OASIS Web Services for Remote Portals (WSRP) TC

Agenda for the 3rd WSRP/WSIA TC Meeting (September 9th-12th, 2002)

(All times are CET)

Monday, September 9th

WSIA Meeting (WSRP Members are invited as well)

09:00 - 09:30	Welcome and Agenda
09:30 - 10:30	Interfaces for properties: Entity and Session Customization and Use Cases
10:30 - 10:45	Break
10:45 - 11:30	Customization Interfaces (continued)
11:30 - 01:00	Metadata for property description
01:00 - 02:00	Lunch
02:00 - 03:30	Coordination use case, interfaces, and metadata
03:30 - 03:45	Break
03:45 - 05:00	Roadmap for publishing Customization interfaces, continuing Coordination

---

Tuesday, September 10th

WSRP Meeting (WSIA Members are invited as well)

09:00 - 09:30	Welcome and Agenda
09:30 - 10:30	WSRP implementation experiences on Tomcat / J2EE
10:30 - 10:45	Break
10:45 - 01:00	WSRP Markup Fragments - Slot1
Starting with presentation of open and closed activities (Chris Braun)	
01:00 - 02:00	Lunch
02:00 - 03:30	WSRP Markup Fragments - Slot 2
03:30 - 03:45	Break
03:45 - 05:00	WSRP Security, Identity, SSO
Starting with presentation of open and closed activities (Mark Cassidy)	

Wednesday, September 11th

09:00 - 10:30	WSRP Interfaces & Protocols - Slot1

Starting with presentation of open and closed activities (Mike Freedman, Alan Kropp)

10:30 - 10:45	Break
10:45 - 01:00	WSRP Interfaces & Protocols - Slot2
01:00 - 02:00	Lunch
02:00 - 03:30	WSRP Interfaces & Protocols - Slot3
03:30 - 03:45	Break
03:45 - 05:00	WSRP Interfaces & Protocols - Slot4

07:00 Dinner

Thursday, September 12th

09:00 - 10:30	WSRP Publish, Find, Bind & Metadata
10:30 - 10:45	Break
10:45 - 12:00	JSR 168 Reference Implementation Overview
12:00 - 01:00	Breakout Sessions for subgroups
01:00 - 02:00	Lunch
02:00 - 03:00	Breakout Sessions for subgroups
03:00 - 03:30	Summary of Breakout Sessions
03:30 - 03:45	Break
03:45 - 05:00	Milestone Update (Final steps to specification, implementation, ...)

and adjourn

WebSphere software

IBM

e business software

WSRP – Publish/Find/Bind/Metadata

WSRP F2F
September 09-12 2002

Richard Cieply
Carsten Leue

IBM Software Group

Agenda

- Publish
  - WSDLs
  - UDDI
- Find
  - Finding WSRP services in UDDI directories
  - Private UDDIs
- Bind
  - Attaching to a service using proxies
- Metadata

Publish WSRP Services - Overview

- WSRP plans to factor its interfaces
  - ‘base’
  - Properties
  - Extensions ?
  - Addtionally allow for vendor specific factors
- Factor
  - Set of operations logically related
  - Defines operation names, argument names/types and return types
  - Corresponds to a Interface definition
  - Corresponds to the ‘portType’ in WSDL lingo
- Binding
  - Defines message format and protocol details defined by a ‘portType’
- Port
  - Individual endpoint or address for a ‘binding’
- Service
  - Group of related ‘ports’
  - Defines an unit fullfilling a task

Publish WSRP Services – Assumptions

- The underlying standards allow for factoring
- Ports may share information, i.e. sessions can be maintained across ports
  - WSDL issue
  - JAX-RPC issue
- Multiple import problem in WSDL solved
  - Seems to be a tooling issue

Publish WSRP Services - Factors

- Each WSRP service MUST implement the WSRP ‘base’ factors
- A WSRP service MAY implement further factors which MAY NOT be part of the WSRP spec
- There MAY be different bindings for each factor
  - SOAP
  - SOAP Messages with Attachments (MIME)
  - SOAP encapsulation in DIME messages
  - HTTP
  - HTTPS
  - ...
- Each producer offered entity is represented as a service in the WSDL sense
- There will only be one common access point for all ports (required for consistent session handling)

Publish WSRP Services – WSDL Definitions

- Each factor is defined by an “interface” WSDL
  - <types>
  - <message>
  - <portType name=`PortName`>
- Each binding is defined by a “binding” WSDL
  - <import „interface“ WSDL> or embed it
  - <binding name=`BindingName` type=`PortName`>
- A service is defined by a “service” WSDL
  - <import „binding“ WSDLs> or embed them
  - <service>
    - <port binding=`BindingName1`>
      - <soap:address location=`AccessPoint`>
    - <port binding=`BindingName2`>
      - <soap:address location=`AccessPoint`>

Example: Interface WSDL

```
<wsdl:types>
 <complexType name="DataItem">
   <sequence>
     <element name="description" nillable="true"
   type="soapenc:string"/>
     <element name="data" type="soapenc:base64"/>
   </sequence>
 </complexType>
 <element name="DataItem" type="tns1:DataItem"/>
</wsdl:types>
<wsdl:message name="echoDataRequest">
  <wsdl:part name="data" type="tns1:DataItem"/>
</wsdl:message>
<wsdl:message name="echoDataResponse">
  <wsdl:part name="return" type="tns1:DataItem"/>
</wsdl:message>
<wsdl:portType name="Echo">
 <wsdl:operation name="echoData" parameterOrder="data">
   <wsdl:input message="intf:echoDataRequest"
   name="echoDataRequest"/>
   <wsdl:output message="intf:echoDataResponse“
   name="echoDataResponse"/>
 </wsdl:operation>
</wsdl:portType>
```

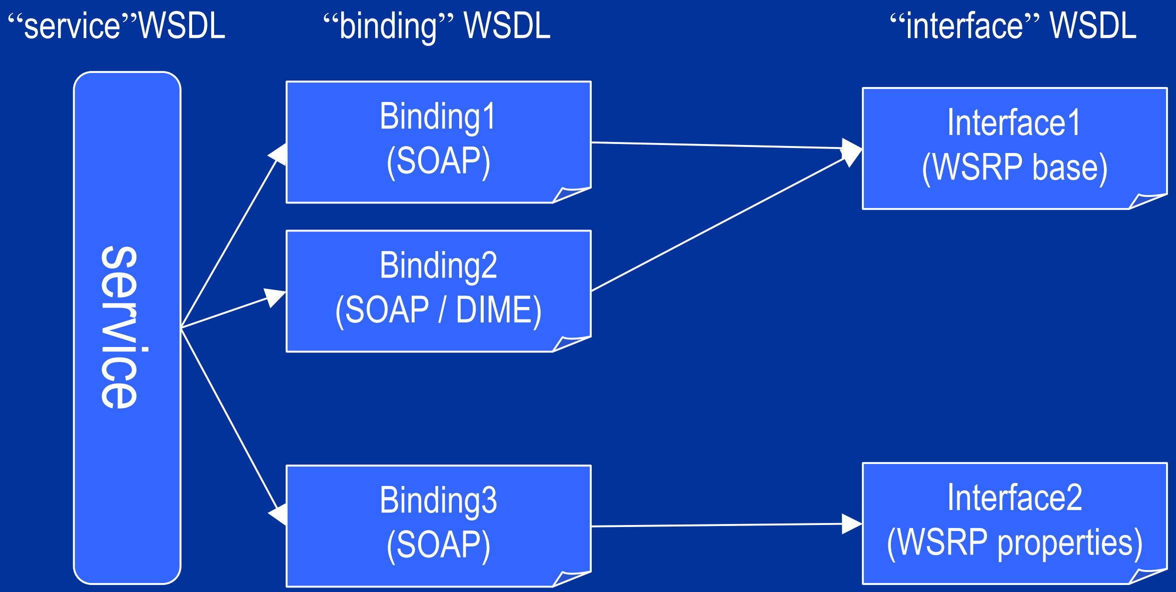
Example: Binding And Service WSDL

```
<wsdl:binding name="EchoSoapBinding" type="intf:Echo">
 <wsdlsoap:binding style="rpc“ transport=“…/http"/>
 <wsdl:operation name="echoData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="echoDataRequest">
     <wsdlsoap:body encodingStyle=“…” namespace=“…”
   use="encoded"/>
   </wsdl:input>
   <wsdl:output name="echoDataResponse">
     <wsdlsoap:body encodingStyle=“…” namespace=“…”
   use="encoded"/>
   </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
```

```
<wsdl:service name="EchoService">
 <wsdl:port binding="intf:EchoSoapBinding" name="Echo">
   <wsdlsoap:address
   location="http://democorp.com:8080/Services/Echo"/>
 </wsdl:port>
</wsdl:service>
```

Publish WSRP Services – WSDL Representation

- WSRP spec defines
  - “Interface” WSDLs for basic WSRP factors
  - “binding” WSDLs for basic WSRP factors
  - Standard WSDLs SHOULD be hosted by oasis-open.org/wrsp ?



Publish WSRP Services – WSDL Naming

- Naming convention allows
  - Humans to find interfaces/bindings they search in directories or the web
  - Consumers to find out if a interface/binding offered by a producer is the appropriate one
- Define a name that contains the protocol, the factor and the binding
- **WSRP.<factor>.<binding>.wsdl**
  - Example: “http://oasis-open/wsrp/wsrp.base.soap.wsdl”
  - Specifies a “binding” WSDL for the WSRP base factor with SOAP binding
- The publisher of the WSDL MUST ensure that the content of the WSDL matches the name
- Naming SHOULD also include a version number

Publish WSRP Services – WSDL Summary

- WSDL allows and is intended for factoring
- WSRP factors, bindings and services are separeted in WSDLs
  - Interface WSDL
  - Binding WSDL
  - Service WSDL
- Naming schema allows for easier search and check

- Information needed to bind to a service can be found in WSDL
  - Access point stored in `<port>` elements
  - Binding descriptions stored in `<binding>` elements
    - Allow to use proxies (precompiled or built on the fly)
  - Interface descriptions stored in `<portType>` elements
- Additionally metadata is needed
  - Contains information about producer offered entities

UDDI - Overview

- Universal Description, Discovery and Integration
  - Defines a way to publish and discover information about web services
  - Relies upon a distributed registry of businesses and their service descriptions
- Registry data consists of
  - White Pages
    - Business names, descriptions
    - Contact information
    - Known identifiers (DUNS, Thomas Register)
  - Yellow Pages
    - Business categories
      - Industry: NAICS
      - Product/Service: UN/SPSC
      - Location: Geographical taxonomy
  - Green Pages
    - Describe the “how to” use a service
    - Service descriptions
    - Binding Information
    - Service type (represented by tModels)

UDDI - tModel

- Represents a technical specification
  - Wire protocols
  - Interchange formats
  - Interchange sequence rules
  - …
- tModel are identified by a global unique tModelKey
  - Specification designers are able to establish a unique technical identity
- Web Services can express compliance to specification(s)
  - By referencing the tModelKey(s) in their service's bindingTemplate data
- Web Service consumers may find compliant services
  - By searching for services that refer to the desired tModelKeys

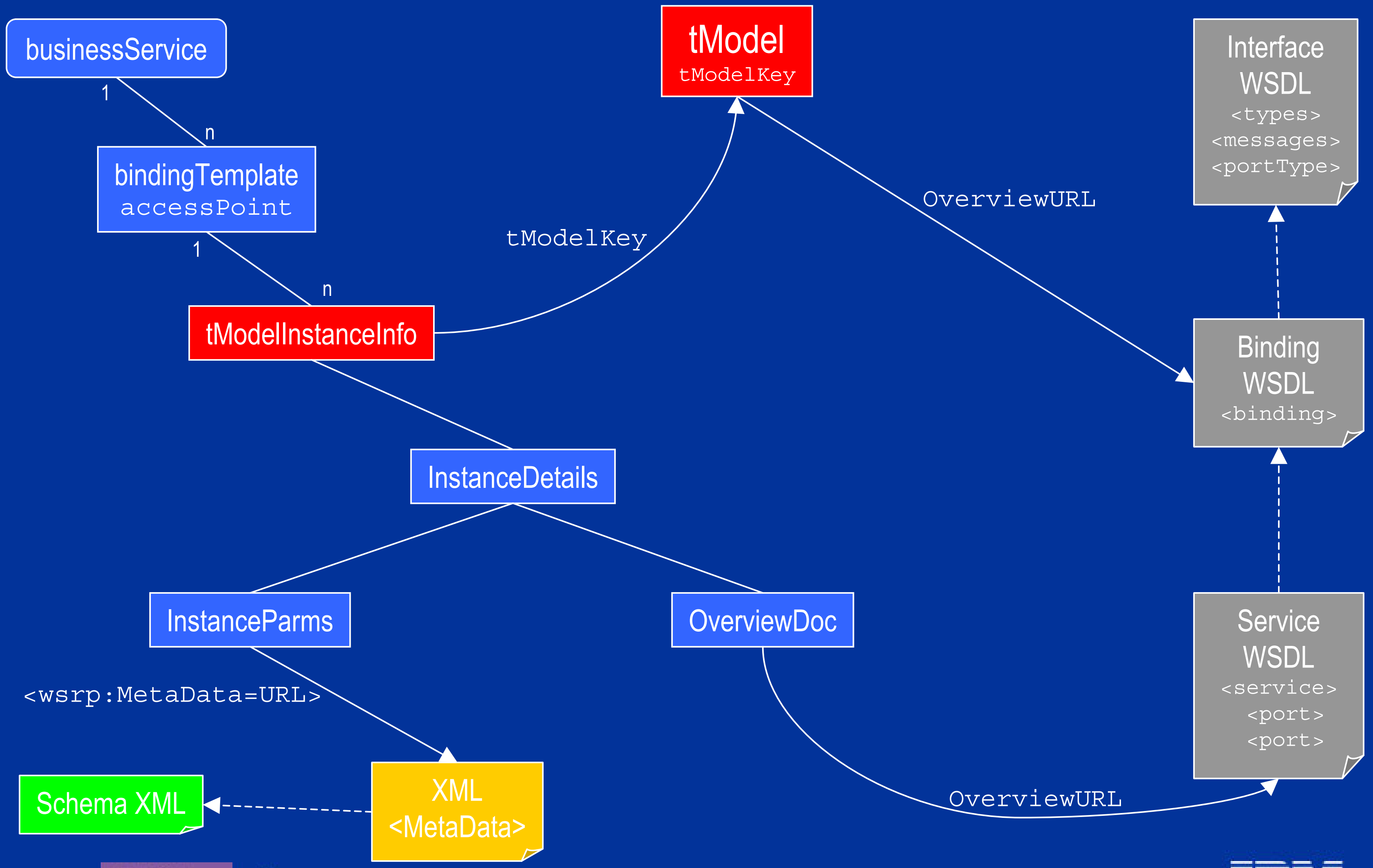
UDDI – Public vs. Private Registries

- Public Registry
  - World wide “Service Cloud”
  - Access is open and public
  - Data may be shared or transferred among other registries
  - Public registries are replicated and visible world wide
  - Keys are identifying data items uniquely world wide
- Private Registry
  - Internal registry
  - Behind a firewall
  - Access is secured
  - Data not shared with other registries

UDDI – Advantages

- Publish
  - Superior to publishing to search engines
  - Full control over which businesses and services are published and when
  - Standardized data formats
  - Exact definitions how to publish
    - Allows automated/programmatical proceeding
- Find
  - Service address doesn’t need to be well known a priori to obtain information about the service
  - Service can be found by
    - Business
    - Name
    - Taxonomy
    - Technical fingerprint
  - Exact definitions how to find
    - Allows automated/programmatical proceeding
- Tools available
  - Implemented in many products

Publish WSRP Services – UDDI Model



WebSphere software

IBM

UDDI Data Structures - businessService

- Name
  - Human readable name which allows the user to find a service by name
  - MAY be defined in multiple languages
  - MUST be adorned with a unique `xml:lang` value to signify the language
  - Mapping to WSDL:
    - N/A
    - UDDI names allow blanks, multiple languages, …
- Description
  - Human readable text describing the service
  - MAY be defined in multiple languages
  - MUST be language qualified with `xml:lang` attribute
  - Mapping to WSDL:
    - `<documentation>` element within the `<service>` element
    - WSDL `<documentation>` MAY contain any attribute thus allows also for language specific documentation
      - `xml:lang` not explicitly defined in WSDL spec

UDDI Data Structures - bindingTemplate

- Description
  - See buisinessService description
  - Mapping to WSDL:
    - `<documentation>` element of the `<port>` element
- AccessPoint
  - Used to convey the entry point address suitable for calling the service
  - Mapping to WSDL:
    - `Location` attribute of `<extension>` element within `<port>` element (at least for SOAP/DIME/MIME and HTTP binding)
- tModelInstanceInfo
  - List of tModels building the fingerprint for this binding
- OverviewURL

UDDI Data Structures - tModelInstanceInfo

- References the tModelKey(s) of the tModel(s) implemented
- OverviewDoc
  - OverviewURL points to “service” WSDL and `<port>` element within the `<service>` element
  - Example: http://my.company.com/myService.wsdl#port
- InstanceParms
  - Contains key-value pair:
    - Key: `wsrp:metadata`, Value: URL to metadata XML

UDDI Data Structures - tModel

- Name
  - Human readable name which allows the user to find a tModel by name
  - WSRP SHOULD standardize a naming schema allowing to find distinct interfaces/bindings
    - WSIA:WSRP.<interface>.<binding>
- Description
  - See buisinessService description
  - Mapping to WSDL:
    - `<documentation>` element within `<binding>` element
- OverviewURL
  - URL to “binding” WSDL
    - If multiple bindings are present in WSDL qualify by binding
    - Example: http://oasis-open.org/wsrp/wsrp.base.soap.wsdl#binding
- categoryBag
  - keyName: `uddi-org:types`, value: `wsdlSpec`

Publishing to UDDI - Summary

- WSRP defines “interface” WSDL(s) and “binding” WSDLs
- “binding” WSDL are published to UDDI as tModels
- Naming schema for WSDL/tModel names:
  `wsia:wsrp.<factor>.<binding>` (example)
- For each access point the businessService contains a bindingTemplate
- Each bindingTemplate MUST refer to according tModelInstanceInfo(s) representing the binding(s) the service exposes
- Each bindingTemplate MAY refer to further not part of the spec tModels
- A WSRP service MUST at least expose the wsrp.base factor and a “base” binding (i.e. `wsia:wsrp.base.soap`)
- Metadata of an entity is presented in a XML defined by a schema XML, instanceParams refer to it

Find WSRP Services – Use of tModelBag

- Goal: Find WSRP compliant services
- UDDI V2.0 API allows to search businessServices by passing a tModelBag
- tModelBag contains tModelKeys which the service MUST implement, i.e. all tModelKeys passed are logically AND’d
- Each WSRP service MUST implement at least the base factor and the “standard” binding (SOAP-RPC)
- Thus searching for WSRP services results in a search for services with the tModelKey of `wsrp.base.soap` tModel in the tModelBag

Find WSRP Services – Use of tModelBag (cont'd)

- Common usage pattern 1:
  - Find all WSRP services
    - Reference wsrp.base.soap tModel in tModelBag
  - Find the service's most sophisticated set of interfaces and bindings the conumer understands
  - Use these „best" bindings to communicate to the service
- Common usage pattern 2:
  - Find all WSRP services the consumer is able to understand
    - Reference tModels the consumer is able to handle
  - Use the bindings to communicate to the service

Find WSRP Services – Use of tModelBag (cont'd)

- Extended search may be accomplished in the same way
- Example 1:
  - search a service which implements the base AND an extension interface AND supports DIME transport
    - Add tModelKey of wsrp.base.dime
    - Add tModelKey of wsrp.ext.dime
- Example 2:
  - Find Service which implements base interface and supports SOAP or DIME binding
  - Results in two searches:
    - One with tModelkey of wsrp.base.soap
    - One with tModelKey of wsrp.base.dime
  - Superset of found services delivers service list to the user

Find WSRP Services – How to find tModelKeys

- WSRP tModels are published a priori to global UDDI
  - tModelKeys are well known for all specified interfaces/bindings
  - tModelKeys need to be published in Spec (or at least a link to them)
- Alternative: allow the user to search UDDI for WSRP tModelKeys
  - This requires a naming schema for tModelNames published by WSRP
  - Problem: tModel Names may not be unique in UDDI
  - Define names which are “namespaced”
  - Example: oasis-open.org/wsia:wsrp.base.soap

Find WSRP Services – UDDI V3.0 outlook

- UDDI V3.0 API allows for `find_services` the `findQualifiers` argument
  - tModelKeys may be logically OR'd
  - One search sufficient to implement previous example 2
- UDDI V3.0 API allows for `find_services` the `find_tModel` argument
  - Alternative or additional way to specify tModelKeys
  - Is treated as embedded inquiry performed prior to `find_service`
  - tModelKeys may then be found by name

Find WSRP Services – Private UDDI Directories

- Generally the same way as in public directories
- Problem: tModelKeys are not well known a priori
- Need of multiple steps:
  - Customer/Portal has to publish tModels first to obtain tModelKeys
  - Use the same tModel name naming schema as mentioned
  - Find tModelKeys by tModel names
  - Publishing services MUST refer to these tModelKeys in their tModelInstanceInfo
  - Find services by tModelKey the same way as in public UDDI

Find WSRP Services - Summary

- UDDI allows sophisticated search methods
  - By business, name, category, tModel
- WSRP compliant services can be found using tModels
- A naming schema for tModels helps searching for distinct tModels
- UDDI V3.0 allows to specify tModels by name in search for service
- Problem in private UDDI directories
  - tModelKey of WSRP services not known a priori
  - Customer/Administrator needs to publish tModels first

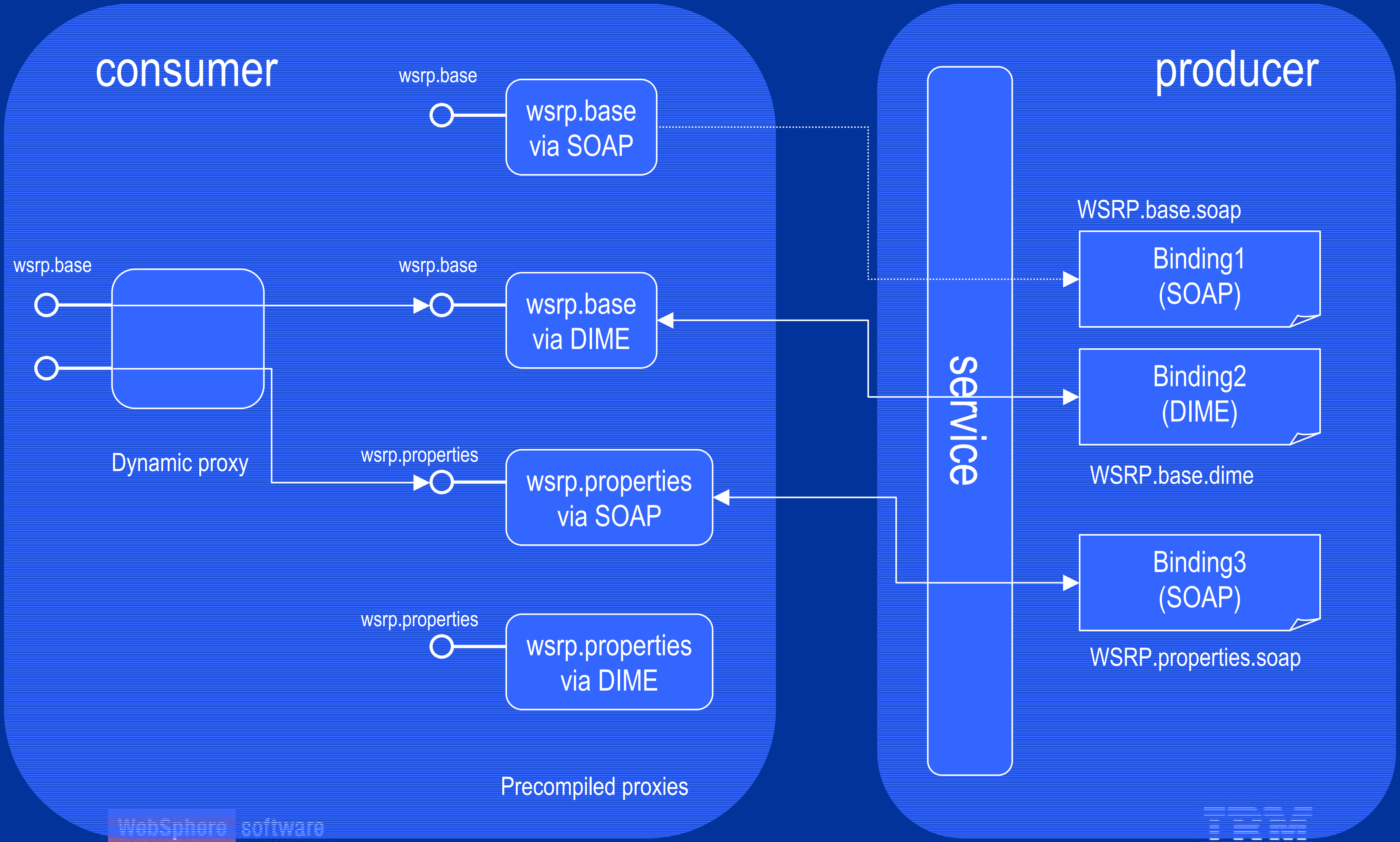
Bind to WSRP services

- Information needed to bind
  - Access Point
  - Binding Information
    - Operations defined
    - Transport methods used
    - Used to generate proxies
    - Used to identify precompiled proxies
  - Interface Information
    - Referenced by binding(s)
  - Metadata

Bind to WSRP Services (cont'd)

- Self description interface
  - ➢ Well known location
  - ➢ Use self description to obtain interface/binding information and metadata
- UDDI
  - ➢ Find WSRP service exploiting tModelBag
    - – Filter by business, names, categories, …
  - ➢ Interface/binding information and metadata contained in UDDI (by URL reference)
- If registration required: `registerConsumer()`
- Optional: `initEnvironment`
  - ➢ Producer may indicate that it is interested in assistence from Consumer
    - – Example: load balanced environment

Bind to WSRP services - Proxies



Metadata - Status

- Current status from Mike?

Metadata - Spec Status

- Entity
  - Supported Locales
  - Supported markup types
    - HTML, XHTML, VoiceML, …
  - Supported modes
    - VIEW, EDIT, HELP
    - CONFIG, DESIGN, PREVIEW (optional)
  - Supported view states
    - MINIMIZED, NORMAL, MAXIMIZED
  - Cachability
    - Expiry, Sharing ?
  - Locale specific:
    - title
    - Short title
    - Description
    - Keywords
    - Supported roles
      - Administrator, page designer

Metadata

- Entity
  - Entity handle in case of UDDI publishing
  - Producer URL rewritung supported
  - Contact Information?
  - isClonable?
- Producer
  - Requires registration?
  - User profile attributes

Publish/Find/Bind/Metadata

Thank you

for your attention

WSRP Security Subgroup

F2F Update

September 10, 2002

Mark Cassidy

Agenda

- Secure transport usage with WSRP
- Roles
- User Profile
- UserContext
- Issues/Discussion

Secure Transport

Client <-> Consumer

- Consumer-written URLs
  - Producer indicates https required via wsia:secureURL parameter
- Producer-written URLs
  - Producer exposes markup template properties:
    - SecureActionTemplate
    - SecureRenderTemplate
    - SecureResourceTemplate
  - Consumer sets these with https values

Secure Transport

Client <-> Consumer(2)

- Initial client request resulting in getMarkup() may not be secure:
  - Consumer URL for an aggregate page may not use https
- SecureClientConnection allows Consumer to indicate status of client connection for current request
- Producer may use this status to generate 'safe' markup where it would normally require a secure client connection for its markup

Secure Transport

Producer <-> Consumer

- Transport bindings declared in service description
- Factoring of interfaces: Potential to have interfaces available over both http and https
  - This would allow selective use of https by Consumer only when markup params dictate or when transferring user profile data
- Issue: can session be preserved across http & https requests? Will load balancers be able to maintain stickiness when moving between http and https?
  - Could simplify and always use secure transport if an entityType requires secure client connections for any operation

Secure Transport Metadata

- needsSecureClientCommunications
  - Indicates requirements for secure client connection
  - Currently specified to have 3 possible values
    - Never – no client interactions require secure transport
    - Sometimes – indicates that some operations could be carried out without secure transport(pending outcome of factoring issue from previous page)
    - Always - a hint to the Consumer that could be used at design time to assign the page an https URL

WSRP Roles

- Roles are designated for use with all entity-related operations
- Operations that do *not* consider Roles:
  - registerConsumer()
  - modifyConsumer()
  - initEnvironment()
  - releaseHandles()

Standard WSRP Roles

- Abstract access levels defined(ordered by level of privilege implied)
  - Admin(unrestricted)
  - Page Designer(creates & modifies entities)
  - User(personalizes entities)
  - Viewer(read-only access to entities)
- Specific behavior associated with roles is producer-defined

Role-related Metadata

- Supported role names are declared in EntityType metadata
- Role descriptions are part of the ServiceDescription structure

Role Issues/Discussion Points

- Recent note from Security JC discouraging defining Role mechanism in WSRP: Possible confusion w/SOAP 1.2 role concept
  - SOAP 1.2 roles are not related to access-control
  - SOAP 1.2 roles appear to be primarily for managing message processing when intermediaries are involved

User Profile Attributes

- WSRP will use the P3P-defined standard user attributes
  - Name
  - Birthdate
  - Gender
  - Employer
  - Home & Work info(location, contact)
- Producer declares which subset it wishes to receive as part of EntityType metadata
- Extensibility mechanism to declare support for named custom attribute schema

Extended User Attributes & Registration

- Consumer indicates its ability to support named extensions at registration time
  - An extension identifies a custom attribute schema
  - Definition/discovery of custom attr schema is outside WSRP scope
- Producer may use this information to alter the service description
  - For example, include or exclude profile attributes it wishes to receive

Consumer Passing of User Attributes

- P3P defines standard user attributes in a hierarchical schema
- Easier to have a flattened array of attribute strings?
- Recommend to maintain support for hierarchy, XML representation of attributes
- Related to property/XFORMS discussion

UserContext Data Object

- Aggregates role, user attribute data into a single context object
- Passed with entity-related operations
- Optimization possible if Producer is able to store UserContext for the life of the session

UserContext Optimization

- Passed when there is no valid sessionID
- Must always pass UserContext for :
  - getPropertyDescription()
  - getProperties()
  - setProperties()
- Once a Consumer has a valid sessionID, UserContext may be omitted
  - Consumer requests carrying a sessionID that has timed out will need to be resubmitted with UserContext data(Producer should indicate a fault)
- Producer can declare support for this optimization in EntityType metadata

User Identity

- WSRP protocol as currently defined doesn't include user identity in any data objects/signatures
  - Proposed to use WS-Security for transferring user identity
  - SAML is another standard alternative for carrying user identity
  - Or, we could define it to be part of our UserContext object
- Issue: None of above is ideal
  - WS-Security is not yet a standard and support not available from commonly used stacks
  - SAML is a complex protocol and not yet widely implemented
  - Our own mechanism will be obsoleted at some point by the above(or other) standards

User Authentication

- Entities that need end-user-supplied credentials for access to back-end systems may obtain via secured markup-driven interactions
- Security token carried via WS-Security header could provide an alternative mechanism for a Producer to authenticate the end-user

User ID/Profile Discussion

- It has been suggested we leverage SAML’s support for userID, user attributes(and roles for that matter).
- Issue is that SAML is going to be complex to use, and there’s no policy mechanism for defining how SAML assertions should be formed.
- Decision we have to make is whether to proceed on our current path with basic, limited functionality that can be implemented with current infrastructure, or to hold back until external standards are solidified and adopt them as they mature?

Other issues

- End-user authentication level
  - Raised for consideration within WSRP
    - SAML has mechanism for Auth assertions
    - WS-Security discussions currently starting on QoS
  - Necessary for v1 or defer until external support can be integrated?

Discussion

WS-Security Background

WS-Security Scope

- Defines three mechanisms for SOAP messages:
  - Inclusion of security tokens
  - Digital signatures on elements of the message
  - Encryption of message elements
- All WS-Security elements are contained within a <wsse:Security> element in the SOAP header

WS-S Security Tokens

- General mechanism to include different types of tokens:
  - Username/password
  - Binary tokens(I.e. X.509 certificates, Kerberos tickets)

WS-S Signature Blocks

- XML-DSIG block referencing some element in the SOAP envelop
- May have multiple signature blocks in the <wsse:Security> element
- Signatures may reference security tokens for keyinfo

WS-S Encryption Blocks

- XML-ENC block referencing an encrypted element in the SOAP envelope
- May have multiple encryption blocks in the <wsse:Security> element
- Encryption key may be directly carried(in encrypted form) in the <wsse:Security> element
- Security token may be used to unencrypt the contained key

How WS-S Fits with WSRP

- Message integrity and confidentiality
  - Encryption and signature mechanisms can be applied to WSRP messages
  - Requires a policy mechanism for this to be plug n play
- User Identity
  - Can provide identity and authentication info about the end-user
  - However, spec doesn’t appear to support using this mechanism to identify/authenticate both Producer and end-user

What WS-S Does not Cover:

- Identity mapping
- Any explicit support for access roles (possibly via extensibility mechanism)
- Standard handling of user attributes (extensibility mechanism provided)