

SOAP-Faults und deren Behandlung		Konvention
		soapfault 1.0.6
		Entwurf öffentlich
Kurzbeschreibung	<p>Im Rahmen der Kommunikation mittels SOAP-WebServices tritt die Problematik über den Umgang mit SOAP-Faults zutage.</p> <p>Dieses Dokument zielt nun darauf ab, eine einheitliche Vorgehensweise zu erstellen, wie SOAP-Faults organisatorisch und praktisch behandelt werden.</p> <p>Weiters soll eine einheitliche Übersicht über Fehlergruppen und über applikationsspezifische Erweiterungen derselben gegeben werden.</p>	
Autor(en)	Michael Liehmann	Projektteam / Arbeitsgruppe
	Franz-Josef Herpers	Kommarch/SoapFaults

1

Stelle	Vorgelegt am	Angenommen am	Abgelehnt am
	08.04.2005		

2

Dokumentklasse:

Konvention
 Empfehlung
 Information

Doku-Stadium:

Entwurf intern
 Entwurf öffentlich
 Empfehlung

3

SOAP-Faults

4

5 Inhaltsverzeichnis

6	(1)	Einleitung	3
7	(2)	SOAP-Faults Syntax.....	3
8	(2.1)	SOAP-Faults in SOAP 1.1.....	3
9	(2.2)	SOAP-Faults in SOAP 1.2.....	5
10	(3)	Fehlerunterscheidung	6
11	(3.1)	Fehler in der Applikation.....	6
12	(3.2)	Fehler auf der Clientseite.....	7
13	(3.3)	Fehler in der Transportschicht	7
14	(4)	SOAP-Faults vs. XML-Fehlermeldung	7
15	(4.1)	SOAP-Fault	7
16	(4.2)	XML-Fehlermeldung bzw. XML-Statusmeldung.....	7
17	(5)	Vorgehensweise bei Fehlern	8
18	(5.1)	Fehler aus Sicht der Transportschicht (System Exception)	8
19	(5.2)	Fehler aus Sicht der angesprochenen Systeme (Application Exception).....	8
20	(6)	Fehler- und Statuscodes.....	9
21	(6.1)	Einteilung der Fehler- bzw. Statuscodes.....	9
22	(6.2)	Fehler- bzw. Statuscodes in SOAP-Faults.....	10
23	(7)	Referenzen.....	11
24	(8)	Literaturvorschläge	12
25			

26 (1) Einleitung

27 Problemstellung und Ziel

28 WeBservices sind Applikationen die einen programmatischen Zugang zu einer
 29 Fachapplikation über das Internet anbieten. SOAP(Simple Object Access Protocol
 30 [SOAP11], [SOAP12]) bezeichnet ein Protokoll, mit dem WeBservices angesprochen
 31 werden können. Eine Vielzahl an Applikationen im Bereiche E-Government verwenden nun
 32 jene SOAP-WeBservices zur Kommunikation zwischen Applikation und Client bzw.
 33 zwischen einzelnen Applikationen. Im Rahmen dieser Verbindungen treten bisweilen
 34 Probleme auf. Etwaige Fehlermeldungen können mittels SOAP-Faults an die aufrufende
 35 Applikation weitergegeben werden. In den üblichen Entwicklungsumgebungen, wie JAVA
 36 AXIS und MS .NET, führen diese SOAP-Faults zu applikationsspezifischen Exceptions.
 37 Manche Fehlermeldungen sollen zu einer solchen Exception führen, bei anderen Fehlern
 38 ist es anzuraten, diese auf andere Weise zu übermitteln.

39 In diesem Papier wird eine technische und applikationsübergreifende Einteilung getroffen,
 40 welche Fehler SOAP-Faults darstellen und somit Exceptions erzeugen und welche
 41 Meldungen besser als ordentliche SOAP-Meldung an die anfragende Applikation
 42 zurückgegeben werden sollen.

43 (2) SOAP-Faults Syntax

44 Laut der SOAP-Spezifikation des W3C soll ein SOAP-Fault Fehlerinformationen innerhalb
 45 einer normalen SOAP-Nachricht in dem Element *env:Fault* übertragen. Da die simple
 46 Existenz dieses Elements in einer SOAP-Nachricht nur auf einen Fehler hinweist, aber
 47 nicht auf welchen bestimmten, ist diesem Element eine Struktur zugeordnet, die genauere
 48 Aufschlüsse zulässt. Die SOAP-Fault Syntax veränderte sich jedoch von der Version in
 49 SOAP 1.1 auf die Version in SOAP 1.2. Zum bessern Verständnis werden hier beide
 50 Strukturen kurz vorgestellt.

51 (2.1) SOAP-Faults in SOAP 1.1

52 Abbildung 1 beschreibt das XML-Element SOAP-Faults laut SOAP-Spezifikation 1.2.

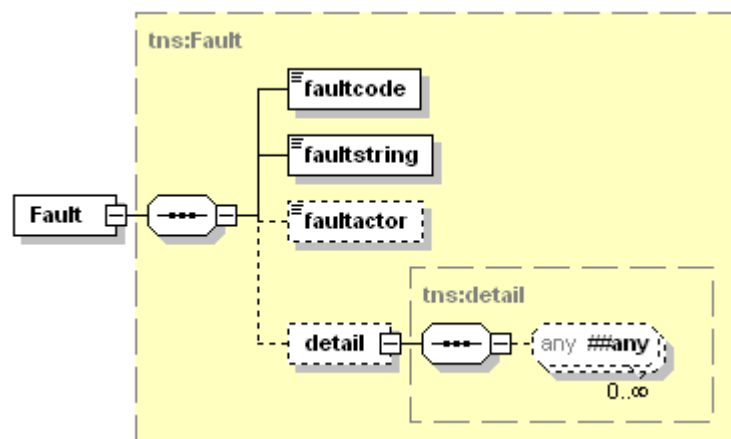


Abbildung 1

55 Im Folgenden eine Beschreibung der einzelnen Elemente:

- 56 • **<faultcode>** (mandatory)
- 57 SOAP-Fault Codes helfen bei einer Klassifizierung der SOAP Fehlermeldungen.
- 58 Mögliche Fault Codes sind:

- 59 ○ VersionMismatch: Der SOAP-Envelope verwendet einen ungültigen
60 Namensraum für das SOAP-Element *Envelope*. Mit diesem Fehler wird in
61 SOAP auf Protokollversionierungsprobleme hingewiesen.
- 62 ○ MustUnderstand: Ein Headerblock, der das Flag *mustUnderstand=true*
63 enthält, wurde vom Empfänger der Nachricht nicht verstanden.
- 64 ○ Client: Der Empfänger hat ein Problem mit der SOAP-Nachricht und kann
65 diese nicht abarbeiten. Üblicherweise liegt bei dieser Art von Fehler das
66 Problem beim Client.
- 67 ○ Server: Es ist ein Fehler aufgetreten, der nicht unmittelbar mit der
68 Verarbeitung der Nachricht in Verbindung gebracht werden kann.

69 Detaillierter Beschreibungen sind der Spezifikation zu entnehmen [SOAP11].

70 • **<faultstring>** (mandatory)

71 In diesem Element befindet sich eine lesbare Erläuterung des Fehlers.

72 • **<faultactor>** (optional)

73 Dieses Element zeigt, mittels eines *xs:anyURI* Elements, auf jenen Knoten,
74 welcher den Fehler verursachte. Es ist durchaus möglich, dass ein Webservice
75 die Verarbeitung der Nachricht an ein anderes Webservice auslagert bzw.
76 abgibt; somit wird eine Kaskadierung von Webservices erstellt. Tritt nun an
77 einer Stelle dieser Kaskade ein Fehler auf, insofern ein Webservice nicht
78 antwortet, so wird jene Adresse des Knotens in diesem Element angegeben.

79 • **<detail>** (optional)

80 Applikationsspezifische Fehlerinformationen, deren Syntax keinem Schema
81 obliegt. Dieses Element muss inkludiert werden, falls der Fehler von einem
82 Element innerhalb des *Body*-Elements verursacht wurde. Er darf nicht
83 inkludiert werden, wenn der Fehler bei der Abarbeitung des Headers auftrat.

84

85 Ein HTTP-Response mit einer SOAP-Fehlermeldung laut SOAP 1.1 hat folgendes
86 Aussehen:

```

HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <env:Fault>
      <faultcode>env:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <faultactor>http://www.foo.com</faultactor>
      <detail>
        <e:myfaultdetails xmlns:e="Some-URI">
          <message>My application didn't work</message>
          <errorCode>1001</errorCode>
        </e:myfaultdetails>
      </detail>
    </env:Fault>
  </env:Body>
</env:Envelope>

```

87

88

89

90

91

92 (2.2) SOAP-Faults in SOAP 1.2

93 Abbildung 2 beschreibt das XML-Element SOAP-Faults laut SOAP-Spezifikation 1.2.:

94

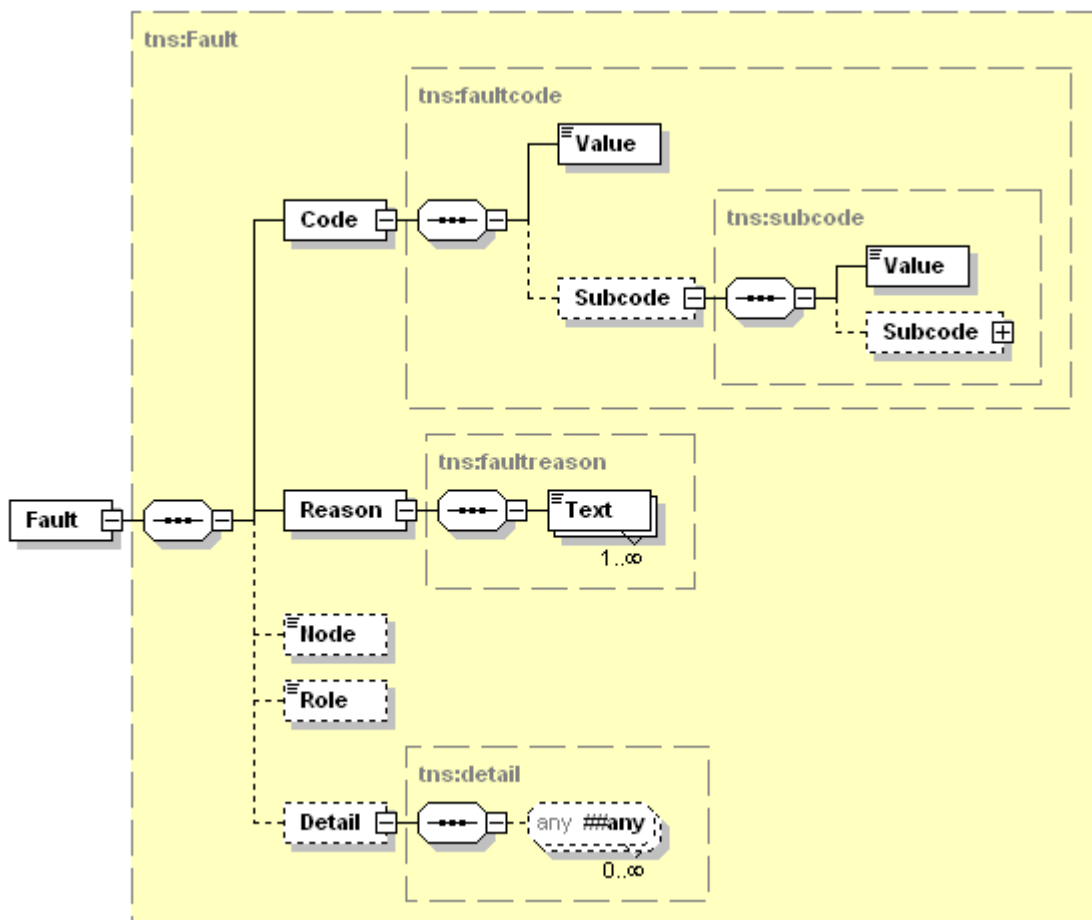


Abbildung 2

95

96

97

98 Im Folgenden eine Beschreibung der einzelnen Elemente:

99

- **<Code>** (mandatory)

100

SOAP-Fault Codes helfen bei einer hierarchischen Klassifizierung der SOAP-Fehlermeldungen. Die Codierung kann beliebig granular mittels Subcodes definiert werden. Mögliche Fault Codes sind:

101

102

103

- VersionMismatch: siehe SOAP 1.1

104

- MustUnderstand: siehe SOAP 1.1

105

- DataEncodingUnknown: Dieses Element zeigt einen Fehler des Empfängers auf, der durch einen nicht erkennbaren Wert des *encodingStyle* Attributs in der SOAP-Message verursacht wurde.

106

107

- Sender: siehe SOAP 1.1

108

- Receiver: siehe SOAP 1.1

109

110

Eine detailliertere Beschreibung ist wiederum der Spezifikation zu entnehmen [SOAP12].

111

- 112 • **<Reason>** (mandatory)
 113 In diesem Element befindet sich eine oder mehrere lesbare Erklärungen des
 114 Fehlers.
- 115 • **<Node>** (optional)
 116 Siehe SOAP 1.1 **<faultactor>**.
- 117 • **<Role>** (optional)
 118 Hier wird die Rolle des in **<Node>** bezeichneten Knotens beschrieben, in dem
 119 der Knoten zum Zeitpunkt des Fehlers agierte.
- 120 • **<Detail>** (optional)
 121 Applikationsspezifische Fehler Informationen, deren Syntax keinem Schema
 122 obliegt.
- 123 Ein HTTP-Response mit einer SOAP-Fehlermeldung laut SOAP 1.2 hat folgendes
 124 Aussehen:

```

HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:m="http://www.example.org/timeouts"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>m:MessageTimeout</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">Sender Timeout</env:Text>
      </env:Reason>
      <env:Detail>
        <m:MaxTime>P5M</m:MaxTime>
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>

```

125

126 (3) Fehlerunterscheidung

127 Sowohl laut der SOAP Spezifikation 1.1, wie auch laut Version 1.2 wird ein SOAP-Fehler
 128 über HTTP mit dem HTTP-Statuscode 500 (InternalServerError) [HTTP] zurückgesandt.
 129 Dies geschieht auch falls der Fehler auf der Clientseite liegt.

130 Da nicht immer ein interner Server Error den Fehler verursacht hat, ist es für eine
 131 detaillierte Beschreibung des Umgangs mit SOAP-Faults wichtig zu verstehen, in welchen
 132 Ebenen der Webservice-Architektur Fehler auftreten können und somit woher diese
 133 Fehler kommen. Die einzelnen Fehlercodierungen in SOAP 1.1 und SOAP 1.2 weisen
 134 schon auf die Fehlerquellen hin.

135 (3.1) Fehler in der Applikation

136 Während der Verarbeitung der Daten, die über das Webservice an die Fachapplikation
 137 gesandt wurden, kann ein Fehler auftreten. Üblicherweise tritt ein Fehler auf, wenn der
 138 Server den eingegangenen Request nicht weiter bearbeiten kann.

139 (3.2) Fehler auf der Clientseite

140 Falls auf der Clientseite Fehler auftreten, so ergeben sich dadurch normalerweise keine
141 SOAP-Faults. Es kann aber durchaus vorkommen, dass der Client eine Anfrage an ein
142 Webservice stellt und diese nicht der vorgegebenen Syntax entspricht. Die Ursache des
143 daraus entstehenden SOAP-Faults ist somit auf dem Client zu suchen.

144 (3.3) Fehler in der Transportschicht

145 Eine weitere Möglichkeit, wo sich Fehler in der Kommunikation ereignen können, besteht
146 in der Transportschicht zwischen Client und Webservice bzw. bei kaskadierenden
147 Webservices zwischen diesen. Hier kann es zu physischen Problemen bei der
148 Übermittlung der Daten kommen oder bei der Erreichbarkeit von anderen Quellen

149 (4) SOAP-Faults vs. XML-Fehlermeldung

150 Folgendes Kapitel behandelt die Unterschiede zwischen einer XML-Fehlermeldung und
151 einem SOAP-Fault.

152 (4.1) SOAP-Fault

153 Beim Bearbeiten des Webservices trifft der Server auf eine für ihn unlösbare Situation. Er
154 erzeugt einen SOAP-Fault, der mit der HTTP Statusmeldung 500 – Internal Server Error
155 an die aufrufende Applikation retour gesandt wird. Wie schon eingangs erwähnt führt ein
156 SOAP-Fault in den allgemein bekannten Webservice Frameworks eine Exception. Diese
157 muss programmatisch abgefangen werden.

158 Um zu erfahren, wo der jeweilige Fehler gelegen hat, muss die Exception-Message, die
159 übermittelt wird, vom Client geparkt werden. Danach kann der Webservice-Client
160 dementsprechende Schritte setzen, um eine etwaige Fehlermeldung an den Benutzer
161 weiterzugeben. Gegebenenfalls kann das Client-Programm auch eine andere
162 Fehlerbehandlung erwägen.

163 Somit liegt bei einem SOAP-Fault die Intelligenz vor allem beim Client, der auf die
164 Fehlermeldungen selektiv reagieren muss.

165 (4.2) XML-Fehlermeldung bzw. XML-Statusmeldung

166 Der Ablauf der Fehlererzeugung bzw. die Behandlung des Fehlers am Client ist bei einer
167 XML-Fehlermeldung anders, als bei einem SOAP-Fault.

168 Der Fehler wird ebenfalls am Server erkannt. Die Exception, auf die der Server bei der
169 Abarbeitung stösst, wird jedoch schon am Server abgefangen.

170 Der Client erhält eine standardisierte Nachricht, die auf einen Fehler hindeutet und die
171 genaue Fehlerbeschreibung und andere beschreibende Daten enthält. Es wird am Client
172 keine Exception geworfen. Nach dem Parsen der übermittelten Daten, wird
173 programmatisch darauf reagiert.

174 Die Intelligenz bei dieser Fehlerbehandlung liegt vordergründig am Server, jedoch muss
175 auch der Client selektiv auf die ihm gesandten Mitteilungen reagieren.

176 Es können durch diesen Prozeß nicht nur standardisierte Fehlermeldungen an den Client
177 geliefert werden, sondern auch reine Statusmitteilungen, die Informationen über die
178 Transaktionen auf Applikationsseite beinhalten.

179 Die vorliegende Empfehlung stellt folgendes Schema zur Verfügung, das über ein
180 `<xs:include>` in ein anwendungsspezifisches Schema importiert werden kann:

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Message" type="CustomFaultType"/>
  <xs:complexType name="CustomFaultType" final="extension">
    <xs:annotation>
      <xs:documentation>CustomFault reporting structure</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Code" type="xs:integer"/>
      <xs:element name="Reason" type="xs:string" maxOccurs="unbounded"/>
      <xs:element name="Help" type="xs:string" minOccurs="0"/>
      <xs:element name="Detail" type="xs:anyType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

181 Applikationsspezifische Schemata können sowohl ein eigenes Element für ihre XML-
 182 Fehlermeldungen definieren und ihm den global deklarierten komplexen Typ
 183 CustomFaultType zuweisen oder direkt das global deklarierte Message-Element
 184 referenzieren.

185 (5) Vorgehensweise bei Fehlern

186 Wir empfehlen Fehlermeldungen in Webservice generell als SOAP-Faults zu erzeugen. In
 187 Ausnahmefällen, insbesondere, wenn trotz eines Fehlers noch sinnvolle Ergebnisse
 188 zurückgeliefert werden können, kann als Alternative die Antwort mit einer standardisierten
 189 XML-Fehlermeldung ausgeliefert werden. Als genaues Unterscheidungsmerkmal sind zwei
 190 Sichtweisen anzuführen:

191 (5.1) Fehler aus Sicht der Transportschicht (System Exception)

192 Eine System Exception tritt immer dann auf, wenn aufgrund eines Fehlers in der
 193 Transportschicht oder eines Fehlers in der Serverinfrastruktur, der Request nicht
 194 bearbeitet werden kann.

195 Beispiele:

- 196 • Ausfall von sekundären Systemen (Datenbank, ...)
- 197 • Fehlerhaftes Routing bei kaskadierenden Webservices
- 198 • HTTP Timeout

199 Durch Behandlung dieser Fehler mit einem SOAP-Fault, kann der Client mit einer, von den
 200 Frameworks erstellten Exception, auf die Problematik hingewiesen werden, dass
 201 momentan keinerlei Ergebnisauslieferung stattfinden kann.

202 Das Auftreten einer System Exception sollte grundsätzlich einen SOAP-Fault erzeugen.

203 (5.2) Fehler aus Sicht der angesprochenen Systeme (Application 204 Exception)

205 Eine Application-Exception bezeichnet einen Fehler, bei dem aus Sicht der Anwendung ein
 206 Request nicht korrekt bearbeitet werden kann, weil entweder der Request nicht korrekt ist
 207 oder die Anwendung einen Request nicht korrekt bearbeiten kann.

208 Beispiele:

- 209 • Datenbankabfrage mit leerer Ergebnismenge
- 210 • Anfrage mit ungültigem Wertebereich

211 Dieser Art von Fehler empfehlen wir ebenfalls mittels SOAP-Faults zu übertragen.

212 Lediglich im Falle, in dem trotz des Auftretens eines Fehlers noch eine sinnvolle
 213 Ergebnismenge zurück geliefert werden kann, kann eine standardisierte XML-
 214 Fehlermeldung erzeugt werden.

215 Dem Client kann mitgeteilt werden, dass es bei der Abarbeitung der Anfrage zu
 216 Problemen gekommen ist, diese jedoch nicht kritisch sind bzw. zu einem möglicherweise
 217 ungewollten Ergebniss führen. Ein typisches Beispiel wäre die fehlende Unterstützung für
 218 ein optionales Feature wie z.B. Sortierung der Ergebnismenge einer Registerabfrage.
 219 Ein Client, der die Sortierung angefordert hat, kann durch eine XML-Meldung darauf
 220 hingewiesen werden, dass dieses Feature nicht unterstützt wird, wobei ihm trotzdem die
 221 Ergebnisse (allerdings natürlich unsortiert) ausgeliefert werden können.
 222 Es ist danach dem Client überlassen, seine Anfrage mit anderen Parametern neu
 223 abzusenden, oder die Daten so zu übernehmen.

224 **(6) Fehler- und Statuscodes**

225 Zu einer standardisierten Kommunikationsarchitektur zählen auch vorgegebene
 226 Statusmeldungen und Fehlercodes. Eine Einteilung der Fehler und der Status in Gruppen
 227 und die durchgehende logische Nummerierung derselben, hilft nicht nur den Entwicklern
 228 der Applikationen, sondern bietet auch bei der Erstellung der Clientapplikationen
 229 Unterstützung.

230 Weiters soll eine Standardisierung der Fehlercodes, durch die Verwendung eines eigens
 231 dafür zu definierenden Namespaces, vorangetrieben werden.

232 **(6.1) Einteilung der Fehler- bzw. Statuscodes**

233 Die auftretenden Status bzw. Fehler können in Anlehnung an die HTTP Statuscodierung in
 234 folgende Gruppen unterteilt werden:

- 235 • Informationen über den momentanen Status
 236 Diese Gruppe beinhaltet keine Fehlermeldungen, sondern ordentliche Mitteilungen
 237 über den momentanen Status einer Transaktion.
- 238 • Erfolgsmeldungen
 239 In dieser Gruppen werden alle Erfolgsmeldungen, die im Laufe einer Kommunikation
 240 übermittelt werden sollen, gesammelt.
- 241 • Zusätzliche Informationen nötig
 242 Die Meldungen in dieser Klasse zeigen auf, dass zusätzliche Informationen nötig sind,
 243 um die Transaktion zu beenden. Dies bedeutet jedoch nicht, dass die Anfrage
 244 fehlgeschlagen ist, sondern nur, dass die Applikation nun mit der vorhandenen
 245 Information versucht die Transaktion zu beenden.
- 246 • Fehler des Clients
 247 In dieser Klasse werden Meldungen gesammelt, die anzeigen, dass dem Client ein
 248 Fehler unterlaufen ist.
- 249 • Fehler des Servers
 250 Fehlermeldungen in dieser Klasse lassen auf interne Fehler des Servers schliessen. Der
 251 Server zeigt auf, dass er nicht in der Lage ist, die Transaktion zu beenden, oder die
 252 Anfrage zu verarbeiten.
- 253 • Andere Fehler
 254 Diese Klasse bietet Fehlern oder Status, die logisch nicht in den vorangegangenen
 255 Klassen Einzug finden, Platz.

256 Durch die obige Standarderteilung ergibt sich eine 5-stufige Fehlereinteilung. Die HTTP
 257 Codierung nutzt eine dreistellige zusätzliche Interpretation dieser 5 Stufen (zB.: 4001 für

258 Unauthorisierten Zugriff). Um den vielfältigen Applikation im Bund eine Möglichkeit zu
 259 bieten, alle Fehler und Status in dieses Schema überzuführen, empfehlen wir eine
 260 vierstellige Interpretation.

261 Daraus ergibt sich folgende Zahlencodierung der Fehler- und Statusgruppen:

262 • XML-Statusmeldungen

263 • Statusmeldungen – **1xxx**

264 1000 – Continue

265 1001 – Idle

266 • Erfolgsmeldungen – **2xxx**

267 2000 – OK

268 2001 – Created

269 2002 – Accepted

270 • XML-Fehlermeldungen

271 • Zusatzinformation – **3xxx**

272 3000 – Multiple Choice

273 3001 – Moved Permanently

274 • Clientfehler – **4xxx**

275 4000 – BadRequest

276 4001 – Unauthorized

277 4003 – Forbidden

278 • SOAP-Faults

279 • Serverfehler – **5xxx**

280 5000 – Internal Server Error

281 5001 – Not implemented

282 • Sonstiger Fehler – **6xxx**

283 Die oben vorgeschlagenen Fehlercodes sind numerisch und sollen in einem
 284 <faultcode>-Element eines SOAP-Faults zur Anwendung kommen. Laut SOAP-Schema
 285 [SOAP12] sind dort aber nur Qualified Names zulässig, d.h. Zeichenketten, die nicht mit
 286 einer Ziffer beginnen dürfen. Aus diesem Grund müssen die Fehlercodes in SOAP-Faults
 287 mit einem führenden „F“ versehen werden (zb.: „F4010“, „F5018“, ...).

288 (6.2) Fehler- bzw. Statuscodes in SOAP-Faults

289 Gemäß WS-I-Empfehlung (s. [WSI], 3.3.6) sollten die Fehlercodes dem Namensraum des
 290 Applikationsschemas zugewiesen werden, indem er das für diesen definierte Präfix erhält.
 291 Ein Beispiel zeigt die korrekte Verwendung der Fehlercodes im <faultcode>-Element
 292 eines SOAP-Faults:

```
<soap:Fault xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:
  sw=" http://reference.e-government.gv.at/namespace/xml-sw/1#">
  <faultcode>sw: F4010</faultcode>
  <faultstring> Required search criteria missing</faultstring>
</soap>
```

293

294

295 (7) Referenzen

296 [HTML]

297 Raggett, Dave; Le Hors, Arnaud; Jacobs, Ian: HTML 4.01 Specification W3C
298 Recommendation 24 December 1999, Abgerufen aus dem World Wide Web am
299 29.06.2004 unter <http://www.w3.org/TR/html4/>

300 [HTTP]

301 Fielding, R. and et al.: Hypertext Transfer Protocol, Juni 1999, Abgerufen aus dem
302 World Wide Web am 21.12.2004 unter
303 <http://www.ietf.org/rfc/rfc2616.txt?number=2616>

304 [SOAP11]

305 Box, D. and et al: Simple Object Access Protocol (SOAP) 1.1 , Abgerufen aus dem
306 World Wide Web am 21. Dez

307 ember 2004 unter <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

308 [SOAP12]

309 Gudgin, M. and et al: SOAP Version 1.2 Part 1: Messaging Framework, Abgerufen
310 aus dem World Wide Web am 20. Dezember 2004 unter
311 <http://www.w3.org/TR/soap12-part1/>

312 [WSI]

313 WebServices Interoperability Organization: WS-I Basic Profile Version 1.1.,
314 Abgerufen aus dem World Wide Web am 25. Jänner 2005 unter [http://www.ws-](http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html#SOAPFAULT)
315 [i.org/Profiles/BasicProfile-1.1-2004-08-24.html#SOAPFAULT](http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html#SOAPFAULT)

316 [VALIDATE]

317 Liehmann, M: Erklärung des Validate Ansatzes im Rahmen der
318 Kommunikationsarchitektur, Abgerufen aus dem World Wide Web am 01. Jänner
319 2005 unter <http://reference.e-government.gv.at/Zwischenergebnisse.571.0.html>

320 [XML-SW]

321 Herpers, F.J.: XML-Search:XML-basiertes Protokoll für Suchanfragen via
322 Webservices, vorgelegt in der ARGE Kommarch am 9.11.2004

323

(8) Literaturvorschläge

325 Using SOAP Faults:

326 [http://msdn.microsoft.com/library/default.asp?url=/library/en-](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service09172002.asp)
327 [us/dnservice/html/service09172002.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service09172002.asp)

328 Einführung in SOAP:

329 <http://www.oreilly.de/catalog/progwebsoapger/chapter/>

330 Using SOAP Faults in WebLogic:

331 http://dev2dev.bea.com/products/wlworkshop81/articles/soap_wlw_ws.jsp

332 Using SOAP with J2EE: (SOAP 1.1)

333 <http://www.informit.com/articles/article.asp?p=169106&seqNum=6>

334 Using the SOAP Protocol with Java: (SOAP 1.2)

335 <http://www.quepublishing.com/articles/article.asp?p=328640&seqNum=11>

336 What's new in SOAP 1.2:

337 <http://www.hadley.net/org/marc/whatsnew.html>

338 **History**

Version 1.0.0	Datum 29.12.2004	Kommentar Initialversion erstellt.
Ersteller Michael Liehmann		
Version 1.0.1	Datum 26.01.2005	Kommentar Änderungen eingearbeitet nach Vorschlägen von Hrn. Herpers. Referenzen ergänzt.
Ersteller Michael Liehmann		
Version 1.0.2	Datum 11.03.2005	Kommentar Änderungen eingearbeitet nach Vorschlägen aus der AG Kommarch. Fehlereinteilung erweitert.
Ersteller Michael Liehmann		
Version 1.0.3	Datum 21.03.2005	Kommentar XML-Fehlermeldungen erweitert.
Ersteller Michael Liehmann		
Version 1.0.4	Datum 22.03.2005	Kommentar Anmerkungen von Hrn. Herpers eingearbeitet.
Ersteller Michael Liehmann		
Version 1.0.5	Datum 4.04.2005	Kommentar Änderungswünsche eingearbeitet, die nach der letzten Aussendung eingegangen sind. SOAP-Faults Fehler sind mit vorangestelltem „F“ anzuführen.
Ersteller Michael Liehmann		
Version 1.0.6	Datum 12.04.2005	Kommentar Titel geändert.
Ersteller Michael Liehmann		

339

340