Institute of
Public Administration

COMP41530 - Web Services in Cloud Computing

Barry Corish
Associate Lecturer, IPA

Lecture 07

Overview

- Review
- UDDI
- JSON and XML

Overview

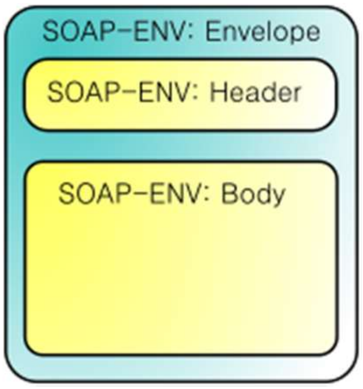
- **Review**
- UDDI
- JSON and XML

SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- Inter-system communication protocol for messages
- XML based, normally carried over HTTP
  – so ideal for Internet
- Vendor, platform and application neutral
- Standard for many Enterprise level WebServices

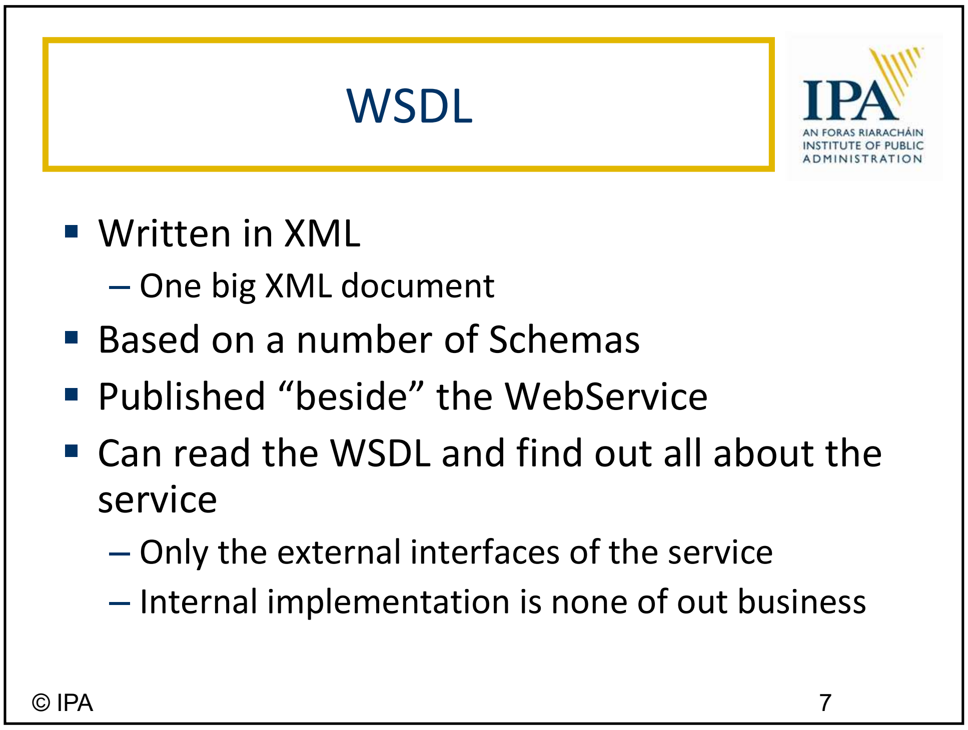
SOAP Envelope

- Each message is put into an "Envelope".



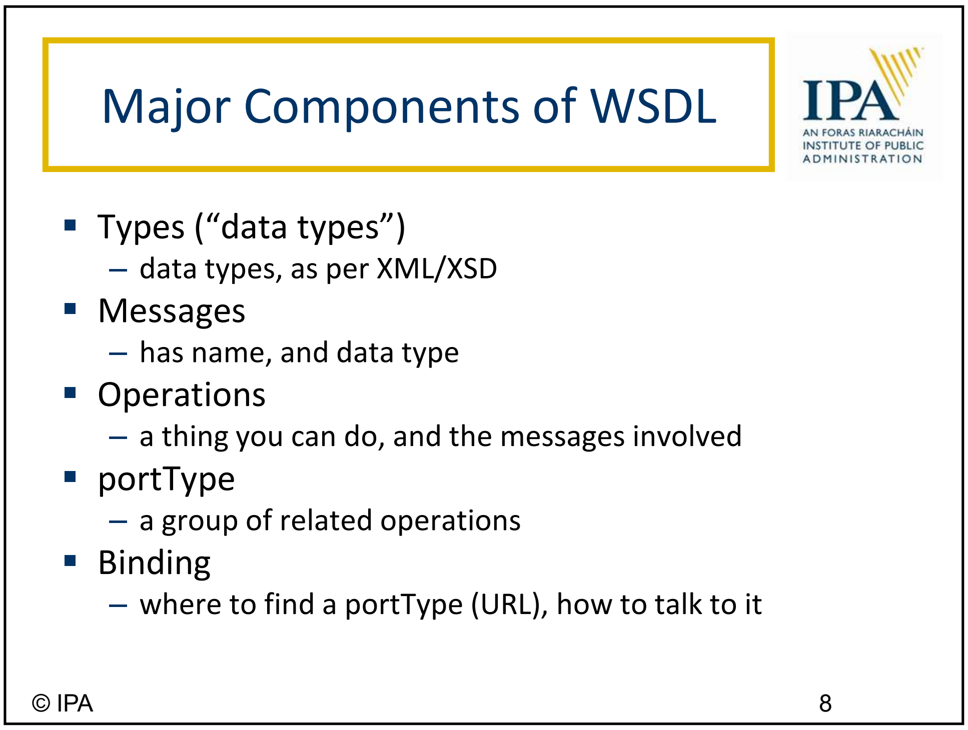
WSDL

- **W**eb **S**ervices **D**escription **L**anguage
- Describes:
  - What a service does
  - Where to find the service
  - How to use the service
- Written by the WebService provider
- Is the “contract” around the WebService



WSDL

- Written in XML
  - One big XML document
- Based on a number of Schemas
- Published “beside” the WebService
- Can read the WSDL and find out all about the service
  - Only the external interfaces of the service
  - Internal implementation is none of out business



Major Components of WSDL

- Types (“data types”)
  - data types, as per XML/XSD
- Messages
  - has name, and data type
- Operations
  - a thing you can do, and the messages involved
- portType
  - a group of related operations
- Binding
  - where to find a portType (URL), how to talk to it

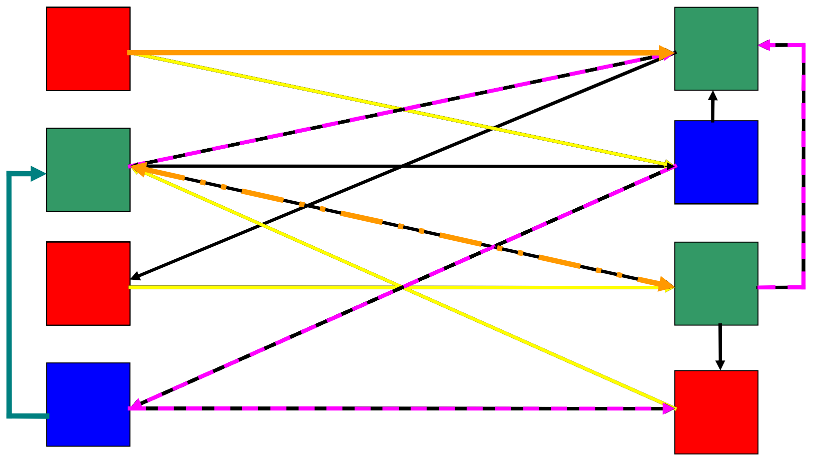
Questions?

?

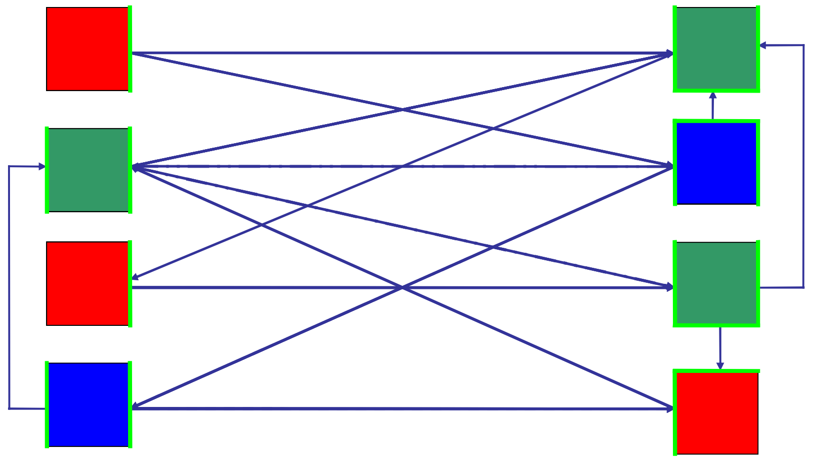
Overview

- Review
- **UDDI**
- JSON and XML

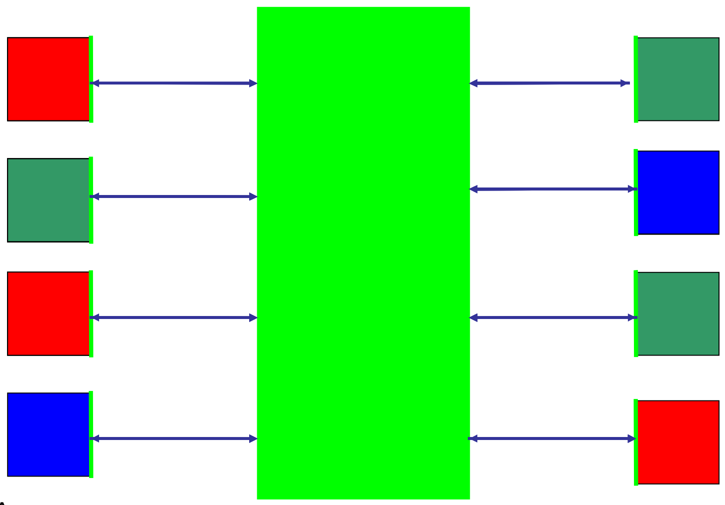
Where we started



Web Services



Web Services + Middleware



3 keys to good architecture

- Appropriate use of middleware
- Make the Services easy to find and use
  - One possible solution for this is UDDI – The "phone directory" for Services
- Proper Governance
  - Next Week

But what about WSDL?

- Doesn't that give us enough?
  - Where it is
  - What it does
  - Etc…
- Yes, but if we don't know where it is, how do we get the WSDL in the first place?

Service Discovery and Re-use

- WebServices may be spread around the cloud
- How to make them "discoverable"?
- Can only implement a true SOA if we can find services so they can be re-used
- If we build the same service twice because we don't know the first instance of it exists, then we've failed!

Be lazy!

"Progress is made by lazy men looking for easier ways to do things"

Robert A. Heinlein

- As with defining your own XML Schema…
- Before doing the hard work of building a WebService, see if someone else has done it first
  – Or has foundations or components available which you can extend or build on

Small Scale

- Easy to keep track of services at first:
  – Small numbers of services to be tracked
  – Just keep a list!
  – Rely on developer/IT Architect knowledge
  – When starting, fewer services to remember

Medium/Larger Scale

- Over time:
  - More services
  - Staff turnover, loss of knowledge
  - Likely to fail to reuse and develop again, unless tightly managed
  - They’re easy to reuse – they must be easy to find!
  - We need a registry of services

What does the Service Registry Store?

- Information about:
  - Businesses
  - Services
  - Technical information

Why do we do this?

- Maximise service re-use
- Allow us to manage and govern the services we have
- Particularly when growing fast!
- Provides "interfaces" to the information
- All key to successful SOA

What should a Service Registry provide?

- Allow services to be “registered”
- Allow search for services by:
  - Function
  - How to interact with them
  - Cost
  - Availability/speed/reliability/other QoS measures
- Provide necessary information to allow the services to be selected and used effectively
- Provide useful interfaces to all of the above

Type of Service Discovery (1/2)

- Static Service Discovery
  - Typical pattern
  - Most common with internal provided services
  - At design/build time, do search, make selection
  - "One off" selection
  - Build the selected service into application

Type of Service Discovery (2/2)

- Dynamic Service Discovery
  - Part of the application as built is to pick the service to be used "on the fly"
  - Mainly for "utility" type services
    - Card Payment Processing
    - Computation
    - Storage
  - Pick whoever is cheap/fast/available today

UDDI

- Universal Description, Discovery and Integration
- A standard for Service Registries
- Designed for use by Developer Tools and applications
  - Not designed for direct usage by humans

UDDI

- UDDI is:
  - XML Based
  - Vendor Neutral
  - Open Standard

UDDI Core Elements

- White Pages
  - Details of Organisations
- Yellow Pages
  - List of Business Services provided by the Organisations
  - Based on industry "standards"
- Green Pages
  - Details of WebServices providing the Services

UDDI Core Elements

- White Pages
  - Who you are
- Yellow Pages
  - What you do
- Green Pages
  - How to work with you

History

- Started around 2000
- Was seen as a global directory
- Several large public directories set up
- Intended to allow organisations find business partners
- Failed to deliver at that scale
- Never took off
- Died off on public basis around 2008

Types of Service Registry

- Public
- Affiliate Group
- Internal
- Internal with external exposure

Types of Service Registry

- Full Public (Dead)
- Affiliate Group (Common)
- Internal (Common)
- Internal with external exposure (Occasional)

Current UDDI Usage

- Inside an organisation
  - Often implemented as Green Pages only
  - Only "Skeleton" Yellow and White pages
- Many larger implementations are seen in Public Service/Government
  - Suits larger scale, but not "public" use

Not always necessary!

- Use if:
  - Large numbers of services
  - Lots of organisational units involved
  - High rate of change in services
  - Services are commodities

UDDI Data Structures

- businessService
- businessEntity
- bindingTemplate
- tModel
- publisherAssertion
- subscription

Service Provider information

- businessEntity (White Pages)
- Contains:
  - businessKey (unique)
  - Name
  - Description
  - Contact(s)
  - Business Services (Green Pages)
  - Identifier Bag
  - Category Bag (Yellow Pages)

Business service information

- businessService (part of businessEntity)
- Contains:
  - serviceKey (unique)
  - Name
  - Description
  - bindingTemplate(s)
- Compare to WSDL:
  - operations, port types, ports

Technical service information

- bindingTemplate (part of businessService)
- Contains:
  - Description
  - bindingKey
  - Describes how/where to access the service
    - (by referring to a tModel)

publisherAssertion

- Describes relationship between different businessEntities
- e.g.:
  - Parent/Subsidiary
  - Group/Division/Department
  - Peer "trust" relationship

WSDL and UDDI

- WSDL specifies:
  - Where a WebService is
  - What the WebService does
  - How to work with the WebService

WSDL and UDDI

- UDDI specifies:
  - Who an Organisation is
  - What Business Services they provide
  - What WebService provides the Business Service
  - ...and the WSDL
  - ...and other documents (SLA/QOS etc).
- WSDL portType is equivalent to the UDDI tModel

Implementation

- UDDI is generally implemented as a set of WebServices
  - Submit query in XML
  - XML returned
- UDDI Query Tools!

Publishing Interface

- Used by Service Providers (publishers)
- Publish details of:
  - businessEntities
  - businessService
  - bindingTemplate
  - etc.
- Also edit, update and delete these

Enquiry Interface

- Allows anyone to search the UDDI model
  - Seach by any criteria ("Browse")
  - Return list
  - Get details of any itme ("Drill Down")

Why did Public UDDI fail?

- Complicated
- Hard to categorise products and services
- Hard to get paid to host a public UDDI service
- No dominant Service Registry emerged
  - Remained fragmented
- Never gained critical mass
- Business generally isn’t done by picking a service provider out of the “phone book”

Where has UDDI worked?

- Large, but controlled environments:
  - Government/Public/Civil Services
  - Specific B2B industry sectors
    - Logistics/Distribution
    - Electronic Components
    - Fruit Wholesale Operations
  - Internal use within large single organisations

Should my Organisation use UDDI?

- Not at first, if at all
  - Try other management/governance structures first
- If you manage:
  - large volumes of services
  - rapid change or expansion in services
  - Large numbers of orgs./depts. etc.
- …then UDDI may be useful

Questions?

?

Overview

- Review
- UDDI
- **JSON and XML**

Criticisms of WebServices

- WebServices in general:
  - XML Required - very verbose
  - Parser slow/resource intensive
- …and criticisim of SOAP in particular:
  - Over complicated for many (most?) purposes
- Some of this is justified
  - …some is due to bad implementation
  - …some due to “over promising” what the technology can deliver

Alternatives: JSON

- JavaScript Object Notation
- Simple key : value pair notation system
- Open Standard
- Less verbose than XML
- Widely used and understood
  - Libraries available in most environments

Sample XML

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Jane</to>
  <from>Dave</from>
  <subject>Reminder</subject>
  <body>Remember the milk.</body>
</note>
```

- 135 characters

Same Sample Data as JSON

```
{
  "documentType" : "note",
  "to" : "Jane",
  "from" : "Dave",
  "subject" : "Reminder",
  "body" : "Remember the milk.",
}
```

- 102 characters (24% less than XML)

JSON Highlights

- Normally less verbose /more efficient than XML
- Freeform and adaptable
- Easy/low resources required to parse
- “Native” parsing into JavaScript in many browsers (fast!)
- JSON Schema is available
- Easier to get started

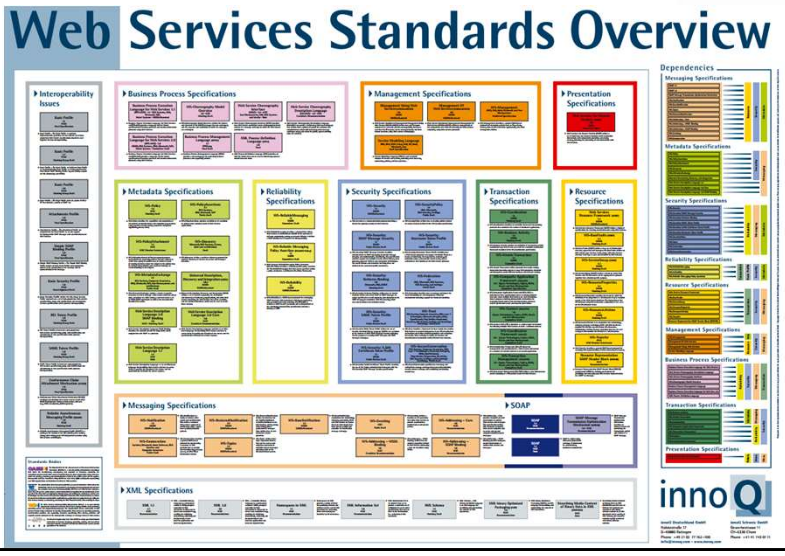
Sample of JSON Schema

```
{ “name": “Document",
   "properties": {
        “documentType": {
                "type": “string",
                "description": “Description of Document Type",
                "required": true
        },
        “to”: {
                "type": "string",
                "description": “email address of recipient”,
        etc…
```

JSON Lowlights

- Freeform and adaptable
- Less developed data types – no formal “date”, no formal handling of errors etc.
  - There are some generally accepted norms for these.
- No generally accepted WSDL equivalent
- Fewer accepted standards than WebServices/XML

Because Standards are good…


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
Dependencies
innoQ

JSON or XML: which is best?

- As usual, it depends on the question
  - What are the priorities?
  - Who do we have to interact with?
  - How many participants are there?
  - How complex is the data type?
  - Is bandwidth/storage a priority?
- Neither is best in all circumstances

Overview

- Review
- UDDI
- JSON and XML