HTTP Binding Use Cases

Table of Contents

1.Retrieve document from HTTP URL

These use cases describe the retrieval of a document from an HTTP URL, which may be constant or have varying elements. The document format may vary, however all cases assume that there is an XML schema representation of the document. There are no semantics associated with the content of the document, in particular any URLs within the document have no special processing.

For all these use cases security and proxy connection are assumed to be orthogonal issues and are dealt with via specific use cases.

Currently there are no use cases for explicitly controlling caching. For the moment that is assumed to be SCA runtime specific.

In general I don't think its possible to determine from the interface and binding which HTTP method should be used so this always needs to be specified in the binding.

1.1 Retrieve document with known schema from fixed HTTP URL

Description	Retrieve document with known schema from fixed HTTP URL
Preconditions	• The retrieval of the document can be done using an HTTP GET request • An XML schema type is available that defines the document structure. • A WSDL portType is constructed with an empty input message and an output message consisting a single XML element of the correct type for the document • Optional: the portType may includes one or more fault messages for application level faults • Optional: certain HTTP status codes are mapped to faults • Optional: certain HTTP status codes are identified as successful • Optional: the HTTP version is 1.0 (default is 1.1) • The accepted formats of the document have corresponding wireFormat elements • An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and accepted wireFormats
What happens	An HTTP GET request is formed using the fixed URL, and with no content. The following HTTP headers are set to specific values: • Content-Length: 0 • Accept: <list of content-types corresponding to the accepted wireFormats> Other headers may be set, however that should not affect the behaviour observed by the calling component. In particular if chunking is enabled subsequent chunks are handled by the interaction between the binding and wireFormat implementation. Chunks are not exposed to the calling component. If HTTP headers are set that result in certain expected non-success response codes (e.g. 304, not modified) then the binding implementation MUST handle those transparently to the calling component.

	Success case 1: An HTTP response is received with response code 200 (success) or other response code configured as indicating success. The response includes a non-zero Content-Length, and a Content-Type that corresponds to one of the accepted wireFormats. The HTTP response is passed to the wireFormat to map the content into the runtime representation. What about other 2xx response codes? For this case I don't believe any other 2xx responses are expected Success case 2: An HTTP response with response code 301 (moved permanently) or 307 (temporary redirect) is received. Behaviour follows above with a new HTTP GET request to the redirected URL. For response code 301 the binding SHOULD use the new URL for future requests; for response code 307 the binding SHOULD use the original URL for future requests. Failure case 1: The HTTP connection fails due to malformed URL or failure to connect to the URL. A system fault is returned to the calling component indicating that the binding URL is invalid. Failure case 2: The HTTP connection times out before a response is received. A system fault is returned to the calling component indicating that the resource is unavailable. Failure case 3: An HTTP response with response code other that 200, 301 or 307 is received. If there is a defined mapping from the response code to an application fault, that application fault is returned (how does the HTTP response message get mapped into that fault? Via a wireFormat?) If there is no defined mapping a system fault is returned to the calling component. The SCA runtime MAY distinguish between client errors (response codes 4xx) and server errors (response codes 5xx). Failure case 4: An HTTP response with response code 200, 301 or 307 is received but the Content-Length is zero, or the Content-Type does not correspond to one of the accepted wireFormats, or the wireFormat is unable to process the response body. A system fault is returned to the calling component indicating that an unexpected response was received.
Post-conditions	Success: The runtime representation of the document is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example	<reference>

definition	<interface.wsdl ...portType with one operation...> <binding.http uri="http://....." httpVersion="1.0" method="GET"> <response> <accepts type="text/xml"> <wireFormat.xml> </accepts> <accepts type="application/json"> <wireFormat.json> </accepts> <fault responseCode="410" faultName="doesNotExist"/> <success responseCode="201"/> <!-- artificial example --> </response> </binding.http> </reference> The accepts element could be extended to include additional items which would map to Accept-Charset, Accept-Encoding, Accept-Language if we feel there is a valid case for any of these. We may also want to allow configuration of the read timeout and number of retries for HTTP GET requests.

1.2 Retrieve document with known schema from variable HTTP URL

Description	Retrieve document with known schema from variable HTTP URL
Preconditions	• As for use case 1.1; in addition the URL has variable parts, specific combinations of which identify different documents with the same schema. The variable parts are either within the URL path or query string. • A WSDL portType is constructed with an input message including a top-level string typed element for each parts of the URL that is to vary and an output message consisting a single XML element of the correct type for the document. • Within the URL the variable parts are specified using a defined syntax that identifies the matching element in the input message definition. This could be a direct use of the input message part name. Do we need to allow for doc/lit wrapped interface style here? For query values we may need some way to express that a constant value appears or not based on a boolean input element. Should look at JAX-RS as an example. • An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and accepted wireFormats • Note: a degenerate version of this use case could have the entire URL passed as an input parameter
What happens	An HTTP GET request is formed using the URL, and with no content. The variable parts in the URL are replaced by the corresponding values from the input message (suitably URL encoded) The behaviour is then consistent with use case 1.1. Failure case 1:

	A response code is returned indicating that an invalid combination of variable parts has been specified. A system fault is returned to the calling component indicating that an unexpected response was received.
Post-conditions	Success: The runtime representation of the document is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ...> <binding.http uri="http://example.org/books/{author}?page={page}" method="GET"/> <response> <accepts type="text/xml"> <wireFormat.xml> </accepts> </response> </binding.http> </reference>

1.3 Retrieve document with known schema from variable HTTP URL based on values within an existing element

Description	Retrieve document with known schema from variable HTTP URL based on values within an existing element. My feeling is that this case, and possible extensions of it to access subelements in the returned document, introduces integration logic into the binding which is not appropriate.
Preconditions	• As for use case 1.2; existing complex type(s) contain the values for the variable elements in the URL. • A WSDL portType is constructed with an input message including the existing complex type(s) and an output message consisting a single XML element of the correct type for the document. • Within the URL the variable parts are specified using a defined syntax that identifies the matching element in the input message definition. This syntax either directly references the elements via Xpath or specifies logical names which are then mapped by the HTTP binding to elements via Xpath. • An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and accepted wireFormats
What happens	An HTTP GET request is formed using the URL, and with no content. The variable parts in the URL are replaced by the corresponding values from the input message (suitably URL encoded) The behaviour is then consistent with use case 1.1. Failure case 1: A response code is returned indicating that an invalid combination of variable parts has been specified. A system fault is returned to the calling component indicating that

	an unexpected response was received.
Post-conditions	Success: The runtime representation of the document is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ...> <binding.http uri="http://example.org/books/{lastName}/{firstName}" method="GET"/> <input name="lastName" element="input/author/lastName"> <input name="firstName" element="input/author/firstName"> <response> <accepts type="text/xml"> <wireFormat.xml> </accepts> </response> </binding.http> </reference>

1.4 Retrieve self-describing document with unknown schema from fixed HTTP URL

Description	Retrieve self-describing document with unknown schema from fixed HTTP URL I'd be happy to remove this use case if no-one supports it.
Preconditions	• As for use case 1.1, except that the type of the document is not known at the time the reference is authored. • A WSDL portType is constructed with an empty input message including and an output message consisting a single XML element of type <anyType>. Is this valid for WSDL interfaces? How would this then appear to a calling Java component for example? • An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and accepted wireFormats
What happens	The behaviour is consistent with use case 1.1. It is expected that either the schema for the retrieved document can be identified and retrieved from that document, or identified and already available to the SCA runtime. Failure case 1: The HTTP response body cannot be parsed due to lack of schema information.
Post-conditions	Success: The runtime representation of the document is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	As per case 1.1

1.5 Retrieve multiple different documents via the same SCA reference

Description	Retrieve multiple different documents via the same SCA reference
Preconditions	• A combination of the above use cases; the documents are available from a common base URL with specific paths and possible variable parts for each document. • A WSDL portType is constructed with multiple operations, one for each different type of document to be retrieved. Each operation has an appropriate input message as per other use cases and an output message consisting a single XML element of the appropriate type for the document to be retrieved. • An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and accepted wireFormats for each operation
What happens	The behaviour is consistent with use case 1.1, where the construction of the HTTP request and processing of the HTTP response is done based on operation-specific values in the binding.http element.
Post-conditions	Success: The runtime representation of the document appropriate for the operation that was invoked is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ...portType with multiple operations...> <binding.http uri="http://...common base URL/"> <response> <accepts type="text/xml"> <!-- all operations accept XML --> <wireFormat.xml> </accepts> <fault responseCode="410" faultName="doesNotExist"/> </response> <operationProperties name="getBook" uri="book/{author}" method="GET"> <accepts type="application/json"> <wireFormat.json> </accepts> </operationProperties> <operationProperties>....</operationProperties> </binding.http> </reference>

1.6 Retrieve document from HTTP URL via proxy

Description	Retrieve document from HTTP URL via proxy
Preconditions	• Proxy configuration is required in order to access the document via its URL, including host, port and authentication • An SCA reference with binding.http specifying the proxy information • Remainder as per previous use cases
What happens	An HTTP GET request is formed using the document's URL

	The connection is made to the proxy address supplied. Remainder of behaviour is as per previous use cases. Failure case 1: A connection to the proxy host could not be made A system fault is returned to the calling component indicating that the proxy configuration is invalid.
Post-conditions	Success: The runtime representation of the document appropriate for the operation that was invoked is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ... /> <binding.http uri="http://...common base URL/" method="GET"> <proxy host="..." port="80"> <authentication ...proxy authentication information.../> </proxy> ... </binding.http> </reference> Authentication information is likely to be runtime-specific. Do we want to include this in the standard schema? For basic authentication having user name and password in the binding element does not seem like a good idea. I'm not aware of authentication information being present in any of the other SCA specs.

1.7 Retrieve document from HTTP URL with basic authentication

Description	Retrieve document from HTTP URL with basic authentication
Preconditions	• Basic authentication configuration is required in order to access the document via its URL • An SCA reference with binding.http specifying the basic authentication information • Remainder as per previous use cases
What happens	An HTTP GET request is formed using the document's URL as in previous use cases. An SCA runtime may choose to include an HTTP Authorization header including "Basic", the base-64 encoded user name and password. Success case 1: If the Authorization header was omitted from the original request and an HTTP response with response code 401 (unauthorized) is received, behaviour follows earlier use cases with a new HTTP GET request including the appropriate Authorization header. Failure case 1: An HTTP response with response code 401 (unauthorized) is received when the original request included the Authorization header, an no specific mapping of the 401 response code to an application faul in the interface. A system fault is returned to the calling component indicating that the authentication information is invalid.

Post-conditions	Success: The runtime representation of the document appropriate for the operation that was invoked is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl .../> <binding.http uri="http://...common base URL/" method="GET"> <authentication ...basic authentication information.../> ... </binding.http> </reference>

1.8 Retrieve document from HTTPS URL

Description	Retrieve document from HTTPS URL
Preconditions	• The document's URL uses the https scheme. • An SCA reference with binding.http specifying the https URL. • Remainder as per previous use cases.
What happens	An HTTP GET request is formed using the document's URL as in previous use cases. The request is sent via a secure connection established using the document's URL and SSL configuration information provided in the binding.
Post-conditions	Success: The runtime representation of the document appropriate for the operation that was invoked is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ...portType with multiple operations...> <binding.http uri="https://...secure URL/"> <authentication ...SSL configuration information.../> ... </binding.http> </reference>

1.9 Retrieve document from HTTP URL including links

Description	Retrieve document with known schema from fixed HTTP URL. The returned document includes links to other resources. Web pages these days are conglomerations of HTTP GET requests - images, CSS, Javascript, frames & iframes. They may be perceived as a single "page", when it is anything but. Retrieving data from these otherwise unmodified web pages actually requires following multiple links to get to the data that the service actually needs. These URLs may point to referenced resources (the book's publisher, the queue's queue manager, the list of referenced documents, etc...), or to uninteresting items (an button's image). These are modelled

	explicitly as URLs in the interface. The binding does not automatically get them too so that they can be represented as nested complex objects. We could possibly provide an endpoint reference or other veneer over these references to avoid exposing the fact that the reference was obtained via use of binding.http rather than some other binding.
Preconditions	As per use case 1.1
What happens	Nothing different. The links are treated as application data. It is possible to then retrieve the document referred to by the link using the degenerate form of use case 1.2 where the variable part of the URL is the entire URL. By convention a reference with an HTTP binding could have a specific operation set up to retrieve linked resources.
Post-conditions	Success: The runtime representation of the document appropriate for the operation that was invoked is returned to the calling component. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ... /> <binding.http uri="http://...base URL/"> <operationProperties name="getLinkedResource" uri="{resourceURL}" method="GET"/> </operationProperties/> ... </binding.http> </reference>

1.10 Retrieve document from HTTP URL using a WSDL port with HTTP binding

Description	Obtaining a document from an HTTP URL using a WSDL port with HTTP binding
Preconditions	Should this be a binding.http use case at all? Use binding.ws for this if it supports the WSDL HTTP binding? • The retrieval of the document can be done using an HTTP GET request • An XML schema type is available that defines the document structure. • A WSDL document including a portType is available that includes an operation to retrieve the document, a WSDL HTTP binding for that operation, and a WSDL port which identifies the endpoint address for the document. • An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http referring to the WSDL port.
What happens	As for earlier use cases with the URL and details of how to retrieve the document derived from the WSDL HTTP binding. More detail required here if we decide this is a valid use case
Post-conditions	Success: The runtime representation of the document appropriate for the operation that was invoked is returned to the calling component. Failure: An appropriate fault is returned to the calling component.

Notes	<reference> <interface.wsdl ... /> <binding.http wsdlPort="identity of WSDL port"/> </reference>

1.11 Other use cases for retrieve document

Any other use cases? Are there cases where an HTTP GET is used with a document in the message body? I assume that's not a pattern we would support.

Also other use cases involving other HTTP methods. It should be possible to name any method desired on the binding/operationProperties, along with the required URL and body mapping to parameters and output.

2.Send document to HTTP URL

2.1 Send document to variable HTTP URL

Description	Send document to variable HTTP URL
Preconditions	• The document can be sent using an HTTP POST request, and expected success response code is 200 (success) or 201 (created) Note: Depending on what I read, this could be POST or PUT • An XML schema type is available that defines the document structure. • A WSDL portType is constructed with an input message including a top-level string typed element for each parts of the URL that is to vary, an element of the correct type for the document, and an empty output message. Note: not no output message, this is not a one-way operation. • Optional: the portType may includes one or more fault messages for application level faults • Optional: certain HTTP status codes are mapped to faults • Optional: certain HTTP status codes are identified as successful • Optional: the HTTP version is 1.0 (default is 1.1) • The URL has variable parts as per use case 1.2. An SCA reference with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and wireFormats for the document.
What happens	An HTTP POST request is formed using the URL. The variable parts in the URL are replaced by the corresponding values from the input message (suitably URL encoded). The HTTP message body is serialized using the identified wireFormat. The following HTTP headers are set to specific values: • Content-Length: <length of the content produced by the wireFormat> • Content-Type: <content type configured on the binding> Other headers may be set, however that should not affect the behaviour observed by the calling component. In particular if chunking is enabled subsequent chunks are handled by the interaction between the binding and wireFormat implementation. Chunks are not exposed to the calling

	component. If HTTP headers are set that result in certain expected non-success response codes then the binding implementation MUST handle those transparently to the calling component. Success case: An HTTP response is received with response code 200 (success) or 201 (created) or other response code configured as indicating success. Do we expect there to be any content in the HTTP response? Should there be a separate use case for that? Failure case 1: The HTTP connection fails due to malformed URL or failure to connect to the URL. A system fault is returned to the calling component indicating that the binding URL is invalid. Failure case 2: The HTTP connection times out before a response is received. A system fault is returned to the calling component indicating that the operation failed. Failure case 3: An HTTP response with response code other that 201 is received. If there is a defined mapping from the response code to an application fault, that application fault is returned (how does the HTTP response message get mapped into that fault? Via a wireFormat?) If there is no defined mapping a system fault is returned to the calling component. The SCA runtime MAY distinguish between client errors (response codes 4xx) and server errors (response codes 5xx).
Post-conditions	Success: The document is now available at the URL. Failure: An appropriate fault is returned to the calling component.
Example definition	<reference> <interface.wsdl ...portType with one operation...> <binding.http uri="http://example.org/books/{author}/title/{title}" method="POST"/> <sends contentType="text/xml"/> <wireFormat.xml/> <response> <success responseCode="200"/> <!-- artificial example --> </response> </binding.http> </reference> I don't think we need to support multiple wireFormats i.e. multiple possible representations of the same document to be sent. How is the content-type best identified in this case? I don't like the idea of having an attribute on the binding, but that then requires an additional element. Is there some way we could make the content-type

	automatically derivable from the wireFormat? Does it make sense to have these settings on the binding or should there always be an operationProperties element containing them?

3.Provide HTTP-based access to information from an SCA service

3.1 Fixed HTTP URL addressability of read-only service information

Description	Fixed HTTP URL addressability of read-only service information Essentially the reverse of use case 1.1 “Retrieve document with known schema from fixed HTTP URL” This is essentially the reverse of the first use-case described above, with one additional caveat. It may be desirable to map to multiple URLs (.xml, .rdf, .xhtml, .txt), so for the service side of this binding a natural thing, I think would be to specify an XSLT file correlated with a particular MIME type. In the XML case, that might be added as a XSL processing instruction.
Preconditions	• The retrieval of the document is provided in response to an HTTP GET request to a specific URL • The requested information is available in a number of different forms. • An XML schema type is available that defines the document structure. • A WSDL portType is constructed with a single operation with an empty input message and an output message consisting a single XML element of the correct type for the document • Optional: the portType may includes one or more fault messages for application level faults that may be returned from the component connected to the service. • Optional: certain faults mapped to HTTP status codes • Optional: a specific HTTP status code represents a successful result • Optional: the HTTP version is 1.0 (default is 1.1) • The accepted formats of the document have corresponding wireFormat elements • An SCA service with interface.wsdl referring to the above WSDL portType, and binding.http specifying the URL, HTTP version and provided wireFormats • An SCA component wired to the SCA service whose implementation returns the required information when the operation is invoked. The operation SHOULD be idempotent. • The URL mapping to operation descirbed here is provided by operationSelector.httpDefault.
What happens	An HTTP request is received at the fixed URL. The following HTTP headers are expected to be set to specific values: • Content-Length: 0 • Accept: <list of content-types accepted by the requester> Other headers may be set, however that should not affect the behaviour

observed by the component wired to the service. In particular if chunking is enabled chunks are handled by the interaction between the binding and wireFormat implementation. Chunks are not exposed to the wired component.
If HTTP headers are set that result in certain expected non-success response codes (e.g. 304, not modified) then the binding implementation MUST handle those transparently to the wired component.

Success case 1:

One of the provided Content-Type values matches one of the values in the Accept header.
The wired component is called and returns an object matching the required document type.
An HTTP response is sent with response code 200 (success) or the specific response code configured as indicating success.
The response includes a non-zero Content-Length, and the matching Content-Type. The corresponding wireFormat is used to construct the HTTP response body from the runtime representation.
Would we ever expect this service to return 301 or 307 or other response codes?

Failure case 1:

There is no provided Content-Type that matches one of the values in the Accept header.
The wired component is not called.
An HTTP response is sent with response code 406 (not acceptable). The response SHOULD include an entity containing a list of available entity characteristics and location(s) from which the user or user agent can choose the one most appropriate [rfc2616]

Failure case 2:

One of the provided Content-Type values matches one of the values in the Accept header.
The wired component is called and returns a fault which has a mapping to a specific response code.
An HTTP response is sent with the mapped response code. The response includes an entity containing the fault information.

Failure case 3:

One of the provided Content-Type values matches one of the values in the Accept header.
The wired component is called and returns a fault which has no mapping to a specific response code.
An HTTP response is sent with the response code 500 (Internal Server Error). The response includes an entity containing the fault information.

Failure case 4:

The request could not be processed due to some error at the HTTP level. An HTTP response is sent with the appropriate

	response code, typically one of the 4xx client errors.
Post-conditions	Success: The required representation of the document is returned to the client. Failure: An appropriate HTTP response is returned to the client.
Example definition	<service> <interface.wsdl ...portType with one operation...> <binding.http uri="http://example.org/getStatusReport/"/> <operationSelector.httpDefault/> <provides contentType="text/xml"> <wireFormat.xml/> </provides> <provides contentType="application/json"> <wireFormat.json/> </provides> <response> <success responseCode="200"/> <!-- artificial example --> </response> </binding.http> </service>

3.2 Variable HTTP URL addressability of read-only service information

Description	Variable HTTP URL addressability of read-only service information
Preconditions	As per use case 3.1, with the URL providing a template mapping to parameters defined in the input message.
What happens	
Post-conditions	
Example definition	<service> <interface.wsdl ...portType with one operation...> <binding.http uri="http://example.org/getBook/{author}/{title}/"> <operationSelector.httpDefault/> <provides contentType="text/xml"> <wireFormat.xml/> </provides> <provides contentType="application/json"> <wireFormat.json/> </provides> </binding.http> </service>

3.3 Variable HTTP URL addressability of update operation on service

Description	Variable HTTP URL addressability of update operation on service.
Preconditions	As per use case 3.2

What happens	Potential different set of response codes and methods – more detail required Eric: Again, I'd love more details. Thinking about a use-case like "updating my preferences" in a standard HTTP-based application, I get a single form submission to update a set of data. Whereas, in an SOAP application, I might wish I had operations for "setEmailDeliveryPreference", "setHomePageDisplayPreference". Obviously, I could provide a single operation that takes a single set of data, as in "updateProfilePreferences", and map that onto the existing HTTP-POST request. My concern from what you said, "the set of operations I want to make available", I'm not sure whether you're suggesting both possibilities, or really just wrappers around already defined URLs to do a "POST", that is, do you conceive of the HTTP binding as mapping N operations onto one or more HTTP verbs to M resources, or is it simply a wrapper around a single HTTP resource with a single verb? (my thinking is that we define a simple 1-1 mapping in the HTTP binding. Any higher level orchestration/sequencing is not the business of a binding in my opinion).
Post-conditions	
Example definition	

3.4 Provide ATOM feeds from SCA service

Description	Stateful "monitor" of an service providing 1 or more atom feeds, supporting three operations "getNewEntriesForFeedFoo", "getLastNEntriesForFeedFoo", "getContentForFoo" - per Atom feed This is a variation on use case 3.2 with the appropriate URL templates, and multiple operations in the portType. This use case needs updating to be consistent with the others
Preconditions	Service that offers up one or more atom feeds Atom feeds are changing slowly enough that polling for data at lazy intervals (1 - 60 minutes) is sufficient Customer has a name for each URL. Operations in the interface follow a known pattern like getNewEntriesForFeedFoo Customer has particular data that they're interested in from each feed, such as author, modification date, ID, other metadata
What happens	A service requests an update from a specific specific feed by calling getNewEntriesForFeedFoo (where "Foo" changes per feed)
Post-conditions	Returns a list of new entries, the list of the last N entries requested, or the content for the particular entry.
Notes	From Eric: It would be much more natural to define the interface with "interface.atom", wherein the relevant metadata would be flagged for each feed, and the content of the root element of a feed entry was established. This would follow known mapping rules to map to a WSDL expression of the same, if that was required. Then again, I filed an

	issue for letting Assembly allow non-WSDL interface descriptions for a reason....

3.5 HTTP access to SCA Java component with JAX-RS annotations

Description	HTTP access to SCA Java component with JAX-RS annotations
Preconditions	• A Java component whose interface is defined by a Java class with JAX-RS annotations. • An SCA service with interface.java referring to the above Java interface, and a binding.http with the JAX-RS wireFormat and operationSelector.
What happens	The behaviour of the HTTP binding is derived from the JAX-RS annotations, overridden by any specific HTTP binding configuration. If this use case is acceptable, then we need to define in detail how the JAX-RS annotations map to the binding.http elements.
Post-conditions	An HTTP client interacts with the Java component via the HTTP binding. It should not be able to tell the difference between this use case and those where JAX-RS annotations are not used.
Example definition	<service> <interface.java ...interface with JAX-RS annotations...> <binding.http> <operationSelector.httpJaxrs/> <wireFormat.httpJaxrs/> </binding.http> </service>

3.6 HTTP access to component using WSDL port with HTTP binding

Description	HTTP access to component using WSDL port with HTTP binding Similar to 3.5 where instead of JAX-RS annotations we have a WSDL HTTP binding.
Preconditions	
What happens	
Post-conditions	
Notes	

4.Requirements on the HTTP binding

This section needs updating once the use cases above have had some review.
In order to satisfy the use cases described above, an HTTP binding element needs to be able to express the following:

- The optional default mapping from content-type to wireFormat for requests. The keys

of the map define the default set of accepted content-types for a service or requested content-type for a reference. Supplied wireFormats include:
  - wireFormat.json
  - wireFormat.atom
  - wireFormat.xml
- The optional default mapping from content-type to wireFormat for responses. The keys of the map define the default set of accepted content-types for a service or requested content-type for a reference. Supplied wireFormats include:
- the operationSelector. Supplied operationSelectors include:
  - operationSelector.httpJAXRS: maps HTTP requests to operations based on JAX-RS annotations in the Java interface
  - operationSelector.httpDefault: maps HTTP requests to operations based on the HTTP binding operationProperties element
  - operationSelector.httpWSDL: maps HTTP requests to operations based on the HTTP binding in the referenced WSDL document.

For each operation in the interface an operationProperties element which includes:

- a URL template which may be absolute or relative to the base URL. The URL template may include replaceable parameters in the path or query string. The syntax defined for JAX-RS seems as good as any to define the URL template. e.g.: /constantPart/{param1}/moreConstant?query1={param2}
- the HTTP method supported for this operation *(more than one needed?)*
- an optional mapping from the replaceable parameters to elements in the input message. The default mapping assumes the replaceable parameters match top-level string-types message parts. A deeper mapping to sub-elements/attributes could be supported using XPath for instance.
- an optional mapping from the HTTP body to an element in the input message for create/update operations, as above.
- an optional mapping from the HTTP body to an element in the output message for retrieve operations, as above.
- an optional specific mapping from content-type to wireFormat
- *mapping from faults to response codes*

HTTP reference binding

- The default mapping from content-type to wireFormat. The keys of the map define the default set of requested content-types. (is more than one actually needed?) Supplied wireFormats include:
  - wireFormat.json
  - wireFormat.atom
  - wireFormat.xml

For each operation in the interface an operationProperties element as for the service binding.