Issue #12 HTTP Status Codes aka 500 v 200

Context: SOAP/XMLP HTTP Binding

Section 6.2 of the SOAP/1.1 [1] and XMLP/SOAP [2] states:

```
"6.2 SOAP HTTP Response

SOAP HTTP follows the semantics of the HTTP Status codes for communicating status
information in HTTP. For example, a 2xx status code indicates that the client's
request including the SOAP component was successfully received, understood, and
accepted etc.
```

`In case of a SOAP error while processing the request, the SOAP HTTP server MUST issue an HTTP 500 "Internal Server Error" response and include a SOAP message in the response containing a SOAP Fault element (see section 4.4) indicating the SOAP processing error."`

There is discussion of whether it is appropriate to use the HTTP 5xx status code to signal the failure of a SOAP processor to process a message delivered in an HTTP POST request message.

Possible Resolutions

- Use 2xx status codes for ALL XMLP/SOAP messages carried in HTTP response messages
- Use 5xx status code for ALL XMLP/SOAP Fault messages in HTTP response messages (and 2xx for ALL non-Fault XMLP/SOAP messages) (status quo I think).
- Use 2xx status code for some (TBD) classes of XMLP/SOAP Fault message and 5xx for some (TBD) classes of XMLP/SOAP Fault message carried in HTTP responses.

Arguments (1)

Pro 2xx (for Faults)	**Pro 5xx (for Faults)**	**Notes**
SOAP and HTTP processing are separate. 5xx should be used only to indicate internal failure of an HTTP server/intermediary. Should NOT be used to signal failure of a layered SOAP processor.	SOAP and HTTP processing for SOAP bound to HTTP are inseparable. The SOAP/HTTP processor IS an HTTP processor and HTTP semantics apply on internal failure.	Seem to hinge fundamentally on whether we view SOAP as layer on-top of HTTP or as an extension of HTTP
Some HTTP intermediaries can be configured to automatically rewrite the body of HTTP 500 response messages, e.g., to provide more helpful error messages; this rewriting will interfere with the delivery of a SOAP fault. Such intermediaries might be HTTP origin servers acting in front of a CGI-based implementation, but may also occur in explicit or transparent proxy servers. (from Larry Masinter)		Some degree of discussion of 1) Whether the HTTP specs. allow re-writing of response content. 2) Whether common implementations allow this behaviour to be configured into origin, proxy and intermediary servers. 3) Does it actually happen in practice.

Arguments (2)

Pro 2xx (for Faults)	**Pro 5xx (for Faults)**	**Notes**
	Correct operation of other web applications: search engines, proxies, annotation engines etc. rely on the correct use of HTTP status codes. Making inappropriate use of 2xx status codes to will damage the operation of these (pre-existing) web applications. Must observe HTTP semantics "It is part of life when living within HTTP (which is a transfer protocol and not a transport protocol) - HTTP owns the message and HTTP status codes are for the complete message - you can't have partial success or partial failure for the HTTP message."	
Streaming: Use of 500 to signal failures forces an irrevocable choice of status codes. Use of 200 only, for success and SOAP failures would aid streaming implementations.		Handling failure 'mid-stream' is a much more complex issue than correctly handling HTTP status codes. Abort of resulting messages; Substitution of Fault Message..
Some practical difficulties in generating HTTP 5xx POST response messages with full control over Content-Type and Body content.		Seems a bit spurious. (Mail thread offered fixes/workarounds).

“On the use of HTTP as a substrate for other Protocols”

(Section 8: draft-moore-using-http-01.txt – expired ID dated 16th October 2000, Keith Moore)

```
   A few guidelines are therefore in order:

  o    A layered application should use appropriate HTTP error codes to
       report errors resulting from information in the HTTP request-line
       and header fields associated with the request.  This request
       information is part of the HTTP protocol and errors which are
       associated with that information should therefore be reported using
       HTTP protocol mechanisms.

  o    A layered application for which all errors resulting from the
       message-body can be classified as either "complete success" or
       "complete failure" may use 200 and 500 for those conditions,
       respectively.  However, the specification for such an application
       must define the mechanism which ensures that its successful (200)
       responses are not cached by intermediaries, or demonstrate that
       such caching will do no harm; and it must be able to operate even
       if the message-body of an error (500) response is not transmitted
       back to the client intact.

  o    A layered application may return a 200 response code for both
       successfully processed requests and errors resulting from the
       request message-body (but not from the request headers).  Such an
       application must return its error code as part of the response
       message body, and the specification for that application protocol
       must define the mechanism by which the application ensures that its
       responses are not cached by intermediaries.  In this case a
       response other than 200 should be used only to indicate errors
       with, or the status of, the HTTP protocol layer (including the
       request headers), or to indicate the inability of the HTTP server
       to communicate with the application server.

  o    A layered application which cannot operate in the presence of
       intermediaries or proxies that cache and/or alter error responses,
       should not use HTTP as a substrate.
```

Questions:

- Should the HTTP status code reflect the outcome (success/failure) of an XMLP/SOAP processing?

  i.e. should HTTP say "Ok, here's the reply" independently from whether XMLP/SOAP processing was successful or not? (Hugo)

- Should ALL SOAP Fault messages carried in HTTP POST responses have the same HTTP status code?

- Are the only SOAP messages carried in an HTTP POST response with a status code of 5xx SOAP Fault messages?

Eric’s Scenario Questions

In the following scenarios (from Eric Jenkins [11]) what HTTP status codes should be returned.

a) HTTP Server attempts to start SoapProcessor to handle a Soap message and and the SoapProcessor crashes.

b) HTTP Server recognizes a Soap message (because of SOAPAction?) but the URL is non-existent, i.e., SoapProcessor might have handled it just fine but the server found some flaw in the header.

c) HTTP Server hands a Soap message to the SoapProcessor but the message envelope is ill-formed

d) HTTP Server hands a Soap message to a SoapProcessor requesting a stockprice for a non-existent stock

e) The SoapProcessor is actually not the destination but is only an intermediary, and the message is being transported via HTTP, and the SoapProcessor generates a mustUnderstand Fault.

Possible Resolutions

- Use 2xx status codes for ALL XMLP/SOAP messages carried in HTTP response messages
- Use 5xx status code for ALL XMLP/SOAP Fault messages in HTTP response messages (and 2xx for ALL non-Fault XMLP/SOAP messages) (status quo I think).
- Use 2xx status code for some (TBD) classes of XMLP/SOAP Fault message and 5xx for some (TBD) classes of XMLP/SOAP Fault message carried in HTTP responses.