OASIS WSIA Technical Committee

Requirements Document
Use Case Report: Embedded Producers

Version <1.7>

Revision History

Date	Version	Description	Author
05/Mar/2002	1.0	Aggregated Producer	Dan Gisolfi, Graeme Riddel, Alan Kropp, Eilon Reshef, Gil Tayar, Rex Brooks, Ravi Konuru, Keven Brinkley, Aditi Karandikar, Monica Martin, Rich Thompson, Charlie Wiecha
06/Mar/2002	1.0	Renamed to Embedded Producer	Charlie Wiecha
20/Mar/2002	1.1	Updated flows, added requirements section	Rich Thompson
22/Mar/2002		Added requirements	Charlie Wiecha
25/Mar/2002	1.2	Added requirements	Rich Thompson
29/Mar/2002	1.3	1) Rework Requirement section to separate discussion of a requirement from a more succinct statement of the requirement. 2) Incorporate changes from 28/Mar/2002 telecon and followon notes from participants.	Rich Thompson
03/Apr/2002	1.4	Accept previous changes Add references to separate requirements document Add current status to each external standards committee (section 7.3)	Rich Thompson
05/Apr/2002	1.5	Namespace, security considerations, partly adapted from WSRP subgroup proceedings.	Alan Kropp
09/April/2002	1.6	Rewording in Section 7 (Lifecycle, Rewriting, Security). Appendices for properties, CSS abstractions	Alan Kropp
11/April/2002	1.7	Rework 7.1, adding Lifecycle section, 7.3.3 Appendix C security standards	Alan Kropp

Table of Contents

Use Case Report: Embedded Producers

1. Definition of the Embedded Producers use case

Definition: Control over selection, configuration, placement of Producers within a Consumer is under manual (i.e. either Administrator or End-user) control. Producers do not publish an interface to the Consumer other than for invocation with generic arguments (not Producer specific). Arguments may include, among other things, user profile, output markup preferences, language, and device preferences. Producer specific configuration is via separate edit pages, i.e. not reflected in the Producer's interface.

1.1 Brief Description

2. Actors

There are three actors in this use case:

- Producer: one or more WSIA web services
- Consumer: a container which instantiates and controls interaction with one or more Producers on behalf of End-Users
- Consumer Administrator: a person who instantiates and configures Producers in one or more Consumers on behalf of End-Users
- End-User: a person who interacts directly with the output of the Consumer

3. Flow of Events

3.1 Basic Flow

While the following discussion is in terms of "portlets", these should be viewed as a special (and common) case of embedded WSIA services where the Consumer is a portal. This use of portal oriented terminology is for clarity and not intended to restrict the set of actors to the special cases of portals and portlets.

3.1.1 Admin created portlets

[Traveler's checks] Travelers Checks applications as-is within a corporate portal

[Portlet] Consumer Administrator finds remote portlet service in UDDI, places portlet on a page for all users, uses the portlet's edit mode to configure the portlet for the Consumer's page, persistently saves the configured portlet. End-User accesses page and views/interacts with displayed output.

- Pages should be cached to reduce load on remote portlet service.
- End-user entered data is transferred to remote portlet imbedded in parameters or properties of the portlet.
- To facilitate the correct mapping of an End-User interaction to the correct instance of a remote portlet, the URLs contained in the page must be rewritten to both refer to the Consumer and provide it with the information necessary to map the interaction to the properties / operations of the correct remote portlet. One possible means to enable this URL rewriting include:

- Producer marks all interaction URLs (distinct from reference URLs such as on an <img> in XHTML) with a prefix of "wsia:". This makes it unambiguous to the Consumer which URLs need to be rewritten.
- Insert an additional parameter (preferably of a well known name such as wsia:mapKey) that will enable the Consumer to locate the correct record in some internal mapping table. This is essential if the Consumer is to correctly identify the Producer for redirecting the interaction. [It has been noted that ebXML has a similar concept throughout much of its API where a UID is passed to enable the lookup of a set of values in a registry. Since the need for this key is entirely internal to the Consumer (though its value is sent to the End-User in the markup and returned on the resulting invocation), requiring an external registry may be more burdensome than useful. Should be reviewed by the committee as a whole.]
- The Consumer may also wish to change the operation name to reflect the need to pass through the remapping logic. [OR do the WSIA portTypes want to explicitly expose such an operation and standardize its signature?]
- [Is there any utility in defaulting the operationName to invokeAction (from WSXL)?]

- User profile transferred to remote portlet in parameters or properties or by reference to an external registry (such as UDDI).
- Markup type configured by the Consumer at page load time based on End-User profile or device type in use by the End-User [Requires the Consumer to detect information about the End-User and the device. Is this information hidden by the web services stack in todays implementations?].

3.2 Alternative Flows

3.2.1 End-user created portlets

[Portlet] End-user locates a portlet in UDDI or through a search implemented by Consumer, places portlet on a page, optionally uses portlet's edit mode to configure its output, persistently saves the configured portlet. End-user accesses page and interacts with displayed output.

- The Consumer Administrator may restrict the set of portlets available for placement on a page even when the search is implemented via UDDI. This enables the Consumer Administrator to manage any business issues related to the portlet's usage.

3.2.2 *Output type configurable for different devices*

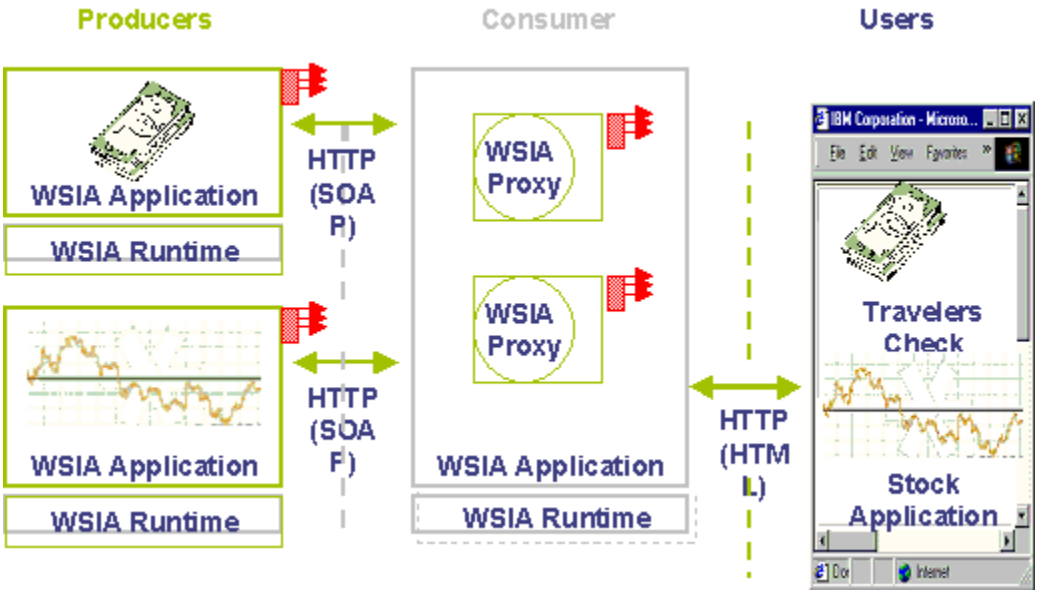
[Multimedia sports portal, Mortgage center] Consumer must indicate to the Producer the device (mime?) type extracted from the End-User connection. Preference would be for this to be a well-known property of all Producers [Perhaps there are a set of these … all using a 'wsia:' prefix]. [Need to define a minimum set of required properties that are adhered to by all web services – this may be guided by context – see some of the work of ebXML CC just as a reference.]

3.2.3 *Non-persistent remote portlets*

[Portlet] Flow as 3.2.1 above, but with no persistence at the Producer. Configuration state returned to the Consumer for persistent storage. The Consumer is then responsible for supplying this information on any subsequent connections to the Producer. [Assumptions and business requirements would direct whether performance and loading overheads are acceptable.]

4. Diagrams

4.1 Relationship between Producers and Consumers in the Embedded Producers Use Case



4.2 < First special requirement >

5. PreConditions

[A precondition (of a use case) is a textual description of any constraints or dependencies that must be satisfied prior to entry of the use case.]

5.1 < Precondition One >

6. PostConditions

[A postcondition (of a use case) is a textual description of any constraints or dependencies that must be satisfied after termination of the use case.]

6.1 < Postcondition One >

7. Requirements

As the separate requirements document gets pulled together, this document will track how these items match the enumerated requirements using the syntax "{R101}". Note that each requirement here may track to multiple in the requirements document.

7.1 Lifecycle

The Consumer interacts with the Producer using the published interface of a WSIA service. This interaction transitions through several states during its Lifecycle.

7.1.1 State 0: Service unknown (unbound)

The Consumer does not yet know about the service.

7.1.2 State 1: Service becomes known (bound)

The Consumer "discovers" the service via registry search, or other method, and acquires the service description. The Consumer uses the description information determine how to integrate the service, by examining and optionally setting service properties.

7.1.3 State 2: Service active

The service is active in the Consumer environment, and is ready to service End User requests.

7.1.4 State 3: Service inactive

Either the Consumer or the Producer has deactivated the service, and it is no longer available to service End User requests. [This is typically a Consumer decision. But what if the Producer terminates the service?]
[Is there a case for returning to State 0, i.e., the service is logically unbound?]

7.1.5 Reference to the Service during its lifecycle

WSIA Services need to provide Consumers with means to refer to particular instances in subsequent calls (This need is related to the stateless character of WSDL services and should be supplanted with WSDL support for stateful services when that support becomes available). We therefore introduce the concept of a 'Handle' as a remote opaque reference to an instance of the service (though an implementation may choose to make this a reference to the equivalent of session data) and define basic operations to create, and destroy those instances.

7.1.5.1 WSIA services SHALL manage their lifecycle such that a Consumer can access a particular instance during the lifecycle of the interactions between them. {R102}

7.1.6 A WSIA service SHALL maintain its condition, i.e. its state [Discussion – question of persistence] {R102, R702, R703}

7.2 Statefulness

It must be possible to operate WSIA services in a stateless or stateful manner. This is a Producer-specified mode of operation. The consumer must interact with the service using the mode the Producer has specified. The Producer may specify that a service supports both stateful and stateless invocations. In this case, the Consumer may select either stateful or stateless interaction. This selection should remain unchanged for the duration of the interaction (session).
Stateless WSIA services may always return the same Handle from the create operation and will ignore any Handle

passed in throughout the remainder of the operations.

7.2.1 A WSIA service MAY maintain a stateful or stateless condition. {R102}

7.2.2 When a WSIA service maintains its condition in a stateless manner, a handle MAY be returned by the lifecycle operations.

7.2.3 Once a handle is returned by a service, no other handles SHALL be accepted by other operations.

7.2.4 Explicit creation of handles is not required. Handles may be created implicitly on requesting output for the first time. Such a handle will be returned along with the output in the response message. While this reduces the overhead of the create / destroy operations, it also requires the Consumer to send a complete set of properties on each operation invocation in order to deal with the possibility that the Producer may have used a timeout to clean up the underlying object. [How does the Producer indicate use of the implicit lifecycle semantics?] [Note: There may be a significant performance impact related to sending the full set of properties on each operation invocation.]

7.2.5 Explicit handle creation SHALL not be required. {R201} [201 is relevant, but doesn't capture full sense of this requirement]

7.2.6 If an implicit creation of a handle occurs, that handle SHALL be made available with the associated output.

7.2.7 A WSIA service MUST indicate whether it operates in a stateful or stateless manner.

7.2.8 The Consumer MUST interact with the service in the mode (stateful or stateless) that the Producer has specified for the service.

7.2.9 If the Producer has specified more than one mode, then the Consumer MUST select one of the supported modes, and use it for the duration of the interaction or session.

7.3 Rewriting

Rewriting applies to those entities in the Producer-generated markup that require the proper context to enable the service to function correctly and/or efficiently in the Consumer environment.

7.3.1 Namespace Encoding

To avoid a clash of named entities when a consumer aggregates two or more services, all such entities in the generated service markup (e.g., URL's, properties, Javascript function names, etc.) should be namespace encoded. There are two possible methods to accomplish this encoding; either the Producer does the encoding, or the Consumer.

- For Consumer encoding, there would need to be a convention for identifying entities in the generated markup that need to be encoded. A placeholder prefix (e.g., wsia-local-namespace:) for each of the entities could be located and rewritten by the Consumer.
- The Producer could be responsible for encoding the entities. The Consumer would need to inform the Producer (parameter or property) as to the desired namespaces to be used for encoding. This may not be a scalable solution, but could suffice for embedding simple services in a simple Consumer.

7.3.2 URL rewriting

URLs need to refer to the Consumer in order to allow the correct state and indirections to be applied when processing an interaction from an End-User. Possible means for accomplishing this mirror the options for namespace encoding above:

- Producer modifications during markup generation based on Consumer supplied information [How does the Producer indicate it is willing to make these modifications? Support for a wsia:BaseURL property?]. The Consumer would need to supply the base URL for referring to itself in a manner that allows for remapping user interactions back to the correct Producer. The Producer would then append information onto this base URL indicating how the interaction should be mapped to it (operationName should use a well-known parameter name such as "wsia:opName" [default value of "invokeAction"?]) and any additional parameters that the operation requires [Are opaque operations allowed to have parameters, or do they just act on the 'current state' of the service with an update to the service's properties being carried with the call?].[Need to address dynamically generated URLs (eg. JavaScript)] [I'd like to consider removing this option, and focusing on Consumer-applied modifications, as I think it aligns better with what's going on in WSRP, and also because it makes sense from an encapsulation perspective]
- Consumer applied modifications. In this case the Producer marks all interaction URLs (distinct from reference URLs such as on an <img> tag in XHTML) with a prefix of "wsia:" This makes it unambiguous to the Consumer that URLs need to be rewritten. [IBM WPS team => a longer marker than wsia: with unusual character sets makes for more efficient parsing.]

[Is this appropriate at the UC/Req level? Implementation detail?] In both cases there is a need to insert an additional parameter (preferably of a well known name such as wsia:mapKey) that will enable the Consumer to locate the correct record in some internal mapping table. [Does this need introduce a problem for multiple tiered chains of Producers/Consumers using the Producer modifications means?] This is essential if the Consumer is to correctly identify the Producer for redirecting the interaction when it arrives from the End-User. The Consumer will need to store any and all information in this mapping table that is needed to invoke the correct operation on the correct Producer.

7.3.3 A Consumer SHALL provide the capability to redirect invocations back to the Producer that sourced the markup causing the invocation.

7.3.4 A Consumer SHALL provide the capability to include input or additional parameters to the Producer.

7.3.5 A unique identifier SHALL be available to identify the correct Producer, the operation and any additional parameters (where applicable) for processing an invocation [is this the Handle?].

7.4 Security

This is a broad area, covering a number of relevant topics:

7.4.1 Establishment of trust between Producer and Consumer. This trust relationship has a few dimensions.

- *Discovery/bind time*. At the point the Consumer first integrates the Producer service, the trust mechanism ensures that the embedded service is authentic (i.e., no spoofing or malicious code), and delivered as specified by the Producer. [XML Signature, SAML]. There may be a precondition for the Producer and Consumer to negotiate some exchange of credentials prior to binding a particular service, perhaps as part of a contractual relationship. This exchange could be human-mediated, and the credentials and related documents hard copy.
- *Service runtime*. When the service is invoked, the trust mechanism ensures that Consumer requests and Producer responses are appropriately credentialed. [SAML, XML Signature, XML Encryption, SSL]
- *Service revocation*. There must be a reliable mechanism to ensure that credentials and authorizations for a revoked service are also revoked.

7.4.2 Identity

Need to consider what information about the End User can pass back to the Producer. There is a two-way dependency. On the one hand, the Producer may specify what information about End Users is required or desired. On the other hand, privacy policy in effect at the Consumer may limit or prohibit the amount of information about End Users and their activities being transmitted. Another dependency could be introduced in the event the End User

themselves chooses to protect or reveal their personal information. [P3P]
There are three use cases.

1. *Anonymous.* No identifying information about the End User is transmitted. Generic or aggregated usage details can be returned (clickstream analysis, for example), but even this may be subject to user privacy concerns, and whatever privacy policy the Consumer enforces. [P3P]
2. *Identified.* End User identifier, as well as other personal profile information, could be transmitted.
3. *Authenticated.* In addition to the profile data, the service invocation may require authentication. This could be a username/password combination. A precondition could be that this username/password has been previously established and disseminated to the End User (or the Consumer, on behalf of the End User).

7.4.2.1 In a chained series of Producer-Consumer interactions, what visibility should an arbitrary upstream Producer have into the identity stream: immediate Consumer, End User, or varying levels in between?

7.4.3 Data Security

Transport and document-level encryption. Need a way to pre-secure the channels for a set of services or at least check them and make sure that the channels are open and available for transmission before a full session is engaged. [XML Encryption, WS-Security (encryption header), SSL, etc., etc.]

7.4.4 Relevant Standards

Other standards efforts are underway for the purpose of defining security and authentication protocols for XML and web services. While WSIA will need to be concerned about these issues, it is expected that concern will be addressed by tracking the work of these other efforts. See Appendix C for a list of these efforts.

7.5 General

Export one or more component interfaces that expose enough information to enable adaptation, aggregation and integration while still allowing the application to evolve. [Not testable … do we need to place some parameters around the types of extensibility?]

7.5.1 The WSIA portTypes SHALL provide a minimum set of criteria to enable adaptation, aggregation, and integration*. {R301, R302}*

Do not preclude that applications will be built out of separate presentation, data, and control components using refinements of this embedded level of interface; this helps developers to separate design issues that if left intermixed in more monolithic objects would make integration for re-use in multiple channels more difficult.

7.5.2 The WSIA portTypes structure SHOULD allow for extensions.

7.5.3 The WSIA portTypes structure SHOULD allow for loose coupling to aggregated operations (i.e. [side note] they provide for flexibly exposed interfaces and operations).

To ensure a fit with existing web-based application architectures, generate markup that can be used by conventional browsers and devices through existing formats and protocols.

7.5.4 Producers SHOULD provide the capability to support legacy applications and infrastructure. [m2: This requires further definition or bounding.]

To ensure markup fragments may be aggregated onto a single page, markup restrictions need to be

defined for common markup types. Note that the potential to republish makes this a recursive issue and that the "wrapper" around the composable markup fragment may not be applied until the tier directly communicating with the End-User's device.

7.5.5 Producers MUST comply with any markup restrictions defined for the markup types they generate. {R602, R603}

7.6 ServiceDescription:

- This category of interface provides basic inquiry operations that allow a client to:Request the WSDL service description document (particularly useful when the service was not discovered through UDDI)
- Inquire whether or not a particular portType is supported.
- Provide information about generated markup. This could be expressed in terms of the look and feel abstractions (CSS classes) that are considered to be common for Embedded services. [see Appendix B]

7.6.1 An alternative to UDDI SHOULD be available to request a service description. *{R201, R301}*

Means to request a Producers attributes SHOULD be available. [Has dependencies on rights – see 7.3] {R301, R302}[Examples of Producer-level attributes?]

7.7 Property Management

For purposes of this section, there are WSIA-specific properties for embedded services. It should be noted that "properties" is a loaded term, and while the concern in this document is to describe a mechanism for managing WSIA-specific properties, the discussion and mechanism MAY be relevant to other classes of properties.

7.7.1 Property Classes

- *Service meta-data* (as discovered in a registry entry…service access points/ports, etc.)
- *Service-level settings*. These are settings that alter the service's general operating characteristics from the Consumer perspective, and apply to all End User invocations through that Consumer. WSIA properties reside in this class.
- *User-level settings*. These are End User specific settings (e.g., a selected stock symbol) for either a single (stateless) invocation of the service, or for all of this End User's interactions with the service (stateful).

7.7.2 A WSIA service MUST implement property management operations whereby Consumers can modify properties at times other than initialization.

7.7.2.1 Producers MUST support property management operations {R302, R303, R401}

7.7.2.2 Consumers MAY modify a Producer's properties during the lifecycle of their interaction. {R401}

7.7.2.3 Producers MAY reject attempts to modify its properties. [Access control, read only properties…] {R403}

7.7.2.4 Producers MUST inform Consumers which properties have actually been modified.

7.7.3 A Producer MUST specify its supported properties (including that particular properties are read-only) so that Consumer may modify them as necessary. Preferably this is done through a schema definition.

7.7.3.1 Producers SHALL specify their supported properties. {R302}

7.7.3.2 Producers MAY specify their supported properties using XML Schema. {R303}

There is a significant set of information that is typically relied upon when a Producer generates its markup (e.g.

desired contentType) and may be a set of useful properties for controlling the interaction between a Producer and a Consumer.

7.7.4 Define a set of WSIA properties: [see Appendix A]

7.8 Output:

Operations related to generating the markup for the service. Returns an opaque representation of the object's output. Consumers must inform Producers of the desired output type. Preferred technique is through a WSIA defined set of properties (deviceType, mimeType, …)

7.8.1 Producer MUST export a WSIA defined operation for requesting the output reflecting its current state.

7.8.2 Consumers SHALL set properties on Producers that control output generation. {R401} [Not fully captured]

7.8.3 The output MUST be testable against conformance rules (see 7.4.5). {R601} [Not capturing that there will be conformance rules]

[Mike F. question raised earlier] Thomas raised the question/need that a producer may generate output that has to be aggregated into different locations in the markup. Is this something that should be covered/identified here?

7.9 User interaction:

The need to direct user interactions back to the Producer of the markup sourcing the interaction request can be addressed either opaquely (as per many portal environments today – see IBM's WSRP submission) or transparently as unique WSDL operations. In order for Consumers to have no Producer-specific programming, the opaque means is essential.

7.9.1 Producers SHALL support an opaque operation by which Consumers may direct user interactions to the Producer. {R202} [This is more detailed …]

7.10 Persistence:

Many interaction between Producers and Consumers occur in well defined contexts (eg. business agreement and/or a configuration preset by an Administrator. This optional portType provides operations to create and destroy these persistently stored configurations.

7.10.1 Producers MAY support a WSIA portType providing Consumers the means to request the persisting of the current state for use in later interactions as well as the destruction of such persisted states. {R702}

7.11 Basic Look and Feel Adaptation:

While a Consumer operating at the Embedded Use Case level is not performing Producer-specific operations, the need to reasonably aggregate the output of disparate Producers requires each to use the same general look and feel values.

7.11.1 Consumers SHALL have the ability to set the values of a standard set of items controlling the appearance of a Producer's output. {R401} [Not fully captured]

7.11.2 A standard set of CSS classes MAY be used by a Consumer to control the appearance of a Producer's output.

7.11.2.1 wsiaBackgroundColor

7.11.2.2 wsiaForegroundColor

7.11.2.3 wsiaFont

7.11.2.4 ...

For client/browsers that do not support CSS, it will be useful for the Producer to have access to the values of the standard CSS classes for the purpose of generating output that may be reasonable aggregated with the output of other Producers.

7.11.3 Consumers MAY set a property on Producers specifying the CSS stylesheet with the Consumer's values for the standard classes.

[m2: Conformance requirement – see OASIS TC on Conformance.
Also as a reference example, see http://www.ebxml.org/specs/index.htm.
(ebTA v 1.04 – Section 9 on Conformance)]

8. Appendix A: WSIA Properties

8.1.1 wsia:output/wsia:deviceType – The device type that will render the output. [Is there a standard set of these defined somewhere?]

[Does it matter whether the Browser is a Internet Explorer or a Netscape Navigator particularly in issues like enabling cookies etc? Are additional properties required??]

8.1.2 wsia:output/wsia:contentType – The type of content requested by the Consumer (WML, XHTML, PDF, …)

8.1.3 wsia:output/wsia:supportedContentTypes (from WSRP) – The types of content the Producer is willing to supply.

8.1.4 wsia:output/wsia:supportedModes (from WSRP) – The modes where the Producer supports generating output.

8.1.5 wsia:baseURL – Proposed means of enabling Producer URL rewriting. The Producer would concatenate any name/value pairs for the querystring portion of the URL onto this Consumer supplied baseURL.

8.1.6 wsia:cssStylesheetURL – The URL for the CSS stylesheet the Consumer will use when sending the output to the End-User's device.

8.1.7 wsia:user/wsia:locale (from WSRP) – The locale of the End-User

8.1.8 wsia:user/wsia:preferredCharacterSet (from WSRP) – The preferred character set of the End-User.

8.1.9 wsia:cacheable (from WSRP) – Boolean (?) indication whether the Producer's output may be cached.

8.1.10 wsia:supportsQueuedOperations (does this get added by another use case?) – This property was added for the scenarios requiring offline operations. It indicates that a Producer supports its operations being queued by consuming applications.

8.1.11 wsia:versionNumber – This property allows a lighter weight versioning capability than registering a new service in a UDDI registry. It is expected this would be useful when the output of the Producer is changing, but its interface (operations/properties/ …) are not. In general this should be read-only to the Consumer.

8.1.12 wsia:urlRewriter – Property indicating whether the Consumer expects the Producer to rewrite URLs based on the baseURL property. Possible values include Consumer, Producer, (Both??).

9. Appendix B: Common Look and Feel Abstractions

This will ultimately become a (normative?) list of the UI abstractions that embedded services MAY support. At this point in time, there is an ongoing exercise to collate common abstractions (CSS classes, etc.) that different committee members are familiar with from their own experience. The resulting matrix will then be normalized into list format. What follows is the matrix, as it stands now. Other columns will be added as members contribute.

	IBM WPS	WSUI		Comments
Links	A A:visited A:hover			
Fonts	.portletText .portletSmText .portletTinyText .buttonText .fieldErrorText	wsui-font wsui-font-small wsui-font-large wsui-dim wsui-dim-small wsui-error wsui-error-small wsui-ok wsui-ok-small wusi-form-label		
Tables	.tableHead .tableText .tableShdRow	wsui-table wsui-table-row-header wsui-table-row-sectionheader wsui-table-row-odd wsui-table-row-even		
Sections	.portletTitle .portletTitleCust	wsui-section-title wsui-trail wsui-trail-current		
General	UL .portletColorBack .portletHead .portletBody .portletBack	wsui-page-title wsui-block-bgcolor		
Menus		wsui-menu wsui-menu-current		
Position		Wsui-right Wsui-left Wsui-top Wsui-bottom Wsui-shape		This describes the position of an element
Media properties		Wsui-media format		Probably for Audio and Video streaming of content
Background properties		Wsui-backgroung color Wsui-background pic		Logo? Or Uniform background color
Margins		Wsui-left margin Wsui-right margin Wsui-top margin Wsui-bottom margin		

Spacing		Wsui-space_between_row (of text)		spacing
Dimensions		Wsui_height_of_elements Wsui_width_of_elements		Could be redundant – linked to Font size, but for general representation of all elements
Language		Wsui_language		? not sure about this

10. Appendix C Security Standards Efforts

Other standards efforts are underway for the purpose of defining security and authentication protocols for XML and web services. While WSIA will need to be concerned about these issues, it is expected that concern will be addressed by tracking the work of these other efforts, including:

10.1 W3C

- XML Signature (joint with IETF) - The mission of this working group is to develop an XML compliant syntax used for representing the signature of Web resources and portions of protocol messages (anything referencable by a URI) and procedures for computing and verifying such signatures.
  - Syntax and Processing => Recommendation (12/Feb/2002)
- XML Encryption – The mission of this Working Group (WG) is to develop a process for encrypting/decrypting digital content (including XML documents and portions thereof) and an XML syntax used to represent the (1) encrypted content and (2) information that enables an intended recipient to decrypt it.
  - XML Encryption Syntax and Processing => Candidate Recommendation (04/Mar/2002)
  - Decryption Transform for XML Signature => Candidate Recommendation (04/Mar/2002)
- P3P – The Platform for Privacy Preferences Project (P3P), developed by the World Wide Web Consortium, is emerging as an industry standard providing a simple, automated way for users to gain more control over the use of personal information on Web sites they visit.
  - The Platform for Privacy Preferences => Proposed Recommendation (28/Jan/2002)
- XML Key Management – The mission of this working group is to develop a specification of XML application/protocol that allows a simple client to obtain key information (values, certificates, management or trust data) from a web service.
  - XML Key Management Specification => Working Draft (18/Mar/2002)

10.2 IETF

- GeoPriv – As more and more resources become available on the Internet, some applications need to acquire geographic location information about certain resources or entities. These applications include navigation, emergency services, management of equipment in the field, and other location-based services. But while the formatting and transfer of such information is in some sense a straightforward process, the implications of doing it, especially in regards to privacy and security, are anything but. The primary task of this working group will be to assess the authorization, integrity and privacy requirements that must be met in order to transfer such information, or authorize the release or representation of such information through an agent.
  - Goal => Be finished by now … no drafts available yet
- Common Authentication Technology - The goal of the Common Authentication Technology (CAT) Working Group is to provide distributed security services (which have included authentication, integrity, and confidentiality, and may broaden to include authorization) to a variety of protocol callers in a manner which insulates those callers from the specifics of underlying security mechanisms.

- GSSAPI SASL mechanisms => draft (30/May/2001)

- XML Digital Signatures – see bullet under W3C

10.3 OASIS

- XML-Based Security Services (SAML) - will produce set of one or more Committee Specifications that cover use cases and requirements, core assertions, protocols, bindings, and a conformance suite, all of the aforementioned to be examined with respect to security considerations. The work will take the S2ML specification and the intended submission of AuthXML, along with any other relevant and timely submissions, into consideration.
  - **SAML 1.0 Specification Set => post-last call (29/Mar/2002)**
- eXtensible Access Control Markup Language (XACML) - XACML is expected to address fine grained control of authorized activities, the effect of characteristics of the access requestor, the protocol over which the request is made, authorization based on classes of activities, and content introspection (i.e. authorization based on both the requestor and potentially attribute values within the target where the values of the attributes may not be known to the policy writer). XACML is also expected to suggest a policy authorization model to guide implementers of the authorization mechanism.
  - XACL Language => draft version 12 (01/Apr/2002)
- Rights Language - The purpose of the Rights Language TC is to define the industry standard for a digital rights language that supports a wide variety of business models and has an architecture that provides the flexibility to address the needs of the diverse communities that have recognized the need for a rights language.
  - TC formed (02/Apr/2002)