



JavaOne™
Sun's 2003 Worldwide Java Developer Conference

Document-Centric Web Services

Michael Leventhal

XML Product Director, Tarari

Sen Zhang

Senior Principal, Dante Consulting

Agenda

- Why document-style SOAP?
- Document-centric Web Services defined
- Best practices for building high-performance document-centric Web Services
- Suggest improvement for JAXM

Challenges at Enterprise Level

- Complex, diverse and large payload
- Loosely coupled service providers and consumers required
- End-to-end, with multiple hops and intermediaries
- Multiple transports and constrained by security policy
- Asynchronous messaging involved
- Reliable transfer of mission-critical data

SOAP Styles Compared

RPC Style	Document Style
<ul style="list-style-type: none">• Emulate function calls• Payload is interpreted with ex-schema encoding style rules• Fine-grained• Synchronous programming model (blocking call)	<ul style="list-style-type: none">• Exchange business documents• XML payload defined by schemas but can carry binary data as well• Coarse-grained• Asynchronous messaging

Document-Centric Web Service Defined

- A message exchange paradigm
- Not just a SOAP style
- Focus on processing of business documents
 - Creation and consumption
 - Packaging
 - Transmission
 - Transformation
- Asynchronous, coarse-grained and loosely coupled services

Selected Best Practices In Building Document-Centric Web Services

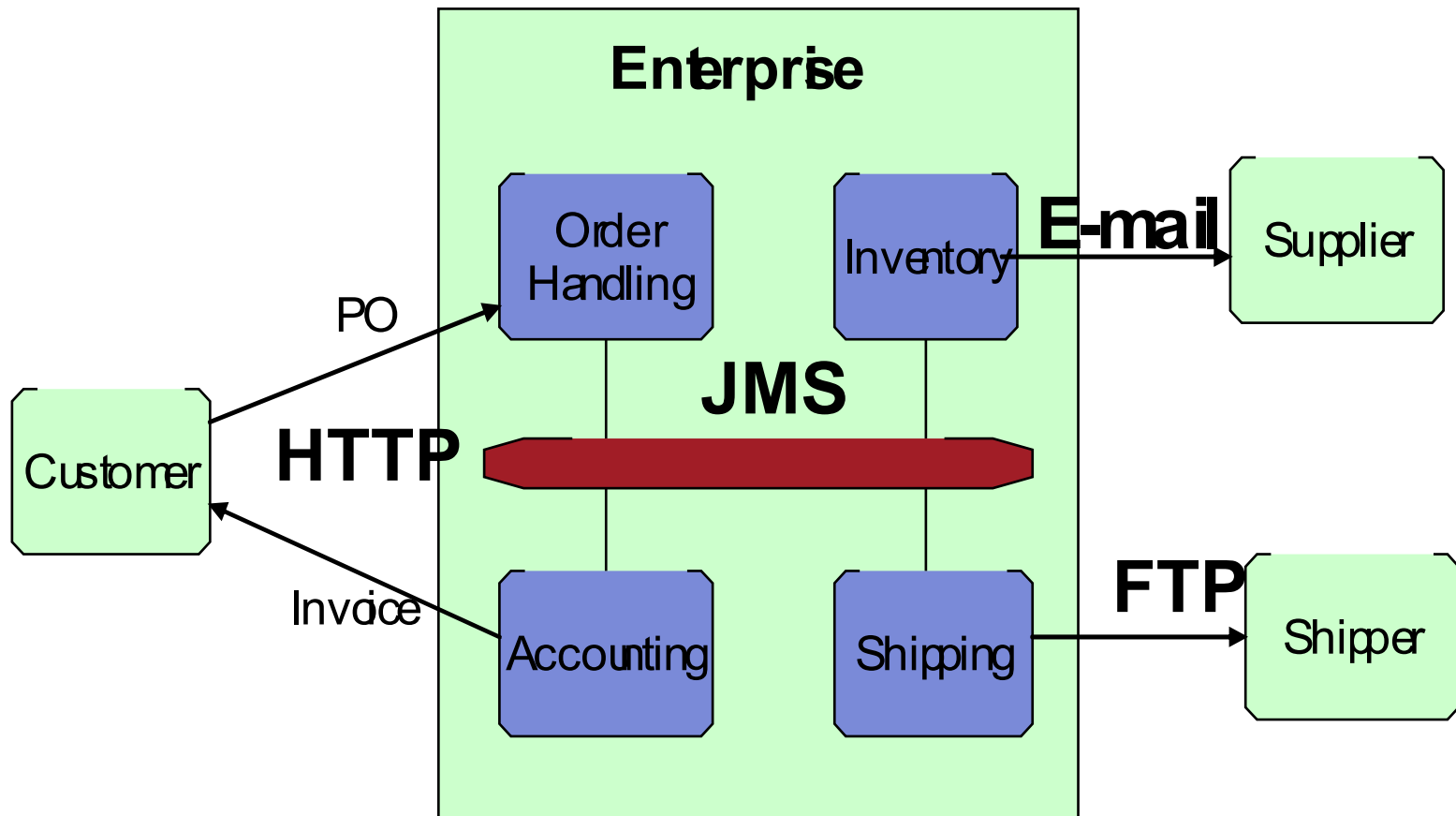
- Transport selection
- Large XML document handling
- XML data validation
- Strong document support

Transport Selection

- Available: HTTP, SMTP, JMS, FTP,...anything!
- Selection criteria:
 - Align with WWW technologies
 - Existing IT support and firewall policy
 - Interoperability, performance and reliability
 - Messaging pattern
 - Trading partners' technology
 - Timeline and skill set

Transport Selection

Case study



Large Data Handling

Best practices

- Apply “lazy” principle to SOAP message processing
 - Keep XML data in stream format as long as possible
 - No all applications need tree-like objects
 - Off-load expensive parsing from SOAP server
- Use the stream-based processing enabled by SAX
- Stream-based XPath and XML transformation technologies are available

XML Validation

Issues

- XML validation is needed to ensure integrity of message payload and adherence to business rules
- Need to retrieve schemas or DTDs for XML Validation (entity resolution)
- Schema retrieval can be a major source of reliability problem

XML Validation

Pitfalls

- **Don't!** Hard code locations to local disk in the XML instance:
 - Not deployable to new local environment
 - Sender and receiver do not share disks over the Internet
- **Danger!** Retrieve from the Internet:
 - Unreliable
 - High latency

XML Validation

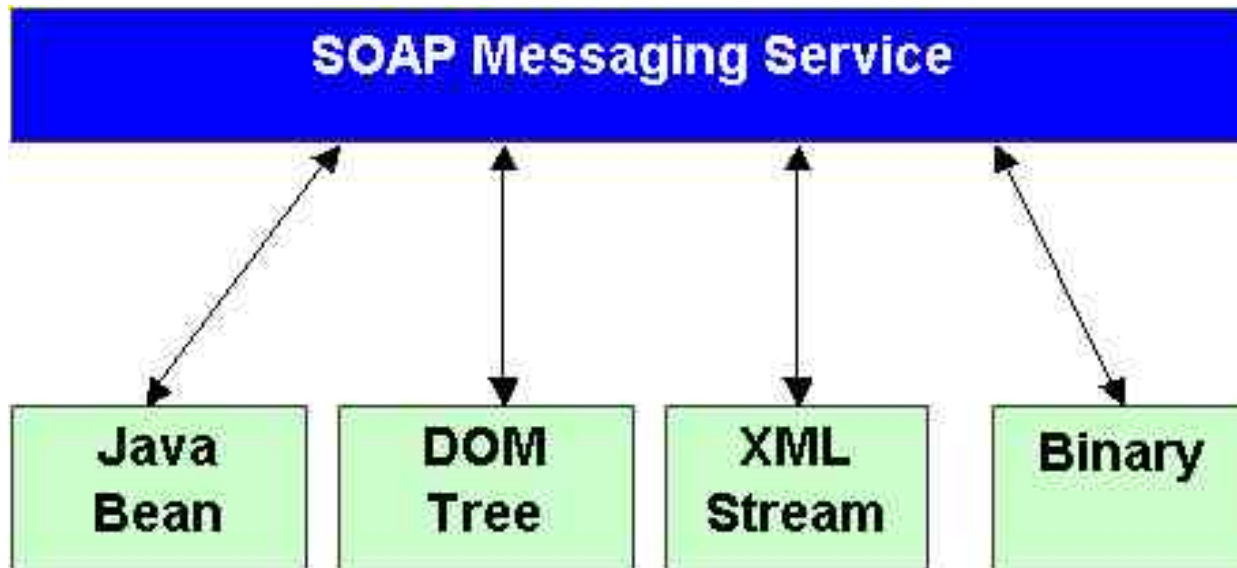
Techniques

- Adopt URI-based namespace for schema design
- Cache schemas and DTDs in local controlled storage: file system, database or LDAP
- Develop customized entity resolution:
 - Feature enabled by JAXP's Entity Resolver interface
 - Map namespace to schema cache
 - Chain entity resolvers to provide “fall-back” support

Document Support

Issues

- There are many possible document representations for XML payloads.
- Each representation has performance and flexibility implications.



Document Support

Usage Techniques

- Choose the least expensive document format that fits your need
 - Binary: least expensive, suitable for passthrough or archiving applications
 - XML Stream: Can be used to test well-formedness, validation, stream-based query, but content manipulation may be tedious
 - DOM Tree: Flexible content manipulation, full XPath support, expensive
 - Java Beans: Most convenient but most expensive

Document Support

Framework Techniques

- User should be able to add any document representation into the message
- Make no assumption on application need, use the least expensive document format
- Support multiple document representations
- Support format conversion and XML document validation “on demand”

What is JAXM?

- Java API for XML Messaging
- Packages: *java.xml.soap* and *java.xml.messaging*
- SOAP message construction via SOAP with Attachments API for Java™ (SAAJ)
- Point-to-point SOAP messaging support
- End-to-end, reliable and asynchronous messaging support via “remote providers”

Potential Improvements for JAXM

- Need better document support
 - Message construction has to follow tree-like object model composed of SOAPElements
 - No support for multiple XML payload representations enabled by: DOM, SAX or JAXB
 - Violate “lazy evaluation” principle and poor performance handling large payloads

Potential Improvements for JAXM

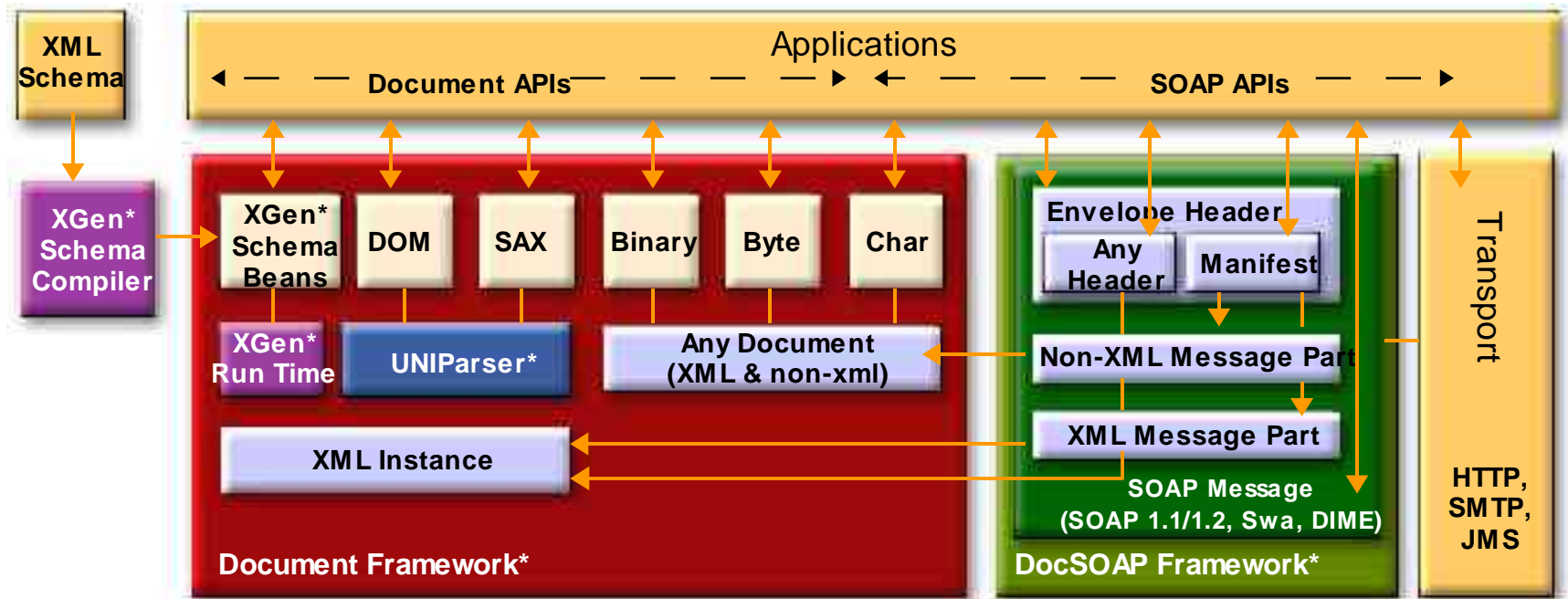
- Need binding independent API for SOAP message construction
 - SAAJ statically bond to SOAP with Attachments (Multipart MIME) model
 - Exclude alternative binding formats such as DIME

Introducing DocSOAP XDK

- Advanced XML and SOAP tools for document-centric Web Services
- Applied and enabling best practices
 - Stream-based handling
 - Manifest-driven SOAP API
 - Uniform document framework
 - Schema-Java mapping
 - Chained entity resolver
- Open-source

<http://www.commerceone.com/developers>

DocSOAP XDK Architecture



* Conductor DocSOAP XDK Component