Norm 260 – Sicherheitsmechanismen

Release und Version

Release 2

Version 2.5.0 vom 15.04.2014, NAUS-Beschluss vom 27.03.2014

Status

Arbeitsentwurf vom 12.08.2008

Potenzielle Norm vom 28.08.2008

Vorgeschlagene Norm vom 08.10.2010

RFC Fristende am 12.01.2011

Offizielle Norm unveröffentlicht vom 27.01.2011

Editor

Geschäftsstelle BiPRO e.V.

Autoren

Martin Bierlein, BiPRO e.V. (bierlein@bipro.net)

Dr. Dieter Ackermann, VOLKSWOHL BUND (dieter.ackermann@volkswohl-bund.de)

Carsten Baehr, VOLKSWOHL BUND (carsten.baehr@volkswohl-bund.de)

Sören Chittka, VOLKSWOHL BUND (soeren.chittka@volkswohl-bund.de)

Fabian Stolz, VOLKSWOHL BUND (fabian.stolz@volkswohl-bund.de)

Dr. Günther vom Hofe, Continentale (guenter.vomhofe@continentale.de)

Dr. Thomas Kippenberg, NÜRNBERGER (thomas.kippenberg@nuernberger.de)

Dr. Torsten Schmale, inubit AG (ts@inubit.com)

Markus Heussen, Unternehmensberatung (heussen@standardisierung.net)

Michael Pollmeier, S&N AG (mpollmeier@s-und-n.de)

Gegenstand der Norm

Die Norm 260 definiert die technischen Sicherheitsmechanismen die im Kontext BiPRO-konformer Schnittstellen zu verwenden sind.

Voraussetzung

Norm 200, Release 2

Norm 210, Release 2

Norm 220, Release 2

Norm 240, Release 2

Norm 250, Release 2

Norm 270, Release 2

Hinweise zum Urheberrecht

Dieses Norm-Dokument, wie auch alle anderen damit im Zusammenhang stehenden Dokumente (z.B. technische Dateien, Datenmodell etc.) von BiPRO unterliegen dem Urheberrecht.

BiPRO-Normen und andere Dokumente stehen während ihrer Entwicklungs- und Qualitätssicherungsphase nur den Mitgliedern des BiPRO e.V. zur Nutzung und Anwendung zur Verfügung. Die Überlassung an diesen geschlossenen Empfängerkreis stellt keine Erstveröffentlichung, zu deren der BiPRO e.V. allein berechtigt ist, dar.

Die BiPRO-Normen werden nach erbrachtem Nachweis der Praxistauglichkeit und mit der Feststellung des Status "Offizielle Norm" (ON) für die allgemeine Veröffentlichung freigegeben (vgl. BiPRO Norm 100: Allgemeine Grundlagen der Normierung) und sodann von BiPRO der Öffentlichkeit als BiPRO-Norm einschließlich der dazugehörigen Dokumente über das vereinsöffentliche Normenportal von BiPRO zur Verfügung gestellt; es gelten die dort veröffentlichten jeweils aktuellen Nutzungsbedingungen www.bipro.net/nutzungsbedingungen.

Inhaltsverzeichnis

86

Sicherheit von Services

Allgemeines

Sicherheit bedeutet im Kontext der BiPRO die Absicherung der Internet-basierten Kommunikation zwischen den Prozessbeteiligten und die Absicherung der Web Services vor unbefugtem Zugriff. Folgende Aspekte der Sicherheit sind zu betrachten:

- **Vertraulichkeit**
  Daten DÜRFEN bei der Übertragung NICHT durch Unberechtigte gelesen werden können.
- **Integrität**
  Daten DÜRFEN bei der Übertragung NICHT unbemerkt durch Unberechtigte verändert werden können.
- **Authentizität**
  Daten und Services DÜRFEN NUR berechtigten Personen oder Systemen zur Verfügung gestellt werden.
- **Verbindlichkeit**
  Stellt die Autorenschaft des Senders sowie den Nachweis der durchgeführten Aktionen sicher.

Vertraulichkeit und Integrität

Nach dem derzeitigen Stand der BiPRO-Aktivitäten ist die Abbildung einer Ende-zu-Ende-Sicherheit aufgrund fehlender Intermediäre zurzeit nicht erforderlich. Aus diesem Grund wird lediglich die Sicherung des Transportweges benötigt. Die Vertraulichkeit und Integrität MÜSSEN über das HTTPS-Protokoll realisiert werden. Wird dennoch später eine Ende-zu-Ende-Sicherheit bei der Kommunikation benötigt, MUSS die WS-Security-Spezifikation von OASIS inkl. WS-Signature und WS-Encryption verwendet werden.

Authentizität

Die Authentizität eines Benutzers ist für die meisten Services, die im Rahmen der BiPRO normiert werden, notwendig, da viele Services nur autorisierten Benutzern zur Verfügung gestellt werden dürfen. Ein Benutzer MUSS sich vor der Nutzung eines Services authentifizieren, falls der Service dies erwartet. Die BiPRO-Norm trifft ausdrücklich keine

Regelungen zur Sicherung der Autorisierung. Die Prüfung, ob eine authentifzierter Benutzer Zugriff auf einzelne Funktionen oder Daten eines Services hat, MUSS durch den Anbieter des Services behandelt werden.

Verbindlichkeit

Die Verbindlichkeit wird in einer späteren Phase im Rahmen der Abbildung einer End-to-End-Security behandelt.

Verwendete Standards

Grundlage der in diesem Kapitel beschriebenen Normen zur sicheren Abwicklung von Prozessen im Kontext der BiPRO sind die Implementierungsrichtlinien der WS-I. Die in diesem Kapitel beschriebenen Inhalte beziehen sich in weiten Teilen auf das **Basic Security Profile**[1] in der Version 1.0.

Die BiPRO-Normen behandeln die Aspekte der Sicherheit, wie sie auf der Basis des HTTP-Protokolls und des Nachrichtenaustausches auf SOAP-Ebene realisiert werden können. Basis für die Abbildung der Sicherheit sind die folgenden OASIS-Spezifikationen.

- **WS-Security** (Version 1.0) Sicherheitsframework für Web Services
- **WS-Trust** (Februar 2005) Definition von Security-Token-Services
- **WS-SecureConversation** (Februar 2005) Abwicklung sicherer Sessions
- **WS-SecurityPolicies** (Juli 2005) Definition der Sicherheitsanforderungen

WS-Security bietet Mechanismen für das Sichern einer einzelnen Nachricht bei einem einfachen Nachrichtenaustausch. Interaktionen zwischen einem Web Service und einem Benutzer führen jedoch meist zum Austausch mehrerer Nachrichten. Obwohl jede Nachricht einzeln gesichert werden kann, ist es effizienter, einen vom Web Service und dem Benutzer gemeinsam verwendeten Kontext zu schaffen und mit diesem die durch das Sichern jeder ausgetauschten Nachricht entstehende Arbeitslast zu reduzieren.

WS-Trust definiert dazu einen Security-Token-Service und WS-SecureConversation das Verfahren, wie ein Sicherheitskontext-Token (Security-Context-Token, SCT) generiert und genutzt wird. Beide Spezifikation sind sehr umfassend und bieten z. B. auch eine Basis für

[1] Download: http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

Gewährleistung von Integrität und Vertraulichkeit. Diese Aspekte der Sicherheit werden bei BiPRO über das HTTPS-Protokoll realisiert. Dies steht nicht im Widerspruch zu WS-Trust und WS-SecureConversation, da diese explizit als Baustein-System entworfen sind.

Authentifizierung

Zur Sicherstellung der Authentizität eines Service Consumers ist seine Authentifizierung erforderlich. Es wird empfohlen, dass ein Service Provider eine angemessene Authentifizierung seiner Services realisiert. Dennoch kann ein Business Service auch ohne eine Authentifzierung BiPRO-konform sein.

Des Weiteren DARF ein Business Service zusätzlich oder anstelle des im Hauptteil dieses Dokumentes definierten Security-Context-Token (SCT) einen ISTS Security Token akzeptieren. Im Fall der Verwendung des ISTS gelten die im Anhang aufgeführten Ergänzungen.

Realisiert ein Service Provider im Kontext der BiPRO eine Authentifizierung für seine Services, so MÜSSEN die nachfolgend definierten Mechanismen zur Authentifizierung verwendet werden.

Der Service Provider MUSS für jeden Service dem Service Consumer die End Point References der zuständigen Security-Token-Services (STSs) (siehe späteres Kapitel) mittels einer in der WSDL-Datei publizierten WS-SecurityPolicy beschreiben. Als Standard für die BiPRO werden zunächst die folgenden Authentifizierungsarten unterstützt.

Authentifizierungsarten (Credentials)

- Benutzerkennung, Passwort
- Benutzerkennung, Passwort, OTP (One Time Password)
- VDG-Ticket
- Zertifikat (x509 Token)

Es MUSS eine sessionorientierte Authentifizierung implementiert werden. Die Authentifizierung bei einem Service Provider MUSS zur Absicherung mehrerer Services mit einer Session-ID auf Ebene des SOAP Protokolls (ungleich SSL-Session) dienen können. Eine (transaktionsbasierte) Authentifzierung DARF NICHT direkt am Business Service erfolgen. Eine Identifizierung des Benutzers am Business Service MUSS über eine Session-ID

erfolgen. Diese Session-ID (wird in WS-Trust und WS-SecureConversation als spezieller Security-Token dargestellt, der als Security-Context-Token bezeichnet wird) MUSS vor der erstmaligen Nutzung der Services eines Providers zunächst über einen eigenständigen Security-Token-Service nach WS-Trust des Providers angefordert werden (siehe unten). Die Session-ID wird vom Provider nicht beendet, sondern MUSS für die von ihm angegebene Zeit gehalten werden. Im Folgenden werden die Begriffe Session-ID und Security-Context-Token (SCT) synonym verwendet.

Da ein Service Provider für seine unterschiedlichen Services möglicherweise verschiedene Authentifizierungsstärken verlangt, DARF es in diesem Fall verschiedene Security-Token-Services bei einem Service Provider geben. Jeder STS MUSS in seiner WSDL-Datei beschreiben, welche Authentifizierungsarten zugelassen sind. Jeder Business Service MUSS in seiner WSDL-Datei beschreiben, welche STS für ihn zugelassen sind.

Nach einer erfolgreichen Authentifizierung am STS des Providers ist bei weiteren SOAP-Requests des Consumers an die Business Services des Providers keine erneute Authentifizierung notwendig. Es MUSS dann nur noch das SCT beim Aufruf des Services mitgegeben werden. Für die Darstellung von Authentifizierungsinformationen in SOAP-Nachrichten MUSS die WS-Security Spezifikation von OASIS verwendet werden. Die Übertragung der Authentifizierungsinformation erfolgt im WS-Security-Element, das im Header einer SOAP-Nachricht übertragen wird.

Template Definition

Die Authentifizierung eines Benutzers erfolgt immer bei einem STS. Dazu stehen mehrere Authentifizierungsarten zur Verfügung.

Im weiteren Verlauf werden folgende Variablen für Templates verwendet:

Variable	Wert
${X1}	Namespace WS-Security: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
${X2}	Namespace BiPRO: http://www.bipro.net/namespace

${X2.1}	Namespace BiPRO Nachrichten: http://www.bipro.net/namespace/nachrichten
${X2.2}	Namespace BiPRO Tarifierung: http://www.bipro.net/namespace/versicherung/tarifierung
${X2.3}	Namespace BiPRO Datentypen: http://www.bipro.net/namespace/datentypen
${X3}	Namespace SOAP-Envelope: http://schemas.xmlsoap.org/soap/envelope/
${X4}	Namespace WS-Trust: http://schemas.xmlsoap.org/ws/2005/02/trust
${X5}	Namespace WS-SecureConversation: http://schemas.xmlsoap.org/ws/2005/02/sc
${X6}	Namespace WS-Security Utilities: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
${X7}	Identifier entsprechend der WS-SecureConversation Language, in Form einer BiPRO-Session-URI: z.B. bipro:sdgdfashd72364
${X8}	An XML Schema DateTime (mit der Möglichkeit weitere Attribute zu enthalten): wsu:AttributedDateTime
${X9}	Namespace WS-Security Extensions: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
${X10}	Benutzername, z. B. mustermann
${X11}	Passwort, z. B. abc123def$
${X12}	Einmalig gültiges, häufig temporär und zufällig generiertes Kennwort, das nach kurzer Zeit seine Gültigkeit verliert (One

	Time Password - OTP), z. B. `162832`
${X13}	Typ des BinärTokens, z. B. `bipro:VDGTicket` für eine starke Authentifizierung nach VDG
${X14}	Binär-Token, z. B. `gz4URcII8kk03FJaSSvt8do...`
${X15}	Namespace WS-Addressing: http://schemas.xmlsoap.org/ws/2004/08/addressing
${X16}	Namespace XML-Signature: http://www.w3.org/2000/09/xmldsig#
${X17}	Die BiPRO-Version für diesen Service (laut Norm 240)

Benutzerkennung und Passwort

Die Authentifizierung eines Benutzers beim STS erfolgt mit seinem Benutzernamen und seinem Passwort. Dafür wird das UsernameToken-Element der WS-Security-Spezifikation verwendet. Im UsernameToken-Element DARF das Passwort NICHT verschlüsselt werden. Es MUSS im Klartext übertragen werden, da sonst die Interoperabilität zwischen .Net und Java nicht gewährleistet werden kann. Ein Sicherheitsproblem stellt sich dadurch nicht, da die Verschlüsselung der Daten (und des Passwortes) auf dem Transportweg über SSL erfolgt.

```
<wsse:UsernameToken xmlns:wsse="${X9}">
        <wsse:Username>${X10}</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        username-token-profile-1.0#PasswordText">${X11}</wsse:Password>
</wsse:UsernameToken>
```

Benutzerkennung, Passwort und OTP

Die Authentifizierung eines Benutzers beim STS erfolgt mit seinem Benutzernamen, seinem Passwort und einem OTP (One-Time-Password, Einmalpasswort). Dafür wird ein UsernameToken verwendet, das als zusätzliches Claim das OTP enthält. Das Passwort MUSS im Klartext übertragen werden (siehe oben).

```
<wsse:UsernameToken xmlns:wsse="${X9}" xmlns:bipro="${X2}">
        <wsse:Username>${X10}</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        username-token-profile-1.0#PasswordText">${X11}</wsse:Password>
```

```
        <bipro:OTP>${X12}</bipro:OTP>
</wsse:UsernameToken>
```

Alternativ DARF ein Provider auch das statische Passwort und das One-Time-Passwort zusammenfassen und innerhalb des Passwort Claims übergeben.

Hinweis: Das Element bipro:OTP ist zwar dem Namensraum der BiPRO zugeordnet (`http://www.bipro.net/namespace`), nicht jedoch in den allgemeinen BiPRO XML Schemata enthalten. Dies stellt aus Sicht des technischen Ausschusses (TAUS) der BiPRO derzeit keine Restriktion bei der Arbeit mit einem OTP dar.

VDG-Ticket

Die Authentifizierung eines Benutzers beim STS eines Providers erfolgt mit dem VDG-Ticket. Für die Übertragung MUSS das BinarySecurityToken-Element der WS-Security-Spezifikation verwendet werden. Das VDG-Ticket für die Authentifizierung beim Service Provider MUSS der Service Consumer zuvor über einen (nicht standardisierten) Web Service vom Authentifizierungsservice des VDG (Identity Provider) (nach WS-Federation Identity Provider) abrufen. Nach dem OASIS Standard wird dieser Vorgang in WS-Federation spezifiziert. Diese Spezifikation wird hier zurzeit noch nicht angewendet.

```
<wsse:BinarySecurityToken xmlns:wsse="${X9}" xmlns:wsu="${X6}"
        EncodingType="wsse:Base64Binary" ValueType="${X13}">
        ${X14}
</wsse:BinarySecurityToken>
```

Zertifikat (x509-Token)

Ein Zertifikat stellt die Verbindung eines öffentlichen Schlüssels mit weiteren Daten (Aussteller, Besitzer, Gültigkeit, etc.) dar. Eine Authentifizierung ausschließlich über ein Zertifikat ist damit nicht möglich. Stattdessen erfolgt die Prüfung der Identität indirekt mit Hilfe einer digitalen Signatur, die anhand des Zertifikats verifiziert wird. Oft wird dazu ein Handshake benutzt, bei dem ein verschlüsseltes Geheimnis in mehreren Schritten ausgetauscht wird (z. B. bei SSL). Dies würde für BiPRO allerdings bedeuten, dass mehrere Aufrufe für die Authentifizierung nötig sind.

Daher wird der einfache Weg genutzt, die Daten im SOAP-Body zu signieren und diese Signatur zu prüfen. Um bei diesem Vorgang Replay-Attacken zu verhindern, wird zusätzlich ein Timestamp in die Signatur eingeschlossen. Der Timestamp MUSS vom Consumer grundsätzlich im Format `wsu:Timestamp` erstellt werden. Der Provider MUSS Anfragen

desselben Users mit dem gleichen Timestamp unterbinden. Diese Regelung führt nicht zu einer Behinderung des Consumers, da dieser Session-Tokens im regulären Betrieb nicht mehrmals innerhalb einer Sekunde anfordert. Außerdem DARF der Provider Anfragen mit einem veralteten Timestamp NICHT bearbeiten. Dabei ist zu beachten, dass die Wahrscheinlichkeit eines erfolgreichen Angriffs steigt, je länger die erlaubte Zeitspanne ist, die Wahrscheinlichkeit für Fehlverhalten (bedingt durch eine nicht exakte Synchronisierung zwischen Consumer und Provider) jedoch mit kürzeren Zeitspannen steigt. Zu empfehlen ist eine Spanne in der Größenordnung von einigen Minuten. Der Provider MUSS im Fall eines durch die Zeit bedingten Fehlers einen Hinweis auf die Ursache in der Fehlermeldung aufnehmen.

Im Header der SOAP-Nachricht MUSS neben dem BinarySecurityToken für das Zertifikat auch eine gültige Signatur über den SOAP-Body, das Zertifikat und der Timestamp übertragen werden:

```
<wsse:Security xmlns:wsu="${X6}" >
	<wsu:Timestamp wsu:Id="timestamp">
		<wsu:Created>yyyy-mm-ddThh:mm:ssZ</wsu:Created>
		<wsu:Expires>yyyy-mm-ddThh:mm:ssZ</wsu:Expires>
	</wsu:Timestamp>
	<wsse:BinarySecurityToken wsu:Id="binarytoken"
		ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
		EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">
		${X14}
	</wsse:BinarySecurityToken>
	<ds:Signature xmlns:ds="${X16}" Id="signature">
		<ds:SignedInfo>
			<ds:CanonicalizationMethod
		Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
			</ds:CanonicalizationMethod>
			<ds:SignatureMethod
		Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
			</ds:SignatureMethod>
			<ds:Reference URI="#body">
			<ds:Transforms>
			<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
			</ds:Transform>
			</ds:Transforms>
			<ds:DigestMethod
		Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
```

```
                        </ds:DigestMethod>
                        <ds:DigestValue>${X17}</ds:DigestValue>
                        </ds:Reference>
                        <ds:Reference URI="#binarytoken">
                        <ds:Transforms>
                        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
                c14n#">
                        </ds:Transform>
                        </ds:Transforms>
                        <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
                        </ds:DigestMethod>
                        <ds:DigestValue>${X18}</ds:DigestValue>
                        </ds:Reference>
                        <ds:Reference URI="#timestamp">
                        <ds:Transforms>
                        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
                c14n#">
                        </ds:Transform>
                        </ds:Transforms>
                        <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
                        </ds:DigestMethod>
                        <ds:DigestValue>${X19}</ds:DigestValue>
                        </ds:Reference>
                </ds:SignedInfo>
                <ds:KeyInfo>
                        <wsse:SecurityTokenReference>
                                <wsse:Reference URI="#binarytoken" />
                        </wsse:SecurityTokenReference>
                </ds:KeyInfo>
                <ds:SignatureValue>${X14}</ds:SignatureValue>
        </ds:Signature>
</wsse:Security>
```

Damit der Bezug auf den Body der SOAP-Nachricht korrekt dargestellt wird, MUSS ferner der Body folgende Id enthalten:

```
<soap:Body wsu:Id="body" xmlns:wsu="${X6}" …>
…
</soap:Body>
```

Security-Token-Service

Jeder Service Provider MUSS zur Authentifizierung seiner Benutzer mindestens einen eigenständigen Security-Token-Service (STS) anbieten, sofern eine Anmeldung am Business Service erforderlich ist. Ein STS führt für alle Business Services, die den STS in ihrer WSDL-

Beschreibung referenzieren, die Authentifizierung der Benutzer durch und stellt nach erfolgreicher Authentifizierung ein Security-Context-Token (SCT) aus. Die Authentifizierung eines Benutzers beim STS MUSS mit einer oder mehreren der obigen Authentifizierungsarten erfolgen. Die WS-SecurityPolicy des STS DARF die zulässigen Verfahren auf eine Teilmenge einschränken. Dabei ist zu beachten, dass ein einzelner STS genau ein Sicherheits-Level darstellt, auch wenn er mehrere Authentifizierungsarten anbietet. Diese werden dann vom Provider als gleichwertig angesehen. Dementsprechend ist es nicht notwendig, dem STS in einer Anfrage mitzuteilen, für welchen Business-Service ein SCT ausgestellt werden soll.

Die Übertragung der Authentifizierungsinformationen erfolgt wie oben zu den einzelnen Methoden definiert innerhalb des SOAP-Headers.

Falls ein Service Provider für seine Services unterschiedliche Authentifizierungsarten verlangt, MUSS er dabei folgende Punkte beachten:

- Für jeden Service, der eine Authentifizierung erfordert MUSS in der WSDL mindestens ein STS spezifiziert sein.
- Jeder STS MUSS in seiner WSDL per Security-Policy spezifizieren, welche Authentifizierungsarten möglich sind.
- Die von einem STS ausgestellten SCTs MÜSSEN mindestens bei allen Services gültig sein, die den STS in ihrer WSDL auflisten.
- Beim Aufruf eines Business Services mit einem gültigen SCT, das jedoch von einem nicht erlaubten STS ausgestellt wurde, MUSS eine entsprechende Standard-Fehlermeldung (siehe unten) generiert werden.
- Ein Upgrade eines bereits erstellten SCT auf eine höhere Authentifizierungsstufe ist nicht möglich. Die Sitzung hat einen technischen Charakter und darf nicht mit langlaufenden, asynchronen fachlichen Prozessen vermischt werden. Außerdem ermöglich das Aufwerten der Authentifizierung für ein und dieselbe Sitzung Angriffe per „Session-Fixation“ und ist daher aus Sicherheitsaspekten nicht sinnvoll.

Das vom STS zurückgelieferte SCT MUSS beim Aufruf von Funktionen (z. B. getQuote) im SOAP-Header des Request übertragen werden.

BiPRO-konforme Services MÜSSEN den zuvor vom STS erhaltenen SCT verstehen.

Die technische Spezifikation der Schnittstelle des STS erfolgt in Norm 410.

Security-Context-Token

Der STS benötigt die Authentifizierung, um ein `SecurityToken` zu generieren und zurückzuliefern. Dies ist als ein Login zu verstehen, das ein Security-Context-Token (SCT) ausliefert. Zur Authentifizierung MUSS die Methode `RequestSecurityToken` verwendet werden. Sowohl die Anfragen des Tokens als auch die Rückgabe werden im Body der SOAP-Nachricht transportiert.

Der erhaltene SCT wird bei den folgenden Aufrufen der Services (z. B. `HausratTarifierung`) im SOAP-Header der Nachricht übertragen.

Der Identifier des SCT MUSS gemäß der WS-SecureConversation Spezifikation eine URI sein. Im BiPRO-Umfeld MUSS dazu eine allgemeine (nicht hiearchische URI) mit dem Schema „`bipro:`“ verwendet werden. Das Token MUSS damit die Form `bipro:xxx` haben, wobei `xxx` eine beliebige alphanumerische Zeichenfolge darstellt, die zur Identifikation der Sicherheits-Session genutzt wird.

Es folgen Beispiele für einen Request und einen Response sowie das Entwerten eines SCT am Security-Token-Service. Danach wird gezeigt wie der erhaltene SCT in anderen Business Services verwendet wird.

Request

Fall A: Es wird ein SCT über Username-Passwort angefordert:

Der Client sendet eine RequestSecurityToken (RST) Nachricht mit der BiPROVersion, dem Request- und dem TokenType an den STS.

```
<soap:Envelope xmlns:soap="${X3}">
  <soap:Header>
    <wsse:Security xmlns:wsse="${X1}">
      <wsse:UsernameToken xmlns:wsse="${X1}" xmlns:bipro="${X2}">
        <wsse:Username>${X10}</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">${X11}</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken xmlns:wst="${X4}">
      <wst:TokenType>http://schemas.xmlsoap.org/ws/2005/02/sc/sct</wst:TokenType>

```

```
<wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</wst:RequestType>
      <allgemein:BiPROVersion xmlns:nachr="${X2.1}">${X17}</nachr:BiPROVersion>
    </wst:RequestSecurityToken>
  <soap:Body>
<soap:Envelope>
```

Fall B: Es wird ein SCT über ein VDG-Ticket angefordert:

```
<soap:Envelope xmlns:soap="${X3}">
  <soap:Header>

    <wsse:security xmlns:wsse="${X1}">
      <wsse:BinarySecurityToken
         xmlns:wsse="${X1}"
         EncodingType="wsse:Base64Binary"
         ValueType="bipro:VDGTicket">${X14}
      </wsse:BinarySecurityToken>
    </wsse:security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken xmlns:wst="${X4}">
      <wst:TokenType>http://schemas.xmlsoap.org/ws/2005/02/sc/sct</wst:TokenType>
      <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</wst:RequestType>
      <nachr:BiPROVersion xmlns:nachr="${X2.1}">${X17}</nachr:BiPROVersion>
    </wst:RequestSecurityToken>
  </soap:Body>
</soap:Envelope>
```

Response

Das angeforderte SCT wird zurückgegeben. Der STS sendet nach erfolgreicher Authentifizierung eine RequestSecurityTokenResponse (RSTR) Nachricht mit der BiPRO-Version und dem SCT an den Client.

Das SCT wird mit einer Information zu seiner Lebenszeit versehen. Die Festlegung der Lebenszeit obliegt alleine dem Service Provider, MUSS jedoch berücksichtigen, dass bestimmte Authentifizierungsvarianten eine erneute Authentifizierung erst nach Ablauf einer Wartezeit (z. B. von 60 Sekunden) erlauben.

```
<soap:Envelope xmlns:soap="${X3}">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityTokenResponse xmlns:wst="${X4}">
      <wst:TokenType>http://schemas.xmlsoap.org/ws/2005/02/sc/sct</wst:TokenType>
      <wst:Lifetime xmlns:wsu="${X6}">
```

```
          <wsu:Created>${X8}</wsu:Created>
          <wsu:Expires>${X8}</wsu:Expires>
        </wst:Lifetime>
        <wst:RequestedSecurityToken xmlns:wsc="${X5}">
          <wsc:SecurityContextToken>
            <wsc:Identifier>${X7}</wsc:Identifier>
          </wsc:SecurityContextToken>
        </wst:RequestedSecurityToken>
        <nachr:BiPROVersion xmlns:nachr="${X2.1}">${X17}</nachr:BiPROVersion>
      </wst:RequestSecurityTokenResponse>
    </soap:Body>
  </soap:Envelope>
```

Entwertung eines Kontextes

Bei der Entwertung wird das SCT sowohl im Header (Funktion Authentifizierung) als auch im Body (Funktion Referenz) übertragen.

```
<soap:Envelope xmlns:soap="${X3}" xmlns:wsse="${X9}">
  <soap:Header>

    <wsse:Security>
      <wsc:SecurityContextToken xmlns:wsc="${X5}" xmlns:wsu="${X6}" wsu:Id="sct">
        <wsc:Identifier>${X7}<wsc:Identifier>
      </wsc:SecurityContextToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken xmlns:wst="${X4}">
      <wst:TokenType>http://schemas.xmlsoap.org/ws/2005/02/sc/sct</wst:TokenType>
      <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Cancel</wst:RequestType>
      <wst:CancelTarget>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#sct"/>
        </wsse:SecurityTokenReference>
      </wst:CancelTarget>
      <nachr:BiPROVersion xmlns:nachr="${X2.1}">${X17}</nachr:BiPROVersion>
    </wst:RequestSecurityToken>
  </soap:Body>
</soap:Envelope>
```

Die Entwertung wird mit folgender Response-Nachricht bestätigt:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <wst:RequestSecurityTokenResponse
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
         <wst:RequestedTokenCancelled/>
```

```
        <nachr:BiPROVersion
xmlns:nachr="http://www.bipro.net/namespace/nachrichten">2.1.0.1.0</nachr:BiPROVersion>
      </wst:RequestSecurityTokenResponse>
   </soap:Body>
</soap:Envelope>
```

Beispiel eines Serviceaufrufes

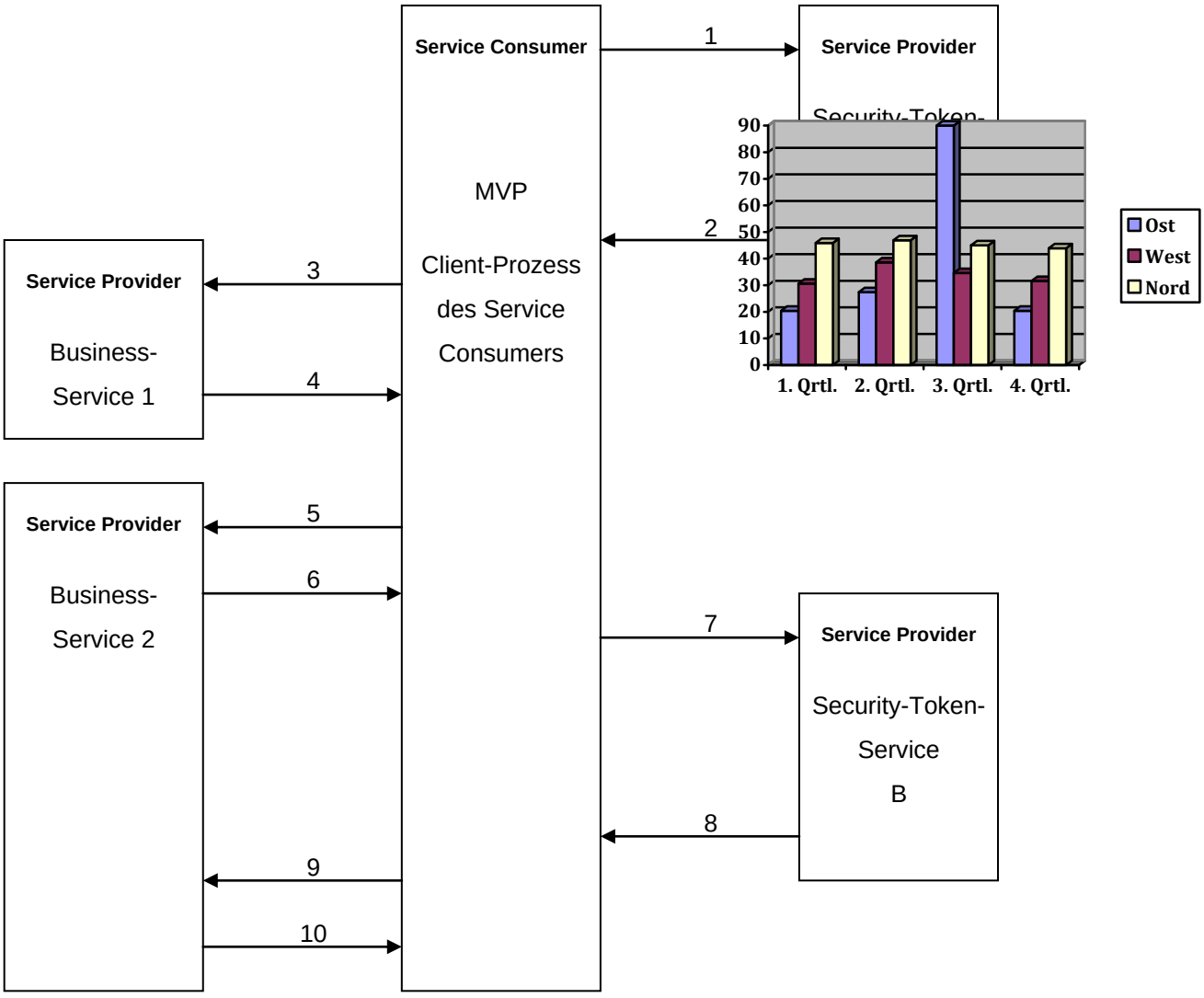
Innerhalb des Security-Teils des SOAP-Headers wird nun bei den Aufrufen der Services der Identifier des SCT gesendet.

```
<soap:Envelope xmlns:soap="${X3}" xmlns:wsse="${X9}>
       <soap:Header>
              <wsse:Security>
                     <wsc:SecurityContextToken xmlns:wsc="${X5}">
                            <wsc:Identifier>${X7}<wsc:Identifier>
                     </wsc:SecurityContextToken>
              </wsse:Security>
       </soap:Header>
       <soap:Body>
              <tarif:getQuote xmlns:tarif="${X2.2}">
                     <tarif:Request>
                            <tarif:Tarifierung>...</tarif:Tarifierung
                     </tarif:Request>
              </tarif:getQuote>
       </soap:Body>
</soap:Envelope>
```

Authentifizierungsprozess

Authentifizierung beim Service Provider

In den folgenden Diagrammen wird der Authentifizierungsprozess eines Service Consumers bei einem Service Provider zusammengefasst dargestellt.



507

508 1. Client möchte den Business-Service 1 nutzen und ruft den STS A auf, der in der
509 WSDL-Beschreibung des Business-Service 1 angegeben ist. In der WSDL-
510 Beschreibung des STS A findet der Client die möglichen Authentifizierungsarten.

511 2. Der STS A liefert nach erfolgreicher Authentifizierung das SCT zurück.

512 3. Der Client ruft den Business-Service 1 mit dem erhaltenen SCT auf.

513 4. Der Business-Service 1 prüft den Security Token und liefert den Response.

514 5. Client ruft den Business-Service 2 mit keinem oder einem ungültigen Security Token
515 auf.

516 6. Der Business-Service 2 prüft das SCT und liefert als Response eine Fehlermeldung

aus. Aus der WSDL des Services enthält die End Point Reference auf den oder die zuständigen STSs für den Business-Service 2 (im Beispiel B)

7. Der Client ruft den STS B auf, der durch die in der Fehlermeldung enthaltenen End Point Reference angegeben ist. In der WSDL-Beschreibung des STS B findet der Client die möglichen Authentifizierungsarten.

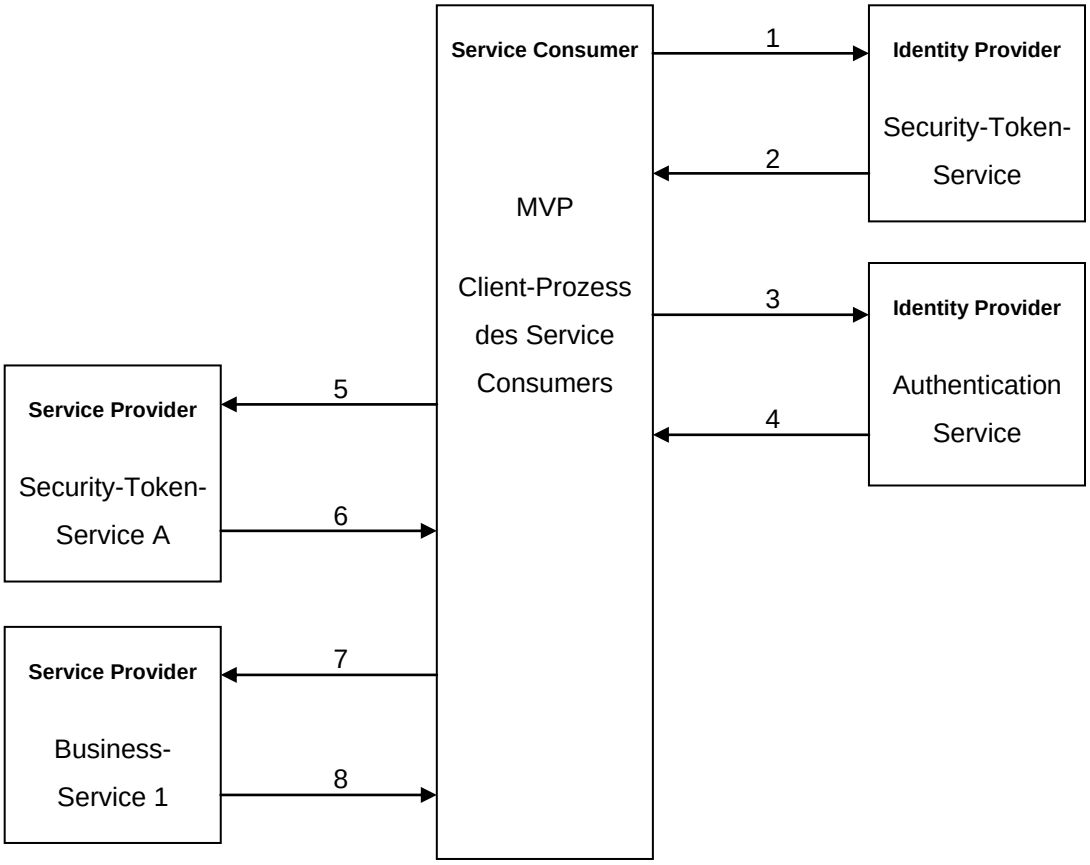
8. Der STS B liefert nach erfolgreicher Authentifizierung das SCT zurück.

9. Der Client ruft den Business-Service 2 mit dem erhaltenen SCT auf.

10. Der Business-Service 2 prüft das SCT und liefert den Response.

Authentifizierung durch einen Identity Provider

Im folgenden Diagramm wird der indirekte Authentifizierungsprozess eines Service Consumers bei einem Service Provider über einen externen Authentifizierungs-Service eines Identity Providers (Ticket-Service) dargestellt. Dieses Modell entspricht der OASIS Spezifkation WS-Federation.



1. Client möchte den Business-Service 1 nutzen und ruft dazu den STS des zuständigen Identity Providers auf, um sich dort zu authentifizieren. Der Consumer nutzt dazu folgende Informationen.

   - Aus der WSDL-Beschreibung des Business-Service 1 erhält er die End Point Reference des zuständigen STS A des Providers.
   - Aus der WSDL-Beschreibung des zuständigen STS A des Providers erhält er die Information über die Authentifizierungsart (z.B. VDG-Ticket) mit einer End Point Reference des Authentication Services des zuständigen Identity Providers (z.B. VDG).
   - Aus der WSDL-Beschreibung des zuständigen Authentication Service des Identity Providers erhält er die End Point Reference des zuständigen STS des Identity Providers.
   - Aus der WSDL-Beschreibung des zuständigen STS des Identity Providers Authentication erhält er die Authentifizierungsart, mit der der Client sich beim Identity Provider authentifizieren muss.

2. Der STS des Identity Providers liefert nach erfolgreicher Authentifizierung das SCT für den Authentication Service des Identity Providers zurück.

3. Der Client ruft mit dem erhaltenen SCT den Authentication Service des Identity Providers auf und fordert einen Security Token (VDG-Ticket) für den STS A des Service Providers an.

4. Der Authentication Service des Identity Providers prüft den Security Token des STS des Identity Providers und stellt ein Security Token (VDG-Ticket) für den STS A des Service Providers aus.

5. Der Client ruft den STS A des Service Providers auf und authentifiziert sich dort mit dem erhaltenen Security Token (VDG-Ticket) des Identity Providers

6. Der STS A des Service Providers liefert nach erfolgreicher Authentifizierung das SCT für den Business-Service 1 des Service Providers zurück.

7. Der Client ruft den Business-Service 1 mit dem erhaltenen SCT auf.

8. Der Business-Service 1 prüft das SCT und liefert den Response.

Gültigkeit von SCTs für verschiedene Services

Der parallele Einsatz von unterschiedlichen Authentifizierungsverfahren durch einen oder mehrere STS-Instanzen ist grundsätzlich unproblematisch. Allerdings bieten diese nicht immer dasselbe Sicherheitsniveau. So ist beispielsweise die Authentifizierung mit Benutzername / Passwort und One Time Password i.d.R sicherer als nur mit Benutzername / Passwort. Um einfacher unterscheiden zu können, welcher Token von welchem STS ausgestellt wurde, DARF der SecurityContextToken um ein eindeutiges TokenPrefix ergänzt werden. In diesem Fall hat der Token dann das Format „`bipro:<tokenPrefix>.<alphaNumericNumbers>`“. Natürlich MUSS der Service weiterhin beim STS anfragen, ob der Token gültig ist. Durch diesen Mechanismus können aber schon vorher die nicht akzeptierten Token ausgefiltert werden. Zusätzlich ermöglicht diese Logik die zentrale Tokenverwaltung mehrerer STS in einem gemeinsamen Pool, in dem alle aktuell gültigen Token verwaltet werden.

Hinweis: Dieses Verfahren ist ein Ersatz für den in WS-Trust spezifizierten Einsatz von `/wst:RequestSecurityToken/wsp:AppliesTo`. Bei diesem weiß der STS, für welche Services das ausgegebene Token verwendet werden darf. Mit dem hier beschriebenen vereinfachten Verfahren wissen die Services, welchen STS und TokenPrefixes sie vertrauen.

Spezifikation der Fehlermeldungen

Im Bereich der Sicherheit gibt es nur wenige verschiedene Fehler, die auftreten können. Dies ist nicht zu verwechseln mit der Tatsache, dass den Fehlern verschiedene Ursachen zugrunde liegen können. Dieser Situation wird Rechnung getragen, in dem in jeder Fehlermeldung genau ein Fehlertext ${1} bis ${4} verwendet werden MUSS. Optional DÜRFEN weitere Hinweise zur Ursache integriert werden.

Variable	Wert
${X1}	Technischer Fehler - Authentifizierungsdaten fehlerhaft
${X2}	Technischer Fehler - Serviceaufruf fehlerhaft
${X3}	Technischer Fehler - Service temporär nicht verfügbar
${X4}	Security Fehler - Authentifizierungsdaten ungültig

Entsprechend der Norm zu Fehlermeldungen (siehe Norm 250) DÜRFEN Meldungen auf Applikationsebene (also oberhalb der XML-Schema-Validierung) NICHT als SOAP-Faults

behandelt werden. Entsprechend MÜSSEN Security-Fehler, die an einem Business Service (z. B. Tarifierung, Angebot, Antrag) eintreten, - sofern nicht ausdrücklich im Business Service anders vorgesehen - im Rahmen einer validen Response-Nachricht inkl. Status- und Meldungsobjekten an den Service Consumer ausgeliefert werden. Das Fehlermanagement des STS ist jedoch ausdrücklich ausgenommen von der Regelung gemäß Norm 250. Ein STS MUSS Fehlermeldungen im Rahmen eines SOAP-Faults abbilden. Der SOAP-Fault MUSS im `faultstring` einen der vier Fehlertexte ${1} bis ${4} tragen, MUSS im `faultcode` einen durch WS-Trust, WS-SecureConversation oder SOAP 1.1 definierten Wert (s. u.) enthalten und MUSS im Abschnitt `detail` ein BiPRO Status-Objekt beinhalten.

Für das BiPRO Status-Objekt gelten folgende Punkte:

- Die `StatusID` lautet `NOK`
- Es MUSS genau ein Meldungs-Objekt der `ArtId=Fehler` enthalten sein
- Optional DÜRFEN weitere Meldung-Objekte der `ArtId=Hinweis` zur Erläuterung der Fehlerursache übermittelt werden.

Die Attribute `MeldungId` und `Text` der Meldungs-Objekte sind im Folgenden tabellarisch aufgeführt. Im Rahmen der BiPRO steht für die `MeldungId` der Zahlenbereich von `00900` bis `00999` zur Verfügung. Aus Sicherheitsgründen wird bewusst auf eine detailierte Beschreibung im Text verzichtet. Dies führt nicht zu einer Einschränkung der Handhabbarkeit, da die Meldungen über die `MeldungId` verschlüsselt sind.

Die Meldungen beim Aufruf des STS und von Business Services im Detail lauten:

ArtId	MeldungId	Meldungs-Text	Faultcode im SOAP-Fault	Erläuterung
1	00900	${X1}	`wst:InvalidRequest` oder einer der Faultcodes der entsprechenden zusätzlichen Meldung	Sicherheitsinformationen im Header nicht vorhanden oder im falschen Format. Ursache beim Consumer. Diese Meldung kann durch Hinweise mit einer MeldungId 00901-00920 ergänzt werden.

3	00901	Ursache	`wst:InvalidRequest`	SOAP-Header nicht ermittelbar
3	00905	Ursache	`wst:AuthenticationBadElements`	Kein WS-Security-Header vorhanden
3	00906	Ursache	`wst:AuthenticationBadElements`	Leerer WS-Security-Header
3	00907	Ursache	`wst:AuthenticationBadElements`	Unzulässiger WS-Security-Header
3	00908	Ursache	`wst:AuthenticationBadElements`	Kein `wsse:UsernameToken` vorhanden
3	00909	Ursache	`wst:AuthenticationBadElements`	Kein SecurityContext-Identifier vorhanden
3	00910	Ursache	`wst:BadRequest`	`wst:TokenType` nicht korrekt oder nicht vorhanden
3	00911	Ursache	`wst:BadRequest`	`wst:RequestType` nicht korrekt oder nicht vorhanden
3	00912	Ursache	`wst:BadRequest`	`wst:CancelTarget` nicht vorhanden
3	00913	Ursache	`wst:BadRequest`	`wsse:SecurityTokenReference` nicht vorhanden
3	00914	Ursache	`wst:BadRequest`	`wsse:Reference` nicht vorhanden
3	00915	Ursache	`wst:BadRequest`	`wsse:Reference` ohne URI-Attribut
3	00916	Ursache	`wst:BadRequest`	SCT im Header stimmt nicht mit Referenz im Body überein.
3	00917	Ursache	`wst:AuthenticationBadElements`	WS-Security-Header enthält kein Security-ContextToken

3	00918	Ursache	wst:AuthenticationBadElements	Kein SecurityContext-Identifier vorhanden
3	00919	Ursache	wst:FailedAuthentication	Kein korrekter SecurityContext-Identifier
3	00920	Ursache	soap:Client	VDG: Keine Berechtigung für die Web-Anwendung des angegebenen Mandanten
3	00921	Ursache	wst:BadRequest	Timestamp im <wsse:Security> Element fehlt
3	00922	Ursache	wst:BadRequest	Zertifikat kann nicht aus <wsse:BinarySecurityToken> entnommen werden
3	00923	Ursache	wst:BadRequest	Signatur des Timestamp im <wsse:Security> Element fehlt
3	00924	Ursache	wst:BadRequest	Signatur des <wsse:BinarySecurityToken> mit Zertifikat fehlt
3	00925	Ursache	wst:BadRequest	Signatur des <soap:body> Elements fehlt
1	00930	${X2}	soap:Client oder einer der Faultcodes der entsprechenden zusätzlichen Meldung	Der Service kann nicht aufgerufen werden. Mögliche Ursachen sind z.B. eine fehlgeschlagene Validierung oder der Aufruf eines ungültigen Services. Diese Meldung kann durch Hinweise mit einer MeldungId 00931-00932 ergänzt werden.

3	00931	Ursache	`soap:Client`	Validierung des Bodys fehlgeschlagen
3	00932	Ursache	`soap:Server`	Fehler bei der Verarbeitung (Technischer Fehler der bei Service-Abarbeitung auftritt)
1	00940	${X3}	`soap:Server` oder einer der Faultcodes der entsprechenden zusätzlichen Meldung	Vorübergehende technische Störung auf Seiten des Providers. Diese Meldung kann durch Hinweise mit einer MeldungId 00941-00949 ergänzt werden.
3	00941	Ursache	`wst:RequestFailed`	Fehler bei der Erstellung der RequestSecurityTokenResponse
3	00942	Ursache	`wst:FailedAuthentication`	VDG: Zum Benutzer ist beim VDG kein Auth.-Verfahren hinterlegt.
3	00943	Ursache	`soap:Server`	VDG: Technischer Fehler beim Zugriff auf die Mandantendaten
3	00944	Ursache	`wst:FailedAuthentication`	VDG: Auth.-Verfahren weder secovid, securid noch schwach
3	00945	Ursache	`wst:FailedAuthentication`	VDG: Für den Benutzer ist auf dem Server kein Passwort hinterlegt → Vergabe eines Passworts durch den Benutzer über die Webapp. per OTP erforderlich
3	00946	Ursache	`soap:Server`	VDG: Fehler bei der Initialisierung des TicketIssuers

3	00947	Ursache	`soap:Server`	VDG: Es kann kein Ticket erstellt werden.
3	00948	Ursache	`soap:Server`	VDG: Bei der Ticket-Herausgabe ist ein Fehler aufgetreten.
3	00949	Ursache	`soap:Server`	VDG: Fehler beim Zugriff auf die VDG-Datenbanken (i.d.R. technischer Fehler)
1	00960	${X4}	`wst:FailedAuthentication` oder einer der Faultcodes der entsprechenden zusätzlichen Meldung	Der Benutzer kann mit den übergebenen Credentials nicht authentifiziert werden. Diverse Ursachen: z.B. falsche Kennung, Passwort oder OTP, ungültiges SCT. Diese Meldung kann durch Hinweise mit einer MeldungId 00961-00969 ergänzt werden.
3	00961	Ursache	`wst:FailedAuthentication`	Credentials (Username, Paßwort, OTP) ungültig
3	00962	Ursache	`wst:InvalidSecurityToken`	Session (SCT) ungültig
3	00963	Ursache	`wst:FailedAuthentication`	Timestamp Signatur ungültig
3	00964	Ursache	`wst:FailedAuthentication`	Account gesperrt
3	00965	Ursache	`wst:FailedAuthentication`	Zu viele Fehlversuche, Account ist ab jetzt gesperrt
3	00966	Ursache	`wst:FailedAuthentication`	Passwort abgelaufen
3	00967	Ursache	`wst:FailedAuthentication`	Timestamp im <wsse:Security> Element ist nicht mehr gülitg (expired)
3	00968	Ursache	`wst:FailedAuthentication`	Signatur des <soap:body>

				Elements ungültig
3	00969	Ursache	wst:FailedAuthentication	Signatur des <wsse:BinarySecurityToken> mit Zertifikat ungültig

soap:Fault MUSS das einzige Kindelement unter soap:Body sein.

Beispiel:

```
<soap:Fault xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nachr="http://www.bipro.net/namespace/nachrichten">
	<faultcode>soap:Client</faultcode>
	<faultstring>Technischer Fehler - Anmeldung kann nicht durchgeführt
werden</faultstring>
	<faultactor>http://example.org/someactor</faultactor>
	<detail>
		<nachr:BiproException>
		<nachr:BiPROVersion>2.1.0.1.0</nachr:BiPROVersion>
		<nachr:Status>
			<nachr:StatusId>NOK<nachr:StatusId>
			<nachr:Meldung>
				<nachr:ArtId>Fehler<nachr:ArtId>
				<nachr:MeldungId>00900</nachr:MeldungId>
				<nachr:Text>Technischer Fehler - Authentifizierungsdaten
fehlerhaft</nachr:Text>
			</nachr:Meldung>
			<nachr:Meldung>
				<nachr:ArtId>Hinweis<nachr:ArtId>
				<nachr:MeldungId>00901</nachr:MeldungId>
				<nachr:Text>Die Ursache ...</nachr:Text>
			</nachr:Meldung>
		</nachr:Status>
		</nachr:BiproException>
	</detail>
</soap:Fault>
```

Allgemeine Hinweise zur Implementierung

Mittels der BiPRO Services werden fachliche Dienste (Business Services) online über das Internet zur Verfügung gestellt. Dies ist in sofern ein Novum, dass es einen relativ direkten Zugriff auf zentrale Systeme der Provider bedeutet. Im Gegensatz zu Webapplikationen existiert z. B. keine getrennte Präsentationsschicht mehr, die die Kommunikation durch einen

Protokollbruch absichert. Bei der Implementierung der Services sollten daher grundlegende Hinweise zur Sicherheit beachtet werden, um die Risiken eines Angriffs zu minimieren.

Validierung

Grundsätzlich MÜSSEN alle Eingaben validiert werden bevor irgendeine Art der Verarbeitung durchgeführt wird. Dabei MUSS beachtet werden, dass einerseits die XML-Schemata der BiPRO dazu nicht immer ausreichend sind und dass andererseits aktuelle Frameworks (Axis, .Net) eingehende SOAP-Nachrichten nicht standardmäßig gegen die referenzierten Schemata validieren. Sinnvoll ist es daher, separat eine formale Validierung zu realisieren, die vor dem Zugriff auf zentrale Systeme als Schutz positioniert wird. Die wird in den BiPRO-Normen an anderer Stelle vorgeschrieben.

Subsysteme

Formal sinnvolle Eingaben können immer noch zu einer Gefährdung angeschlossener Subsysteme führen, sofern sie potentielle Metazeichen der Subsysteme enthalten. Bekanntes Beispiel sind z. B. Namensfelder, die Hochkommata erlauben und deren Inhalt an eine Datenbank 1:1 weitergereicht wird. In diesem Fall wäre eine SQL-Injection möglich. Schutz vor dieser Art von Angriffen erreicht man, in dem im ersten Schritt alle Subsysteme identifiziert werden (Datenbanken, Ausgabesysteme, Betriebssysteme, etc.) und im zweiten Schritt alle Metazeichen in der Kommunikation mit diesen Systemen konsequent durch die entsprechenden Escape-Sequenzen ersetzt werden.

Sitzungen

Die im Rahmen dieser Spezifikation definierten Sitzungen berücksichtigen bereits wesentliche Sicherheitsaspekte. Sie werden erst nach einer Authentifizierung ausgestellt, laufen nach einer definierten Periode ab und werden nur verschlüsselt übertragen. Wichtig ist außerdem die Verwendung eines guten Zufallsalgorithmus für die Generierung der Ids im SCT. Ferner sollte sichergestellt sein, dass SCTs nur für die Business Services verwendet werden können, die den austellenden STS in ihrer Policy referenzieren.

Sicherheitspolicies

Zur Umsetzung der in dieser Norm beschriebenen Sicherheitsmechanismen MÜSSEN im WSDL-Dokument des Security-Token-Services und in der WSDL-Datei der fachlichen Services (Business Services) die folgenden Security Policies verwendet werden.

Security Policy des Security-Token-Service

Zur Spezifikation, welche Authentifizierungsverfahren erlaubt sind, um ein SecurityContext-Token beim STS eines spezifischen Providers zu beziehen, werden Security Policies eingesetzt. Die Security-Policy des STS MUSS unter Anwendung des folgenden Templates beschrieben und innerhalb der WSDL-Datei zum Service veröffentlicht werden.

Das folgende Template zeigt eine Policy, die alle vier verschiedenen Authentifizierungs-methoden (Username-Passwort, Username-Passwort-OTP, X509 Token und VDG-Ticket) unterstützt. Je nach Provider DÜRFEN dies auch weniger sein, es MUSS jedoch mindestens eine der möglichen Authentifizierungsarten angegeben werden.

```
  <!-- Policy fuer STS, der alle vier BiPRO Authentifizierungsverfahren unterstuetzt-->
   <wsp:Policy wsu:Id="BiPROAuthSecurityPolicy" xmlns:wsp="..." ...>
     <wsp:ExactlyOne>
       <wsp:All>
        <!-- Definition des Transportbindings als HTTPS -->
        <sp:TransportBinding>
          <wsp:Policy>
            <sp:TransportToken>
              <wsp:Policy>
                <sp:HttpsToken RequireClientCertificate="false"/>
              </wsp:Policy>
            </sp:TransportToken>
          </wsp:Policy>
        </sp:TransportBinding>
        <!-- Definition der moeglichen Tokens, je nach Anbieter auch nur einer
Teilmenge derselben-->
        <sp:SupportingTokens>
          <wsp:Policy>
            <wsp:ExactlyOne>
              <!-- Authentifizierung mit Username und Passwort -->
              <wsp:All>
                <sp:UsernameToken wsu:Id="BiPROBasicToken"/>
              </wsp:All>
              <!-- Authentifizierung mit Username, Passwort und Token -->
              <wsp:All>
```

```
                    <sp:UsernameToken wsu:Id="BiPROOTPToken"/>
                  </wsp:All>
                  <!-- Authentifizierung mit X509 Zertifikat -->
                  <wsp:All>
                    <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Alwa
ysToRecipient">
                      <wsp:Policy>
                        <sp:WssX509V3Token11/>
                      </wsp:Policy>
                    </sp:X509Token>
                  </wsp:All>
                  <!-- Authentifizierung mittels VDG Ticket -->
                  <wsp:All>
                    <sp:IssuedToken>
                      <sp:Issuer>
                        <wsa:Address>${X17}</wsa:Address>
                      </sp:Issuer>
                    </sp:IssuedToken>
                  </wsp:All>
                </wsp:ExactlyOne>
              </wsp:Policy>
            </sp:SupportingTokens>
          </wsp:All>
        </wsp:ExactlyOne>
      </wsp:Policy>
```

Die Unterscheidung, ob ein UsernameToken ein One-Time-Password enthalten soll, wird über das Attribut `wsu:Id` getroffen. Dieser Umweg ist derzeit nötig, da die Spezifikation WS-Security-Policy keine Methoden anbietet, explizit die geforderten Claims zu spezifizieren.

Variable	Wert
${X17}	URL des zugehörigen Providers im Fall eines VDG-Tickets

Die Referenzierung im binding Element erfolgt mit:

```
<wsdl:binding>
  <wsp:PolicyReference URI="#BiPROAuthSecurityPolicy" xmlns:wsp="..."/>
  <wsdl:operation .../>
</wsdl:binding>
```

Security Policy der Business Services

Die Sicherheitsanforderung des jeweiligen BiPRO-Services MUSS ebenfalls durch eine Policy definiert werden, die festlegt, dass ein SCT übertragen werden muss. Der SCT MUSS vom STS stammen. Die Angabe eines Issuers ist nötig, da ein Provider evtl. verschiedene STS zur Verfügung stellt.

```
<!-- Policy fuer Business Service -->
<wsp:Policy wsu:Id="BiPROAuthSecurityPolicy" xmlns:wsp="..." ...>
  <wsp:ExactlyOne>
    <wsp:All>
      <!-- Definition des Transportbindings als HTTPS -->
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken RequireClientCertificate="false"/>
            </wsp:Policy>
          </sp:TransportToken>
        </wsp:Policy>
      </sp:TransportBinding>
      <!-- Definition der Anforderung an ein SecurityContext Token -->
      <sp:SupportingTokens>
        <wsp:Policy>
          <sp:SecureConversationToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Alwa
ysToRecipient">
            <sp:Issuer>
              <wsa:Address>${X18}</wsa:Address>
            </sp:Issuer>
          </sp:SecureConversationToken>
        </wsp:Policy>
      </sp:SupportingTokens>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

${X18}	Adresse des STS, der das benötigte SecurityContext-Token austellt.

Die Referenzierung im binding Element erfolgt mit:

```
<wsdl:binding>
  <wsp:PolicyReference URI="#BiPROAuthSecurityPolicy" xmlns:wsp="..."/>
```

```
    <wsdl:operation .../>
</wsdl:binding>
```

Sofern eine Authentifzierung benötigt wird, MUSS diese Referenzierung der Security Policy am Business Service entweder auf Ebene des Service (unterhalb des wsdl:binding) oder auf Ebene der Methode (unterhalb von wsdl:operation) erfolgen.

Bei Referenzierung auf Ebene der Methode ist es möglich, spezifische Authentifizierungsverfahren für einzelne Methoden zu fordern.

Hinweis: Der Name der Policy (SecureConversationToken) ist missverständlich, da er die Anforderung nach einem „*SecureConversation*-Token“ suggeriert. Tatsächlich entspricht die Policy jedoch der Anforderung nach einem *SecurityContext*-Token unter Angabe des Issuers.

Anmerkung: WS-SecureConversation begreift ein SCT als einen Kontext, der Informationen zur Sicherung (Verschlüsselung) der Nachricht enthalten kann. Im Kontext der BiPRO wird das SCT lediglich als ein Identifier verwendet.

Anhang

ISTS Security Token

Im Rahmen der Trusted German Insurance Cloud (TGIC) wird durch den Insurance Security Token Service (ISTS) ein Security Token definiert, das auf dem OASIS-Standard SAML 2.0 basiert und in diesem Sinne zum Austausch von Authentifikationsinformationen genutzt wird.

Im Folgenden werden zu den entsprechenden Abschnitten im Hauptteil dieses Dokuments notwendige Ergänzungen definiert, damit ein Business Service auch dann BiPRO-konform ist, wenn er neben dem Security-Context-Token (SCT) auch einen ISTS Security Token zur Identifikation des aufrufenden Nutzers akzeptiert.

Die in den entsprechenden Abschnitten im Hauptteil dieses Dokuments für Templates bereits definierten Variablen werden unverändert übernommen. Neu eingeführte Variablen erhalten den Präfix „A“ und eine in diesem Anhang eindeutige Nummer.

Verwendete Standards

Zusätzlich zu den bisher im Dokument genannten Standards wird der folgende Standard vorausgesetzt:

- **SAML (Version 2.0) Definition eines Security-Token-Formats**

SAML (Security Assertion Markup Language) ist eine auf XML basierende Spezifikation von Datenstrukturen, die den Austausch von Authentifikations- und Autorisierungsinformationen ermöglicht.

Authentifikation

Die im Rahmen der sessionorientierten Authentifikation eingeführte Session-ID wird als spezieller Security Token nach WS-Trust dargestellt, der neben der Ausprägung eines Security-Context-Token (SCT) entsprechend WS-SecureConversation auch in der Form eines ISTS Security Token entsprechend SAML 2.0 genutzt werden DARF.

Security Policy der Business Services

Die Sicherheitsanforderung des jeweiligen BiPRO-Services MUSS durch eine Policy definiert werden, die festlegt, dass ein entweder ein SCT oder eine SAML-Assertion als Security Token übertragen werden muss.

```
  <!-- Policy fuer Business Service -->
  <wsp:Policy wsu:Id="BiPROAuthSecurityPolicy" xmlns:wsp="..." ...>
    <wsp:ExactlyOne>
      <wsp:All>
        <!-- Definition des Transportbindings als HTTPS -->
        <sp:TransportBinding>
          <wsp:Policy>
            <sp:TransportToken>
              <wsp:Policy>
                <sp:HttpsToken RequireClientCertificate="false"/>
              </wsp:Policy>
            </sp:TransportToken>
          </wsp:Policy>
        </sp:TransportBinding>
        <!-- Definition der Anforderung an ein Security Token -->
        <sp:SupportingTokens>
          <wsp:Policy>
            <!-- [ERGAENZUNG-START] -->
            <wsp:ExactlyOne>
            <!-- [ERGAENZUNG-ENDE] -->
              <sp:SecureConversationToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Alwa
ysToRecipient">
                <sp:Issuer>
                  <wsa:Address>${X18}</wsa:Address>
                </sp:Issuer>
              </sp:SecureConversationToken>
              <!-- [ERGAENZUNG-START] -->
              <sp:SamlToken
sp:IncludeToken=http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Alway
sToRecipient>
                  <sp:Issuer>
                      <wsa:Address>${A1}</wsa:Address>
                  </sp:Issuer>
                  <wsp:Policy>
                    <sp:WssSamlV20Token11 />
                  </wsp:Policy>
              </sp:SamlToken>
            </wsp:ExactlyOne>
            <!-- [ERGAENZUNG-ENDE] -->
          </wsp:Policy>
```

```
        </sp:SupportingTokens>
      </wsp:All>
    </wsp:ExactlyOne>
  </wsp:Policy>
```

Die beiden in der Policy durch die XML-Kommentare „[ERGAENZUNG-START]“ bzw. „[ERGAENZUNG-ENDE]“ eingefassten Bereiche ergänzen dabei die bisherige Spezifikation.

Variable	Wert
${X18}	Adresse des STS, der das benötigte Security-Context-Token ausstellt.
${A1}	Adresse des alternativen Insurance Security Token Service (ISTS), der ISTS Security Token auf Basis von SAML 2.0 ausstellt. Dieser ISTS muss nicht konform zur BiPRO-Norm 410 sein.

Beispiel eines Serviceaufrufes

Analog der Übertragung eines SCT innerhalb eines WS-Security-Elements im SOAP-Header wird dieses Vorgehen auch für die Übermittlung des ISTS Security Token vom Consumer an den Business Services genutzt.

```
<soap:Envelope xmlns:soap="${X3}" xmlns:wsse="${X9}">
	<soap:Header>
		<wsse:Security>${A2}</wsse:Security>
	</soap:Header>
	<soap:Body>
		<tarif:getQuote xmlns:tarif="${X2.2}">
			<tarif:Request>
				<tarif:Tarifierung>...</tarif:Tarifierung
			</tarif:Request>
		</tarif:getQuote>
	</soap:Body>
</soap:Envelope>
```

Variable	Wert
${X2.2}	Namespace BiPRO Tarifierung: http://www.bipro.net/namespace/versicherung/tarifierung
${X3}	Namespace SOAP-Envelope:

	http://schemas.xmlsoap.org/soap/envelope/
${X9}	Namespace WS-Security: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
${A2}	Das ISTS Security Token entsprechend SAML 2.0.

882