Should WSRP Leverage WSRF?

1. What is WSRF?
2. How would we leverage WSRF?
3. What would be the impacts of leveraging it?
4. What would be the benefits?
5. Setting out a roadmap.

What is WSRF?

- WSRF is an OASIS TC (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf) working on the WS-Resource framework
- This framework is a factored set of specification seeking to provide standardized means for web service interactions with stateful entities following its pattern.
- Work began at the end of April, 2004
- Target for first standard (WS-ResourceProperties) is end of year.
- Concepts and initial spec derived from existing GridForum work (i.e. already reasonably vetted).

What is WSRF? Composable specs

- **Resource Framework:** Current framework uses WS-Addressing to provide a URI for a web service and a set of "Reference Properties" the web service will use to locate stateful entities relevant to processing an invocation.
- **Resource Properties:** (see http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-04.pdf) This spec defines an attribute that can be added to a PortType definition that provides a QName for a schema definition of the XML serialization of the Resource's properties into a document and message exchange patterns for interacting with those properties.
- **Resource Lifetime:** (see http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-03.pdf) This spec adds lifetime operations and controls to WS-Resources. These involve immediate destruction, scheduled destruction and renewal of scheduled destruction semantics. Scheduled destruction is controlled through a Resource property named "TerminationTime".
- **Service Groups:** (see http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ServiceGroup-1.2-draft-02.pdf) This spec defines interactions with a collection of web services or WS-Resources.
- **Base Faults:** (see http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-02.pdf) This spec defines a set of base fault types that any specific fault within the other specs will extend.

WSRF definition

- A WS-Resource looks like (exploits WS-Address):
  - <wsa:EndpointReference>
    <!- WSRF has an open issue of the need for a URL to the WSDL for this interface -->
    <wsa:Address>
    <!- service endpoint for message delivery -->
    http://someOrg.com/aWebService
    </wsa:Address>
    <wsa:ReferenceProperties>
    <!- stateful information that must be supplied when interacting with the resource -->
    <tns:resourceID> C </tns:resourceID>
    <tns:resourceID> F </tns:resourceID>
    </wsa:ReferenceProperties>
    </wsa:EndpointReference>

WSRF Runtime message

- Message to a WS-Resource looks like:
  - <soap:Envelope>
    <soap:Header>
    <tns:resourceID> C </tns:resourceID>
    <tns:resourceID> F </tns:resourceID>
    </soap:Header>
    <soap:Body>
    … some message (i.e. getMarkup)
    </soap:Body>
    </soap:Envelope>

How would WSRP leverage WSRF?

- **WSRP defines many kinds of state:**
  - RegistrationHandle
  - PortletHandle
  - SessionID
  - NavigationalState
  - InteractionState
- Largest value modeling one of these as a WS-Resource would be for portlets. Proposal is to explore casting portlets as WS-Resources and review impacts, values and costs.

Impacts of leveraging WSRF?

- When discussing how to model the web service interface for WSRP v1, it was noted that web services were defined to be stateless by their very nature. This (+ other reasons) led to the interfaces being defined at the container level and the statefulness of portlets being handled idiosyncratically by the protocol. The hope was that emerging work on web service interactions with stateful entites would eventually prove useful to the TC, though there was no desire to wait for it to appear.

Impacts of leveraging WSRF?

- The "grid" efforts have defined a means for web service interactions with stateful entities. They often slip into describing these as objects as the paradigm fits well, but the defined semantics also work for stateless interfaces that access stateful data entities per invocation (common model for servlets and portlets).
- Proposal is to explore recasting the WSRP interfaces in terms of portlets as WS-Resources.
- Using a WSRP defined element for pointing at a series of CustomizationRecords could provide a natural means for supporting portlet hierarchies.

Impacts of leveraging WSRF?

- Portlets as WS-resources: Impacts on the API
  - Markup portType (and possibly portions of PortletManagement portType) get recast to remove the portletContext parameter and have the operation targeted directly at the portlet.
  - Metadata moves to defined portlet properties (perhaps just one property called PortletMetadata) and WSRP-specific operations are dropped.
  - Portlet properties become WS-Resource properties and the operations to deal with them dropped in favor of the WS-ResourceProperties portType.
- Clearly a recast of syntax, but not of semantics.

Impacts of leveraging WSRF?

- Portlets as WS-resources: Impacts on Feature Proposals
  - Leasing Handles: This is covered by the WS-ResourceLifetime spec which defines a property and operations for immediate and scheduled destruction of the resource.
  - Single portlet Metadata: If the metadata is all in a portlet's properties, this is just getting the value of a well-known property.
  - Security/Policy/Privacy: Other efforts are defining how to declare and apply security concerns to a web service. All portlet-specific security issues can be direct leveraging of those definitions if we leverage WSRF, but require WSRP-specific extensions if we do not.
  - Property usage field: This would be a requirement we would pass onto the WS-ResourceProperty specification.
  - Support portlet hierarchies: Could be a natural result of defining our own reference property element, including semantics.

Why now? Benefits:

- WSRP v1 has demonstrated the value of a single defined interface for interacting with a remote UI component.
- WSRP v1 restricted itself to treating the remote portlet as if it were an independent web app in order to keep the effort simple.
- Adding the complexities of coordination and message level security moves us significantly out of this simple territory and the amount of effort to fully define interacting with a remote component looms in front of us.
- Need to keep our labors focused on the WSRP-specific issues and leverage other efforts for general questions (e.g. how to carry security information within a SOAP message).
  - List of Feature Proposals that become simple if we leverage WSRF indicates current value to making this switch.

Why now? Benefits:

- WSRP v1 has taken 6-12 months for significant announcements of support by portal vendors and will take another year or two for significant adoption by customers.
- As customers gain experience, I believe we will hear a loud clamoring for the advanced functionality we have put on our own plate (particularly coordination and security)
- With reduced manpower available for pressing the WSRP effort forward, maximum leveraging of other work should be encouraged.
- Leveraging WS-Resource going forward will make portlets composable with other specs in timeframes we could not hope to accomplish as a TC.

Roadmap?

- Leverage WS-Resource in WSRP v2
  - When are we likely to get a v2 standard approved vs when is WS-Resource likely to be a supported std?
  - Note: Early leveraging of WSRF could be a strong marketplace message that WSRP is the remote UI component interface and that portals are good platforms for integrating web UI components into an application.
- Leverage WS-Resource in WSRP v3
  - Leaves us less composable with other specs for 2-3 years.
  - Reasonable if v2 leveraging would delay adoption by __.