Was kann ein Web Service – oder muss er können?

Die neuen W3C Standards „Web Services Policy 1.5 – Framework“ und „Web Services Policy 1.5 – Attachment“ bilden die Grundlage zur Beschreibung von Eigenschaften und Anforderungen von Web Services.

Felix Sasaki

Web Services zu verstehen ist nicht leicht – weder für Menschen, noch für Maschinen. Letztere können nur erfolgreiche Interaktionen eingehen, wenn die Eigenschaften und Anforderungen eines Service Providers und Requester bekannt sind, z.B. hinsichtlich Sicherheit, verlässlicher Nachrichtenübermittlung oder Optimierung. Das W3C hat hierzu vor kurzem zwei neue „Recommendations“ verabschiedet: „Web Services Policy 1.5 – Framework“ ist ein Rahmenwerk, um Eigenschaften von Services als so genannte Web Services „Policy“ zu beschreiben. „Web Services Policy 1.5 – Attachment“ definiert, wie eine Policy mit Einheiten (z.B. einem Endpoint) verknüpft wird, auf die sie sich bezieht. Der Artikel beschreibt den Hintergrund von Web Services Policy, Eigenschaften der Policy Beschreibungssprache und Policy Verknüpfungsmechanismen, sowie verschiedene Anwendungsszenarien.

Warum Web Services Policy?

Die Motivation für Web Services Policy zeigt sich bei einem Blick auf die Interoperabilität von SOAP-basierten Web Services. Grundlegender Bestandteil der Interaktion sind SOAP Nachrichten. SOAP ist auf einfache Erweiterkeit ausgelegt: SOAP Header und SOAP Body können verschiedene, benutzedefinierte Informationen beinhalten. Um das Format dieser Information zu definieren, wird XML Schema verwendet. Des weiteren dient WSDL zur Beschreibung der Protokolle der Interaktion, oder von „message exchange pattern“.

Mit XML Schema und WSDL ist Interoperabilität hinsichtlich Nachrichten und Protokolle gewährleistet. Web Services Policy fügt nun Meta-Informationen hinzu, die über diese Ebenen hinausgehen. Diese Informationen betreffen Eigenschaften von Web Services hinsichtlich Sicherheit, verlässlicher Nachrichtenübermittlung, Optimierung von Nachrichten etc. Solche Anwendungsbereiche von Web Services Policy heißsen in der Policy-Terminologie „Domains“.

Die Spezifikation „Web Services Policy 1.5 – Framework“ [1] ist das Rahmenwerk zur Erstellung von Policies und „Policy Assertions“, d.h. den Beschreibungen einer Eigenschaft oder Anforderung von Web Services. „Web Services Policy 1.5 – Attachment“ [2] definiert zudem Verfahren, wie Policies mit „Policy Subjects“, z.B. einem Service Endpoint, verknüpft werden können. Die Nutzung des Rahmenwerks und der Verknüpfungsmechanismen wird in separaten, für jede Domain spezifischen Dokumenten beschrieben. Für die drei angesprochenen Domains sind bereits von verschiedenen Organisationen Policy Assertions erstellt worden: die WS-SecurityPolicy [3], Web Services Reliable Messaging Policy Assertion [4] und WS-MTOMPolicy [5]. Es ist zu erwarten, dass verschiedene Organisationen mit der Verabschiedung von Web Services Policy 1.5 weitere Domains in Angriff nehmen und entsprechende Assertions verfassen werden.

Für wen ist Web Services Policy gedacht?

Policy Assertions werden wie gesagt als eigene Spezifikationen entwickelt und reflektieren industrielle Erfahrungen und Anforderungen in der jeweiligen Domain. Die Erstellung von

Assertions ist keine simple Aufgabe, da viele Entscheidungen getroffen werden müssen: Gibt es schon bestehende Assertions der Domain? Interagiert die neue Assertion mit Assertions anderer Domains? Gibt es optinale Assertions in der Domain? Um die Beantwortung solcher Fragen zu erleichtern, wird zur Zeit das Dokument „Web Services Policy 1.5 - Guidelines for Policy Assertion Authors“ [6] erstellt.

Der Entwickler eines Services oder eines Clients bekommt möglicherweise die Policy Assertions selbst nicht zu Gesicht. Je nach Tool sind sie vor dem Entwickler „verborgen“, um unnötige Komplexität zu vermeiden. Die domänenspezifischen Eigenschaften oder Anforderungen stehen dann als Menge von Optionen der Service- oder Clienterstellung in einem Benutzerinterface zur Verfügung.

Die Wahl solcher Optionen, egal ob im GUI oder als Policy Quellcode, bedeutet die Erstellung so genannter „Policy Expressions“ (oder einfach „Policy“), d.h. einer Sammlung von Policy Assertions und ihrer Strukturierung mittels anderer Konstrukte der Policy Beschreibungssprache (vgl. folgende Abschnitte). Ein Entwickler eines Web Services muss wissen, wie eine Policy Expression aufgebaut ist, d.h. sie lesen und schreiben können. Implementierer von Web Services Tools müssen die Policy Beschreibungssprache hinreichend verstehen, um ihre Verarbeitung zu gewährleisten. Für Web Services Entwickler und Tool Developer wird derzeit das Dokument „Web Services Policy 1.5 – Primer“ [7] erstellt, das Antworten auf diese Fragestellungen gibt.

Grundlegende Bestandteile einer Policy

Abbildung 1 zeigt einige grundlegende Bestandteile einer Policy und eine Möglichkeit, diese mit einer WSDL-Beschreibung zu verknüpfen. Die Namensräume in dem Beispiel sind durch drei Präfixe gekennzeichnet: „wsp“ für die Policy Beschreibungssprache, „wsam“ für die „Web Services Addressing Metadata“ Spezifikation [8], und „sp“ für WS-SecurityPolicy.

```
POLICY 1
<wsp:Policy… wsu:Id="common">
  <wsam:Addressing wsp:Optional="true">…</wsam:Addressing>
</wsp:Policy>
POLICY 2
<wsp:Policy… xml:id="commonPlusSecurity">
<wsp:All>
  <wsp:ExactlyOne>
   <sp:TransportBinding>…</sp:TransportBinding>
   <sp:AsymmetricBinding>…</sp:AsymmetricBinding>
  </wsp:ExactlyOne>
  <PolicyReference URI="#common"/>
 </wsp:All>
</wsp:Policy>
NUTZUNG IN WSDL
<wsdl:binding…>
  <Policy>
    <PolicyReference URI="#commonPlusSecurity"/>
  </Policy>
  …
</wsdl:binding>
```

Abb. 1: Grundlegende Bestandteile einer Policy und ihre Nutzung in WSDL

Die Abbildung enthält zwei Policies bzw. „Policy Expression“. Eine Policy Expresssion ist die XML Repräsentation einer Policy. Web Services Tools, die Policies vor Web Services Entwicklern „verbergen“, nutzen eventuell Tool interne Repräsentationen. Policies bzw. Policy Expressions beinhalten so genannte „Policy Alternatives“. Dies sind die eigentlichen Sammlungen der Policy Assertions.

Zunäachst zur ersten Policy in Abbildung 1: Das <wsp:Policy> Element fungiert als Wrapper Element. Die Attribute „wsu:Id“ (oder alternativ „xml:id“) dient der Identifikation einer Policy. Die

Policy umfasst eine Policy Alternative mit genau einer Policy Assertion. Diese Policy Assertion wird ausgedrückt durch das Element <wsam:Addressing>. Sie zeigt an, dass die Nutzung von Web Services Addressing für eine erfolgreiche Web Services Interaktion notwendig ist. Das am <wsam:Adressing> Element vorhandene Attribut wsp:Optional wird im nächsten Abschnitt vorgestellt.

Die zweite Policy demonstriert, wie verschiedene Policy Alternatives und die in ihnen enthaltenen Policy Assertions miteinander kombiniert werden. Dies geschieht durch die Gruppierung von Policy Alternatives mittels der drei Policy Operator <wsp:Policy>, <wsp:All> und <wsp:ExactlyOne>. <wsp:All> und <wsp:Policy> haben die gleiche Funktion: alle eingebetteten Policy Alternatives und deren Policy Assertions stellen ein notwendiges Verhalten dar. <wsp:ExactlyOne> hingegen zeigt an, dass genau eine der eingebetteten Policy Alternatives gegeben sein muss. Die Nutzung diese Operatoren in der zweiten Policy zeigt an, dass Web Services Addressing und genau eine Form von Security genutzt werden muss: „Transport-level Security“, ausgedrückt durch die <sp:TransportBinding> Assertion, oder „Message-level Security“, ausgedrückt durch die <sp:AsymmetricBinding> Assertion. Die Policy Assertion für Web Services Addressing wird dabei als Bestandteil der zuerst definierten Policy, mittels des <wsp:PolicyReference> Elements, eingebunden.

Schließlich zeigt Abbildung 1, wie eine Policy mit einer WSDL-Beschreibung verknüpft wird. In dem Beispiel wird die Policy als Kindelement eines <wsdl:binding> Elements eingefügt. Dies bedeutet, dass sich die Policy auf den Endpoint bezieht, der durch das Binding definiert ist. In der Web Services Policy Terminologie bezeichnet man den Endpoint als „Policy Subject“. Weitere Policy Subjects werden in einem späteren Abschnitt vorgestellt.

Ein näherer Blick auf Policy Assertions: Attribute und Kindelemente, Nested Policy Expressions und leere Policies

Policy Assertions sind gekennzeichnet durch einen eindeutigen Elementnamen. Die Elemente können weitere Attribute besitzen oder Kindelemente.

Es gibt zwei Attribute im Policy Namensraum: „wsp:Optional“ und „wsp:Ignorable“. wsp:Optional mit dem Wert „true“ dient der Identifikation von optionalen Policy Assertions. Wird z.B. das <wsam:Addressing> aus Policy 1 in Abbildung 1 mit „wsp:Optional='true'“ versehen, bedeutet dies: Die Nutzung von Web Services Addressing ist optional. wsp:Ignorable steht für Policy Assertions, die keine Auswirkung auf die Interaktion besitzen, wie z.B. eine Assertion, die das Logging eines Providers ausdrückt. Dieses Attribut wurde erst in der Version „1.5“ von Web Services Policy eingeführt. Es gibt noch keine standardisierten Assertions, welche „wsp:Ignorable“ benutzen.

Attribute oder Kindelemente aus anderen Namensräumen werden als „Assertion Parameter“ bezeichnet. Die Web Services Policy 1.5 Spezifikationen sagen nichts über die Semantik dieser Attribute und Elemente aus. Dies bleibt den jeweiligen Domänen überlassen.

Die Domäne von Web Services Security macht regen Gebrauch von Assertion Parameters und einer weiteren Möglichkeit, Policy Assertions näher zu spezifizieren: so genannte „Nested Policy Expressions“. Nested Policy Expressions sind Policy Expressions als Kindelemente von Policy Assertions. Abbildung 2 zeigt als Beispiel für eine Nested Policy Expression eine Erweiterung der <sp:TransportBinding> Policy Assertion aus Abbildung 1.

```
<sp:TransportBinding>
  <Policy>
    <sp:TransportToken>
```

```
      <Policy>
        <sp:HttpsToken>
          <wsp:Policy/>
        </sp:HttpsToken>
      </Policy>
    </sp:TransportToken>
    <sp:AlgorithmSuite>
      <Policy>
        <sp:Basic256Rsa15/>
      </Policy>
    </sp:AlgorithmSuite>
    …
  </Policy>
</sp:TransportBinding>
```

Abb.2 : Nested Policy Expression

In der Nested Policy Expression wird die <sp:TransportBinding> Policy Assertion weiter spezifiziert mittels zweier Assertions: hinsichtlich der Art des Transport Token (vgl. das <sp:TransportToken> Element) und einer Algorithm Suite für Verschlüsselungsoperationen (vgl. das <sp:AlgorithmSuite> Element). Die Policy Assertion für das Transport Token enthält zudem eine Nested Policy Expression <sp:HttpsToken> mit einer weiteren, leeren Nested Policy. Die Semantik von leeren, nested Policies (oder Policy Alternatives) bedeutet: das Verhalten ist hier nicht näher bestimmt und somit auch nicht gefordert. Wenn also ein Requester in seiner Policy eine bestimmte Form von Authentifizierung verlangt, ist dies nicht kompatibel zu der leeren Policy eines Providers.

Auf zur Service Interaktion: Policy Normal Form und Policy Intersection

Bisher wurde nur die Definition von Policies und ihre Verknüpfung mit Web Services Einheiten am Beispiel von WSDL vorgestellt. Der eigentlich Zweck von Policies zeigt sich jedoch in ihrer Anwendung innerhalb der Web Services Interaktion. Ein Requester soll feststellen können, ob eine Policy Alternative seiner Policy mit einer Alternative des Providers kompatibel ist, vgl. die im letzten Abschnitt erwähnte Kompatibilitätsbedingung der leeren, nested Policy. Nach dieser Kompatibilitätsprüfung kann der Requester entscheiden, ob er mit dem Provider interagiert, und welche Policy Alternative er dazu nutzt.

Zur Kompatibilitätsprüfung stellt „Web Services Policy“ die Möglichkeit der „Policy Intersection“ bereit. Voraussetzung für die Durchführung der Policy Intersection ist, dass die beiden Policies in „Normal Form“ vorliegen. Abbildung 3 enthält die Policy aus Abbildung 2 in Normal Form.

```
<Policy>
  <ExactlyOne>
    <All> <!-- - - - - - - - - - - - - - - Policy Alternative (a) -->
       <wsam:Addressing>…</wsam:Addressing>
       <sp:TransportBinding>…</sp:TransportBinding>
    </All>
    <All> <!-- - - - - - - - - - - - - - - Policy Alternative (b) -->
      <wsam:Addressing>…</wsam:Addressing>
      <sp:AsymmetricBinding>…</sp:AsymmetricBinding >
    </All>
    <All> <!-- - - - - - - - - - - - - - - Policy Alternative (c) -->
       <sp:TransportBinding>…</sp:TransportBinding>
    </All>
    <All> <!-- - - - - - - - - - - - - - - Policy Alternative (d) -->
          <sp:AsymmetricBinding>…</sp:AsymmetricBinding>
    </All>
  </ExactlyOne>
</Policy>
```

Abb. 3: Normalform einer Policy

Die bisher in diesem Artikel vorgestellte Form ist die „Compact Form“, die dem Erstellen (nicht zuletzt vom „menschlichen“ Web Services Entwickler) von Policies dient. In der vom Web Services Tool generierten Normal Form gibt es ein <wsp:Policy> Wrapper Element mit einem

<wsp:ExactlyOne> Kindelement. Dieses beinhaltet 0 oder mehr <wsp:All> Kindelemente. Jedes <wsp:All> Element beinhaltet 0 oder mehr Policy Assertions. Nested Policies haben genau eine Policy Alternative. <wsp:PolicyReference> ist in der Normal Form nicht vorhanden. „wsp:Optional“ wird ersetzt durch ein <wsp:ExactlyOne> Element mit zwei Kindelementen: der Policy Assertion und einem leeren <wsp:All> Element. Unter Berücksichtigung der <wsp:PolicyReference> zu POLICY 1, und des wsp:Optional Attributes an wsam:Addressing, führt die POLICY 2 aus Abbildung 1 zur Repräsentation von vier Policy Alternatives in der Normal Form. Diese vier Policy Alternatives waren in Abbildung 2 auch gegeben, allerdings nicht explizit und nicht unmittelbar der Policy Intersection zugänglich.

Intersection bedeutet: Ist eine kompatible Alternative (im Beispiel a), b), c) oder d)) vorhanden, so wird sie in das Ergebnis der Intersection mit aufgenommen. Kompatibel bedeutet, dass der Namen der Assertions identisch sein müssen, und die Nested Policies ebenfalls. Assertion Parameter werden nicht berücksichtigt.

Für Intersection gibt es zwei Modi: Lax und Strict. Es bleibt den jeweiligen Tools überlassen, wie oder ob sie die Modi dem Benutzer anzeigen. Im Strict Modus hat „wsp:Ignorable“ keinen Einfluss auf Intersection. Im Lax Modus werden Policy Assertions mit wsp:Ignorable="true" bei der Intersection ignoriert, d.h. die Assertions sind nicht Bestandteil des Ergebnisses der Intersection. Wenn z.B. ein Requester nicht daran interessiert ist, ob ein Provider Logging betreibt, kann der Requester Lax Intersection nutzen, um entsprechende Assertions nicht berücksichtigen zu müssen.

Verknüpfung von Policies mit Policy Subjects und Effective Policy

Neben WSDL 1.1 oder WSDL 2.0 beschreibt „Web Services Policy 1.5 – Attachment“ Verknüpfungsmechanismen für UDDI. Als Policy Subject dienen die UDDI Elemente <uddi:businessEntity> (Service Provider Policy Subject), <uddi:businessService> (Service Policy Subject), und <uddi:bindingTemplate> bzw. <uddi:tModel> (Endpoint Policy Subject).

In Abbildung 1 wurde vorgestellt, wie eine Policy mit dem Policy Subject, auf das sie sich bezieht, verknüpft werden kann. Eine Policy kann sich auch auf mehrere Policy Subjects beziehen. In WSDL gibt es zusätzlich zu dem Endpoint normalerweise zwei Policy Subjects. Dies sind eine für einen Endpoint spezifische Operation (d.h. das <operation> Element), oder eine für eine Operation spezifische Message (d.h. <input> oder <output> Elemente). Zudem ist es möglich, einen kompletten Service als Policy Subject zu bestimmen und mit einer Policy zu verknüpfen. Dies geschieht jedoch seltener. Der Grund liegt darin, dass die Wahl von z.B. sicherheitsrelevanten Policies oft nicht spezifisch für den kompletten Service ist, sondern nur für einen Endpoint, eine Operation oder eine Message.

Die „Effective Policy“ ist die Kombination aller Policies, die mit einem Policy Subject verknüpft sind. Die „Attachment“ Spezifikation beschreibt, wie die Effective Policy für WSDL 1.1, WSDL 2.0 und UDDI berechnet wird. Ein Beispiel aus WSDL 1.1: die Effective Policy eines Endpoint Policy Subject ist die Kombination aller Policies, die mit den folgenden Elementen verknüpft sind: WSDL <port> Element, von diesem <port> Element referenzierte <binding> Elemente, und von dem <port> Element referenzierten <portType> Elemente.

Zusammenfassung

Dieser Artikel hat in die Hintergründe, Anwendungsszenarien und Grundlagen von „Web Services Policy“ eingeführt. Es gibt zahlreiche, potentielle Nutzer dieser neuen Spezifikationen – vom Policy Assertion Author bis zum GUI geleiteten Web Services Entwickler. Diese Spannbreite der Leserschaft eines Artikels abzudecken ist nahezu unmöglich. Dennoch hoffe ich, dass genug Wissen vermittelt wurde um zu zeigen was ein Web Services kann – oder können muss.

Felix Sasaki arbeitet beim World Wide Web Consortium. Er beschäftigt sich mit den Bereichen Internationalisierung und Web Services.

Links & Literatur

- [1] http://www.w3.org/TR/ws-policy/
- [2] http://www.w3.org/TR/ws-policy-attach
- [3] http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702
- [4] http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01.pdf
- [5] http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization/
- [6] http://www.w3.org/TR/ws-policy-guidelines/
- [7] http://www.w3.org/TR/ws-policy-primer/
- [8] http://www.w3.org/TR/ws-addr-metadata/