JavaOne℠

Java™ EE Connector Architecture 1.6 Technology (JSR 322)

Sivakumar Thyagarajan and Binod PG
Sun Microsystems
TS-4733

Presentation Goal

Learn about the new features planned for the Java™ EE Connector Architecture 1.6 technology

Agenda

- Introduction
- Connector 1.6 themes
  - Ease of Development (EoD) features
  - Generic Work Context
  - Security Inflow
  - Miscellaneous improvements
- Resources and Summary
- Q & A

Agenda

Introduction

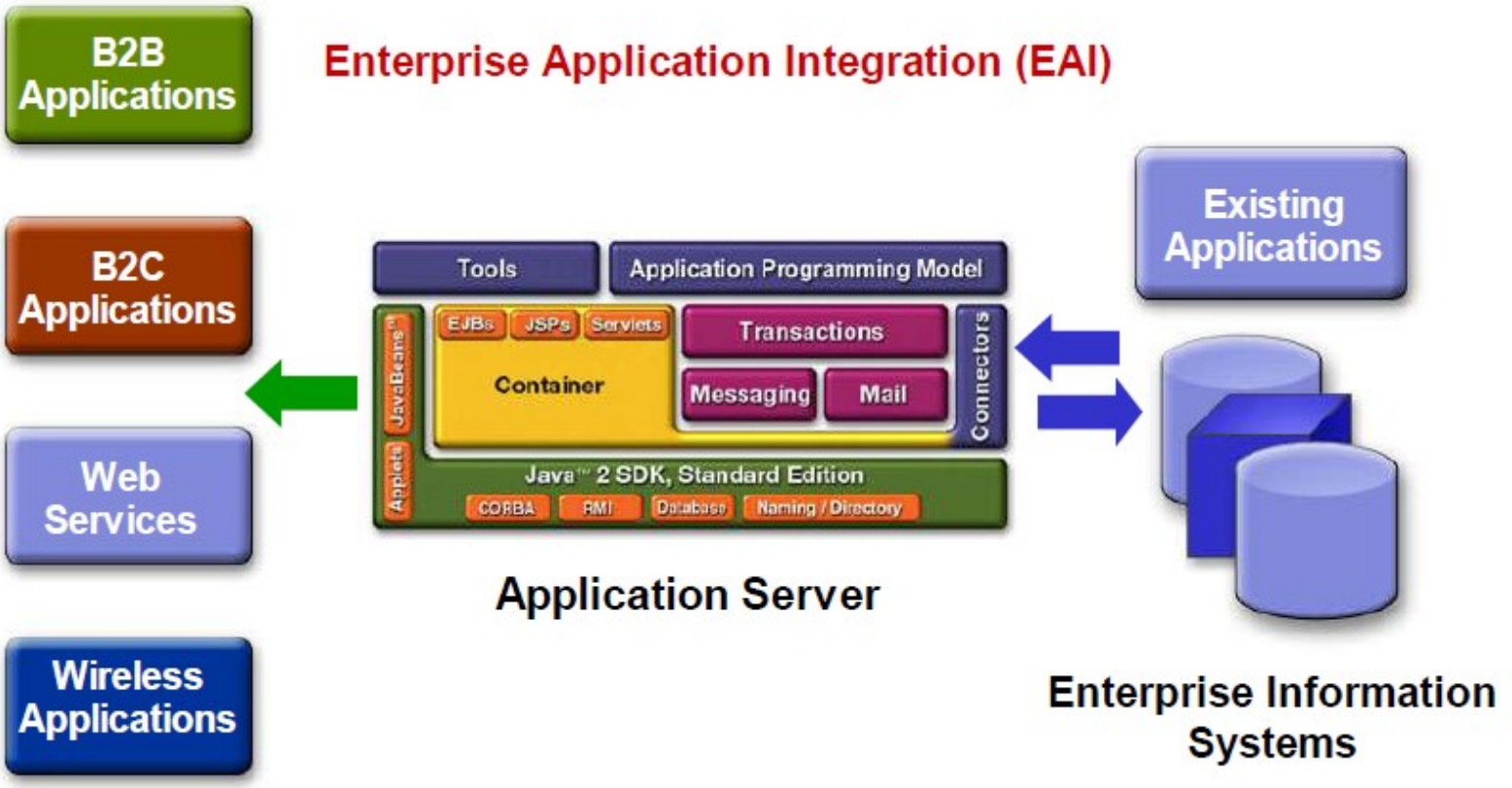
Connector 1.6 themes

- Ease of Development (EoD) features
- Generic Work Context
- Security Inflow
- Miscellaneous improvements

Resources and Summary

Q & A

Java EE Connector Architecture
Overview



Connector Architecture 1.0
Capabilities

Outbound communication

- Mechanism for Java EE components to connect to external systems
- Connection Management: Lifecycle, Pooling, Sharing
- Security Contract
- Exporting Transaction and Security Context to EIS

Connector 1.5 (JSR 112)
Introduction

- Final Release Nov 2003
  - Part of J2EE (Java 2 Enterprise Edition) 1.4
- Major themes
  - Asynchronous Integration with EISs
    - Enabled bi-directional asynchronous interactions with EIS
  - JMS Provider Pluggability

Connector Architecture 1.5
Capabilities

- Lifecycle Management
- Inbound Messaging
  - Mechanism for EIS to call Message Endpoints (MDBs)
  - Import transaction context from EIS
- Work Management

Connector 1.6 Specification
Introduction

Work in progress by the expert group of JSR 322

Part of Java EE 6

Progress so far

- Filed JSR in Dec 2007
- Early Draft Review: July 2008
- Public Review: Nov 2008
- Proposed Final Draft (PFD) **available** from Feb 2009
- Final dates to be aligned with Java EE 6

Connector 1.6 Specification
Major Themes

Driven by community and Connector EG feedback

Themes

- Generic Work Context mechanism
- Security Context Inflow during Message Delivery and Work Submission
- Ease of Development (EoD)
- Misc. improvements to the specification

Disclaimer: APIs/Features shown in this presentation are as per the PFD version of the specification. Subject to change with feedback.

Agenda

Introduction

Connector 1.6 themes

- **Ease of Development (EoD) features**
- Generic Work Context
- Security Inflow
- Miscellaneous improvements

Resources and Summary

Q & A

Ease of Development (EoD)
Goal and approach

Goal

- to simplify the development of RAs for programmers who are either just starting with Connector technology or developing RAs of small to medium complexity

Approach

- Align with the EoD approach adopted by EJB/JPA
  - Make extensive use of Java language annotations
  - Reduce the need for a descriptor (ra.xml), better defaults
  - Allow sparse descriptors, mixed-mode development etc.

Ease of Development
Metadata annotations

@Connector

- @AuthenticationMechanism
- @SecurityPermission

@ConfigProperty

- Automatic discovery of configuration properties

@ConnectionDefinition, @ConnectionDefinitions

@Activation

@AdministeredObject

Ease of Development
@Connector sample usage

Optionally provide a *ResourceAdapter* class

- Required only if RA supports inbound, requires lifecycle callbacks, access to *BootstrapContext* etc

```
//A simple resource adapter that does not support transactions
@Connector()
public class MailResourceAdapter implements ResourceAdapter{
    // Define common configuration properties.
    //... other methods
}
```

Ease of Development
@ConnectionDefinition sample usage

```
//A simple ManagedConnectionFactory implementation that is part of
//a connection definition, could be defined as follows
@ConnectionDefinition(connectionFactory=CF.class,
    connectionFactoryImpl=CFImpl.class,
    connection=Conn.class,
    connectionImpl=ConnImpl.class)
public class ManagedConnectionFactoryImpl implements
  ManagedConnectionFactory {
    //...
}
```

Ease of Development
@Activation sample usage

```
@Activation()
public class MyActivationSpec {
    //Use of Bean Validation annotations to express
    //validation requirements
    @Size(min=5, max=5)
    private int length;
    //... other methods
}
```

Ease of Development
Demo time !

Demonstration: mail-connector sample

- Showcase the ease of development features
- Compare the same resource adapter developed in Connector 1.5 and 1.6
- Deploy on Java EE 6 RI (Project GlassFish v3)

Agenda

Introduction

Connector 1.6 themes

- Ease of Development (EoD) features
- **Generic Work Context**
- Security Inflow
- Miscellaneous improvements

Resources and Summary

Q & A

Generic Work Context
Rationale

A generic mechanism for the RA to propagate other contextual info from EIS during message delivery or Work submission

- Examples: Security, Conversational Context, Availability/QoS etc

Enables

- an application server to support new message inflow and delivery schemes
- provides a richer Contextual Work execution environment to the RA

Generic Work Context
Design Model

RA submits a *Work* that implements *WorkContextProvider*

Before calling *run* on the *Work* instance, *WorkManager* iterates through the *Collection* of *WorkContext*s provided by the *Work* instance

Establishes Context based on the *WorkContext*

Generic Work Context

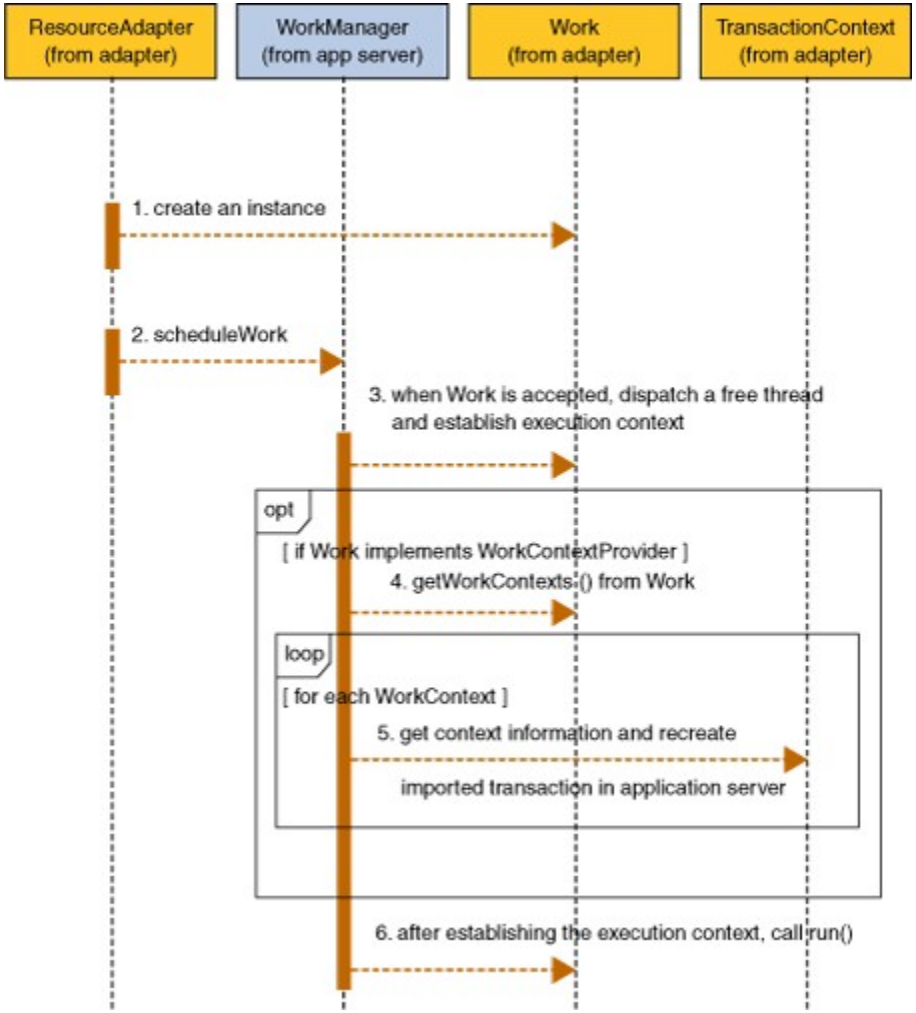
WorkContextProvider and WorkContext interface

```
package javax.resource.spi.work;

public interface WorkContextProvider {

    List<WorkContext> getWorkContexts();

}
```

```
package javax.resource.spi.work;

public interface WorkContext {

    String getName();

    String getDescription();

}
```

Generic Work Context

Context assignment sequence diagram



Generic Work Context
Features

Compatible with the Connector 1.5 Work submission and context assignment model

Independent of the Connector Work Management contract

Standardized *TransactionContext, SecurityContext* and *HintsContext*

RA can require the AS to support, check if the AS supports, and get lifecycle notifications during the assignment of a WorkContext

Generic Work Context
Standardized *WorkContexts*

```
public class TransactionContext extends ExecutionContext implements
   WorkContext {

     public TransactionContext(Xid xid) { ... }

     public TransactionContext( Xid xid, long timeout){ ... }

     public String getName(){

          return "TransactionContext";

     }

     ... other methods

}

public abstract class SecurityContext implements WorkContext {

     public String getName(){

          return "SecurityContext";

     }

     .... other SecurityContext related methods

}
```

Generic Work Context
Sample usage

Scenario

A business use-case requires that the work done by the application components, during a message inflow, be automatically enlisted as part of the imported transaction .

Solution

Use the new *TransactionContext* to propagate transaction information from the EIS to the *MessageEndpoint* as part of the *Work* instance performing the message delivery

Generic Work Context

Sample usage – ResourceAdapter implementation

```
public class MyResourceAdapterImpl
             implements ResourceAdapter {
    ...
    public void start(BootstrapContext ctx) {
        bootstrapCtx = ctx;
    }
    ...
    {
        WorkManager workManager =
          myRA.bootstrapCtx.getWorkManager();
        workManager.submitWork(new MyWork());
    ...
    }
}
```

Generic Work Context

Sample usage – Work implementation

```
public class MyWork implements Work, WorkContextProvider {

    void release(){ ..}

    List<WorkContext> getWorkContexts() {

        TransactionContext txIn = new TransactionContext(xid);

        List<WorkContext> icList = new ArrayList<WorkContext>();

        icList.add(txIn);

        // Add additional WorkContexts

        return icList;

    }

    void run(){

        // Deliver message to MessageEndpoint;

    }

}
```

Agenda

Introduction

Connector 1.6 themes

- Ease of Development (EoD) features
- Generic Work Context
- **Security Inflow**
- Miscellaneous improvements

Resources and Summary

Q & A

Security Inflow
Rationale

Security inflow context absent in Connectors 1.5

Resource adapters not being able to deliver end-to-end application level security while

- executing a task in a *Work* instance
- delivering a message to a *MessageEndpoint*

Security Inflow
Goals

Enable an end-to-end security model for Java EE application-EIS integration

Support the execution of a *Work* instance in the context of an established identity.

Support the propagation of *Principal* information from an EIS to a *MessageEndpoint* during message inflow

Security Inflow
Design Model

- A standard *WorkContext*, *SecurityContext* that may be provided by the RA while submitting a *Work* for execution
  - AS establishes security context for the *MessageEndpoint*/*Work* during context assignment
- Leverage the work done in JSR 196: Java Authentication Service Provider Interface for Containers
  - RA uses the various JSR-196 Callbacks to establish caller identity

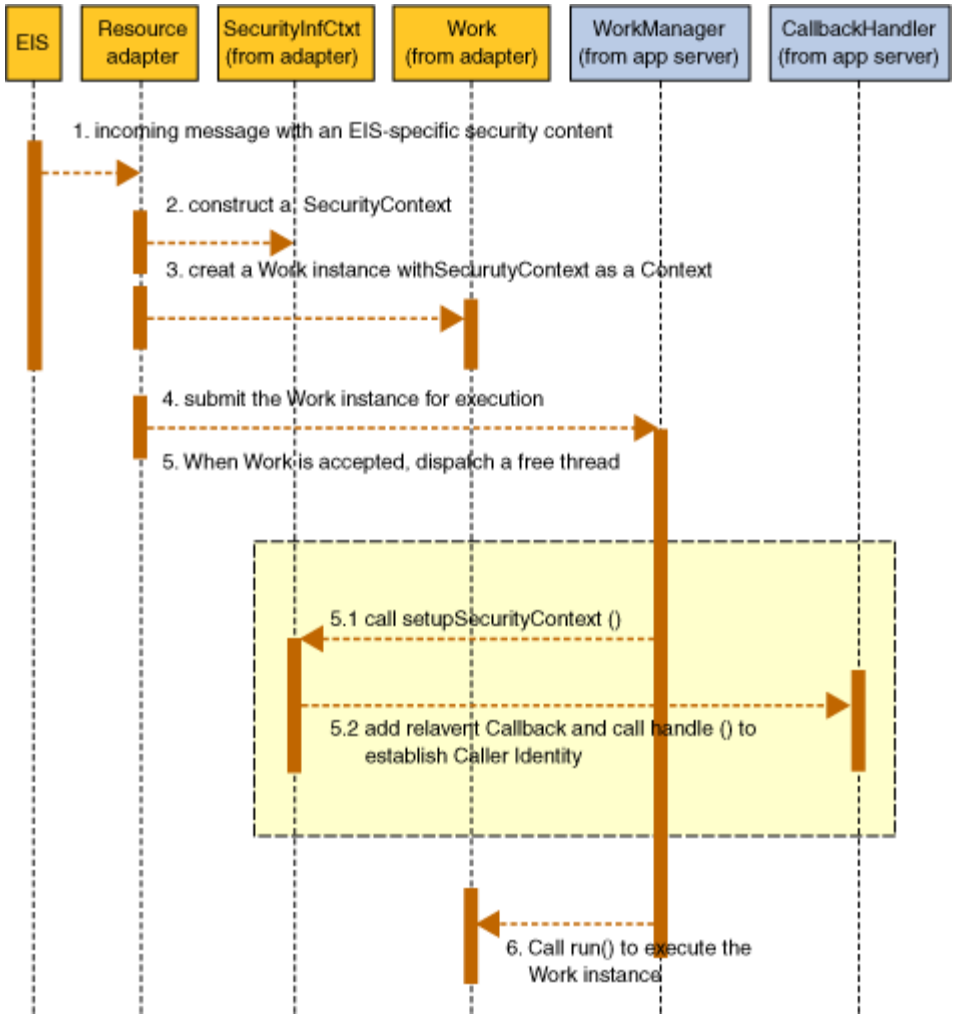
Security Inflow

SecurityContext abstract class

```
package javax.resource.spi.work;

import javax.security.auth.Subject;

import javax.security.auth.callback.CallbackHandler;

public abstract class SecurityContext

                    implements WorkContext {

    //Establish caller identity through the use of JSR196
      Callbacks

    public abstract void setupSecurityContext(

        CallbackHandler handler,

        Subject executionSubject,

        Subject serviceSubject);

}
```

Security Inflow

Security Context establishment – sequence diagram



Security Inflow

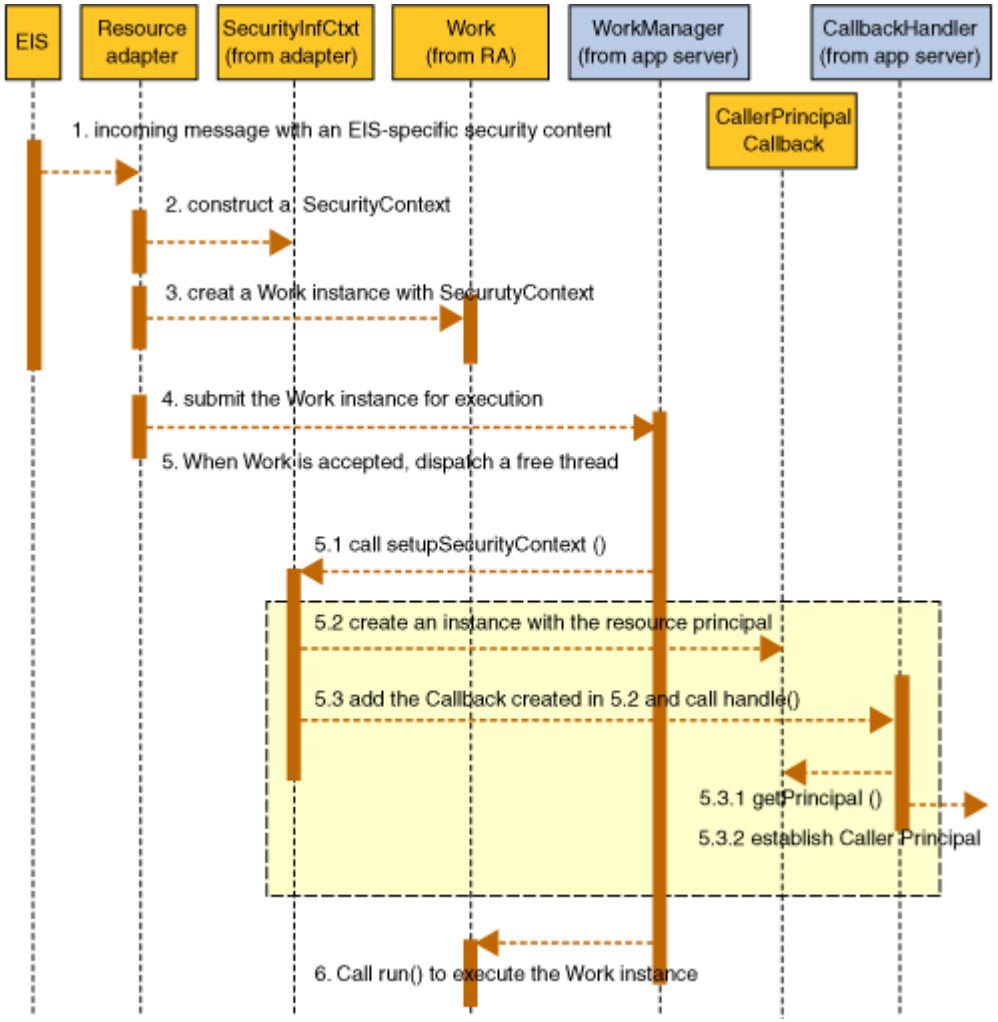
Caller Identity inflow scenarios

Based on whether the Caller identity is part of the AS security domain

- **Case 1**: RA flows-in an identity in the AS's security policy domain: AS may just use the initiating principal from the RA as the caller principal
- **Case 2**: RA flows-in an identity belonging to the EIS' security domain: A translation from one domain to the other needs to be performed.

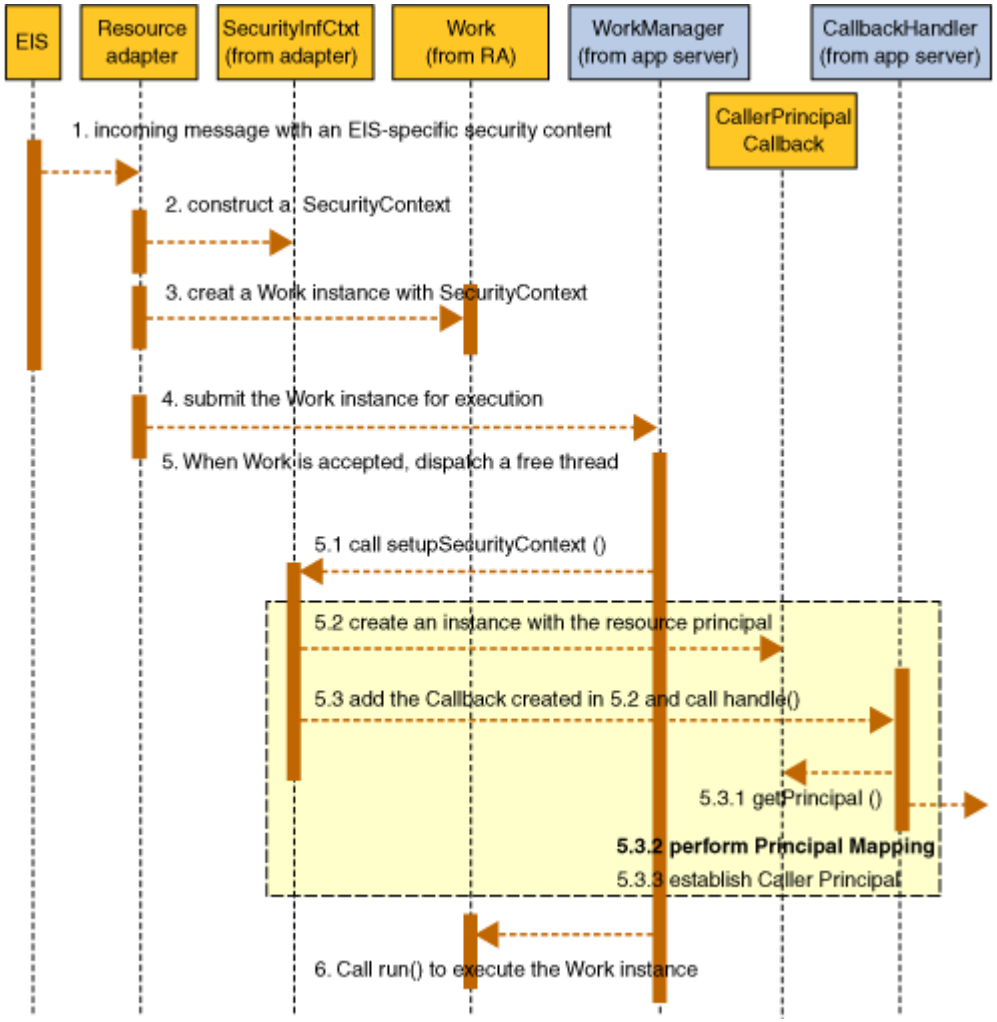
Security Inflow

Security Context establishment – Case 1



Security Inflow

Security Context establishment – Case 2



Agenda

Introduction

Connector 1.6 themes

- Ease of Development (EoD) features
- Generic Work Context
- Security Inflow
- **Miscellaneous improvements**

Resources and Summary

Q & A

Miscellaneous improvements

- Transaction Management
  - Specification of Transaction Support Level at Runtime
- Work Management
  - Distributed Work Processing
  - HintsContext
- Message Inflow
  - createMessageEndpoint timeouts
  - Retryable exceptions

Miscellaneous improvements

- Bean Validation integration (JSR 303)
- Standalone Connector Environment
- Configuration Property processing
  - range specification through JSR 303
  - dynamic reconfiguration and confidential params support
- Classloading requirements
  - clarified scenarios where deployed standalone RARs must be made visible to applications

Agenda

Introduction

Connector 1.6 themes

- Ease of Development (EoD) features
- Generic Work Context
- Security Inflow
- Miscellaneous improvements

Resources and Summary

Q & A

Resources – To learn more & get started

JSR 322 page http://jcp.org/en/jsr/detail?id=322

- Download proposed final draft spec and send comments to jsr-322-comments@jcp.org

Reference implementation at http://GlassFish.org

- Download and try **GlassFish v3** – Java EE 6 release
- Try Connector 1.6 samples as part of GlassFish v3 and build new 1.6 resource adapters

Blogs

- Siva: http://blogs.sun.com/sivakumart
- Binod: http://weblogs.java.net/blog/binod/

Summary

Part of Java EE 6 and reference implementation available through GlassFish v3

Simplifies resource adapter development

Adds powerful new capabilities to assign contexts during Work execution and Message delivery

Enhances enterprise application integration capabilities in Java EE 6

Agenda

Introduction

Connector 1.6 themes

- Generic Work Context
- Security Inflow
- Ease of Development (EoD) features
- Miscellaneous improvements

Resources and Summary

Q & A

Sivakumar Thyagarajan and Binod PG
jsr-322-comments@jcp.org

Sun Microsystems