008

Client

Provider

OrderID = 3824 Books X,Y dispatched 10/06/2008
Package ID = 478842110

OrderID = 3824 Books Z dispatched 13/06/2008
Package ID = 479567736

Why Asynchronous Services?

- Business processes involve long running multi-step sequences
- Clients & Services cannot afford to wait on long request/response operations
  - impact on processing resources, memory, communication resources

Agenda

- Service Component Architecture
- Why Asynchronous Services?
- **Facilities for Async Services**
- SCA support for service interactions
- Example Async Service & Client
- Events and Async Services compared

Synchronous Services

- plenty of support for synchronous programming
- call-and-return typical for most programming languages

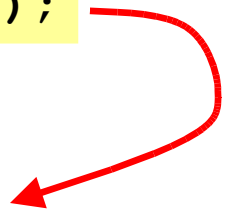
- `OrderResponse placeOrder( OrderRequest oRequest );`
  for clients & service providers

Asynchronous Services

- Facilities less common
  - Java™ concurrency classes
  - JAX-WS asynchronous client
  - JMS messaging
  - BPEL

JAX-WS Async Client API

```
OrderResponse placeOrder( OrderRequest oRequest );
```



```
Future<?> placeOrderAsync( OrderRequest oRequest,
            AsyncHandler<OrderResponse> responseHandler );
```

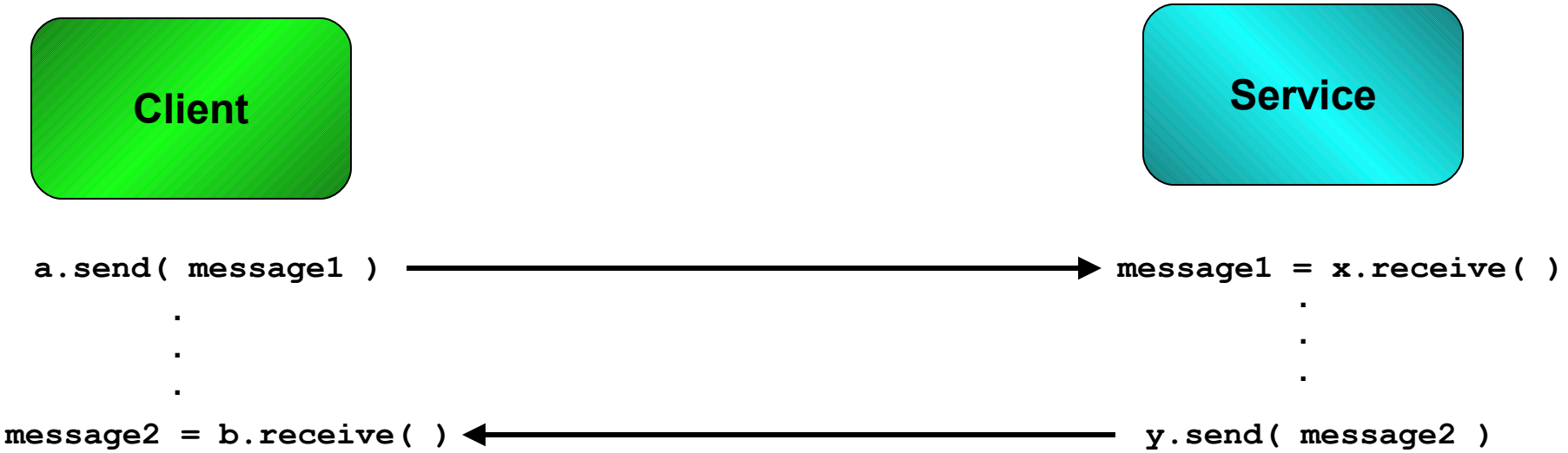
+

```
class OrderResponseHandler
        implements AsyncHandler<OrderResponse> {

    public void handleResponse( Response<OrderResponse> theResponse )
    {
        OrderResponse orderResponse = theResponse.get();
    }
}
```

JMS/Messaging

- Supports async clients and async services



+ messageListener onMessage listener interface

BPEL and Async Processing

- BPEL designed to handle:
  - long running processes
  - with asynchrony
  - multiple partner relationships

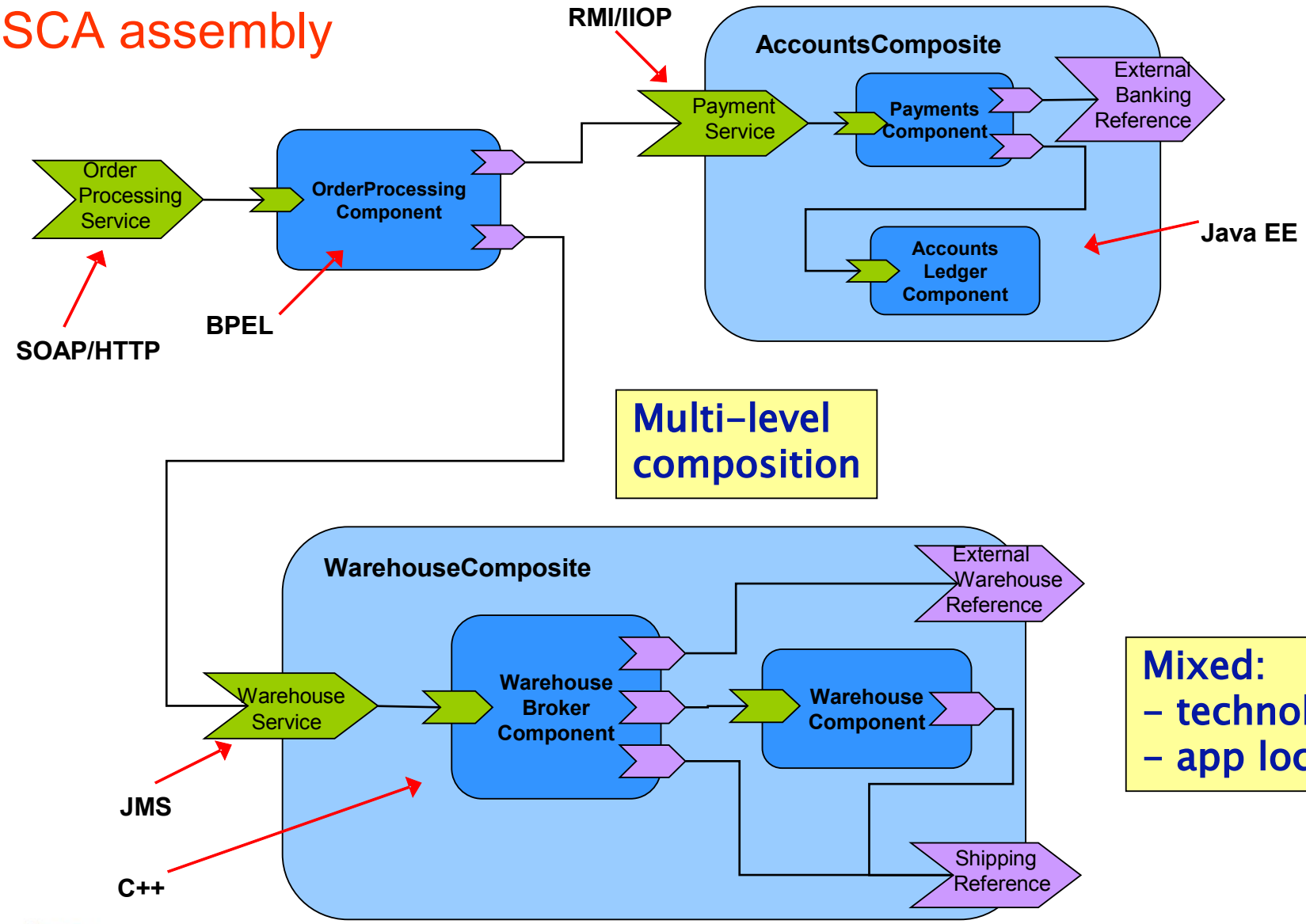
Agenda

- Service Component Architecture
- Why Asynchronous Services?
- Facilities for Async Services
- **SCA support for service interactions**
- Example Async Service & Client
- Events and Async Services compared

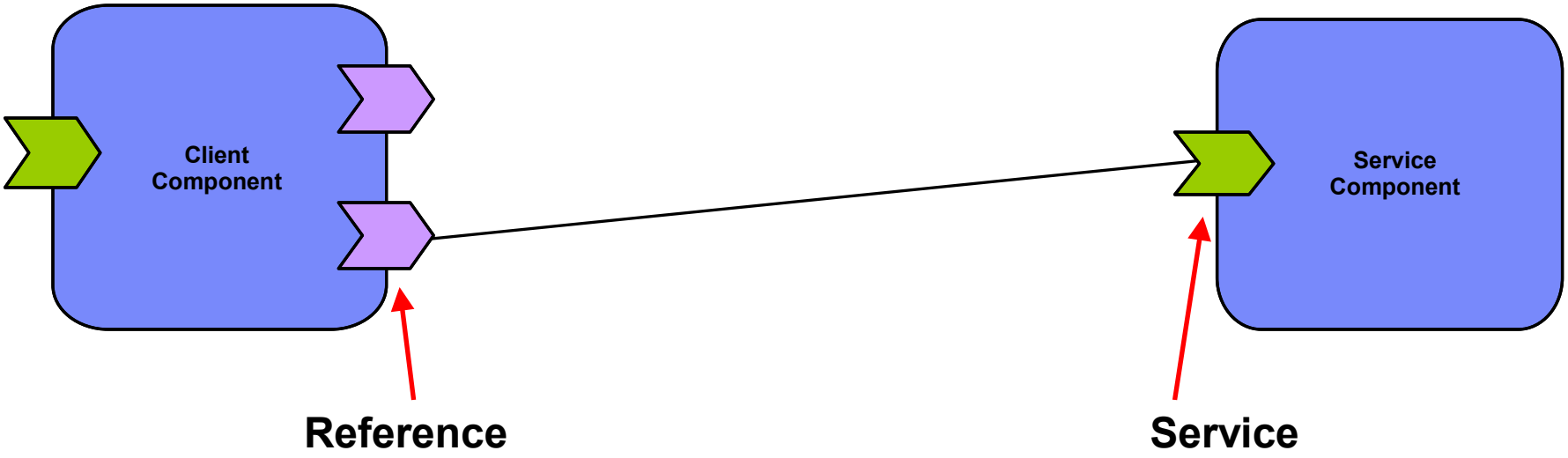
Service Component Architecture (SCA): Simplified Programming Model for SOA

- 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



SCA Service Interaction



Synchronous **or** ***Asynchronous*** **interaction styles supported**
- depends on the interface definition

SCA Callbacks

- SCA supports Async services using ***Callbacks***
  - service has regular forward call interface
  - + callback interface on which service makes calls back to clients
    - clients implement the callback interface

Sync and Async Interfaces

```
public interface SyncOrderService {
  public OrderResponse placeOrder( OrderRequest oRequest );
}
```

```
public interface AsyncOrderService {
        public void placeOrder( OrderRequest oRequest );
}

public interface AsyncOrderCallback {
        public void placeOrderResponse( OrderResponse oResponse );
}
```

Callback Interfaces

- Callback interface
  - may have multiple operations/methods
  - operations may be called at any time after initial forward call to service
  - number and sequence of callback operations for given forward call is variable (0, 1, many…)

Agenda

- Service Component Architecture
- Why Asynchronous Services?
- Facilities for Async Services
- SCA support for service interactions
- **Example Async Service & Client**
- Events and Async Services compared

The interfaces

Forward service interface:

```
public interface AsyncOrderService {
        public void placeOrder( OrderRequest oRequest );
}
```

Callback interface:

```
public interface AsyncOrderCallback {
        public void placeOrderResponse( OrderResponse oResponse );
}
```

Order Service implementation

```
import org.osoa.sca.annotations.*;
public class OrderServiceImpl implements OrderService {
         // A field for the callback reference object
        private OrderCallback callbackReference;

        // The place order operation itself
        public void placeOrder( OrderRequest oRequest ) {

                // …do the work to process the order…
                // …which may take some time…
                // when ready to respond…
                OrderResponse theResponse = new OrderResponse();

                callbackReference.placeOrderResponse( theResponse );
        }

        // A setter method for the callback reference
        @Callback
        public void setCallbackReference( OrderCallback theCallback ) {
                callbackReference = theCallback;
        }
}
```

Order client application

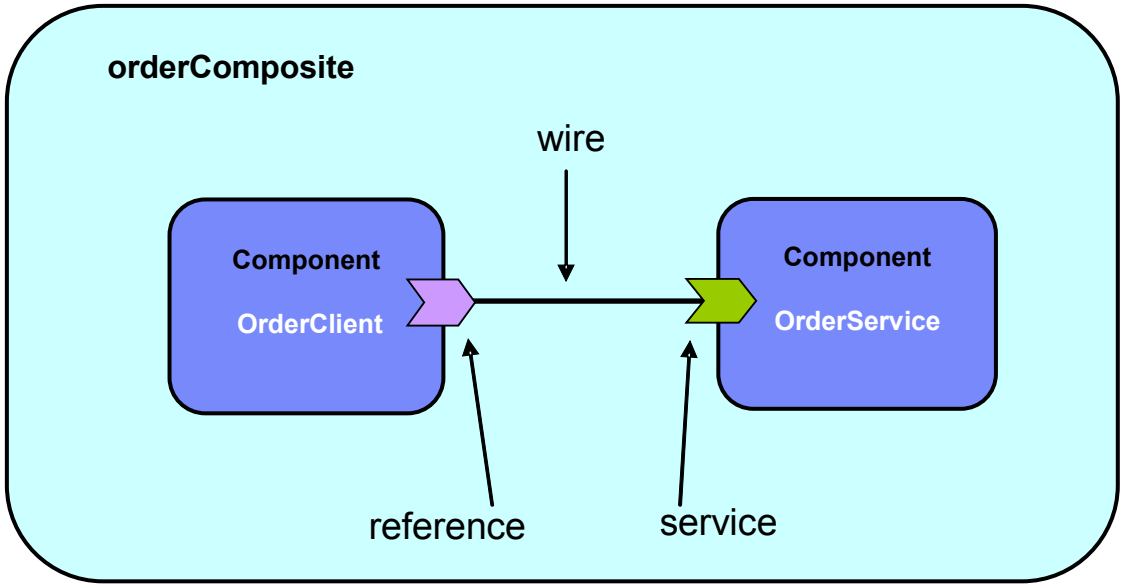
```
import org.osoa.sca.annotations.*;
public class OrderServiceClient implements OrderCallback {
         // A field to hold the reference to the order service
         private OrderService orderService;

         public void doSomeOrdering() {
                  OrderRequest oRequest = new OrderRequest();
                  //… fill in the details of the order …
                  orderService.placeOrder( oRequest );
                  // …the client code can continue to do processing
         }

         // callback method…
         public void placeOrderResponse( OrderResponse oResponse ) {
                  // …handle the response as needed
         }

         // A setter method for the order service reference
         @Reference
         public void setOrderService( OrderService theService ) {
                  orderService = theService;
         }
}
```

Composite diagram



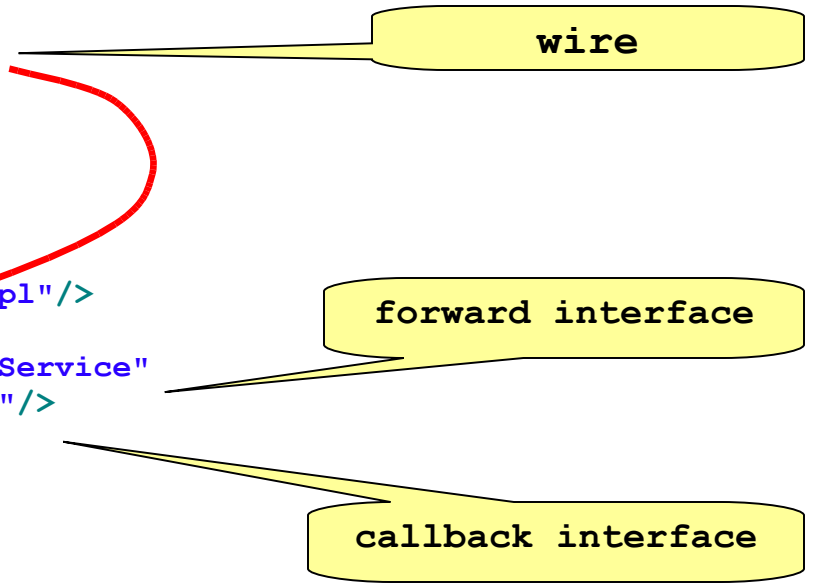
Composite

```
<?xml version="1.0" encoding="UTF-8"?>
<composite name="orderComposite"
          xmlns:sca="http://www.osoa.org/xmlns/sca/1.0">

  <!-- The OrderClient component -->
  <component name="OrderClient">
          <implementation.java class="OrderServiceClient"/>
          <reference name="OrderService"
                 target="OrderService/OrderService">
                    <binding.ws/>
          </reference>
  </component>

  <!-- The OrderService component  -->
  <component name="OrderService">
          <implementation.java class="OrderServiceImpl"/>
          <service name="OrderService">
                    <interface.java interface="OrderService"
                    callbackinterface="OrderCallback"/>
                    <binding.ws/>
          </service>
  </component>

</composite>
```



"under the hood"

- How do the callbacks work in practice?
  - implemented using suitable communication methods, such as:
  - JMS messaging
  - Web services using WS-Addressing

Agenda

- Service Component Architecture
- Why Asynchronous Services?
- Facilities for Async Services
- SCA support for service interactions
- Example Async Service & Client
- **Events and Async Services compared**

Asynchrony & Events

- Another way to handle asynchrony is to use ***events***
- components communicate by producing and consuming events
  - these are one-way
  - asychronous
  - arbitrary sequences
- eg JMS messages

Events or Callbacks

- Both allow asynchrony
- Calls target single specific service
- Callbacks ensure responses go back to original requester
- Event handling is more flexible
  - pub/sub, broadcasting of events
  - no guarantee that anyone listens

Summary

- Asynchronous services are a fact of life
- SCA makes providing async services and their clients simpler and easier
- SCA supports this on a range of communication methods

Useful links

- Articles about SCA:
- http://www.infoq.com/articles/setting-out-for-sca
- http://www.osoa.org/display/Main/SCA+Resources

- Open Source implementation of SCA:
- http://cwiki.apache.org/TUSCANY/

- SCA Specifications in OASIS:
- http://www.oasis-opencsa.org/

- Email address:
- mike_edwards@uk.ibm.com

JavaOne℠

Thank You

Mike Edwards
mike_edwards@uk.ibm.com

IBM Hursley Park

Sun microsystems