REST and WS-*: Myths, Facts and Lies

Paul Fremantle
CTO, Co-founder, WSO2
paul@wso2.com

About me

- Paul Fremantle
  - 17 years experience in Software and Middleware
  - CTO and Co-founder of WSO2 – the open source SOA middleware company
  - Member of the Apache Software Foundation
  - Chair, Apache Synapse PMC
  - Co-chair of the WSRX Technical Committee at OASIS
  - Previously a Senior Technical Staff Member at IBM

What is WS-*

- The set of specifications proposed through
  - W3C, OASIS, WS-I
- SOAP, WSDL, WS-Security, etc
- Supported by IBM, Microsoft, BEA, Tibco, etc
- Designed as a technical implementation of Service Oriented Architecture

A Sample SOAP Message (cont)

```
<soap:Envelope xmlns:soap="http://
  schemas.xmlsoap.org/soap/envelope/">
 <soap:Header/>
 <soap:Body>
  <getProductDetails          xmlns="http://
  warehouse.example.com/ws">
     <productID>827635</productID> </
  getProductDetails>
  </soap:Body>
</soap:Envelope>
```

Myth #1

You need WS-* to implement a Web Service

Fact #1
In many cases HTTP is good enough

- If you have simple requirements, then SOAP is overkill
  - If you just need point-to-point encryption and username/password authentication then XML/HTTP works fine
  - If you have <soap:Header/> with nothing in it, then SOAP isn’t getting you any benefit

What is REST?

- REpresentational State Transfer
  - Coined by Roy Fielding in his PhD thesis
  - Identified as the “true architecture of the web”
- The basic concept is that everything is a “Resource”
  - The HTTP verbs allow transfer of a specific representation (e.g.HTML, XML) of the resource
    - POST, GET, PUT, DELETE
    - Create, Read, Update, Delete

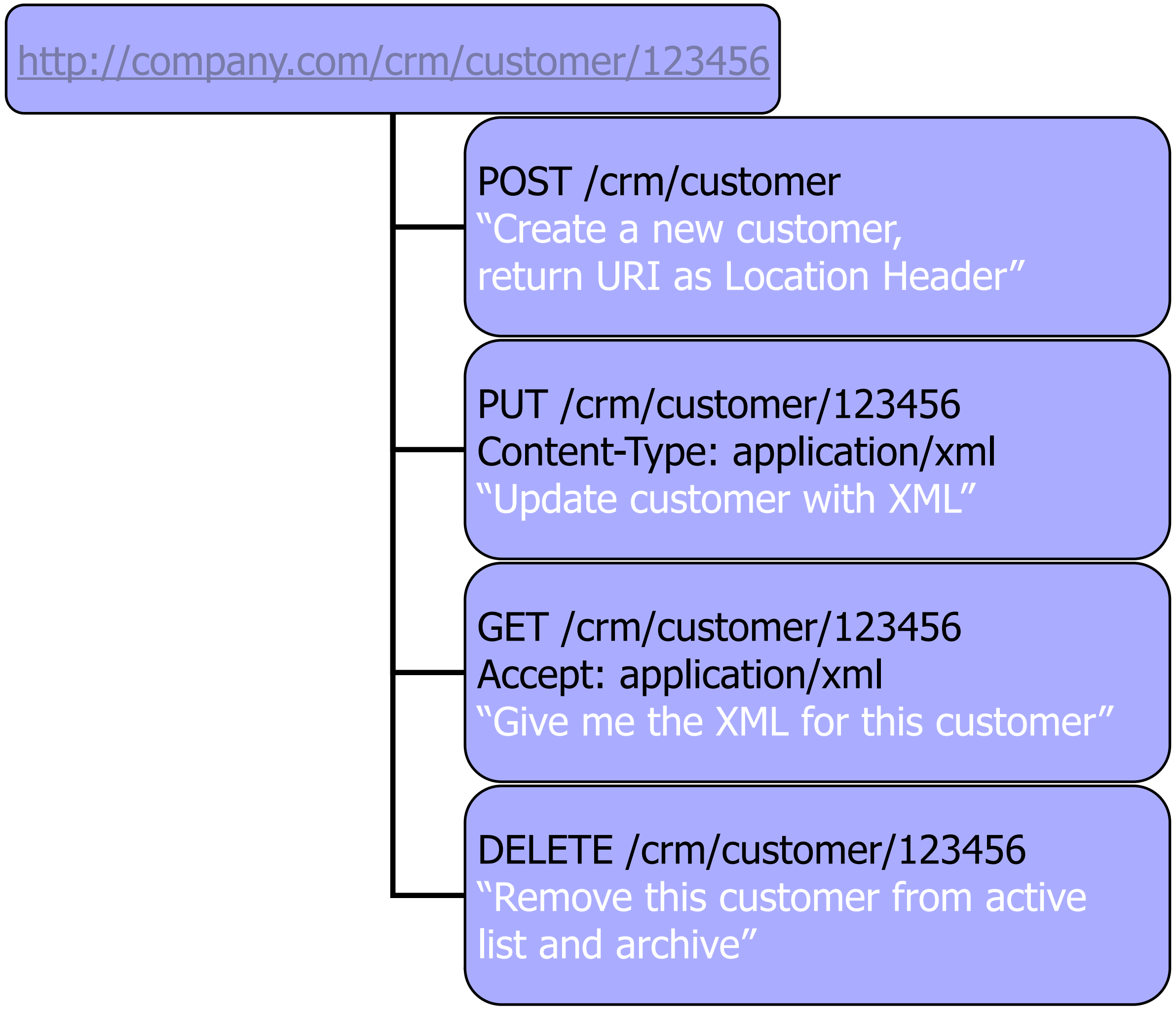
Myth #2

REST is simple

Digging into REST some more

- Everything is a Resource, identified by a URI
- Everything has a Uniform Interface (PUT, POST, GET, DELETE)
- The representation you get is based on Content-Type
  - e.g. text/xml, image/jpeg
- Interactions are stateless
- Links are key!
  - “Hypermedia as the engine of application state”

An example



http://company.com/crm/customer/123456

POST /crm/customer
"Create a new customer,
return URI as Location Header"

PUT /crm/customer/123456
Content-Type: application/xml
"Update customer with XML"

GET /crm/customer/123456
Accept: application/xml
"Give me the XML for this customer"

DELETE /crm/customer/123456
"Remove this customer from active
list and archive"

FACT #2 REST is full of subtleties

- Method Safety
  - GET, HEAD, OPTIONS, TRACE will not modify anything
- Idempotency
  - PUT, DELETE, GET, HEAD can be repeated and the side-effects remain the same
- Caching
  - Correct use of Last-Modified and ETag headers
- Content-negotiation
  - In theory, Accept headers allow this, in practice, it doesn’t work well

Myth #3

REST must be the most scalable, powerful and best model because the whole Web is REST

Fact #4 - most Web Applications are not REST

- You can't bookmark them
- Most application flow is completely based on session scope and form parameters
- Many proxies block PUT requests
- POST is used as "get out of jail free"
- Hardly anyone implements ETags and Last-modified properly
  - not even Google Docs!

Sub-myth

- No-one actually uses SOAP for real stuff

eBay does 50,000,000 SOAP transactions a day (on the web with PowerSellers)

Hyatt does hotel bookings via SOAP over the web with partners

Windows Live links MSN Messenger to mobile gateways using SOAP and SecureConversation

UK website www.thetrainline.com gives partners train information via the web

FACT #5

Well-designed REST applications are very powerful

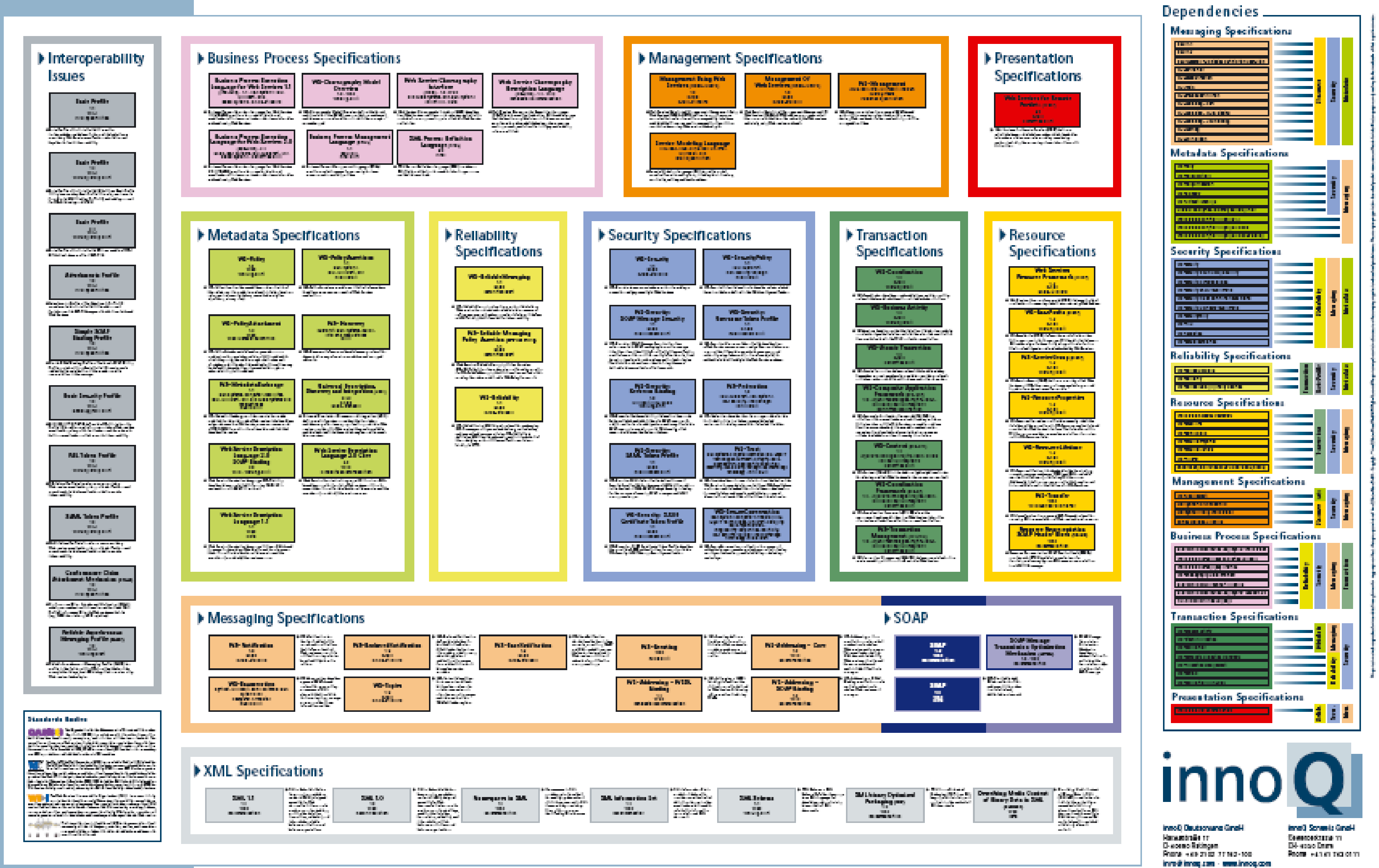
The benefits of a well-designed REST app

- Bookmarkability
  - Each URI really points to a unique entity
  - Every entity can be referenced
- Multiple representations are powerful
  - Allowing one view of a resource for users and one for systems makes application development simpler and more logical
- Having well defined links
  - Does improve the semantic richness of an application
  - By comparison WSDL is very flat and doesn’t show the links between operations and services

Myth #4

WS-* is far too complex

Web Services Standards Overview



Interoperability Issues
Business Process Specifications
Management Specifications
Presentation Specifications
Metadata Specifications
Reliability Specifications
Security Specifications
Transaction Specifications
Resource Specifications
Messaging Specifications
SOAP
XML Specifications
Standards Bodies
Dependencies
Messaging Specifications
Metadata Specifications
Security Specifications
Reliability Specifications
Resource Specifications
Management Specifications
Business Process Specifications
Transaction Specifications
Presentation Specifications
Version 3.0 · February 2007

innoQ Deutschland GmbH

innoQ Schweiz GmbH

info@innoq.com · www.innoq.com

Comparison: A few REST Specifications

- HTTP 1.0/1.1, PEP, HTML, XHTML
- Media Types, MIME, S/MIME
- JSR 311 – JARWS
- POST Once Exactly
- SSL/TLS
- URL, URI, URN, IRI
- WebDav, DeltaV
- XForms, XML, XML Schema, XPath, XSLT, CSS
- JSON
- WebAPI, XMLHttpRequest, AJAX, Comet
- RDDL, Microformats, GRDDL, etc…
- Atom, Atom Publishing Protocol, GData, etc…
- RFCs 1945, 2068, 2069, 2109, 2145, 2169, 2227, 2295, 2296, 2518, 2616, 2617, 2774, 2817, 2818, 2935, 2936, 2964, 2965, 3143, 3205, 3229, 3230, 3310, 4130, 4169, 4229, 4236, 4387, 4559, 4918…

Fact #6

- WS-* standards are *quite* complex
  - Still not enough “out-of-the-box” interoperability despite several years effort
- The WS-* standards offered too many choices
  - WS-I has done a reasonable job of cutting down the choices

Fact #6a:

- “Pay as you go”
- For example,
  - No need to understand WS-ReliableMessaging until you need assured delivery

Myth #5

You can use REST to implement any service

Fact #7
You need WS-* for interoperable security and reliability

- There is no commonly accepted REST model for:
  - Message Signing / Non-repudiation
  - Reliable Messaging
- There are some proposals
  - Mainly require modifying business logic and coding directly
  - Not implemented by any middleware solutions
  - WS-Security, SecureConversation and WS-RM are widely implemented and proven to interoperate

Fact #8: A standard WS-* profile is emerging

- SOAP
  - A transport agnostic messaging model
- WSDL and WS-Policy description
  - A framework for describing services
- WS-Addressing
  - A routing and addressing model
- MTOM
  - How to efficiently include binary data
- WS-Security/SecureConversation
  - Efficiently add encryption, signatures and authentication
- WS-ReliableMessaging
  - Assured delivery of messages

WS-I Reliable Secure Profile

http://www.ws-i.org/deliverables/workinggroup.aspx?wg=reliablesecure

Myth #6

SOAP is just RPC spelt in XML

Fact #9:
SOAP is a *messaging* specification

"SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns by combining such one-way exchanges"

SOAP 1.2 Primer, W3C

Fact #10: WS-* fully supports Asynchronous Messaging

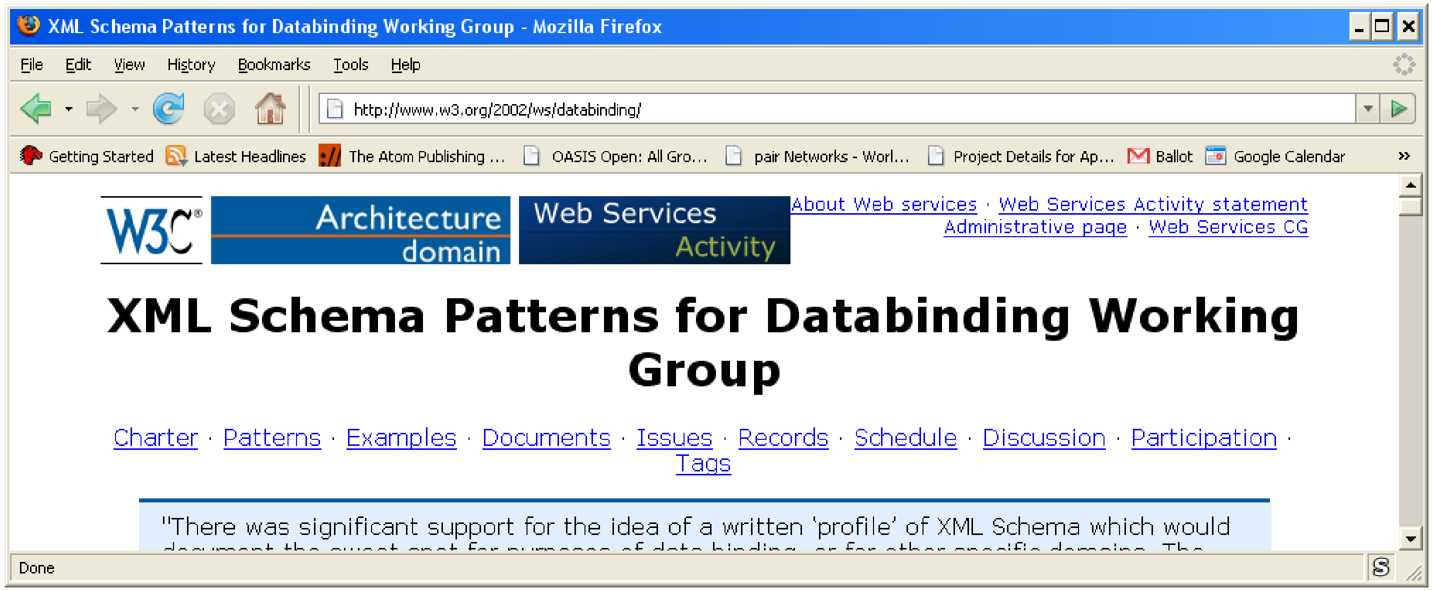
- WS-Addressing specification defines the concept of a ReplyTo
- Allows SOAP interactions to become long-running and more loosely coupled
- Asynchronous behaviour is an important factor in scalability and resilience

Sub-myth #6 - “You need WSDL to use SOAP”

- WSDL makes it easier for programmers to use remote services, but its not “normative”
- SOAP Web Services are “Duck Typed”
  - If you don’t like the WSDL they provide, use another equivalent one!
- WSDL is complex, but its also one of the most useful specifications when used wisely

Sub-Myth #6 (part 2)

- The problems of WSDL go away with REST
- The biggest problem is XML Schema binding
  - Doesn’t go away with REST
  - Although RELAX-NG offers a better option, Schema is still king


XML Schema Patterns for Databinding Working Group - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.w3.org/2002/ws/databinding/
Getting Started
Latest Headlines
The Atom Publishing ...
OASIS Open: All Gro...
pair Networks - Worl...
Project Details for Ap...
Ballot
Google Calendar
W3C
Architecture domain
Web Services Activity
About Web services · Web Services Activity statement
Administrative page · Web Services CG
Charter · Patterns · Examples · Documents · Issues · Records · Schedule · Discussion · Participation · Tags
Done

XML Schema Patterns for Databinding Working Group

"There was significant support for the idea of a written 'profile' of XML Schema which would

Another option to minimize binding issues

- Use XPath

```
invokeOperation(xml) {
   price = xml.xpath(//order/price);
   quantity = xml.xpath(//order/
   @quantity);
   …
}
```

Myth #7

You don’t need a description language – content negotiation is enough

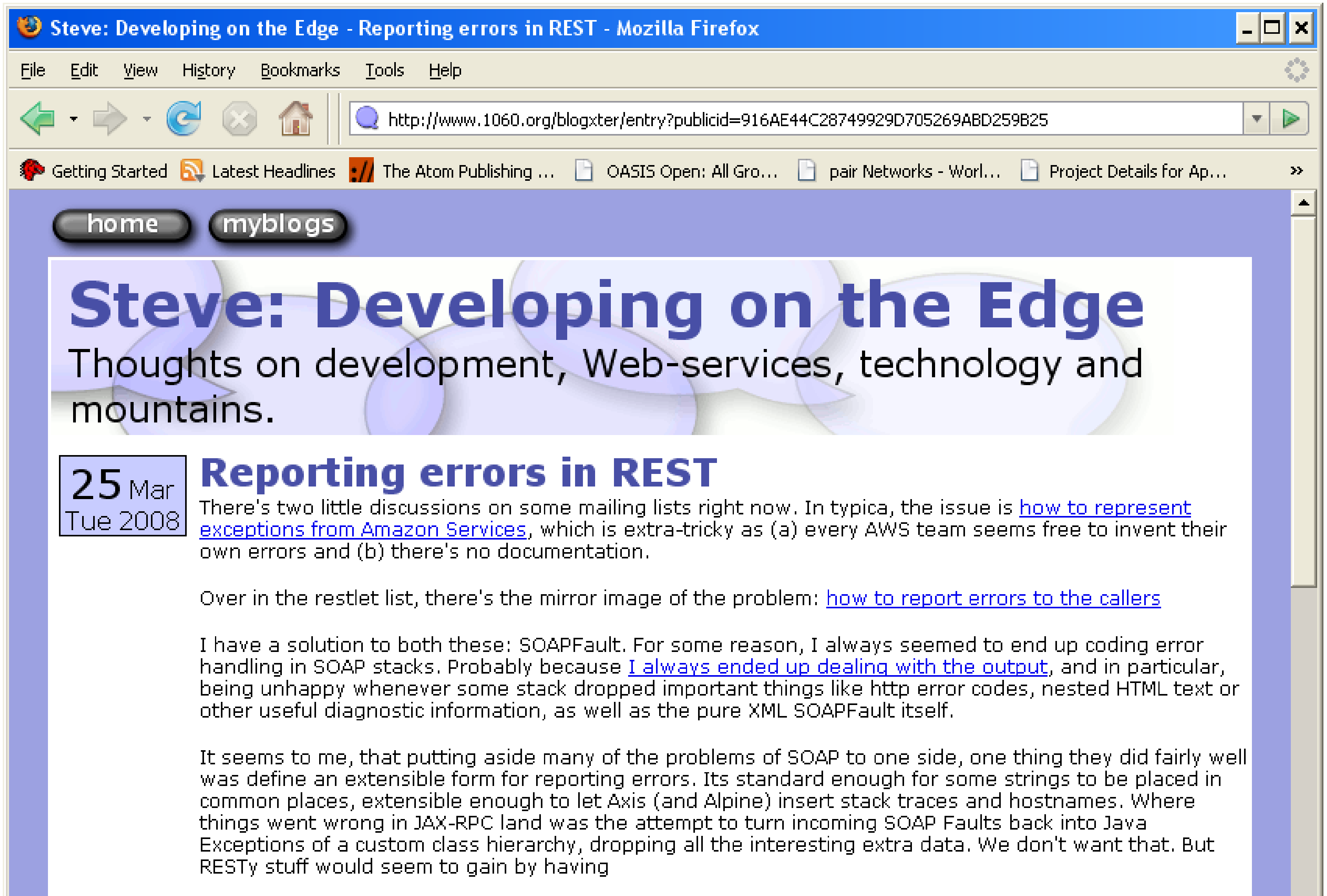
Fact #11: Content-type isn't enough

- Firstly, most XML types come as "application/xml"
- Content-type negotiation is not a successful aspect of REST
- No way of describing the linkages
  - "Hypermedia as the engine of application state"
- Very hard to replace a REST system because there is no well-defined interface specification
- Proposals to improve REST description:
  - WADL
  - WSDL 2.0 can be used to describe any HTTP application

WADL

```
<resources>
  <resource uri="http://.../NewsSearchService/V1/newsSearch">
    <operationRef ref="tns:NewsSearch"/>
  </resource>
</resources>
<operation name="NewsSearch" method="get">
   <request>
      <parameter name="appid" type="xsd:string" required="true"/>
    </request>
    <response>
    <representation mediaType="text/xml" element="yn:ResultSet">
       <parameter name="totalResults"
```

Errors in REST


Steve: Developing on the Edge - Reporting errors in REST - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.1060.org/blogxter/entry?publicid=916AE44C28749929D705269ABD259B25
Getting Started Latest Headlines The Atom Publishing ... OASIS Open: All Gro... pair Networks - Worl... Project Details for Ap...
home myblogs

Steve: Developing on the Edge

Thoughts on development, Web-services, technology and mountains.

25 Mar Tue 2008

Reporting errors in REST

There's two little discussions on some mailing lists right now. In typica, the issue is how to represent exceptions from Amazon Services, which is extra-tricky as (a) every AWS team seems free to invent their own errors and (b) there's no documentation.

Over in the restlet list, there's the mirror image of the problem: how to report errors to the callers

I have a solution to both these: SOAPFault. For some reason, I always seemed to end up coding error handling in SOAP stacks. Probably because I always ended up dealing with the output, and in particular, being unhappy whenever some stack dropped important things like http error codes, nested HTML text or other useful diagnostic information, as well as the pure XML SOAPFault itself.

It seems to me, that putting aside many of the problems of SOAP to one side, one thing they did fairly well was define an extensible form for reporting errors. Its standard enough for some strings to be placed in common places, extensible enough to let Axis (and Alpine) insert stack traces and hostnames. Where things went wrong in JAX-RPC land was the attempt to turn incoming SOAP Faults back into Java Exceptions of a custom class hierarchy, dropping all the interesting extra data. We don't want that. But RESTy stuff would seem to gain by having

Hypermedia as the Engine of Application State

- The links are what matters
- WADL provides a way of identifying links
- But "browseability" is the best model

Myth #8

HTTP is the only protocol you need

Lots of protocols

- Enterprise:
  - JMS, SMTP, TCP, IIOP, MQSeries, etc
- Cool:
  - Jabber/XMPP, YahooIM, SIP, etc

Fact #12

WS-* layers well on top of lots of protocols
For example, the Danish Government OIO project is using Secure Reliable SOAP over SMTP

Fact #13
Even for simple resource-oriented applications HTTP isn't enough

- Two initiatives prove it:
  - WebDAV, DeltaV
    - Extending HTTP to be used as a real repository for documents, code, etc
  - Atom Publishing Protocol
    - Simple way of publishing entries to a blog server
- In both cases, you need to extend the core model to support even simple publishing capabilites effectively

Sub-myth #8a

- REST is an architectural style that can be used with any protocol

Fact #14

HTTP is the only example

Myth #9

REST naturally allows caching and is therefore more scalable

Fact #15
Most applications can’t be cached

- HTTP *was* designed with Caching in mind
  - GET If modified, ETags
- BUT
  - As soon as you **secure** an HTTP connection with SSL/TLS then there is no caching
  - You can have caching OR security but not both

The big lie (#1)

Distributed computing is easy with {SOAP, REST, …}

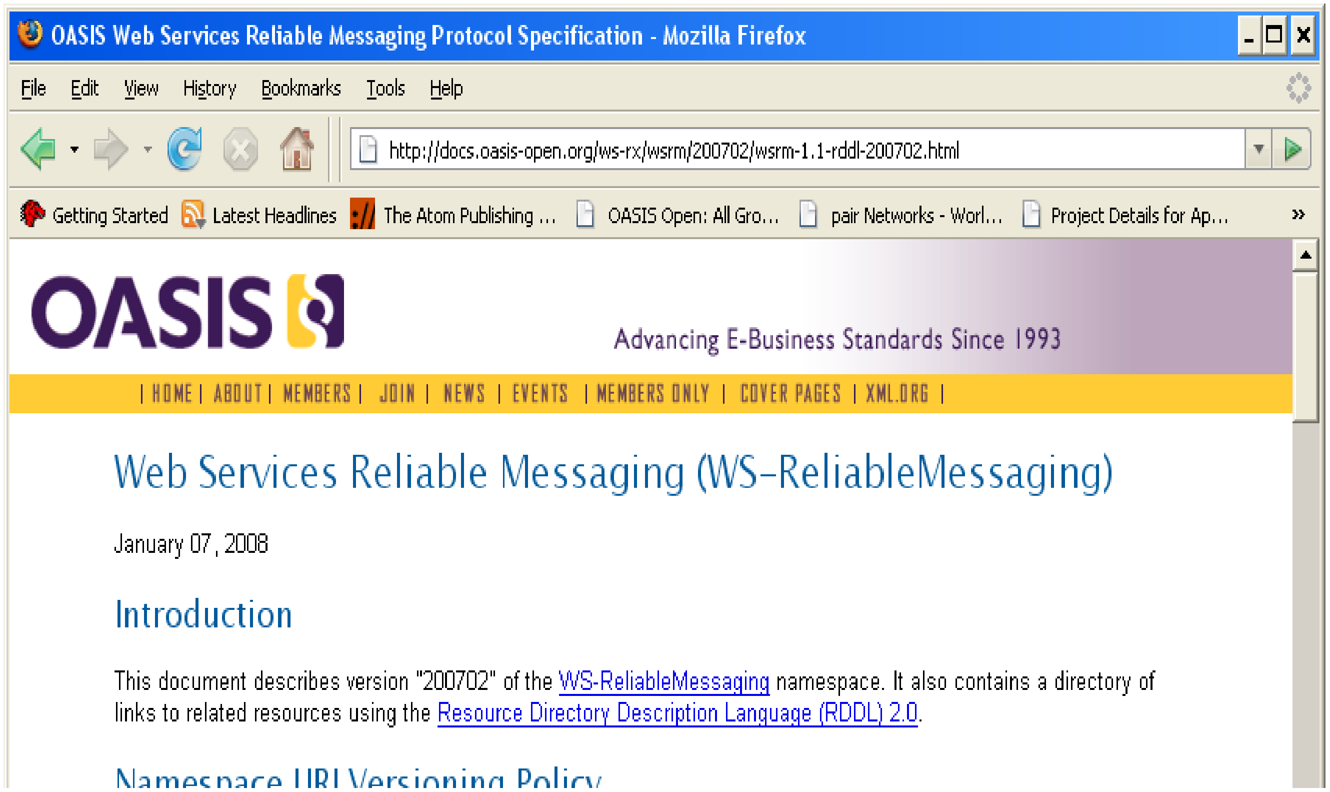
Fact #16
Distributed computing is hard

- Whichever approach you take you need to consider complex issues
    - Security
    - Reliability, latency, failure cases
    - Caching
    - Encoding
    - Description and discoverability
    - Mobility and re-implementability

My recommendations

- Stick to well known profiles
  - .NET, RASP, RSP, etc
  - AtomPub
- Use as much of the Web Architecture whether or not you are using WS or REST
  - e.g. RDDL
  - Make your URIs real
- Use what works

RDDL



OASIS Web Services Reliable Messaging Protocol Specification - Mozilla Firefox

http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-rddl-200702.html

Web Services Reliable Messaging (WS-ReliableMessaging)

January 07, 2008

Introduction

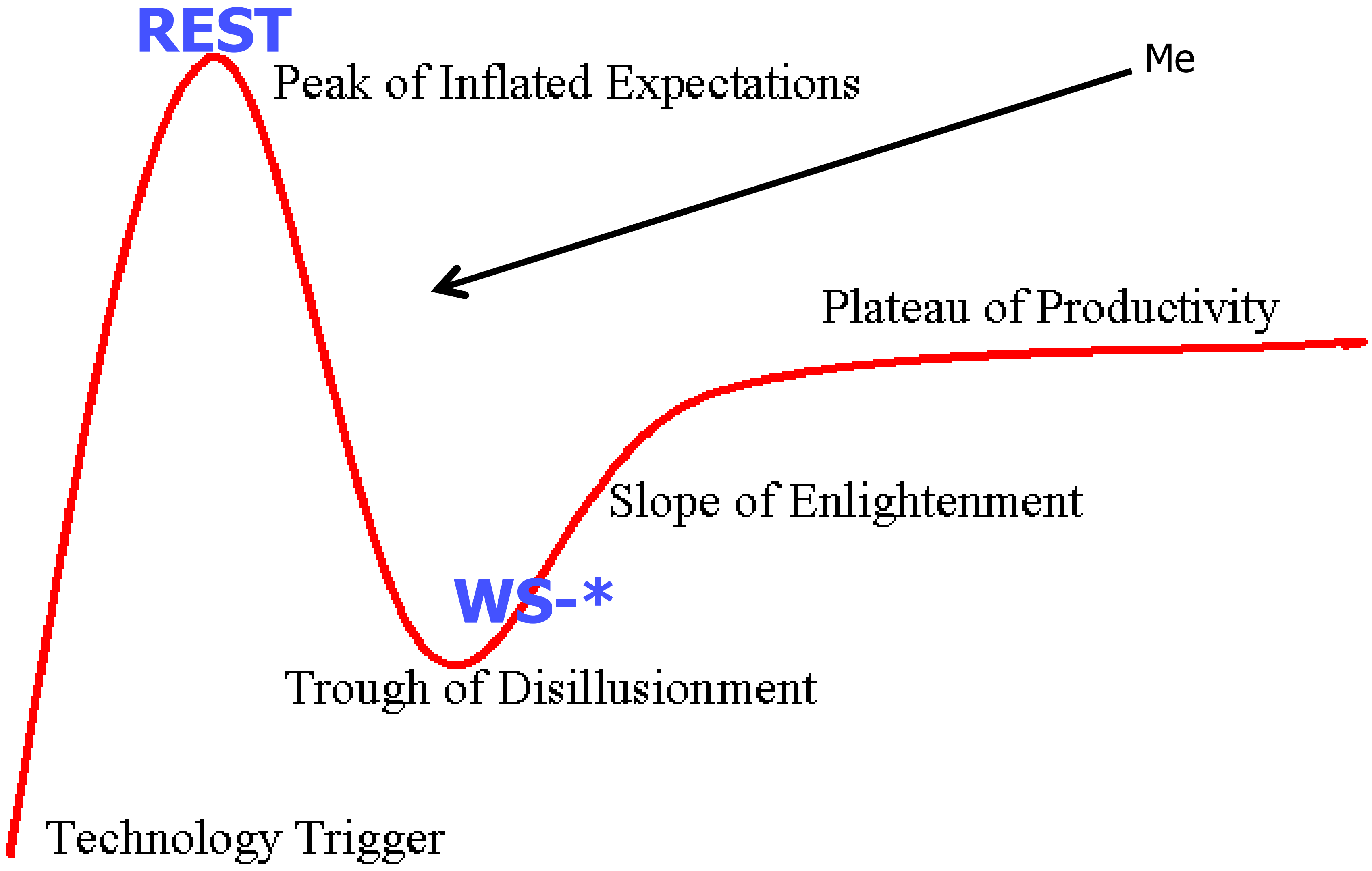
This document describes version "200702" of the WS-ReliableMessaging namespace. It also contains a directory of links to related resources using the Resource Directory Description Language (RDDL) 2.0.

Namespace URI Versioning Policy

File Edit View History Bookmarks Tools Help
Getting Started
Latest Headlines
The Atom Publishing ...
OASIS Open: All Gro...
pair Networks - Worl...
Project Details for Ap...
OASIS
Advancing E-Business Standards Since 1993
| HOME | ABOUT | MEMBERS | JOIN | NEWS | EVENTS | MEMBERS ONLY | COVER PAGES | XML.ORG |

```
<a href="http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-schema-200702.xsd"
 rddl:nature="http://www.w3.org/2001/XMLSchema"
 rddl:purpose="http://www.rddl.org/purposes#schema-validation">
         wsrm-1.1-schema-200702.xsd
</a>
```

Hype Cycle


REST
Peak of Inflated Expectations
Me
Plateau of Productivity
Slope of Enlightenment
WS-*
Trough of Disillusionment
Technology Trigger

Resources

- Roy Fielding's thesis
  - http://roy.gbiv.com/pubs/dissertation/top.htm
- InnoQ WS Poster
  - http://www.innoq.com/resources/ws-standards-poster/
- SOAP Primer
  - http://www.w3.org/TR/soap12-part0
- Atom Publishing Protocol
  - http://bitworking.org/projects/atom/draft-ietf-atompub-protocol-17.html
- RESTful Web Services, book by Leonard Richardson and Sam Ruby
  - http://www.amazon.com/Restful-Web-Services-Leonard-Richardson/dp/0596529260
- WS-I Reliable Secure Profile
  - http://www.ws-i.org/deliverables/workinggroup.aspx?wg=reliablesecure
- My blog
  - http://pzf.fremantle.org

Thanks for listening!