Deliverable
D63

TrustCoM Framework V4 Appendix A: Profiles

AL1 – TrustCoM Framework

Michael D. Wilson, CCLRC
Alvaro Arenas, CCLRC
Lutz Schubert, HLRS

04/11/2006

V0.9

TrustCoM

A trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organisations

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2002-2.3.1.9

Networked business and governments

Deliverable datasheet

Project acronym: TrustCoM

Project full title: *A trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organisations*

Action Line: AL1

Activity: 1.2

Work Package: WP27

Task:

Document title:	**TrustCoM Framework V4 Appendix A: Profiles**
Version:	**v0.9**
Document reference:	
Official delivery date:	**31/01/2007**
Actual publication date:	
File name:	D63 TrustCoM Framework V4 - Appendix A.doc
Type of document:	Report
Nature:	official deliverable

Authors:

Reviewers:

Approved by:

Version	Date	Comments
V0.9	05/11/2006	Moved Profile section from main document to (this) appendix

Table of Content

Introduction EMIC, CCLRC

TrustCoM will focus on, and expects to have its main impact with respect to standardisation in the creation of profiles. A profile identifies how different specifications should be used together to support complex applications. This specifically applies to (but is not limited to) interoperable web services. If individual web services standards are metaphorically seen as pieces of a jigsaw puzzle that each capture some autonomous functionality, then profiles can be seen as recommended designs of jigsaws and “best practice” guidelines that support work towards implementing comprehensive and potentially complex business functions. Profiles are created in response to the ever-growing number of interrelated specifications, all at different version levels and different stages of development and adoption, and often with conflicting requirements. Profiles integrate and refine dominant web services standard specifications by resolving potential conflicts between them, constraining their extensibility options where necessary, and exploiting their complementarity and composability characteristics.

Specific emphasis goes to the potential of creating TrustCoM profiles that integrate existing standards within and across the different areas. The project will concentrate on integration profiles, bringing together the isolated subsystem developments; while we have refined the potential standardisation contributions within each specific TrustCoM research and development area, the most immediate result of the TrustCoM standardisation activity is expected to be in the integration of existing standards across the different areas.

The specification of the TrustCoM Framework for implementation in software draws upon many open specifications for three reasons:

- to transparently show how it operates in order to build trust in it as a technology;
- to ease implementation by anybody who wishes to do so;
- to improve the probability that the technology will interoperate between a wide range of platforms.

Consequently, there are many combinations of open specifications that could be the subject of profiles. In order to have an impact, only a small set of specifications have been selected as the basis of profiles which are both likely to be adopted, and where the project has mature input resulting from significant experience. These are:

- WS-Agreement/WSLA – to revive the structural detail required to specify SLA’s lost in the development focus on WS-Agreement
- WS-Trust – to refine the interaction of WS-Trust with other specifications
- WS-CDL – to demonstrate the integration of choreography and orchestration as methods of co-ordination of distributed business processes.
- XACML – to introduce delegation into the security specification
- EDA-Policies – to refine the policy representation as used in TrustCoM

Each of these will be described below as a proposal for wider adoption.

I WS-Agreement SICS

This section defines a profile for the use of the Web Service Agreement (WS-Agreement) specification to describe service level agreements (SLAs) within a TrustCoM Virtual Organization.

I.1 Background

The SLA technology analysis performed in the state-of-the-art evaluation and the experience accumulated so far in the project has resulted in the selection of WS-Agreement as the main SLA specification formalism to be used within TrustCoM.

Although the specification is at times too general, it can be extended with relevant elements from the well known Web Services Service Level Agreement (WSLA) specification developed by IBM. The resulting specification is rich enough to match the needs of SLAs within the TrustCoM framework, naturally matching the distribution of tasks and responsibilities in the SLA Management subsystem as discussed in Architecture Deliverable (D9).

One design decision affects considerably the way the WS-Agreement specification is extended and used in the Framework. Loosely coupled components are to make as much use as possible of the notification mechanisms supported by the EN/VO infrastructure. An agreement must therefore be explicit about the way these mechanisms are to be used during SLA management. Furthermore, application and supporting services are to be virtualized as VO Resources before they can be shared in a VO. The virtualization mechanisms, also provided by the EN/VO infrastructure, introduce a level of indirection in the representation of service addresses which has to be taken into account when describing services and their QoS requirements/guarantees.

I.1.a Summary

The *Web Service Agreement* (WS-Agreement) specification from the Global Grid Forum, enriched and extended with elements of the *Web Service Level Agreement* (WSLA) specification from IBM, has been chosen as service level agreements description language. The resulting specification has a rich set of elements that is suitable for describing the distribution of tasks and responsibilities in the SLA Management subsystem.

I.2 Profile Definition

I.2.a Namespaces

For this profile, the namespace prefixes are defined as follows:

```
xmlns:wsag="http://schemas.ggf.org/graap/2005/09/ws-agreement"

xmlns:wsla=”http://www.ibm.com/wsla”

xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"

xmlns:uddi=”urn:uddi-org:api_v3”
```

```
xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xmlns:slaEval=”http://www.sics.se/TrustCoM/SLAEvaluator”
xmlns:emicfpi="http://www.microsoft.com/emic/SAFe/#FederationPartners"
```

I.2.b SLA Identification

The identifier of an SLA document is contained in the `AgreementID` attribute of the root element (`wsag:Agreement`). This identifier should be globally unique, i.e. no two distinct SLA documents may have the same identifier. By means of this identifier the SLA document can be recovered from the SLA Repository.

I.2.c Signatory Parties

In WS-Agreement, an SLA relates the consumer to the provider of a service. These two parties are expected to sign the agreement, and they are described with elements of type `emicfpi:FederationPartnerIdentifier`. This represents an extension of the WS-Agreement specification.

I.2.d Supporting Parties

The WS-Agreement specification does not define an element for supporting parties, i.e. contributors to the execution of the SLA that are neither consumers nor providers. The WS-Agreement specification has been extended to accommodate these elements by importing the relevant types from WSLA (the notion of supporting party exists in WSLA). The extended specification defines three types of supporting parties: `MeasurementService`, `ConditionEvaluationService` and `ManagementService`.

Measurement Service. Corresponds both to the simple and aggregating monitors in the conceptual model (cf. deliverable D16 – Conceptual Models).

1. When a `<Metric>` element of type `wsla:MetricType` contains a `<MeasuringDirective>` sub-element, then the metric value is to be produced by a Simple Monitor. The responsibility of making available this value is assigned to the Supporting Party given by the `<Source>` element. The following example tells us that supporting party “YMeasurement” is responsible for producing the integer value of a MeasurementDirective of type tc:StatusRequest (an application-specific concretization of the abstract MeasurementDirective type).

Example

```
          <wsla:Metric name=”MeasuredStatus” type=”integer” unit=””>
               <wsla:Source>YMeasurement</wsla:Source>
               <wsla:MetricURI>http://www.ymeasurement.com/status</wsla:MetricURI>
               <wsla:MeasurementDirective xsi:type=”tc:StatusRequest”
                                     resultType=”integer”>
               </wsla:MeasurementDirective>
          </wsla:Metric>
```

Observe that the directive optionally identifies the URI (`<MetricURI>`) that shall be used to access this monitor. The monitor should implement a manageability interface (e.g. according to WSDM), or its manageability interface should be recoverable from the `MetricURI` or from the `<action>` elements of its supporting party definition. Using a manageability interface it must be possible to access and modify the configuration of the monitor.

2. When a `<Metric>` element of type `wsla:MetricType` contains a `<Function>` sub-element, then the metric value is to be produced by an Aggregating Monitor. The following example tells us that supporting party "HLRS2" is responsible for producing the double integer value that results of dividing the result of metric clock_speed by (100 – process_Cpu_Load).

Example

```
<wsla:Metric name="performance_metric" type="double" unit="GHz">
     <wsla:Source>HLRS2</wsla:Source>
     <wsla:MetricURI>http://csharp.hlrs.de/TrustCoM/agmonitor</wsla:MetricURI>
     <wsla:Function xsi:type="wsla:Divide" resultType="double">
          <wsla:Operand>
               <wsla:Metric>clock_Speed</wsla:Metric>
          </wsla:Operand>
          <wsla:Operand>
               <wsla:Function xsi:type="wsla:Minus" resultType="double">
                    <wsla:Operand>
                         <wsla:LongScalar>100</wsla:LongScalar>
                    </Operand>
                    <wsla:Operand>
                         <wsla:Metric>process_Cpu_Load</wsla:Metric>
                    </wsla:Operand>
               </wsla:Function>
          </wsla:Operand>
     </wsla:Function>
</wsla:Metric>
```

Observe that the example does not specify how party HLRS2 is expected to make the metric value available to other SLAM components. In the case of the Condition Evaluation service this is done by the use of element `<SLAParameter>` as explained below.

3. Some Monitors (i.e. MeasurementServices) are also producers of SLA Parameters. A `SLAParameter` contains a sub-element `SLAParameter/Metric` indicating how the parameter is defined. The party responsible for providing the parameter is indicated by the sub-element `SLAParameter//Source`. The sub-element `SLAParameter//Pull` may be used to define which parties are allowed to pull the SLA Parameter from the monitor (by invoking its `GetSLAParameterValue` operation). The sub-element `SLAParameter//Push`, if not empty, indicates that:

1. The Monitor is expected to produce notifications according to the metric schedule.
2. The SLA Management subsystem shall subscribe the list of parties in the `<Push>` element to receive those notifications.

3. A Monitor publishes notifications using the simple topic dialect for topics. A topic is formed in the following way in order to identify an SLA Parameter uniquely:

   { *value of* `SLAParameter//<Source>` } *value of* `SLAParameter/@name`

   For instance, if element `SLAParameter//<Source>` has value "`http:/example.com:monitor1`" (a URI) and the SLA parameter attribute `name` has value "`averageResponseTime`" then the topic is: "{http:/example.com:monitor1}averageResponseTime".

4. A notification shall contain a message of type `slaEval:SLAParameterWrapperType` that contains element `ArrayOfSLAParameters`, which in turn contains an array of `SLAParameter` elements of type `slaEval:SLAParameterType`. The type `slaEval:SLAParameterType` is based on type `wsla:SLAParameterType` in order to be compatible with the WSLA specification. Thus it has element `value` of type `xsd:double` and has attribute `name` of type `xsd:string`, attribute `type` of type `xsd:string` and attribute `unit` of type `xsd:string`.

Condition Evaluation Service. There is a one to one relationship between this type of supporting party and SLAEvaluators (see D16 – Conceptual Models). An SLAEvaluator is responsible for notifying the violation of a set of service level objectives (see below).

Management Service. The approach of the TrustCoM Framework establishes an SLA Management infrastructure as part of the constitution of the VO, regulated by the GVOA, so it is unnatural to let specific SLAs define how they are to be managed. For this reason, the present profile deprecates the use of ManagementServices.

I.2.e Service Description

The WS-Agreement specification `<ServiceDescriptionTerm>` has been enriched with the addition of elements from the WSLA specification under the sub-element `<ServiceDefinition>`. This extension allows to accommodate service term definitions like the `SLAParameter` and the `Metric` elements mentioned above, service scheduling definitions, and others.

I.2.f Obligations

WS-Agreement `GuaranteeTerm` elements are used only to encode obligations on the SLAEvaluator and the Monitors. These elements have been extended in order to use the rich Service Level Objective (SLO) expression language found in the WSLA specification.

An SLAEvaluator is responsible for the evaluation of an SLO expression, and for the notification of a term violation whenever the expression is not satisfied.

Example

```
          <wsag:GuaranteeTerm wsag:Name="SLO_PERFORMANCE"
                              wsag:Obligated="ServiceProvider">
                    <wsag:QualifyingCondition>
                    <wsla:Validity>
                         <wsla:Start>2005-02-15T14:00:00</wsla:Start>
                         <wsla:End>2007-06-15T14:00:00</wsla:End>
                    </wsla:Validity>
                    <wsla:EvaluationEvent>NewValue</wsla:EvaluationEvent>
               </wsag:QualifyingCondition>
               <wsag:ServiceLevelObjective>
                    <wsla:Expression>
                         <wsla:Predicate xsi:type="wsla:Greater">
                         <wsla:SLAParameter>Performance</wsla:SLAParameter>
                         <wsla:Value>600</wsla:Value>
                         </wsla:Predicate>
                    </wsag:Expression>
               </wsag:ServiceLevelObjective>
          </wsag:GuaranteeTerm>
```

In this example, an SLAEvaluator will compute SLO g1. Whenever this SLO is violated (the predicate evaluates to false), the SLAEvaluator is obliged to send a notification. The notification message is published with a simple topic "{http://wsrf.notification.de}SLA_violation". The message contains the identifiers of the violating SLA Parameters, the identifier of the SLA (equal to the SLA name), the identifier of the SLO (qualifying the SLO name using the SLA name), the identifier of the SLA template (that was used to create the SLA), the service operation name and the violating partner (see discussion on topics for notifications emitted by MeasurementServices). There is also a message id that is a timestamp of type `xsd:datetime`.

In case the SLO is fulfilled (the predicate evaluates to true), a similar message is sent with topic "{http://wsrf.notification.de}SLA_fulfilment".

The `GuaranteeTerm` elements have also been extended with WSLA sub-elements to define the time period through which a `GuaranteeTerm` is valid (`wsla:Validity`), and how often the guarantee should be checked for violations (`wsla:EvaluationEvent`). In the example above `NewValue` means that the SLO will be evaluated each time a new value is received from the corresponding monitor.

II WS-Trust & SAML EMIC, ETH

This section describes a WS-Trust and a SAML token profile for virtual organizations as implemented in the FP6 TrustCoM project. The purpose of this document is to specify how web service components communicate with security token services (STS) to request an STS to issue and validate 'cross-organizational' security tokens.

This chapter does not intend to present a final profile, nor does it intend to present a mandatory profile for use outside the "scoped federations" context as implemented in the FP6 TrustCoM project.

We differentiate between two types of security tokens:

- Organization-internal security tokens and
- Cross-organizational security tokens.

This differentiation is necessary because each VO partner organization may use arbitrary security tokens inside the organization's own network, so that a standardization and unification of these types is not possible. For example, one VO partner organization may solely use Kerberos tokens to authenticate and protect messages inside the company's network, whereas other VO partners may use username/password or X.509 certificates inside their organization. Even in scenarios where all organizations use long-term tokens such as X.509 certificates, it may not be possible to use these tokens cross-organizationally, because the companies may not have a common root of trust (e.g., no X.509 cross-certification).

For the above reasons, it is necessary to agree on a common format of cross-organizational security tokens. Inside TrustCoM, we agreed to use SAML assertions as security tokens.

The objective is to draft a profile in which all the parameters are clearly justified, and correspond to a concept from the framework. The current draft is not fully there yet, primarily because the profile originally suggested uses symmetric encryption (whereas other SAML profiles with which this should be consistent use asymmetric encryption) and it also contained parameters whose purpose is not immediately obvious.

It is important that it can be clearly seen how the profiles and their parameters fit the framework, and what the relationship between profile parameters and framework elements/concepts are.

Here is a summary of what has been agreed so far:

i. The WS-Trust profile will send SAML attribute assertions

ii. The parameters to be used from SAML attribute assertions are

   a. the issuer field is mandatory and contains the name of the issuer of the attribute assertion/security token.

   b. advice is optional and probably wont be used

   c. the signature field is optional and isnt needed when X.509 ACs are passed as the attributes, or when symmetric encryption is used

d. conditions are optional, but when present will contain the validity time of the attribute assertions (notBefore and notOnOrAfter)

e. the subject statement holds the name of the entity that the attributes are being assigned to

f. the set of attributes contain the attributes being assigned to the subject

Whereas the following issues are still outstanding and not agreed so far:

i. how symmetric tokens and tickets are encoded

ii. how obligations are encoded

iii. how delegation permission is encoded

iv. how "no assertion" is incoded

These will be addressed in the next development cycle of six months before the next release of V3 of this Framework.

II.1 Namespaces and supported specifications

Inside this document, the namespace prefixes are defined as follows:

```
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"


xmlns:emic="http://www.microsoft.com/emic/SAFe/"
xmlns:emicfrc="http://www.microsoft.com/emic/SAFe/#FederationRestrictions"
xmlns:emicfpi="http://www.microsoft.com/emic/SAFe/#FederationPartners"
xmlns:wstx="http://www.microsoft.com/emic/SAFe/#WSTrustExtensions"
```

II.2 WS-Trust

This profile is based on the WS-Trust specification from February 2005 (http://msdn.microsoft.com/ws/2005/02/ws-trust/).

II.2.a Issuance Binding Profile

For requesting a new cross-organizational security token, we use the “Issuance Binding” as defined by the WS-Trust specification from February 2005.

- wst:TokenType

The WS-Trust token type for cross-organizational SAML assertions is defined as follows:

```
<wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1</wst:TokenType>
```

- wsp:AppliesTo

In this profile, the requestor of a security token MUST specify a wsp:AppliesTo element as part of the wst:RequestSecurityToken. This element may have the following components:

wsp:AppliesTo/wsa:EndpointReference/wsa:Address (MAY)

> The URI of the web service where the token will be used.

wsp:AppliesTo/wsa:EndpointReference/wsa:Action (MAY)

> The action that is invoked on the web service where the token will be used.

wsp:AppliesTo/wsa:EndpointReference/wsa:ReferenceProperties/emic:FederationUUID (MUST)

> The FederationUUID is an identifier of the VO inside which the token will be used. We expect that an issue request for a cross-organizational token MUST contain a VO identifier (such as a FederationUUID). That is necessary because the STS must be able to lookup whether the requesting client has available claims for that particular VO.
>
> In this profile document, we defined an own format for a VO identifier. The model would allow to use other types of identifiers with equivalent functionality, for example from UDDI space.

wsp:AppliesTo/wsa:EndpointReference/wsa:ReferenceProperties/emicfpi:FederationPartnerIdentifier (SHOULD)

> That federation partner identifier is an identifier of the VO partner organization that performs token validation for the service. Such a partner identifier could be a long-term credential of the partner’s STS (such as an X.509 certificate or a reference to a certificate), a UDDI business entity key or some other unique identifier.
>
> The STS needs the federation partner identifier for different purposes: In a symmetric-key based (Kerberos-like) model, the STS requires that information to

determine the service's organization's security token (key), so that the STS can include a session key inside the cross-organizational token. In addition, the partner identifier may be used for client-side security decisions.

- wst:RequestedSecurityToken

The wst:RequestedSecurityToken MUST contain a cross-organizational saml:Assertion element.

- wst:RequestedProofToken

The wst:RequestedProofToken SHOULD contain the private or secret key material associated with the saml:Assertion. In the current "scoped federations" prototype, the wst:RequestedProofToken contains an xenc:EncryptedKey element. The xenc:EncryptedKey contains a symmetric key encrypted for the requestor of the token, i.e., the key is encrypted under the client's organization-internal key.

II.2.b Validation Binding Profile

The current prototype adopts the WS-Federation "U-model". To validate an existing cross-organizational security token at the service side, we use the "Validation Binding" as defined by the WS-Trust specification.

- wst:TokenType

The WS-Trust token type for validation SAML assertions is defined as follows:

```
<wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1</wst:TokenType>
```

- wsp:AppliesTo

See Issuance Binding Profile. For token validation, the service MUST provide wsa:Address and wsa:Action elements in the wst:RequestSecurityToken/wsp:AppliesTo.

- wstx:ValidateTarget

The wstx:ValidateTarget element refers to the target of validation. The wstx:ValidateTarget element MUST contain the cross-organizational saml:Assertion that should be validated.

- wst:RequestedSecurityToken

The wst:RequestedSecurityToken MUST contain a saml:Assertion that contains the validation results.

- wst:RequestedProofToken

The wst:RequestedProofToken SHOULD contain the public or secret key material with which the service can verify the signature of the received message as well as decrypt the received message. In the current "scoped federations" prototype, the wst:RequestedProofToken contains an xenc:EncryptedKey element. The xenc:EncryptedKey contains the symmetric key associated with the SAML token, now re-encrypted for the service.

- wst:Status

The wst:Status element MUST be included in the RSTR as specified by WS-Trust. The predefined URIs, as specified in WS-Trust, are used in the current prototype.

II.3 SAML Assertion Profile

This profile is based on the SAML 1.1 Assertion specification (http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf) and the Web Services Security SAML Token Profile 1.1 (http://www.oasis-open.org/committees/download.php/15256/Web%20Services%20Security%20SAML%20Token%20Profile-11.pdf).

II.3.a SAML cross-organizational token

The cross-organizational security token is a SAML 1.1 saml:Assertion. The saml:Assertion MUST include saml:Conditions, saml:AttributeStatement, and ds:Signature elements.

- saml:Assertion

The @Issuer attribute SHOULD contain the URI of the issuing STS.

- saml:Conditions

In addition to the @NotBefore and @NotOnOrAfter attributes which MUST be included, the saml:Conditions element MUST include a emicfrc:FederationRestrictionCondition.

emicfrc:FederationRestrictionCondition (MUST)

> The FederationRestrictionCondition defines the federation scope in which the cross-organizational SAML assertion can be used. Validation in other scopes must fail.

- saml:AttributeStatement

The saml:Assertion MUST contain exactly one saml:AttributeStatement. That saml:AttributeStatement element MUST contain one saml:Subject and a saml:Attribute element.

saml:Subject (MUST)

> The subject is the owner of the token and is identified by a saml:SubjectConfirmation/saml:ConfirmationMethod urn:oasis:names:tc:SAML:1.0:cm:holder-of-key as specified in the WSS SAML Token Profile 1.1.
>
> The key is included in a ds:KeyInfo element which contains an xenc:EncryptedKey with a symmetric key encrypted for the receiving VO partner organization.

saml:Attribute Claims (MUST)

> The AttributeName is "Claims" and the AttributeNamespace is "http://schemas.xmlsoap.org/ws/2005/02/trust". The saml:AttributeValue element MUST contain a wst:Claims element.
>
> The wst:Claims element contains the claims that the client possesses in the particular VO. These claims may be xacml11 attributes.

- ds:Signature

The Signature MUST contain exactly one ds:Reference referencing the saml:Assertion/@AssertionID attribute. This Reference MUST have exactly two transforms:

1. The first transform is "Enveloped Signature" (http://www.w3.org/2000/09/xmldsig#enveloped-signature)
2. The second transform is "Exclusive XML Canonicalization without Comments" (http://www.w3.org/2001/10/xml-exc-c14n#)

To support cross-organizational validation of the signature of the token, the KeyInfo element MAY contain various references to the signing certificate of the issuing STS, including a wsse:SecurityTokenReference/wsse:KeyIdentifier, a wsse:SecurityTokenReference/wsse:Embedded, or a emicfpi:FederationPartnerIdentifier.

II.3.b SAML validation token

The validation response is a SAML 1.1 saml:Assertion. The saml:Assertion MUST include saml:Conditions, saml:AttributeStatement, and ds:Signature elements. In addition, a saml:Advice SHOULD be included.

- saml:Assertion

The Issuer attribute SHOULD contain the URI of the validating STS.

- saml:Conditions

In addition to the NotBefore and NotOnOrAfter attributes which MUST be included, the saml:Conditions element MUST include a emicfrc:FederationRestrictionCondition.

emicfrc:FederationRestrictionCondition (MUST)

The federation scope in (and only in) which this SAML assertion is to be considered.

- saml:Advice

The saml:Advice SHOULD contain the original cross-organizational saml:Assertion that has been validated.

- saml:AttributeStatement

The saml:AttributeStatement element MUST include a saml:Subject and at least one saml:Attribute element.

saml:Subject (MUST)

The subject is the owner of the original cross-organizational token that is validated, and is identified by a saml:SubjectConfirmation/saml:ConfirmationMethod urn:oasis:names:tc:SAML:1.0:cm:holder-of-key as specified in the WSS SAML Token Profile 1.1.

The key is included in a ds:KeyInfo element which contains a wst:BinarySecret with a cleartext symmetric key (this assumes that the RSTR is properly protected!), or an xenc:EncryptedKey with a symmetric key encrypted for the receiving VO partner organization.

saml:Attribute FederationPartnerIdentifier (MAY)

The AttributeName is “FederationPartnerIdentifier” and the AttributeNamespace is "http://www.microsoft.com/emic/SAFe/#FederationPartners". This attribute MAY be included to explicitly indicate to the service the VO partner organization the service request is originating from. If this attribute is present, the saml:AttributeValue element MUST contain an emicfpi:FederationPartnerIdentifier element.

saml:Attribute Status (MUST)

The AttributeName is “Status” and the AttributeNamespace is "http://schemas.xmlsoap.org/ws/2005/02/trust". This attribute MUST be included to indicate the result of the security token validation. The saml:AttributeValue element MUST contain a wst:Status with one of the predefined wst:Code status codes.

saml:Attribute Claims (SHOULD)

The AttributeName is "Claims" and the AttributeNamespace is "http://schemas.xmlsoap.org/ws/2005/02/trust". This attribute SHOULD be included to pass the validated (and possibly transformed) claims to the service. If this attribute is present, the saml:AttributeValue element MUST contain a wst:Claims element.

A policy enforcement point (PEP) may forward these validated claims to a policy decision point (PDP) to support the policy decision.

saml:Attribute ValidationMessage (MAY)

The AttributeName is "ValidationMessage" and the AttributeNamespace is "urn:string". This attribute MAY be included to pass a human-readable validation result message to the service.

II.4 Custom elements

The following custom namespace prefixes are defined in the current “scoped federations” prototype in TrustCoM:

```
xmlns:emic="http://www.microsoft.com/emic/SAFe/"
xmlns:emicfpi="http://www.microsoft.com/emic/SAFe/#FederationPartners"
xmlns:emicfrc="http://www.microsoft.com/emic/SAFe/#FederationRestrictions"
```

- emic:FederationUUID

The emic:FederationUUID represents a universal and unique identifier for the federation scope.

- emicfpi:FederationPartnerIdentifier

The emic:FederationPartnerIdentifier identifies a VO partner organization. A partner organization can be identified in various ways as indicated in the Type attribute.

- X509SubjectName Type

X509Data/X509SubjectName (MUST)

The X.509 DN of the certificate of the issuing STS of the partner.

- emicfrc:FederationRestrictionCondition

The emicfrc:FederationRestrictionCondition is a custom SAML condition which intends to indicate the “scope” within which the SAML cross-organizational or validation token MUST be considered.

- wsp:AppliesTo

The FederationRestrictionCondition MUST contain a wsp:AppliesTo element.

wsp:AppliesTo/wsa:EndpointReference/wsa:Address (SHOULD)

The URI of the web service that is invoked.

wsp:AppliesTo/wsa:EndpointReference/wsa:Action (SHOULD)

The action that is invoked on the web service.

wsp:AppliesTo/wsa:EndpointReference/wsa:ReferenceProperties/emic:FederationUUID (MUST)

The VO identifier inside which the assertion can be used.

II.4.a Role semantics

In the first prototype, we used a self-defined role claim with proprietary semantics to represent roles. In TrustCoM, we will use XACML 1.1 attribute values to convey role information.

III WSCDL SAP

This section defines a profile for the use of W3C's Web Service Choreography Description Language (WS-CDL) specification to describe the business process modelling aspects of collaboration definitions within a TrustCoM Virtual Organization.

III.1.a Background

The technology analysis on collaborative business processes performed in the state-of-the-art evaluation and the experience accumulated so far in the project has resulted in the selection of WS-CDL as the business process specification to be used for the holistic view on collaborative business processes within TrustCoM, i.e., the single view on the collaborative process that includes the activities at and interactions between all involved parties. Although much critique has been issued against the specification [3], [4], [5] and it is not yet a standard, the advantages over other available choreography models outweigh the issues. WS-CDL matches the needs for collaborative business processes within the TrustCoM framework, due to the following reasons: It specifies the control flow over interactions and local activities between multiple roles from a high-level perspective, and is conceptually close enough to single-party business process languages to be matched with them. A choreography language that allows for the modeling of complex interaction patterns would mostly be good for design, not for execution as a business process, because its execution should include executable business processes as well as more flexible programming models and human interaction, e.g., for distinctive choice points with a high economical impact.

Most other choreography languages state single-partner processes and connect them, at the cost of hard legibility and high risk of incoherency. WS-CDL always offers a combined view on all partners' activities, making it much easier to realize and observe coherence in the various parties' behavior

Alternative specifications include WSCI, WSCL, and BPSS:

- WSCI[1] is also a specification by W3C, which allows the definition of choreographies by extending WSDL interfaces to express business process semantics over the web service operations and connecting such extended WSDLs to form a choreography. There are multiple points to note here: WSCI is more a web service technology than a business process technology. Its most natural use would be to connect existing web services, thus suggesting a bottom-up approach – instead of the here-anticipated top-down approach. The distribution of the choreography specification over multiple documents does not feature a global view on the collaborative business process as a whole, which is supposedly very helpful for consistence and coherency in the understanding of the overall control and data flow. Last, the level of detail is fairly high: on the choreography level, the exact WSDL interfaces of each partner do not yet have to be present.

[1] WSCI: Web Service Choreography Interface

- ebXML[2] is a business collaboration framework, which offers related mechanisms to specify collaborations between partners. The focus here is rather on the business level with its functional and legal implications and not process integration and execution.

Still, an ideal choreography language is not available yet. Potentially, ongoing and future developments will strongly influence the choice for a choreography language in future implementations of the TrustCoM architecture.

III.1.b Summary

The *Web Service Choreography Description Language (WS-CDL)* [1] is the main effort of W3C's WS Choreography Working Group. Still being on the way of becoming a standard, it offers the most promising, currently available way to describe business processes for multiparty collaborations from a high-level perspective. Similar to an abstract BPEL process, a choreography in WS-CDL only describes the externally observable aspects of a collaborative business process. It is important to keep in mind that a choreography is not meant for execution, but resembles a design artifact.

III.1.c Scope

In TrustCoM, WS-CDL is used to model the collaborative business process (CBP) spanning all members of a VO and describing the interplay of their local activities and communication during the operation phase of a VO. This description is given from the high-level perspective of the whole VO with an emphasis on interactions, omitting the details about internal implementations of business services. In other words: While many components in the TrustCoM framework deal with the administrative aspects of the cooperation between the VO members, the choreography describes the actual work to be performed by the VO and how the members align their efforts.

Due to the current usage of WS-CDL, which is to generate WS-CDL code from UML diagrams via the UML2CDL service, and to generate BPEL code from the CDL via CDL2BPEL, WS-CDL could in principle be replaced with moderate effort.

III.2 WS-CDL Language Elements and Representation

One or many choreographies form a cdl:package. Exactly one of them is marked as the "root choreography", and thus is the starting point for a package. Having its roots in the Pi-calculus, a choreography in CDL describes the control flow around basic activities through structuring activities. A choreography can have variables, exception handlers, and finalisers, which define communication and the like at the end of a choreography. Due to the point of view taken by CDL, there are only few basic activities, with the interaction as the centre piece, since the focus of choreographies is to describe the how and when of communication. All basic activities, conditional expressions, and variables can be defined for only a subset (sometimes of size one) of the available roles.

[2] ebXML: Electronic Business using eXtensible Markup Language , see http://www.ebxml.org/

In CDL, the concept used for referring to one of the parties is always the role type (or, in short, the role). A party that wants to participate in a choreography can be required to play multiple roles by specifying a cdl:participant subsuming these roles. Note that each role can belong to zero or one participant. Also, a role can be defined to show more than one behaviour. Each behaviour can be refined in a WSDL document, and, if it is not, has no deeper meaning for the details of the choreography. However, both, a WSDL and a CDL document, describe the behavioural interface of entities, although a choreography includes far more information. Thus, our impression is that the redundancy in providing an additional WSDL per role behaviour alongside with a choreography yields no significant advantage. Note, that CDL has a closed-world assumption, meaning that interactions are always bilateral between two roles specified in the choreography.

Example

```
<roleType name="AnalysisPartner">
      <behavior name="Analyzer"/>
</roleType>
<roleType name="StoragePartner">
      <behavior name="StorageProvider"/>
</roleType>
```

In the above code snippet from a WS-CDL package in Collaborative Engineering, two role types are defined: the *AnalysisPartner* and the *StoragePartner*, each showing a single behaviour with no assigned WSDL interface. The choreography corresponds to the UML Activity Diagram in Figure 1.

III.2.a WS-CDL Activity Elements

Starting with the structuring ones, the list of activities is shown below.

- **sequence** - Sequential order of activities.
- **parallel** - Parallel execution of activities.
- **workUnit** - As the most unusual structuring activity, the workUnit specifies conditions under which an enclosed activity is executed or repeated. Its guard condition is similar to an if-condition in standard programming languages, and can contain various XPath expressions or CDL supplied functions. The guard can be evaluated either immediately or deferred (e.g., when a variable becomes available) by setting the block attribute to true or false, respectively. Furthermore, the repeat condition states if a workUnit is considered for execution again after completion.
- **choice** - Exclusive branching: at most one of the enclosed activities (which may itself be a structuring activity) is to be performed. A cdl:choice is intended to contain workUnits as children, with a guard condition. If there are non-workUnit children in a choice, the branching condition is said to be non-observable or not relevant at the choreography.
- **interaction** - Used for communication between two roles. In data exchanges the submitted variables are specified. Timeout conditions can be defined directly in an interaction, as well as assignments with reference to the data exchanges. If an interaction's "align" attribute is true, transactionality for an interaction is enabled, in

the sense that the interaction only shows effect if the involved roles have the mutual understanding that the interaction completed successfully.

Example

```
<interaction name="getRawDataReq" operation="getRawDataOp"
  channelVariable="chVarGet">
  <participate relationshipType="AnalysisStorageRel"
    fromRoleTypeRef="AnalysisPartner"
    toRoleTypeRef="StoragePartner"/>
  <exchange name="exRawDataAddr" informationType="uriType"
    action="request">
    <send variable="cdl:getVariable('varRawDataAddr_Ana','','')"/>
    <receive variable="cdl:getVariable('varRawDataAddr_Sto','','')"/>
  </exchange>
</interaction>
```

This code example shows the information exchange between the AnalysisPartner (AP) and StoragePartner (SP) from the first CDL code example. The Web service operation ‘getRawDataReq’ at SP is called by AP. The information exchanged is the raw data’s address, available in the variable ‘varRawDataAddr_Ana’ at the AP and stored in the variable ‘varRawDataAddr_Sto’ at SP after the transmission.

- **noAction** - Explicit “no operation” for a specified role. The respective party must remain idle.
- **silentAction** - Partner-internal action, whose details are of no interest to the choreography as a whole. The comment, by default in natural language, specifies what a partner is assumed to do at that instant, e.g., “analysis of aircraft antenna”.
- **assign** - Variable value modification. Can be used to trigger exceptions.
- **perform** - Execution of another choreography. With CDL’s binding mechanism, variable values from the outer choreography can be carried over to the inner choreography.

The link between WS-CDL and the Pi-calculus is strong, and also becomes apparent in the availability of channels in CDL. There, channel variables are of a channel type, which allows the definition of identity and reference tokens, restrictions on the channel usage, and the receiving role at the end of a channel. However, the way channels can be used in CDL as well as certain activities and more allow for several points of critique. This critique is subject to [3], [4], [5] and shortly summarized below.

III.2.b Graphical Notation

UML activity diagrams offer a good visualization for choreographies, as justified in [2]. Where common business process modelling languages deal with only one party per process, in a choreography there are always multiple roles. The distinction between activities of the various roles is achieved by using a swim-lane (large, rectangular boxes) per partner. In contrast to WS-CDL, UML activity diagrams do not know a single activity for the interaction as a whole, so each cdl:interaction is represented by a pair of send and receive activities.

III.2.c Summary of Critique Against WS-CDL

The main points of critique in [3] (p.16-18) are: the not explicitly stated link to a formalism as the Pi-calculus on the one hand, and the conceptual limits of linking WS-CDL to WSDL, WSDL-MEPs[3], and WSBPEL on the other hand; the not anticipated runtime selection of participants; the restriction to binary interactions; the dissimilarity of the sets of control flow constructs of WS-CDL and WSBPEL with respect to the fact that WSBPEL is the most promising orchestration language; and the discrepancy of WS-CDL being a design-level language and having no graphical representation. These are all very good points and - since they are deeply positioned in the concepts of the language - question the future of WS-CDL as a whole.

In WS-CDL, communication (cdl:interaction) is always bilateral, and built-in transactionality is restricted to the guaranteed mutual agreement of single variable values at one point in time. Therefore, WS-CDL most likely is unable to express the majority of the 15 “Service Interaction Patterns” from [4]. It thus seems not suitable for modelling related use cases, like a broad request for proposals with unknown outcome.

Also, the redundancy in certain WS-CDL elements makes writing a choreography with a general-purpose editor inconvenient. For instance, an attribute whose content has to be a variable, still needs to use the cdl:getVariable function.

III.3 Annotation of Trust, Security, and Contract (TSC) Tasks

As an augmentation of WS-CDL documents a conceptual model for a collaborative business process security concept was introduced in D16, the TrustCoM conceptual models V1, and is further refined in the Appendix. The goal of this concept is to inject security controls where required into the role specific executable public/private business processes. To achieve this, the collaboration definition activities and interactions are annotated with so-called TSC Extension Roles. This concept serves its purpose if, at collaboration definition modelling time, it is at least known, that a TSC control has to be enforced at a specific interaction in collaboration. This is realised by adding an empty TSC Extension Role only containing the header data, the specific role can be deployed at runtime by the BPM service.

[3] WSDL 2.0 Message Exchange Patterns

IV XACML SICS

XACML is an OASIS standard for access control policies. This document describes how XACML is used in TrustCoM. The aim is to define a common method of applying XACML in order to provide for interoperability and easy to use guidelines which save time and effort for the TrustCoM partners.

TrustCoM uses XACML 3.0[4].

XACML is based on the concept of attributes. Subjects and resources are defined in terms of their attributes, for instance the role of a user is an attribute of the subject and the name of a service is an attribute of the resource. Policies are written in terms of these attributes and the attributes of the subject and resource that is being accessed are made available to the PDP, which can then calculate whether the access should be permitted or not.

One important part of this profile is to define which attributes are available for policy writers to refer to in their policies. Another part of this profile makes recommendations on the overall structure of policies and how the delegation features fit in the overall picture of TrustCoM.

XACML itself does not define any kind of transport formats, but the SAML profile of XACML defines transports formats[5].

IV.1 Attributes

In the TrustCoM PDP there are two sources of attributes. The PEP will add attributes to the request it sends to the PDP. These attributes concern the access entities, that is the subject, resource, action and environment[6]. In addition to this, the PDP will get attributes from the tokens that policies have been signed with. These attributes concern the issuers of policies and are used to verify that the policies have been issued in an authorized manner.

The attributes that the PEP fills in the request can be divided into two categories: application independent attributes and application specific attributes. The application independent attributes are derived from the SOAP header of the service invocation that is under access control and the WS-Trust token from the SOAP message. The application specific attributes may be based on content from the SOAP body.

[4] XACML 3.0 is still work in progress, but the latest draft is expected to be close to the final version. The latest draft and the final version when it is ready are available at the XACML homepage at www.oasis-open.org/committees/xacml. For a brief and easy to understand overview see http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

[5] The SAML profile for XACML is available at the XACML homepage at www.oasis-open.org/committees/xacml. The profile is currently being updated for XACML 3.0.

[6] XACML.3.0 replaces the request sections Subject, Resource, Action and Environment from XACML 2.0 with named attribute categories. At the time of writing this, the official identifiers for the categories which correspond to the old sections have not yet been assigned, so we refer to the categories using the old section names.

IV.1.a Attributes based on the SOAP header

The following attributes are derived from the SOAP header.

Description	Address of the invoked service. Value of <wsa:To> element.
XACML request attribute category	Resource
Attribute id	urn:oasis:names:tc:xacml:1.0:resource:resource-id
Value	Content of /soap:Envelope/soap:Header/wsa:To element
Type	http://www.w3.org/2001/XMLSchema#anyURI

Description	Value of <wsa:Action> element
XACML request attribute category	Action
Attribute id	urn:oasis:names:tc:xacml:1.0:action:action-id
Value	content of /soap:Envelope/soap:Header/wsa:Action element
Type	http://www.w3.org/2001/XMLSchema#anyURI

The XML fragments below show how an example SOAP header translates to XACML attributes in the request.

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <soap:Header>
<wsa:Action>http://tempuri.org/RepositoryMngSoap/getContentsByProjectRequest</ws
a:Action>
    <wsa:MessageID>urn:uuid:a1543b3d-451c-458c-b528-8b6e67df00d5</wsa:MessageID>
      <wsa:ReplyTo>
<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</ws
a:Address>
    </wsa:ReplyTo>
  <wsa:To>http://localhost:3998/SP_WS/RepositoryMng.asmx</wsa:To>
  </soap:Header>
  <soap:Body>...</soap:Body>
</soap:Envelope>
```

```
<Request xmlns=" urn:oasis:names:tc:xacml:3.0:schema:os">
  ...
  <Attributes Category=”Resource”>
    ...
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
      <AttributeValue DataType=http://www.w3.org/2001/XMLSchema#anyURI
```

```
>http://localhost:3998/SP_WS/RepositoryMng.asmx</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category=”Action”>
    ...
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://tempuri.org/RepositoryMngSoap/getContentsByProjectRequest</Attribu
teValue>
    </Attribute>
  </Attributes>
  ...
</Request>
```

IV.1.b Attributes from Policy Signatures

An XACML policy is signed in the form of signed SAML 2.0 assertion which contains an XACMLPolicyStatement element as defined by the SAML profile for XACML.[7] Policies are signed using private keys which are paired with X.509 public key certificates. The subject of the X.509 certificate is translated into an attribute in the PolicyIssuer element of the XACML 3.0 policy. The policy which is contained in the SAML assertion does not have a PolicyIssuer element. The PolicyIssuer element is generated internally by the PDP based on the signature when the policy is loaded.

The attribute id is the standard subject-id and the data type is an x500Name. The XML fragment below shows an example PolicyIssuer element:

```
<PolicyIssuer>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
    <AttributeValue
      DataType="urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
      >CN=TrustcomAdminSample,O=SICS,ST=Stockholm,C=SE</AttributeValue>
  </Attribute>
</PolicyIssuer>
```

It is the responsibility of the PDP to validate the correctness and trustworthiness of a signature of a Policy.

IV.1.c Policy Identifiers

An XACML policy has a policy id. This identifier needs to be unique for each policy. In order to prevent collision based attacks, the identifier has to consist of a string which begins the name of the issuer, as it appears in the PolicyIssuer element, followed by some unique string. The PDP has to verify that the policy identifier matches the signature of the policy and reject any policy which does not do so.

[7] SAML 2.0 is available from the SAML homepage at www.oasis-open.org/committees/security. The SAML profile for XACML is available at the XACML homepage at www.oasis-open.org/committees/xacml. The profile is currently being updated for XACML 3.0.

IV.1.d Application specific attributes

Access control and delegation policies, as processed by the PDP, may refer to application specific attributes. For the case where the values of these attributes are to be recovered from the bodies of SOAP messages (corresponding to service invocations and responses), this profile suggests two solutions: (1) the PEP can forward the whole body of the message to the PDP, thus letting the PDP extract attribute values using XPath expressions specified in its policies; or (2) the PEP examines the message body itself, extracts attribute values and places them in the authorization request.

Alternative (1) may result in excessive communication costs, depending on the size of the SOAP message bodies, but has the advantage over (2) that the PEP does not need to be configured with application-specific information. Otherwise, in case (2), the Policy Service, i.e. the service that uploads policies to the PDP, could also be in charge of configuring the PEP with information on how to extract attributes from the message bodies.

IV.1.e Coordination context datatype

To facilitate policies which base permissions on a TrustCoM coordination context, we define a custom datatype to hold a coordination context and an equality function. The PEP includes this in the request.

Description	Coordination context
XACML request attribute category	Environment
Attribute id	http://eu-trustcom.com/xacml/attr/coordinationContext
Value	The coordination context in which the access is made.
Type	http://eu-trustcom.com/xacml/type/coordinationContext

http://eu-trustcom.com/xacml/func/coordinationContextMatch is the id of the equality function. This function returns true if two coordination contexts are equal.

IV.2 Policies

IV.2.a Delegation

When a service is deployed, a root policy must be installed in the PDP which will serve the new service. The root policy should contain a full delegation right for the owner of the service. The access policies will be created by having the service owner issue them, having the service owner delegate the right to do so to some external party. The root policy is not modified during normal operations. If the policies need to be changed, new signed policies may be added or removed. This way daily administration can be decentralized as needed by means of the delegation model.

The right to delegate is expressed by means of conditions on delegation chains. A delegation chain is expressed with special attribute categories (Delegete and IndirectDelegate). A request with a Delegate attribute category is a request for verifying the authority of a policy issuer, a so called administrative request. A request without a Delegate attribute category is a request for verifying the right of a particular subject to access a particular service. In case of multiple step delegation, the attribute category IndirectDelegate is used to provide attributes of policy issuers further down a delegation chain.

By writing conditions on different attribute categories we can differentiate between rights to issue policies (administrative rights) and access rights, and also specify limits on further delegation of administrative rights. For TrustCoM we limit the policies to three kinds: access policies, administrative policies which do not allow further delegation and administrative policies which allow further delegation.

IV.2.b Access policies

An access policy shall refer to attributes in the Subject, Resource, Action and Environment sections.

IV.2.c Administrative policies without further delegation

An administrative policy without further delegation shall refer to the Delegate, DelegatedSubject, DelegatedResource, DelegatedAction and DelegatedEnvironment attribute categories. It shall also contain a condition on the standard maximum delegation depth attribute in the DelegationInfo attribute category to limit the depth of delegation to one.

IV.2.d Administrative policies with further delegation

An administrative policy with further delegation shall refer to the Delegate, DelegatedSubject, DelegatedResource, DelegatedAction and DelegatedEnvironment attribute categories.

IV.3 Transport formats

Policies are signed according to the SAML profile for XACML which is produced by the XACML technical committee. The signed policy consists of a SAML 2.0 assertion containing an XACMLPolicyStatement from the profile.

The XML fragment below shows an example of a signed policy.[8] The actual policy content has been removed for ease of presentation.

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="ID_303286ff-c40b-4fe2-86d0-57b335a4740a" IssueInstant="2007-01-17T09:13:36Z"
Version="2.0">
<saml:Issuer
Format="http://www.w3.org/2001/XMLSchema#string">CN=TrustcomAdminSample, O=SICS,
ST=Stockholm, C=SE</saml:Issuer>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
```

[8] The SAML profile for XACML is currently being updated for XACML 3.0, so some namespaces may change in the final version of the profile.

```
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#WithComments"></ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"></ds:SignatureMethod>
<ds:Reference URI="#ID_303286ff-c40b-4fe2-86d0-57b335a4740a">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"></ds:Transform>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#WithComments"></ds:Transform>
</ds:Transforms>
<ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>z9f/BOgC4rCesMB8dBIQTB1+pl4=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
zmWp/yih84saaXDbeITwnv5rUIyEnsRW3/KyxbBCWf2vk8cB34i8VQd2LfVK1qV5tZMmzl9NAu7x
HFk66IwAfsE//j5RgrAKxMky2cz8sgQHisCmKZRww+aTTgnMJ3MtwV0izTzYUP4aDBV07N+uUjQN
qfG7Efyy/qVjGNN8jkg=
</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDbjCCAtegAwIBAgIBCjANBgkqhkiG9w0BAQQFADCBhTELMAkGA1UEBhMCU0UxEjAQBgNVBAgT
</ds:X509Certificate>
</ds:X509Data>
<ds:KeyValue>
<ds:RSAKeyValue>
<ds:Modulus>
7DKNroJM5icFSRnjNmakukmsLhhozo297UGswVCAeJi4y8b48sJnBDa0XUfqKScLfc880cuOKHlS
vADktBCBz689Qo/Eq3hSE5ZmnV8326VlKOKwzQzZWq+VzvMLf/Z7xhLr9n5XuWQghJkskc1M4R4w
h1f98Mx7r5r7mEw7E4c=
</ds:Modulus>
<ds:Exponent>AQAB</ds:Exponent>
</ds:RSAKeyValue>
</ds:KeyValue>
</ds:KeyInfo>
</ds:Signature>
<saml:Statement xmlns:xacml-saml="urn:oasis:xacml:2.0:saml:assertion:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xacml-
saml:XACMLPolicyStatementType">
<xacml:Policy xmlns="urn:oasis:names:tc:xacml:3.0:policy:schema:os"
xmlns:xacml="urn:oasis:names:tc:xacml:3.0:policy:schema:os"
PolicyId="CN%75TrustcomAdminSample,%20O%75SICS,%20ST%75Stockholm,%20C%75SE_585d5
5a2-4829-4993-b4c2-3f31b5e59822"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides" Version="1.0">
.........
</xacml:Policy>
</saml:Statement>
</saml:Assertion>
```

References

[1] W3C. Web Services Choreography Description Language, 2005. W3C Latest Working Draft from October 8th, 2005, work in progress

[2] Marlon Dumas and Arthur H. M. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In UML ’01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, pages 76–90, London, UK, 2001. Springer-Verlag.

[3] Alistair Barros, Marlon Dumas, and Phillipa Oaks. A Critical Overview of theWeb Services Choreography Description Languages (WS-CDL). BPTrends Newsletter, Vol. 3, March 2005

[4] Alistair Barros, Marlon Dumas, and Arthur H.M. ter Hofstede. Service Interaction Patterns: Towards a Reference Framework for Service-Based Business Process Interconnection. http://sky.fit.qut.edu.au/ dumas/ServiceInteractionPatterns.pdf, April 2005.

[5] Roberto Gorrieri, Claudio Guidi, and Roberto Lucchi. Reasoning about interaction patterns in Choreography. In Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM ’05), 2005.