Federated Web Services with Mobile Devices

Rajeev Angal

Architect
Sun Microsystems

Pat Patterson

Architect
Sun Microsystems

Session TS-6673

Goal of This Talk

Learn how to secure Federated Web Services using the Java™ Platform, Micro Edition (Java ME)

Agenda

Mobile Web Services: The Problem

Identity-Enabling Web Services

Java ME Technology JSRs: 172, 177, 279

Putting It Together

Demo

Next Steps

Agenda

Mobile Web Services: The Problem

Identity-Enabling Web Services

Java ME Technology JSRs: 172, 177, 279

Putting It Together

Demo

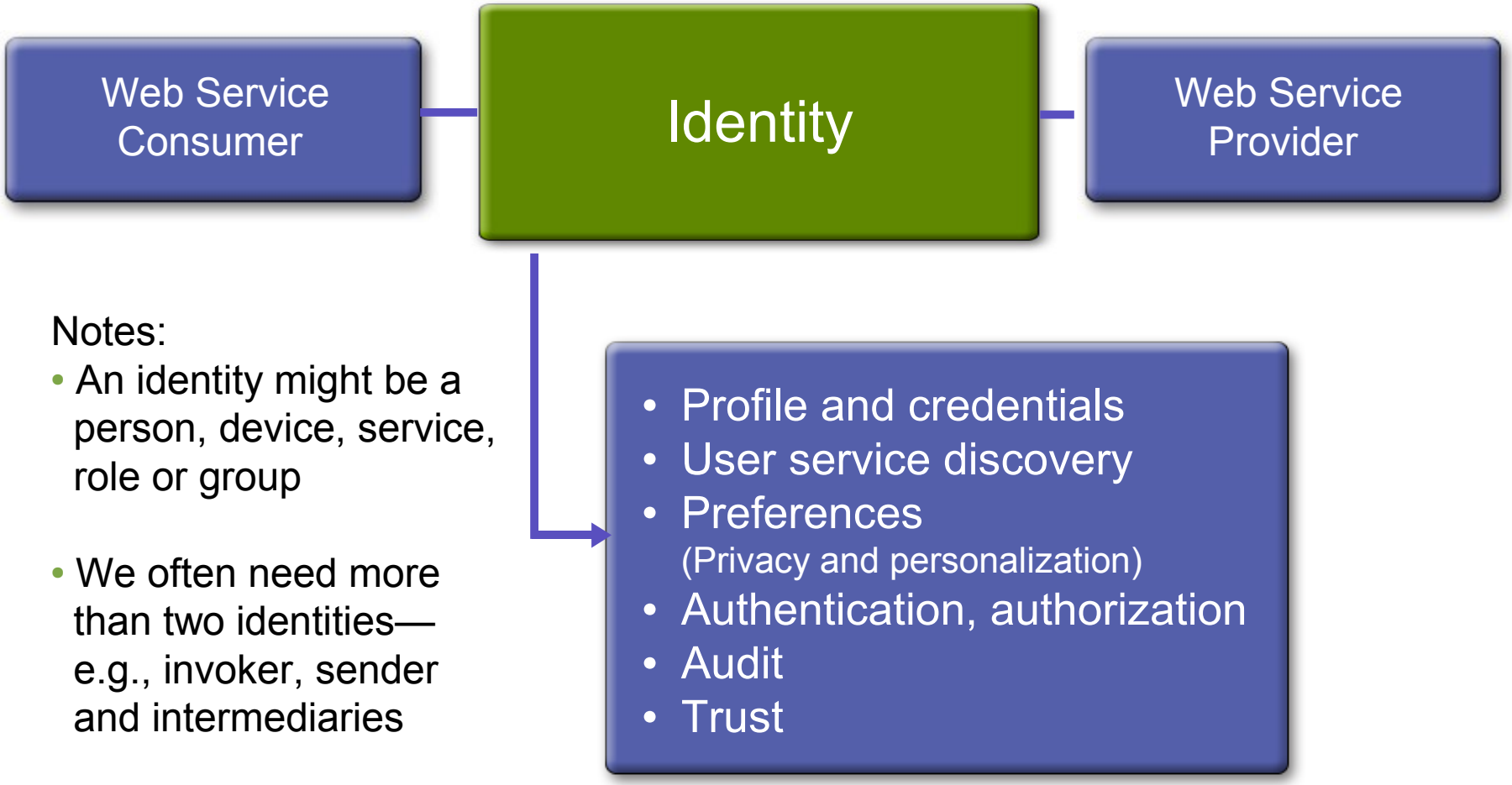
Next Steps

The Problem

- Business problem:
  - Need to leverage mobile commerce opportunities
  - Careers and operators looking to differentiate themselves
  - Need to be balanced with legal and regulations
- Technical problem:
  - Web Services + rich apps is a logical choice: “mSOA”
  - Need for end-to-end security and privacy
  - Need the solution to scale

Solution: Identity enabling mobile web services through security and privacy-focused specifications from Liberty Alliance Project and WS-*

Defining Identity



Notes:

- An identity might be a person, device, service, role or group
- We often need more than two identities—e.g., invoker, sender and intermediaries

Defining Federation

- Federation: the agreements, standards, and technologies that make identity and entitlements portable across autonomous domains;
  for example, an enterprise and third parties providing services to its employees
- We federate identities when we create associations between identities in different domains; a federation must be in place before we can federate identities

Key Security Questions

- How does a Web service consumer find the Web service provider?
- How does Web service consumer obtain the credentials it needs to invoke services at the web service provider?
- How is the trust relationship established?
- How is “invoker” identity passed?
- Privacy, confidentiality, non-repudiation at each hop?
- How are federated identities resolved?
- How to avoid vendor lock-in?

Solution 1: WS-* Specifications

- OASIS Security Assertion Markup Language (SAML) 2.0
- OASIS WS-Security, WS-I BSP
- W3C WS-Addressing (released 5/9/06!)
- WS-Policy (submitted to W3C)
- 'WS-SX' (submitted to OASIS)
  - WS-Trust
  - WS-SecureConversation
  - WS-SecurityPolicy
- Composable framework

Solution 2: Liberty Alliance Project Specifications

- Identity Federation Framework (now subsumed in OASIS SAML 2.0)
- Identity Web services framework 1.1
- Built on existing industry standards: WS-Security, SAML, XML Signature, XML encryption etc.
- Built-in privacy
- Late binding via the discovery service
- “Data Services Template” to build web services
- Interoperable: Mandatory Conformance Program

Liberty Architectural Modules

Identity Federation Framework (ID-FF)
Enables Identity federation and management via identity/account linkage, SSO, and session management

Identity Service Interface Specifications (ID-SIS)
Enables interoperable Identity-based web services through schema and service interface specifications

Identity Web Services Framework (ID-WSF)
Provides the framework for building interoperable identity-based Web services

SAML | HTTP | WSS | WSDL | XML Enc

WAP | XML | SOAP | SSL/TLS | XML Sig

Agenda

Mobile Web Services: The Problem

Identity-Enabling Web Services

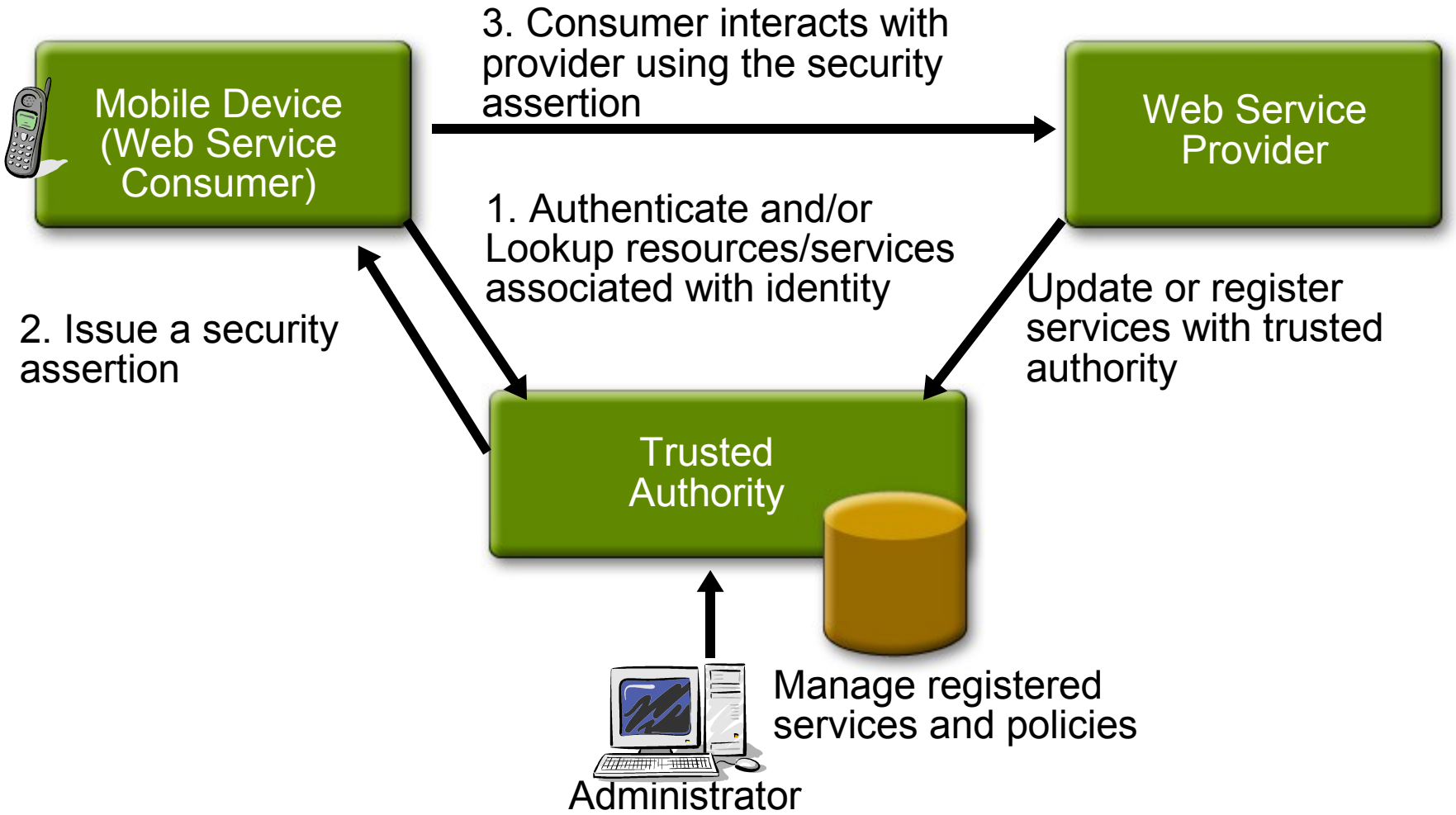
Java ME Technology JSRs: 172, 177, 279

Putting It Together

Demo

Next Steps

Identity-Enabled Web Services



Agenda

Mobile Web Services: The Problem

Identity-Enabling Web Services

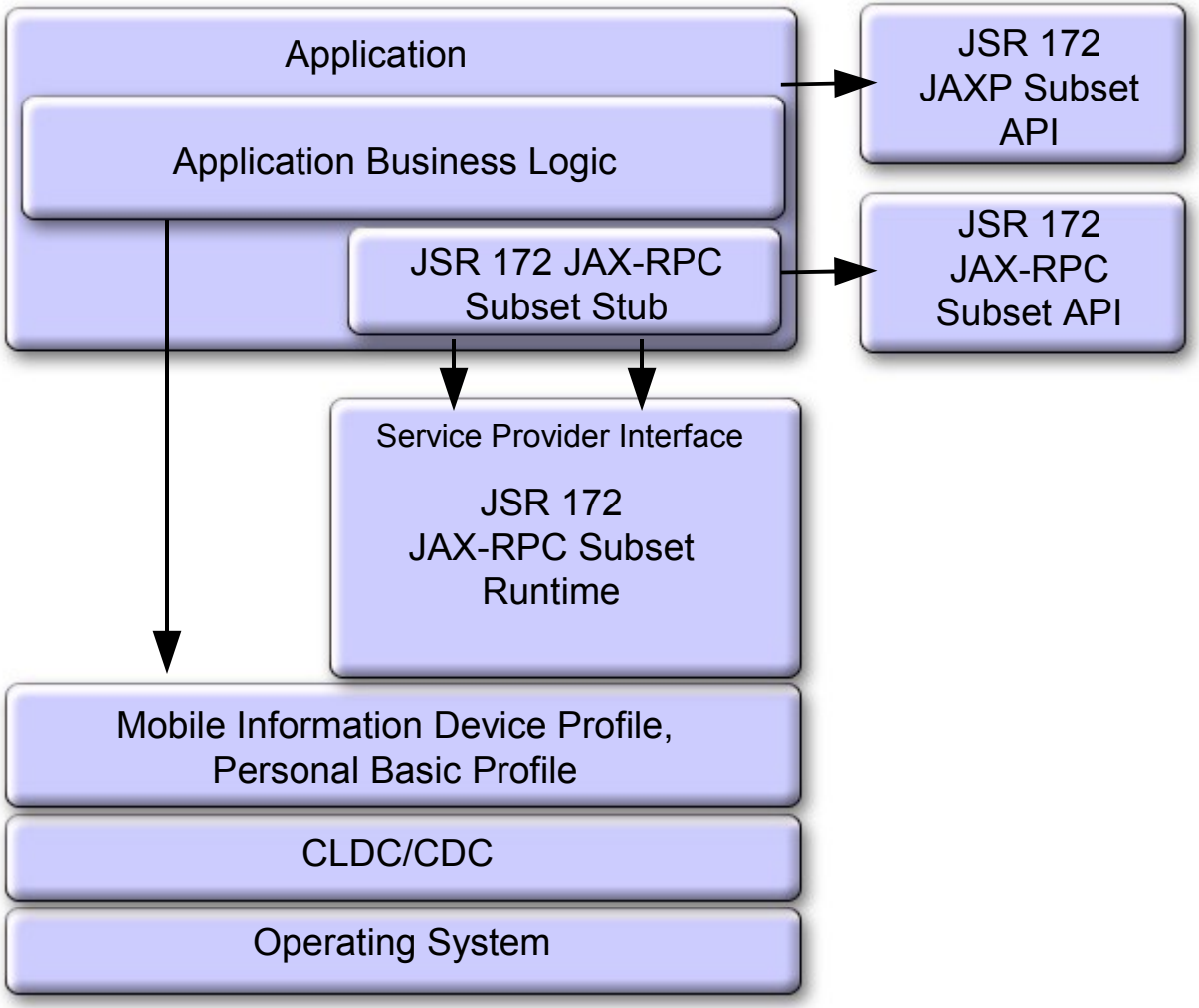
Java ME Technology JSRs: 172, 177, 279

Putting It Together

Demo

Next Steps

JSR 172: XML + Web Services



JSR 172: Programming Model

- Generate a JSR 172 Java API for XML-based remote procedure calls (JAX-RPC) stub class from a WSDL XML document that describes a remote Web service
- In your code, create an instance of the generated stub
- After instantiation, invoke methods of the generated stub; these methods correspond to the service endpoint's wsdl:operation in the WSDL XML document

Code Sample

```
// Instantiate the service stub.
PubService_Stub service = new PubService_Stub();
// Set up the stub.
service._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,
                     serviceURL);
service._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,
                     new Boolean(true));
...
try {
  // Invoke the PubService method getArticleByID()
  WirelessArticle article = service.getArticleByID(articleID);

} catch (RemoteException e) {
    // Handle RMI exception.
}
```

JSR 177: Security and Trust Services

- Smart card access and cryptographic capabilities to applications running on small devices; the SATSA specification defines four distinct APIs:
- SATSA-APDU: communicate with smart card applications using a low-level protocol
- SATSA-JCRMI: communicate with smart card applications using a remote object protocol
- SATSA-PKI: use a smart card to digitally sign data and manage user certificates
- SATSA-CRYPTO cryptographic API for message digests, digital signatures, and ciphers

Java ME Technology Limitations

- XML signing and encryption spec missing; some third-party vendors provide these APIs
- No JAX-RPC handler support
- JSR 279 is still under development

Agenda

Mobile Web Services: The Problem

Identity-Enabling Web Services

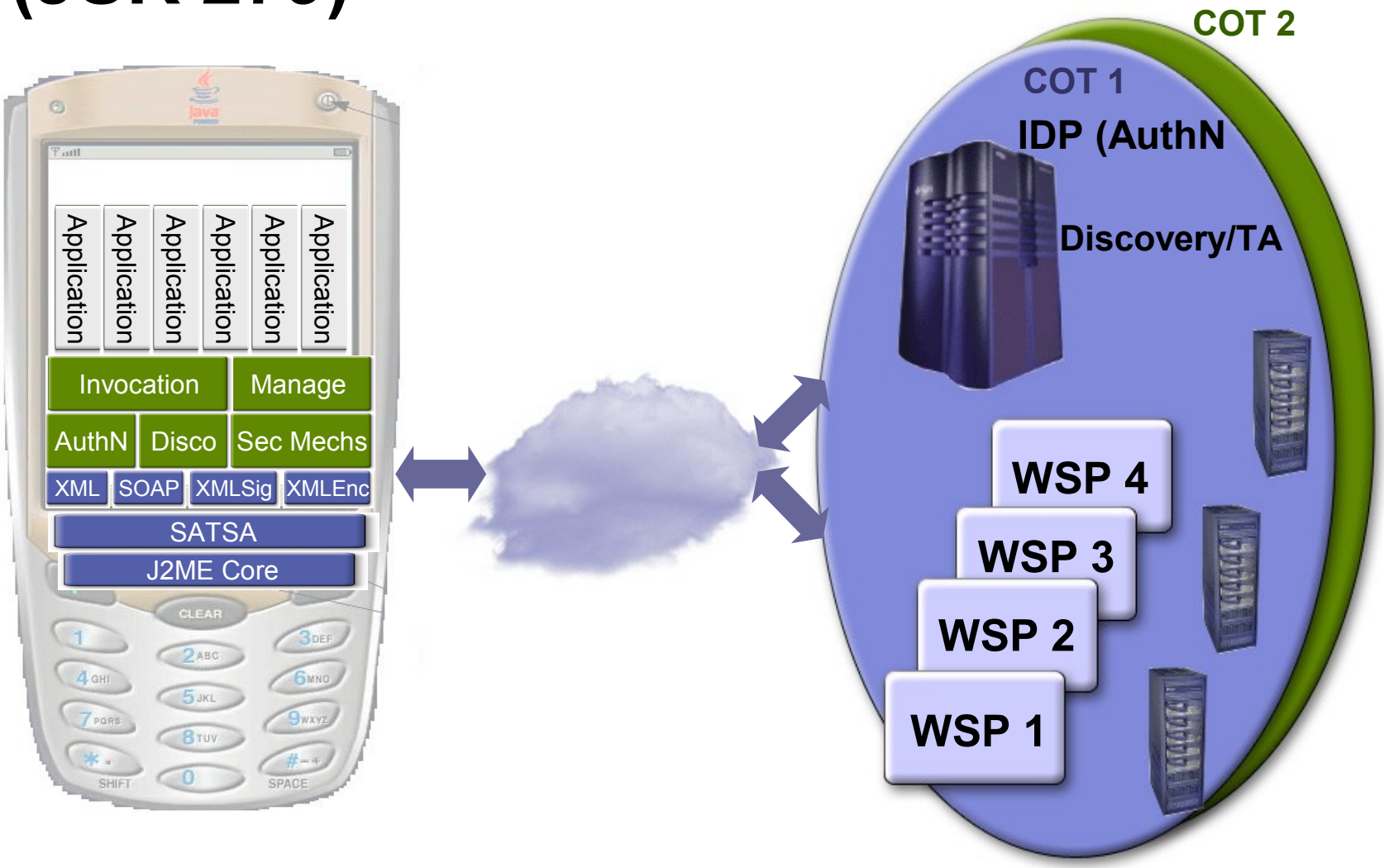
Java ME Technology JSRs: 172, 177, 279

Putting It Together

Demo

Next Steps

Web Services Security Framework (JSR 279)



Leveraging What Exists Today

- Provision user certificate on device [JSR177]
- Use PKI/Cert Authentication for authenticating user to IDP/ID-WSF AuthN service [JSR 172,ID-WSF]
- Obtain bootstrap containing BEARER SAML token [JSR 172, ID-WSF Authn]
- Query Disco web service for WSP to be accessed [JSR 172, ID-WSF Disco]
- Obtain BEARER SAML token [JSR 172, ID-WSF]
- Invoke WSP [JSR 172, ID-WSF SIS]
- Obtain and consume response [JSR 172]

Possible Enhancements

- ClientSSL for all WSC → WSP calls [JSR 177]
- Cryptographically tying the Bearer token with SOAP body with SATSA Crypto: proprietary, both ends have to understand signed/encrypted data
- UserID/Passwd authentication + userid/passwd basic authentication during SOAP invocations could be used in place of PKI

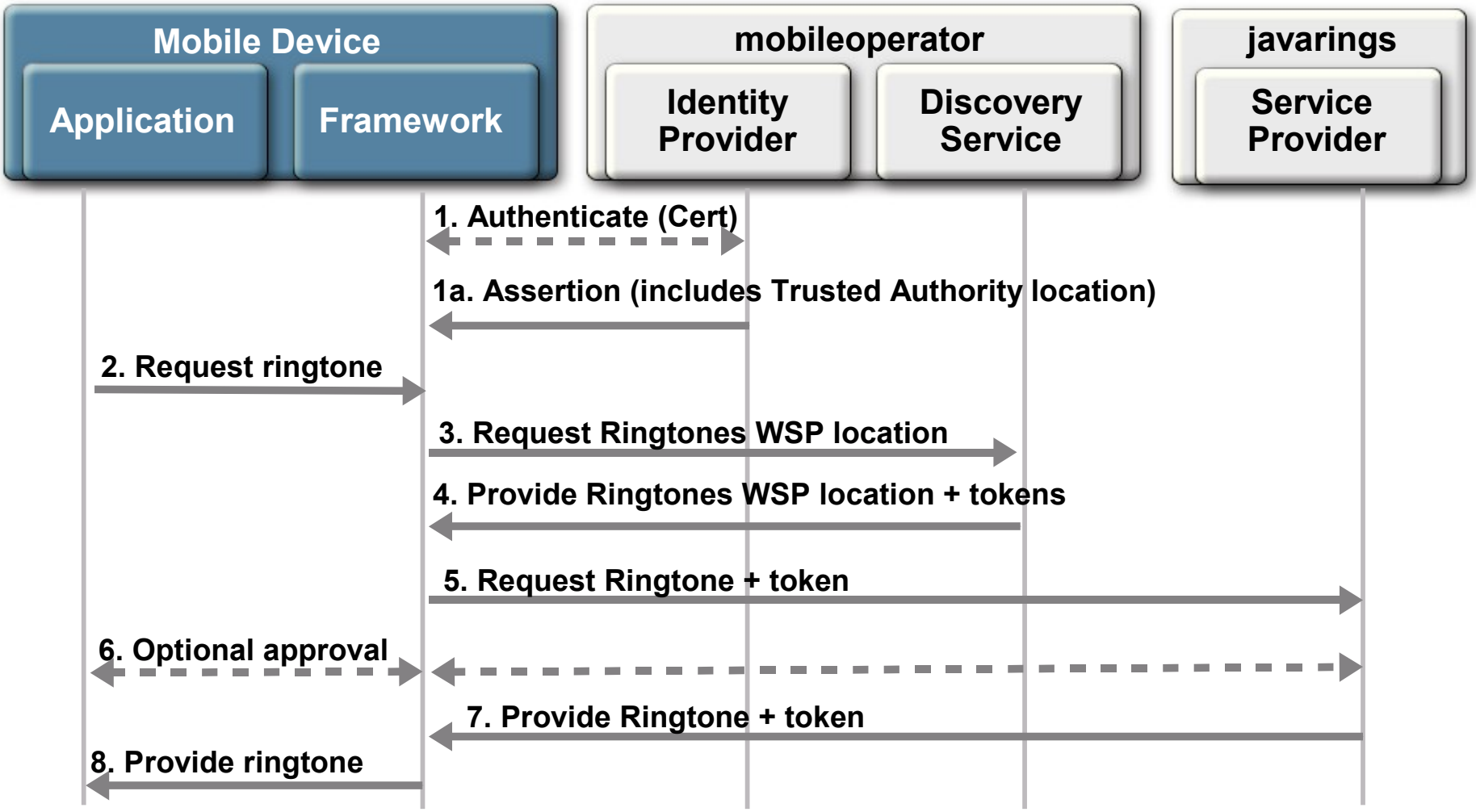
DEMO

Buying Ringtones and View Bill

Demo Summary

- Prerequisites:
  - COT setup:
    - www.mobileoperator.com: IDP+AuthN+Disco+Billing WSP
    - www.javarings.com: Ringtones WSP
  - User's mobile provisioned with certificates from a CA trusted by mobileoperator.com
- Sample MIDP app on phone demonstrating:
  - Cert based user authn with mobileoperator.com
  - A ringtone purchase from www.javarings.com
  - Billing query to see current bill
- Key points to note:
  - AuthN ONCE: Web services "SSO"
  - Secure access + No personal/identifiable info to WSP

What's Happening



Other Interesting Use Cases

- Multi COT access from the same device
- Infocard-like user interface to select IDP's/ credentials to supply based on user choice
- PAOS usage—obtaining profile/preference attributes stored on the phone

Agenda

Mobile Web Services: The Problem

Identity-Enabling Web Services

Java ME Technology JSRs: 172, 177, 279

Putting It Together

Demo

Next Steps

Java ME Technology Enhancements Needed!

- Good news: mobile devices are becoming more powerful, can accommodate larger footprint
- XML Signing and Encryption needs to be addressed
- JAX-RPC—SOAP header manipulation apis need to be provided to ease development
- Common Liberty + WS-*based service invocation framework needs to be in place (JSR 279)
- Isolation/sandboxing between disparate apps on the same device needs to be looked into
- Infocard equivalent for Java ME technology compliant devices

Summary

- Importance of security and privacy in mCommerce applications
- Liberty Alliance Project and Java ME technology specs together addressing the problem today
- Call for future enhancements needed

For More Information

- Identity @ Sun—http://www.sun.com/identity
- Liberty Alliance Project—http://www.projectliberty.org/
- Project Tango—http://wsit.dev.java.net/
- JSRs
  - http://www.jcp.org/en/jsr/detail?id=172
  - http://www.jcp.org/en/jsr/detail?id=177
  - http://www.jcp.org/en/jsr/detail?id=279
- Superpatterns—http://blogs.sun.com/superpat

Q&A

Federated Web Services with Mobile Devices

Rajeev Angal

Architect
Sun Microsystems

Pat Patterson

Architect
Sun Microsystems

Session TS-6673