PROC SOAP, PROC HTTP und der ganze REST Webservices und SAS

Martin Haffner
HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Martin.Haffner@analytical-software.de

Andreas Mangold
HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Andreas.Mangold@analytical-software.de

Bernhard Braun
HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Bernhard.Braun@analytical-software.de

Zusammenfassung

Keine Softwareanwendung ist so universell, dass sie für alle Aufgaben einsetzbar ist. Das Zusammenspiel vieler Softwaresysteme ist in der Wirtschaft vorherrschend und auch in der Forschung und Entwicklung häufig nötig. Webservices bilden eine universelle Schnittstelle zwischen den Systemen, sozusagen eine Lingua franca, eine "gemeinsame Sprache" für den Aufruf von Funktionen zwischen verschiedenen Softwareanwendungen.
Für SAS-Anwender haben Webservices zwei Anwendungsgebiete: das Konsumieren und das Bereitstellen von Webservices. Zum einen können Webservices, die von anderen Anwendungen oder öffentlich im Internet bereitgestellt wurden, durch ein SAS-Programm konsumiert (aufgerufen) werden. Zum anderen kann man jedes SAS-Programm als Webservice bereitstellen und damit anderen Anwendungen den Aufruf ermöglichen. Dieser Beitrag konzentriert sich auf die Funktionen der SAS-Software, die das Konsumieren von Webservices ermöglichen, und zwar anhand eines Beispiels aus der Geographie.

Schlüsselwörter: Webservices, SOAP, REST, PROC SOAP, PROC HTTP, Stored Processes

1 Einführung zu Webservices

1.1 Definition Webservice

Einführend einige Anmerkungen zu Webservices allgemein – beginnend mit der Fragestellung: Was ist das überhaupt, ein Webservice? Die Frage ist nicht so trivial, wie sie klingt, denn selbst das World Wide Web Consortium (W3C) bemerkt immerhin: "*Es gibt auf der Welt eine ganze Menge Dinge, die man als Webservices bezeichnen könnte*"[1].

Bei Wikipedia findet sich eine etwas formelle Definition: Ein Webservice ist "*eine Software-Anwendung, die mit einem Uniform Resource Identifier (URI) eindeutig identifizierbar ist und deren Schnittstelle als XML-Artefakt definiert, beschrieben und gefunden werden kann*"[2].
Die wichtigsten Punkte dieser Definition sind: Erstens handelt es sich um eine Software. Zweitens lässt sich der Webservice durch einen URI bzw. eine URL eindeutig abrufen und drittens lässt sich ihre Schnittstelle mit XML beschreiben. Ganz korrekt ist das übrigens nicht: Es gibt auch Webservices, die ganz ohne XML auskommen; dazu später mehr.
Das W3C ergänzt noch einen weiteren wichtigen Punkt: Ein Webservice ist "*a software system designed to support interoperable machine-to-machine interaction over a network*"[1]. Es geht also um die Kommunikation zwischen Computern und nicht um die Kommunikation zwischen Computer und Endanwender, wie beispielsweise bei einer Webseite.
Möglicherweise klingen diese Definitionen zu abstrakt. Wenn man es etwas pragmatischer möchte, könnte man sich auch auf die folgende Definition einlassen: "*Da stellen wir uns mal ganz dumm und sagen: Ein Webservice, das ist ein großer, runder, schwarzer Raum – also eine Blackbox – , die hat zwei Löcher – also Schnittstellen. In das eine geben wir Anfragen rein, aus dem anderen – das kommt später – kommen die Ergebnisse wieder raus. Es steht alles bei Wikipedia, was ich sage, nur nicht so schön!*"[1]
Auch wenn diese Definition die Sachlage sehr vereinfacht, beschreibt sie doch die wesentlichen Komponenten eines Webservice: Es sind ein Client ("Konsument") und ein Server ("Anbieter") beteiligt. Der Client schickt eine Anfrage mit Parametern an den Server; der Server bearbeitet diese Anfrage und schickt eine Antwort zurück an den Client, die der Client dann wiederum weiter verarbeiten kann.

[1] Sehr frei zitiert nach Prof. Bömmels Ausführungen zur Dampfmaschine in dem Film "Die Feuerzangenbowle".

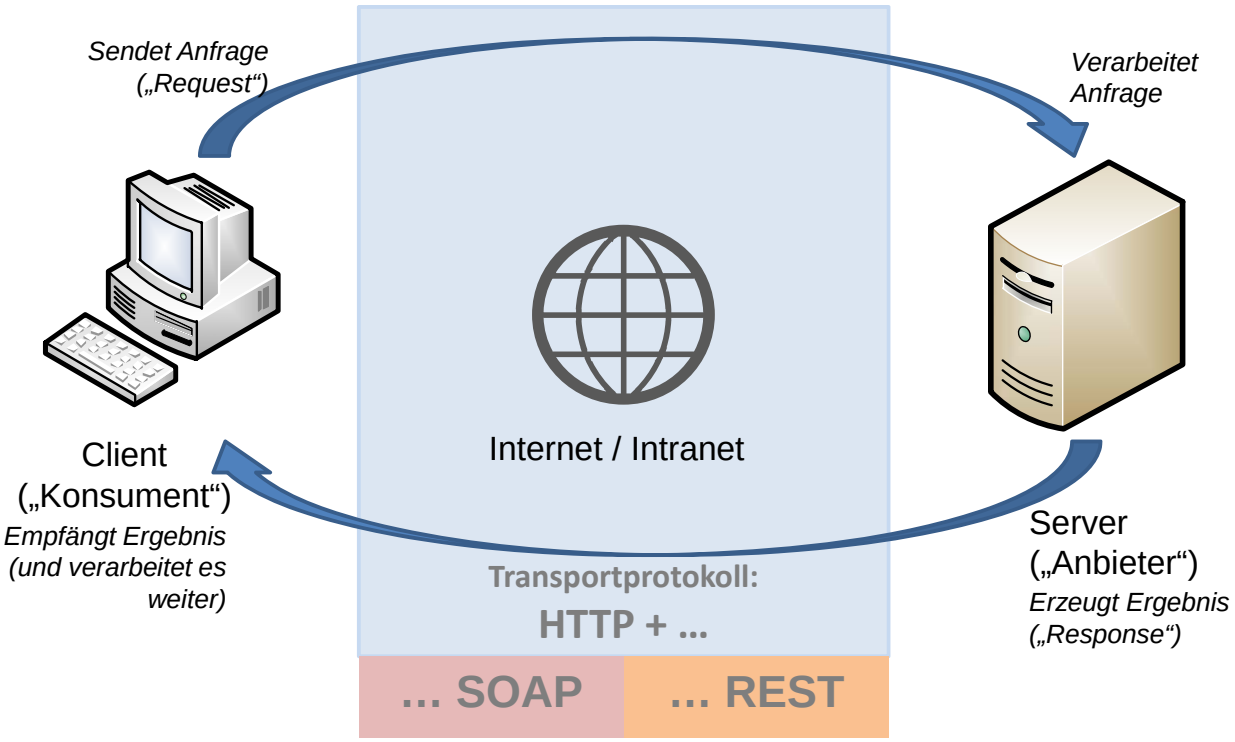


Abbildung 1: Webservice-Konsument und -Anbieter

Zwischen Client und Server liegt das Internet, wenn der Webservice öffentlich zugänglich ist; oder das Intranet einer Firma, wenn es sich um einen nur intern zugänglichen Webservice handelt. Das Transportprotokoll, das für die Übermittlung von Request und Response verwendet wird, ist HTTP, das Protokoll des World Wide Web – daher also auch die Bezeichnung *Webservice*. Oberhalb des HTTP-Protokolls wird allerdings noch eine weitere, Webservice-spezifische Technologie eingesetzt: Entweder SOAP (*Simple Object Access Protocol*) oder REST (*Representational State Transfer*)[2].

1.2 Webservice-Requests mit SOAP oder REST

Ein Webservice-Request soll hier an einem Beispiel von geonames.org veranschaulicht werden. Geonames.org bietet eine ganze Reihe von Webservices rund um Fragestellungen aus der Geographie an. Das hier gezeigte Beispiel nimmt als Anfrage-Parameter geographische Koordinaten (Länge und Breite; hier: 42°N und 42°E) entgegen, und gibt als Ergebnis Name und Land des Ortes zurück, der möglichst nahe an diesen Koordinaten liegt. In der Realität wird dieser Webservice nur auf Basis von REST angeboten; das hier gezeigte Beispiel für SOAP ist eine fiktive Übertragung in einen analogen SOAP-Webservice.

Der SOAP-Request beginnt, wie jeder HTTP-Aufruf, mit dem Namen der HTTP-Methode und der anschließenden URL, an die der Aufruf gesendet werden soll. Für SOAP-

[2] REST ist, wenn man es ganz genau nimmt, keine Technologie, sondern ein Architekturstil.

Requests wird immer die HTTP-Methode POST verwendet, da der eigentliche Inhalt der Nachricht als XML an den Aufruf angehängt werden soll – dies kann nur bei der POST-Methode so durchgeführt werden, andere HTTP-Methoden übergeben ihre Parameter innerhalb der URL.
Die eigentliche SOAP-Nachricht enthält zunächst einmal den so genannten SOAP-Envelope: Einen Umschlag, der die eigentliche Nachricht kapselt, und eventuell noch Metadaten enthält. Schließlich kommt die eigentliche Nachricht, der *Request Body*: Ein XML-Fragment, das die zu übergebenden Parameter – Länge, Breite und Benutzernamen – enthält.

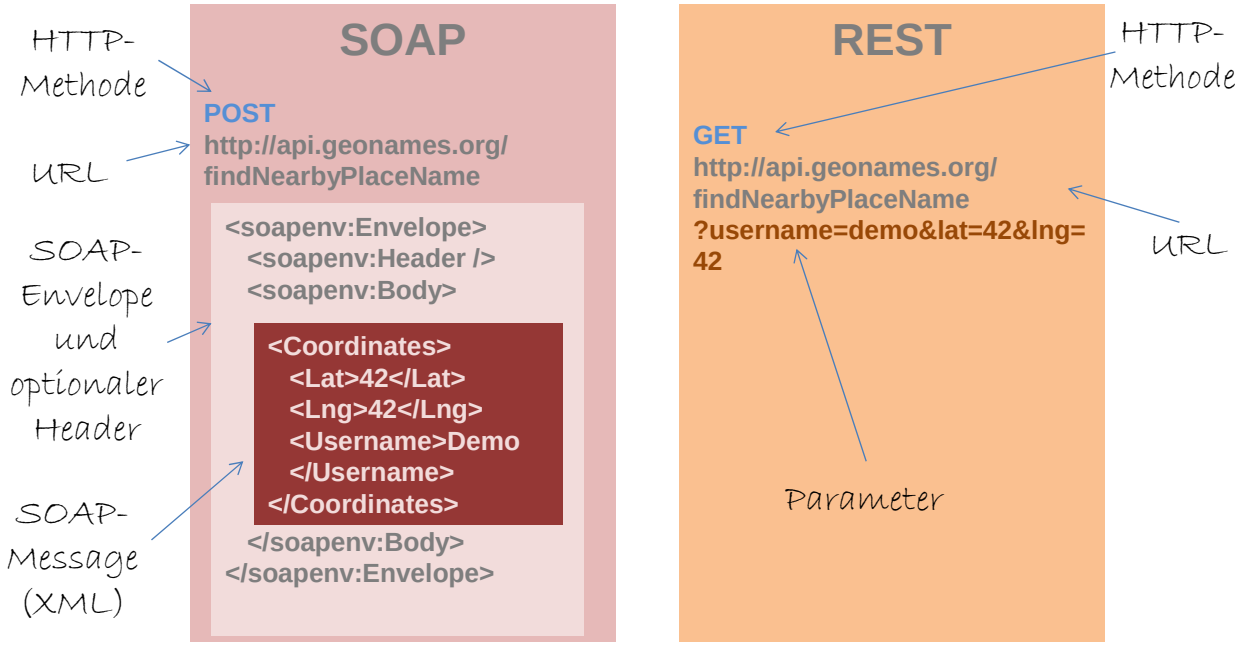


Abbildung 2: Webservice-Anfrage mit SOAP oder REST

Bei REST sieht der Anfang ähnlich aus: Es werden ebenfalls eine HTTP-Methode und die URL des aufzurufenden Webservice angegeben. Hier sind aber alle HTTP-Methoden möglich; im Beispiel wurde die GET-Methode verwendet. Die Parameter sind direkt an die URL angehängt. Man sieht, dass hier die gleiche Information deutlich kompakter übertragen wird.

1.3 Webservice-Responses mit REST oder SOAP

Die Response enthält, wie bei HTTP üblich, immer zunächst den Content Type. Im Falle einer SOAP-Nachricht ist dies automatisch XML. Weiterhin ist die SOAP-Response, genau wie der Request, von einem SOAP-Envelope umschlossen, der das eigentliche Ergebnis kapselt. Dieses Ergebnis liegt wiederum als ein XML-Fragment vor, und verrät uns, dass es sich bei dem gesuchten Ort um *Silauri* in Georgien handelt.
Bei der REST-Response sind zwei unterschiedliche Content Types möglich: XML oder JSON. Bei JSON handelt es sich um das Objektformat der Programmiersprache Ja

vaScript, das im Umfeld von Webanwendungen häufig zum Datenaustausch eingesetzt wird. Die hier abgebildete Response ist im JSON-Format; wie man sieht, ist sie geringfügig kompakter als ihr XML-Äquivalent. Eine XML-Response von einem REST-Webservice entspräche im Wesentlichen dem XML-Body des SOAP-Webservice.

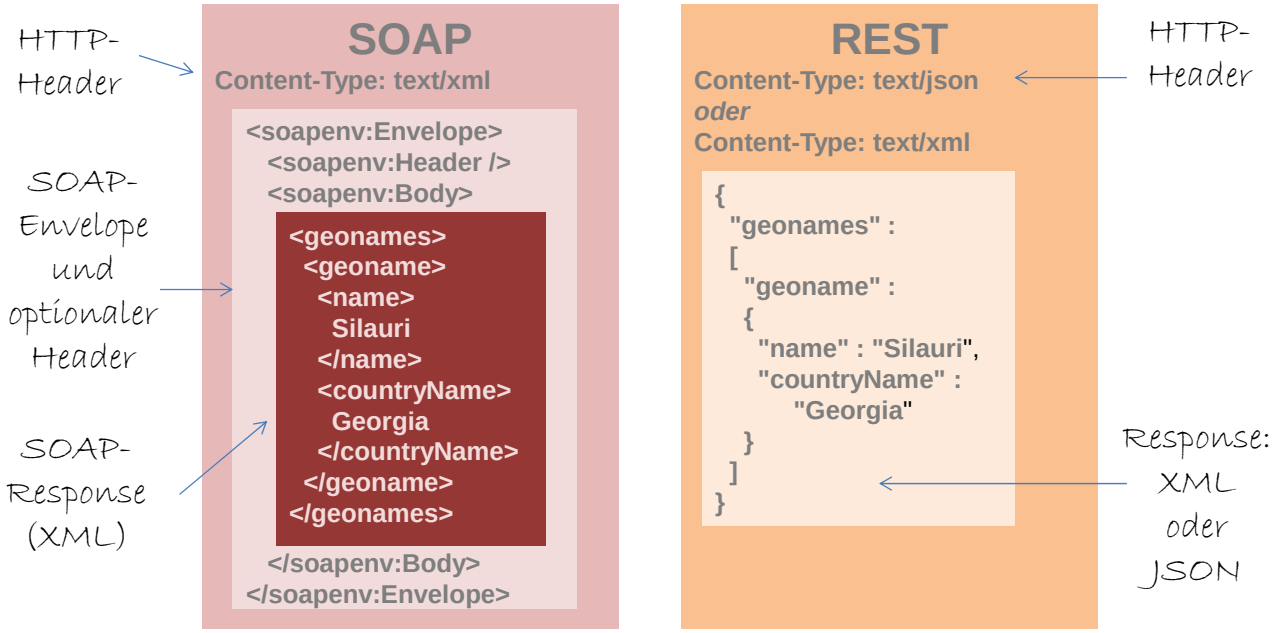


Abbildung 3: Webservice-Response mit SOAP oder REST

1.4 Welche Struktur haben die XML/JSON-Inhalte von Request und Response?

Der Webservice-Aufruf besteht also, wie gesehen, aus dem Senden und Empfangen von XML- oder JSON-Daten. Woher kennt man nun die Struktur dieser Daten – d.h. woher weiß man, welche XML-Tags der Webservice erwartet, und welche XML-Tags er auf der anderen Seite innerhalb seiner Antwort zurücksendet?

Für SOAP-Webservices ist diese Frage recht einfach zu beantworten: Jeder Service beschreibt seine Schnittstelle exakt in Form einer WSDL-Datei. Diese WSDL-Datei wird mit dem Webservice zusammen veröffentlicht, und lässt sich, genauso wie der Webservice selbst, aus dem Internet oder Intranet abrufen (in der Regel hängt man an die URL den Parameter "?wsdl" an).

Bei WSDL handelt es sich um ein standardisiertes XML-Format, das alle relevanten Aspekte der Schnittstelle beschreibt (Welche Parameter gibt es? Welche Datentypen haben sie? Usw.) In der Regel muss man nicht selbst versuchen, den Inhalt einer WSDL-Datei zu interpretieren: Es gibt Tools, die einem diese Arbeit abnehmen, wie z.B. soapUI (http://www.soapui.org). Diese Tools lesen den Inhalt der WSDL-Datei und generieren daraus beispielhafte Requests und Responses.

Bei REST gibt es eine solche standardisierte Schnittstelle bisher noch nicht, auch wenn mit dem WADL-Format (ebenfalls XML) Ansätze dafür vorhanden sind, die sich aber noch nicht flächendeckend durchgesetzt haben. In der Regel wird man sich daher also bei Verwendung von REST-Webservices auf die Dokumentation verlassen müssen, die der Anbieter dieses Webservice bereitgestellt hat.

1.5 Zusammenfassung: Webservices, die "Lingua Franca" der IT

Der entscheidende Vorteil von Webservices ist ihre universelle Einsetzbarkeit: Beim Aufruf eines Webservice werden letztendlich nur Textnachrichten über das HTTP-Protokoll versendet. Jedes System, das HTTP verarbeiten kann, ist somit im Stande, mit Hilfe von Webservices mit anderen Systemen zu kommunizieren.
Dabei ist es nicht notwendig, Details über den jeweiligen Kommunikationspartner zu kennen: Weder Betriebssystem noch Middleware noch Datenbank noch Implementierungsdetails müssen bekannt sein, solange nur die Beschreibung der Schnittstelle vorliegt. Auch wenn sich die Implementierung des Webservices einmal ändert oder der Anbieter eines Webservice auf eine komplett andere Umgebung umstellt, muss der Client keine Anpassungen vornehmen, solange nur die Schnittstelle gleich bleibt.

2 Webservices und SAS

2.1 Motivation

Für die Einbindung von Webservices in SAS gibt es zwei Einsatzszenarien:

a) Man will innerhalb eines SAS/Base-Programms die Funktionalität nutzen, die ein Webservice anbietet, der auf einem anderen System läuft.
b) Man hat ein eigenes SAS-Programm geschrieben, das man zum Aufruf über einen Webservice zur Verfügung stellen möchte.

Für den ersten Fall (Aufruf eines Webservice) stellt SAS/Base mehrere Möglichkeiten zur Verfügung: PROC SOAP (ab SAS 9.2), Data Step-Funktionen (ab SAS 9.3) sowie WSDL-Libnames (nur SAS 9.2) für SOAP-Webservices und PROC HTTP (ab SAS 9.2) für REST-Webservices. Das Bereitstellen eigener Programme (die in Stored Processes eingebunden sind) als Webservices wird ebenfalls unterstützt.

2.2 PROC SOAP

Bei der Verwendung von PROC SOAP hat man in der Regel seine Ausgangsdaten im SAS-Format vorliegen. Mit diesen Daten will man einen Webservice aufrufen, der anhand der übermittelten Daten bestimmte Aktionen ausführen und schließlich ein Ergebnis zurückliefern soll. Die vom Webservice zurückgelieferten Daten benötigt man letztendlich wieder im SAS-Format.
Da bei SOAP XML-Nachrichten versendet und empfangen werden, muss der erste Schritt sein, die an den Webservice zu übermittelnden Daten von einer SAS-Tabelle in eine XML-Datei zu transformieren. Dabei muss nur der Request-Body geschrieben werden; um den Header braucht man sich nicht zu kümmern, den ergänzt PROC SOAP automatisch. Mit dieser XML-Datei ruft man PROC SOAP auf.
PROC SOAP verpackt die übergebene XML-Datei in einen SOAP-Envelope und sendet beides an den Webservice. Nachdem dieser mit seiner Ausführung fertig ist, sendet er sein Ergebnis an PROC SOAP zurück. PROC SOAP packt nun den eigentlichen Inhalt der Nachricht wieder aus dem SOAP-Envelope aus, und schreibt das Ergebnis in eine

XML-Datei. Aus dieser XML-Datei müssen nun im letzten Schritt wieder SAS-Daten extrahiert werden.

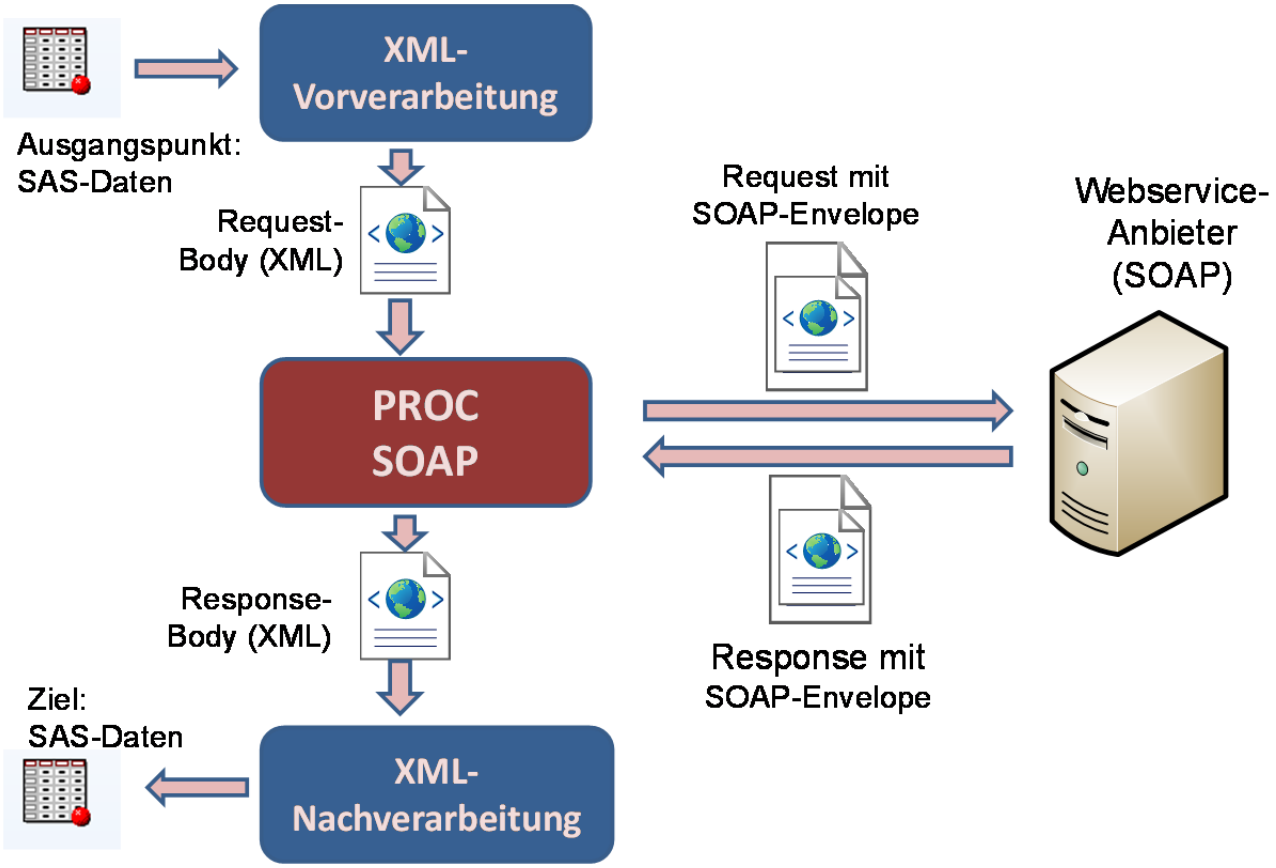


Abbildung 4: Ablauf von PROC SOAP

Der eigentliche Aufruf von PROC SOAP enthält im Wesentlichen eine Referenz auf die einzulesende und die zu erzeugende XML-Datei, und die aufzurufende URL des Webservice. Optional kann man noch eine SOAP-Action angeben, falls ein bestimmter Webservice unter der gleichen URL verschiedene auszuführende Aktionen anbietet. (Die Liste der zur Verfügung stehenden Aktionen lässt sich ebenfalls aus der WSDL-Datei auslesen.)
Das folgende Beispiel zeigt einen Aufruf eines Webservice, der einer IP-Adresse einen bestimmten geographischen Ort zuordnet.

```
FILENAME Request "C:\KSFE12\Request1.xml";
FILENAME Response "C:\KSFE12\Response1.xml";

PROC SOAP IN   = Request
          OUT  = Response
          URL  = "http://ws.cdyne.com/ip2geo/ip2geo.asmx?wsdl"
           SOAPACTION = "http://ws.cdyne.com/ResolveIP"
          ;
RUN;
```

Abbildung 5: Ablauf von PROC SOAP

Über weitere Parameter können auch Benutzername und Passwort übergeben werden. Zur vollständigen Dokumentation von PROC SOAP siehe [3].
Interessant zu wissen ist, dass PROC SOAP auf relativ bekannten und weit verbreiteten Java-Frameworks basiert. Dabei handelt es sich in SAS 9.2 um Apache Axis2, in SAS 9.3 um Spring. Für PROC SOAP ist zwischen SAS 9.2 und 9.3 auch ein deutlicher Performance-Gewinn zu erkennen – es ist zu vermuten, dass dieser auf die Umstellung des zu Grunde liegenden Java-Frameworks zurückzuführen ist.
Noch nicht näher beschrieben wurde die XML-Verarbeitung zum Generieren des XML-Requests und dem Einlesen der XML-Response. Bekanntlich bringt SAS verschiedene Tools mit, mit denen sich XML erzeugen und verarbeiten lässt. Eine Übersicht dazu bietet [4]. Welches dieser Tools empfiehlt sich nun besonders für Webservices?
Generell kann hier keine pauschale Antwort gegeben werden. Allerdings können einige Empfehlungen abgegeben werden, an die man sich halten kann. Zunächst einmal zum Verarbeiten von Webservice-Responses:

a) Mit einer XML-Map kommt man eigentlich meistens relativ gut ans Ziel.
b) Oftmals reicht es aus, nur einen XML-Libname ohne Map zu verwenden. Ob das im konkreten Einzelfall funktioniert, muss man einfach ausprobieren.
c) Für komplizierte Fälle kann man natürlich auch ganz schwere Geschütze auffahren und PROC XSL verwenden. Allerdings sind Webservice-Ergebnisse von der Struktur her meistens nicht besonders komplex, und lassen sich auch ohne XSL-Transformationen verarbeiten.
d) Wenn gar nichts mehr hilft, kann man natürlich immer noch die Webservice-Ergebnisse im Data Step direkt parsen. Das kann natürlich, je nach Komplexität der zu verarbeitenden XML-Struktur, einen immensen Aufwand bedeuten, und sollte wirklich nur ins Auge gefasst werden, wenn sonst gar nichts mehr geht.

Beim Erstellen eines XML-Requests gilt grundsätzlich das Gleiche. Allerdings kommt man hier mit Mitteln der SAS-Makrosprache häufig noch einfacher ans Ziel, indem man folgendermaßen vorgeht:

1. Man erzeugt sich eine XML-"Vorlage", d.h. eine XML-Datei, die die Struktur des Requests enthält, aber in der die einzelnen Parameter durch SAS-Makrovariablen ersetzt sind.
2. In einem SAS-Programm erzeugt man aus der Vorlage den endgültigen XML-Request. In diesem Programm sind die Makrovariablen, die sich in der Vorlage finden, vorhanden und mit ihren konkreten, zu verwendenden Werten belegt. Das Auflösen der Makrovariablen-Bezüge aus der Eingabedatei erfolgt mit der Resolve-Funktion
3. Als Ergebnis hat man eine XML-Datei, in der die Parameter nun ihre konkreten Werte aus SAS enthalten.

In der nachfolgenden Grafik sind diese Abläufe schematisch dargestellt.

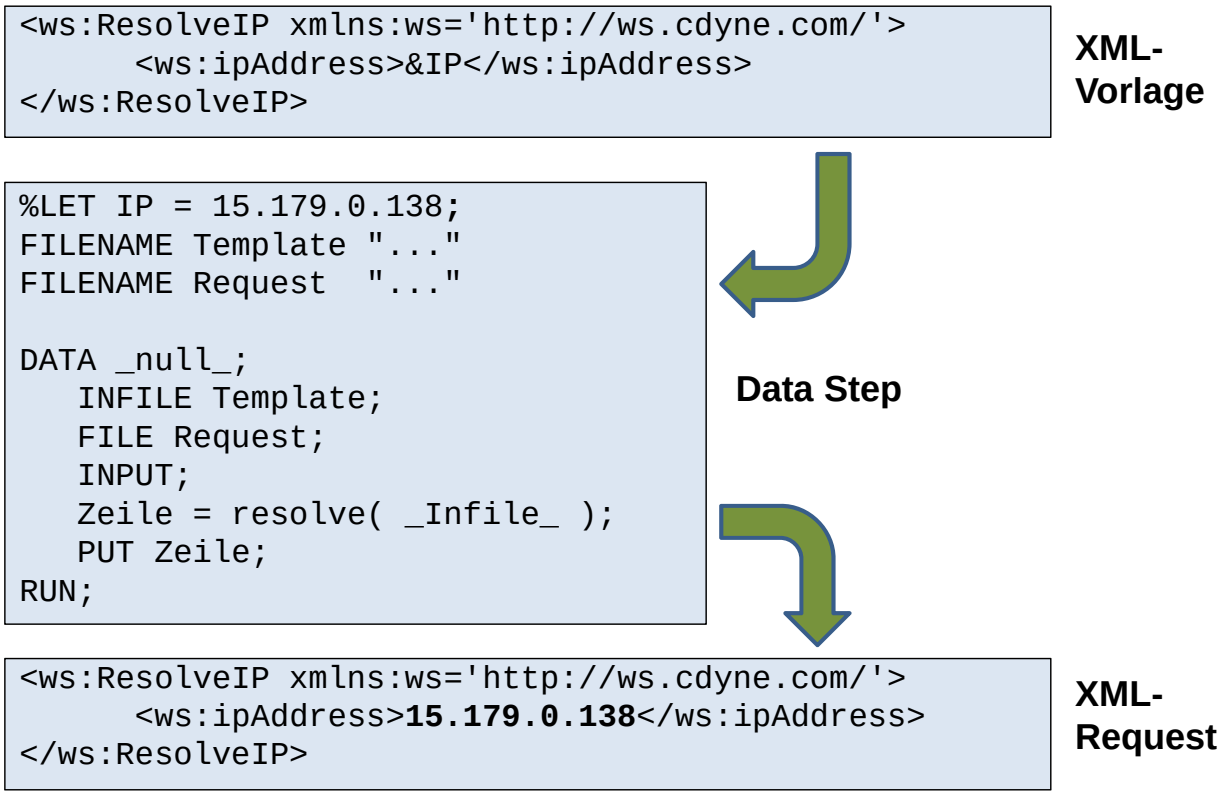


Abbildung 6: Erzeugen von XML-Requests mit Makrovariablen

2.3 Weitere Methoden zum Zugriff auf SOAP-Webservices

In SAS 9.3 ist die Möglichkeit neu hinzugekommen, Webservices aus dem Data Step heraus aufzurufen. Die SOAPWEB-Funktion und die verwandten Funktionen sind beschrieben in [5].
Grundsätzlich entspricht die Vorgehensweise bei Verwendung der SOAPWEB-Funktionen der Vorgehensweise bei PROC SOAP: Es müssen XML-Requests erzeugt werden, und nach der Funktionsausführung XML-Responses verarbeitet werden. Für die Verwendung von XML-Tools gilt das gleiche, was bei PROC SOAP gesagt wurde.
Im folgenden Beispiel ist der Quelltext für den Aufruf einer SOAPWEB-Funktion dargestellt analog zu dem PROC SOAP-Beispiel in Abschnitt 2.2..

Als weitere Schnittstelle zu SOAP-Webservices gibt es in SAS 9.2 die Möglichkeit, einen WSDL-Libname zu verwenden. Dabei wird in einem Libname-Statement die URL der WSDL-Datei eines SOAP-Webservice angegeben; die Libname-Engine bildet die in dieser WSDL-Datei beschriebene Schnittstelle und ihre Parameter auf einen SAS-Dataset ab.
Was in der Theorie eine gute Idee ist, da es die XML-Verarbeitung überflüssig machen würde, läuft leider in der Praxis nicht stabil. In SAS 9.2 ist daher von der Verwendung von WSDL-Libnames eher abzuraten; in SAS 9.3 werden XML-Libname offiziell gar nicht mehr unterstützt.

```
FILENAME Request1  "C:\KSFE12\Request1.xml";
FILENAME Response1 "C:\KSFE12\Response1.xml";

DATA _NULL_;
   InFileRef  = "Request1";
   OutFileRef = "Response1";
   URL        = "http://ws.cdyne.com/ip2geo/ip2geo.asmx?wsdl";
   SOAPAction = "http://ws.cdyne.com/ResolveIP";
   /* Aufruf */
   rc         = SOAPWeb( InfileRef, URL, OutFileRef, SOAPAction );
RUN;
```

Abbildung 7: Verwendung der SOAPWEB-Funktion

2.4 Zugriff auf REST-Webservices mit PROC HTTP

Das Einsatzszenario von PROC HTTP entspricht im Wesentlichen dem von PROC SOAP: Man hat zu Beginn SAS-Daten, transformiert diese zu den benötigten Parametern des Webservice, und muss am Ende aus den Ergebnissen wieder SAS-Daten extrahieren.
Bei Aufrufen von REST-Webservices werden die Parameter üblicherweise als Teil einer URL übergeben (bzw. bei Verwendung der POST-Methode im Request Body angehängt). Zur Konstruktion dieser URL ist in der Regel die SAS-Makrosprache das Mittel der Wahl.
Als Ergebnis liefert ein REST-Webservice üblicherweise Daten im XML- oder JSON-Format zurück. Wenn XML-Ergebnisse weiterverarbeitet werden sollen, gilt genau das Gleiche, was zu PROC SOAP bereits gesagt wurde. Die Weiterverarbeitung von JSON ist problematisch: SAS/BASE bietet hier noch keine Unterstützung an, man müsste sich im Zweifelsfall seine JSON-Ergebnisse selbst parsen. Wenn man bei einem Webservice zwischen mehreren Ausgabeformaten die Wahl hat, dann sollte man zur Weiterverarbeitung in SAS/Base das XML-Format bevorzugen.
Die folgende Abbildung liefert eine Übersicht über den Aufrufprozess von PROC HTTP. Die Übersicht aller Parameter findet sich unter [6]. Wie bei PROC SOAP können auch hier Benutzername und Passwort übergeben werden. Bei POST-Anfragen muss zusätzlich eine komplette Request-Datei als Parameter übergeben werden. Wichtig ist allerdings, dass diese Datei nur den Request-Body enthält; die Header-Elemente

müssen separat übergeben werden, und der Content Type über den Parameter CT= gesetzt werden.

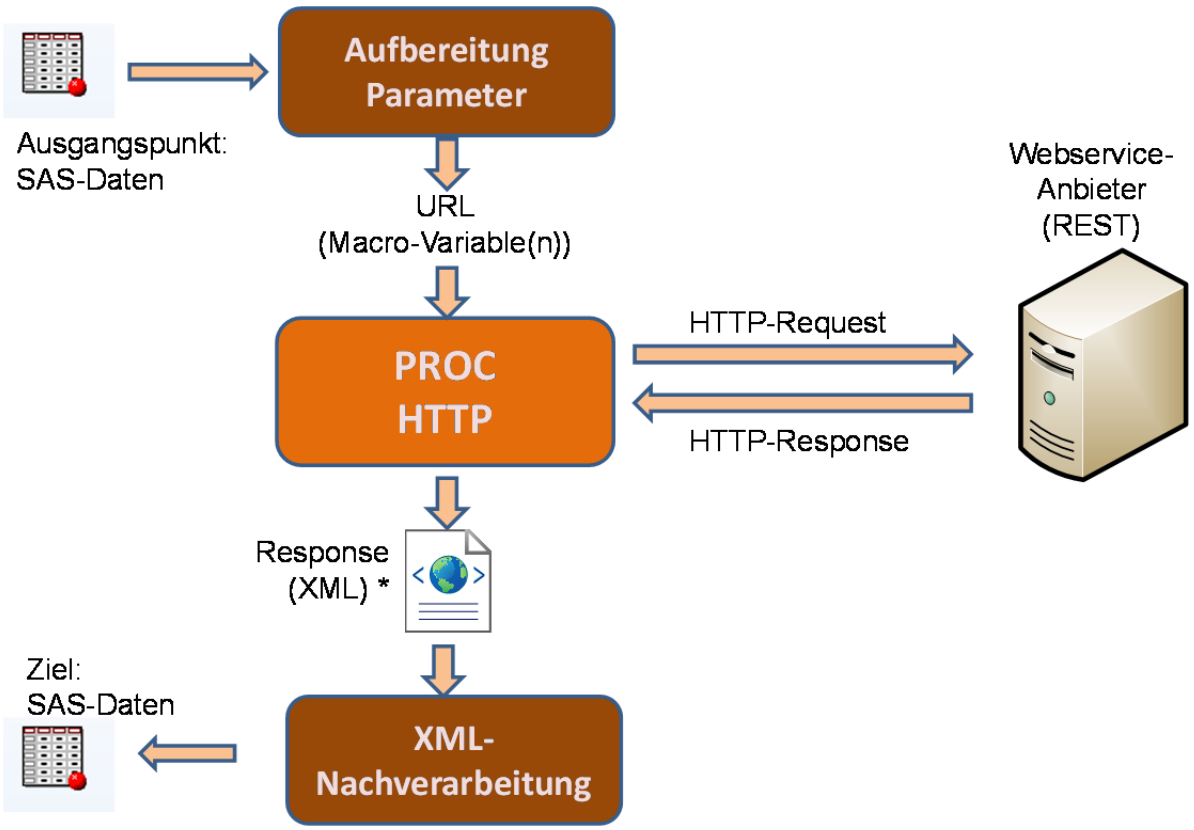


Abbildung 8: Ablauf beim Aufruf von PROC http

```
FILENAME Response "C:\KSFE12\Response1.xml";

PROC HTTP URL     =
'http://api.geonames.org/findNearbyPlaceName?lat=42&lng=42&username=demo'
          OUT     = Response
          METHOD = 'GET'
;
RUN;
```

Abbildung 9: Quellcode für PROC HTTP-Aufruf

2.5 Anmerkungen zur Performance

Es ist relativ schwierig, konkrete Aussagen zur Performance der verschiedenen SAS-Tools zu treffen, da die effektive Performance von einer Vielzahl von Faktoren abhängt. Insbesondere sind dies:

- Die Performance des Netzwerks, über das die Daten übertragen werden
- Die Performance des Webservers, auf dem der Webservice läuft

- Die Performance weiterer Komponenten, wie z.B. Datenbanken, die serverseitig von der Webservice-Implementierung aufgerufen werden.

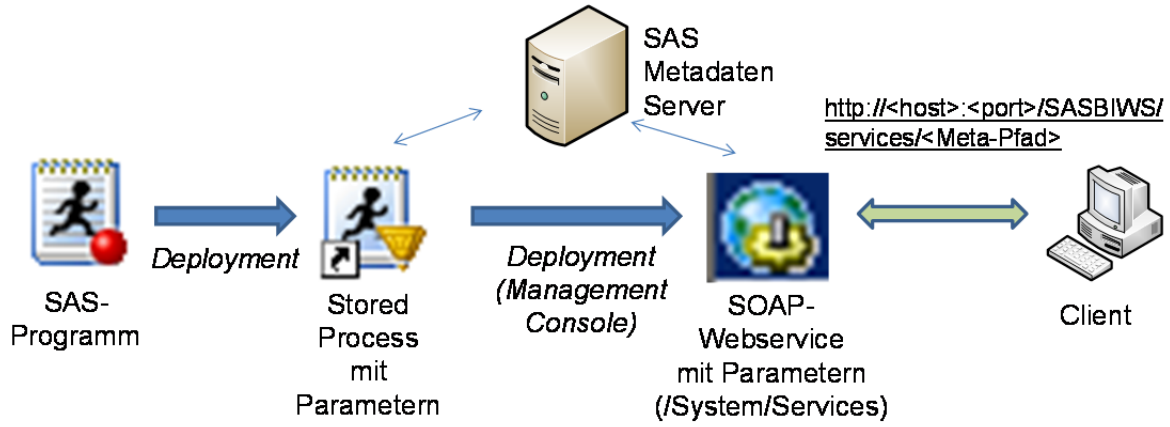
Wenn also beispielsweise eine Datenbank-Abfrage, die im Rahmen der Webservice-Ausführung abgearbeitet wird, aufgrund eines fehlenden Indexes ineffizient ist, dann hat es keinen Sinn, sich Gedanken über ein SAS-seitiges Tuning des Webservice-Zugriffs zu machen.
Über die Performance der SAS-seitigen Komponenten kann man also zwar einige Aussagen treffen, nur sollte man nie außer Acht lassen, das dies eben nur einen Teil der gesamten Performance darstellt:

- Die Implementierung von PROC SOAP wurde in SAS 9.3 im Vergleich zu SAS 9.2 deutlich beschleunigt; wahrscheinlich durch einen Wechsel des zu Grunde liegenden Java-Frameworks. In einer optimalen Konstellation, d.h. bei einem schnellen Netzwerk und bei einem Webservice, der sehr schnell ausgeführt wird, haben wir bei uns zeitliche Verbesserungen von 30-40 Prozent festgestellt.
- Der Unterschied zwischen PROC SOAP und den Data Step-Funktionen ist aus Performance-Sicht vernachlässigbar.
- Der Zugriff auf REST-Webservices ist aufgrund des schlankeren Protokolls in der Regel schneller als der Zugriff auf vergleichbare SOAP-Services. Allerdings wird man meistens nicht in der Situation sein, dass man zwischen SOAP und REST entscheiden kann: man muss die Technologie verwenden, die der Webservice-Betreiber vorgibt.

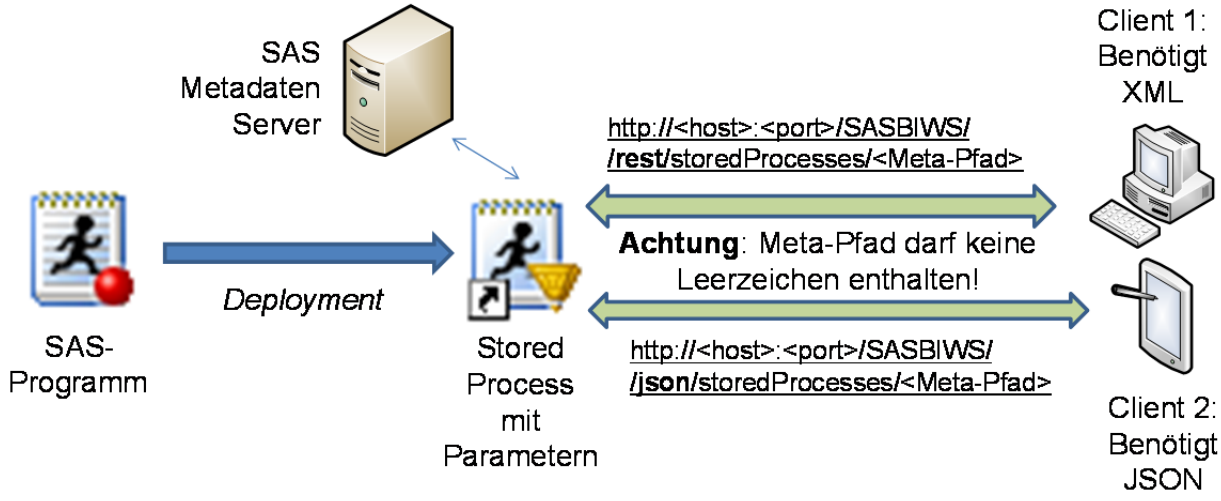
2.6 Bereitstellen von SAS-Programmen als Webservices.

Nachdem die letzten Kapitel den Aufruf von Webservices unter SAS/Base behandelt haben, soll zum Abschluss nun besprochen werden, wie sich eigene SAS/Base-Programme als Webservice bereitstellen lassen. Die Entwickler-Dokumentation zu diesem Thema findet sich unter [7]. Da die Möglichkeiten zur Bereitstellung von Webservices in SAS 9.3 deutlich erweitert wurden, beziehen sich die Inhalte dieses Kapitels ausschließlich auf SAS 9.3.
Grundsätzlich gilt, dass jedes SAS-Programm, das als Webservice ausführbar sein soll, zunächst als Stored Process bereitgestellt werden muss. Dabei müssen Eingabe-Parameter und (neu in SAS 9.3) auch Ausgabeparameter festgelegt werden. Diese Eingabe- und Ausgabeparameter bilden später auch die Schnittstelle des Webservice: D.h. der Webservice erwartet die definierten Eingabeparameter im Request, und füllt die Response mit den angegebenen Ausgabeparametern.
Zu beachten ist, dass, wenn der Stored Process selbst Web-Content erzeugt (z.B. durch Schreiben nach _WEBOUT), dieser Content nicht Bestandteil der Webservice-Ausgabe werden wird – der Webservice gibt nur das zurück, was explizit als Ausgabe-Parameter definiert ist. Weiterhin muss das SAS-Programm nicht selbst XML oder JSON als Ausgabe schreiben – diese Aufgabe übernimmt SAS.
Um einen bereitgestellten Stored Process über SOAP aufrufbar zu machen, muss er aus der Management Console explizit als Webservice bereitgestellt werden. Dabei werden

benötigte Metadaten, wie die WSDL-Datei, automatisch aus den vorhandenen Parametern des Stored Process generiert.



Bei REST sieht die Sache noch etwas einfacher aus: Ein Stored Process ist automatisch auch über eine REST-Schnittstelle aufrufbar, ohne dass er hierzu extra als Webservice bereitgestellt werden muss. Man darf diesen REST-Aufruf nicht mit dem Aufruf über die Stored Process Web Application verwechseln: Bei einem Aufruf über die Stored Process Web Application wird der Stored Process ausgeführt, und schreibt ggf. eine Ausgabe nach _WEBOUT, welche man dann als "Ergebnis" des Aufrufs enthält. Bei einem REST-Webservice-Aufruf umfasst die Ausgabe hingegen nur die Ausgabeparameter des Stored Processes.
Die REST-Schnittstelle erlaubt die Übergabe von Parametern und Ergebnissen sowohl im XML- als auch im JSON-Format. Um zwischen diesen beiden Formaten zu wechseln, muss nur eine andere URL verwendet werden. So können also unterschiedliche Clients, die jeweils andere Formate erwarten, den gleichen Stored Process verwenden, ohne dass dieser wissen muss, dass er gerade zwei unterschiedliche Ausgabeformate bedient.



Eine Besonderheit bei der Bereitstellung von REST-Webservices mit SAS ist, dass Parameter ausschließlich mit POST-Requests übergeben werden. Dies ist normalerweise eigentlich nicht üblich: Parameter für Requests, die den Server nicht verändern, sollten

eigentlich auch GET verwendet werden können; SAS verwendet hier eine etwas eigenwillige Implementierung. Ebenfalls beachten sollte man, dass bei Verwendung der XML-Schnittstelle auch die Requests im XML-Format (und nicht einfach als im Request Body stehende URL-Parameter) übergeben werden müssen.

3 Fazit

Mit den SAS-Prozeduren PROC SOAP und PROC HTTP bzw. den SOAPWEB-Data Step-Funktionen ist es möglich, auf jeden SOAP- oder REST-basierten Webservice zuzugreifen. Man muss nicht wissen, auf welchem System der aufgerufene Webservice läuft, und der aufgerufene Webservice muss nichts davon wissen, dass er gerade von SAS aufgerufen wird. Die Implementierung aufwändiger Schnittstellen entfällt.
Umgekehrt ist es auch möglich, über die Stored Process-Schnittstelle eigene SAS-Programme als Webservices bereitzustellen. So kann SAS-Funktionalität einfach in andere Systeme eingebunden werden, ohne erst neue Schnittstellen entwickeln zu müssen. Insbesondere die Möglichkeit, die Ausgabeparameter von Stored Processes direkt ins JSON-Format auszugeben, ohne dass dies im SAS-Programm extra entwickelt werden muss, dürfte zukünftig interessante neue Möglichkeiten in der Koppelung von Webfrontend und SAS-Backend bereitstellen.

Literatur

[1] Word Wide Web Consortium: Web Services Architecture
http://www.w3.org/TR/ws-arch/

[2] http://de.wikipedia.org/wiki/Webservice

[3] SAS Institute Inc.: Base SAS(R) 9.3 Procedures Guide – SOAP Procedure.
http://support.sas.com/documentation/cdl/en/proc/63079/HTML/default/viewer.htm#n0vft359cr5yyon15n7b6zvrv102.htm

[4] Andreas Adlichhammer: XML mit SAS leicht gemacht. Vortrag auf der KSFE 2011. Abzurufen unter
http://www.analytical-software.de/de/wissensbibliothek/tutorials/xml-sas/

[5] SAS Institute Inc.: SAS(R) 9.3 Functions and CALL Routines: Reference – SOAPWEB Function
http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0e2pridpof9epn1es3ow9klt1jh.htm

[6] SAS Institute Inc.: Base SAS(R) 9.3 Procedures Guide – HTTP Procedure
http://support.sas.com/documentation/cdl/en/proc/63079/HTML/default/viewer.htm#n0bdg5vmrpyi7jn1pbgbje2atoov.htm

[7] SAS Institute Inc.: SAS 9.3 BI Web Services Developer's Guide
http://support.sas.com/documentation/cdl/en/wbsvcdg/62759/PDF/default/wbsvcdg.pdf