OSCI-Transport, Version 2

Technical Features Overview

OSCI Steering Office

Status: Final
Pulished 1st of June, 2009
Editor: Jörg Apitzsch, bremen online services GmbH & Co. Kg

Table of Contents

1 Introduction

This document gives on overview of the principles of the OSCI 2 architecture and specification. Uts intention is, to give readers a base background for the document "OSCI Transport, Version 2.0 – Web Servcies Profiling and Extensions Specification".

Online **S**ervice **C**omputer **I**nterface (OSCI) is a message standard for eGovernment. Since 2002, version 1.2 of this standard has been increasingly used for confidential and legally binding communication via the Internet[1] in Germany by the public administration, in certain business sectors and their customers and some other European countries, too. OSCI builds up of two parts – part A or "OSCI Transport" addresses message exchange, part B addresses the development of interoperable exchange schemes on functional, business scenario specific levels. Objective of part A of OSCI are payload-agnostic message exchange mechanisms.

OSCI Transport was designed to enable the complete and legally binding handling of transactions in the field of eGovernment via the Internet and on the basis of the digital signature. However, scenarios for exchanging messages and documents are also supported, which are not necessarily demanding with a view to their legally binding nature, but which still require a certain level of confidentiality, verifiability and secured integrity of communications. This requires comprehensive interoperability, both on the level of the content data and on the level of transport and security functions. Furthermore, procedures and rules must be taken into consideration, which apply to the area of public activities. The resulting requirements have determined the design objectives for OSCI. OSCI Transport follows the communication paradigm of Web Services; version 1.2 is based on SOAP 1.1 with its own modelling for security mechanisms based on XML Digital Signature and XML Encryption.

Meanwhile much of the principles and objectives of OSCI Transport are addressed by the building blocks of the Web Services protocol family – the so called "WS-Stack". This generic protocol stack is designed to serve arbitrary business scenario requirements for message exchange over open networks. Profiling has to be done for manifesting the general requirements of eGovernment communication scenarios. For the objective of maximised interoperability, restrictions to the degrees of freedom offered by the WS-Stack must be defined.

In consequence, profiling of relevant protocols of the WS-Stack is the major objective of "OSCI Transport Version 2.0 – Web Services Profiling and Extensions specification" – short "OSCI2" in this document. This profiling is intended to be aligned with similar efforts actually driven by a couple of other EU member states; common goal is establishing interoperable eGovernment communication infrastructures between those countries. Efforts have been made by the Web Services Interoperability Organization (WS-I)[2] to establish a baseline for interoperable Web Services implementations. Profilings of relevant WS-Specifications specify a minimum set of clarifications, recommendations and restrictions that Web Services should support to ensure interoperability across diverse platforms. For OSCI2, the WS-I Basic Profile Version 2.0 [WSI-Basic] and WS-I Basic Security Profile Version 1.1 [WSI-BP11] accordingly apply as references. Version 2.0 of the WS-I Basic Profile is the first covering the set of relevant specifications OSCI2 builds on, but still is a working draft whilst writing OSCI2. WS-I Basic Security Profile Version 1.1 is in the state of a working group approval draft at the same time. Hence, the actual version of the OSCI2 may be subject to refinements, if the forthcoming approved versions of mentioned WS-I Profiles should contain relevant changes.

WS-I Basic Profile itself refers to profilings fixed in WS-I Basic Profile Version 1. These profilings are incorporated by OSCI2, too.

[1] Refer to [OSCI]

[2] see www.ws-i.org

To ensure interoperability and data protection issues, collections of template policies for classes of business scenarios will be developed and published as addendum to the OSCI specification. This will be done by a special project starting mid 2009.

In addition, OSCI2 defines some extensions to the WS-Stack for functionalities which are identified as indispensable in German eGovernment but not yet addressed by the Web Services protocol family.

For sake of interoperability, these extensions may be marked optional in concrete scenarios where messages have to be exchanged with implementations of the WS-Stack protocol family not able to serve OSCI2 specialities. This is considered to be the main anchor for OSCI2 conformant implementations to exchange messages with standard industry solution environments for Web Services based communication.

2 OSCI Messaging: Adoption and Extension of Web Services Specifications

2.1 General SOAP Message Structure

The general message structure conforms to [SOAP12]. All data needed for message transport and message security is placed in the SOAP header. SOAP header blocks are defined by specifications of the WS-Stack incorporated and some of them profiled by OSCI2. For requirements going beyond today's WS-Stack capabilities custom headers are defined, referred to as OSCI header. All these SOAP header blocks together constitute the **Communication Data**.

From the OSCI2 point of view the SOAP body is opaque. Information carried herein is specific to application scenarios and referred to as **Content Data** in OSCI2.

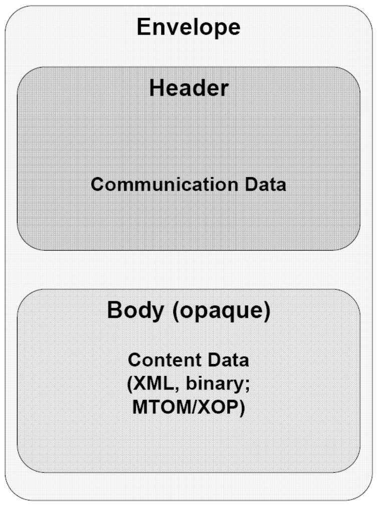

Figure 1: General Message Structure

Attachments in any format can be part of Content Data. For handling of attachments the specification "SOAP Message Transmission Optimization Mechanism" [MTOM] is incorporated without restriction.

2.2 Role and Communication Model

The underlying generalized model of entities involved in the communication and message exchange patterns (**MEP**) is the one defined in [SOAP12]. In concretion, **Source Applications** produce Content Data to be consumed by **Target Applications**. Both are using **OSCI Gateways** for secured, reliable and traceable message exchange. These gateways act in the role of SOAP endpoints. Depending on the individual endpoint capabilities and needs of a concrete MEP, further SOAP intermediary nodes may be passed on the message route acting in OSCI2 specific roles defined in the OSCI specification.

2.2.1 Secured Web Service Paradigm – Authentication and Authorization

Every logical node a message is targeted to is seen as Web **Service Provider (SP)** node; in open, potentially unsecured networks, access to such services must be secured on base of sufficient authentication and authorization of the respective **Service Requestor (SR)**.

We assume closed groups of participants in OSCI2-based communication networks (anonymous allowed only in special cases). Besides use of X509-Certificates and Public Key Infrastructure (PKI), we incorporate the concepts of Web Services Trust [WST] and Web Services Federation Language [WSFED] like **Trust Domains (TD)**, **Identity Provider (IdP)**, **Security Token Service (STS)** and **Attribute Service (AS)**. OSCI2 communication networks are grouped in Trust Domains, where all endpoints and nodes must be registered in a trustworthy manner, whereby the strongness of authentication mechanisms must be scalable according to specific business scenarios needs. Verifiable security tokens proving identity, authentication and authorization of Service Requestors must be carried along with the request – respective those of the Service Provider in the resulting response message.

If not based on agreements between communication endpoints out-of-band of the OSCI specification, addressable nodes in an OSCI2 communication network must expose their service capabilities and security needs based on machine readable policies. With regard to these needs, OSCI makes the only assumption that those policies should be exposed in form of a formal description according to Web Services Description Language [WSDL11] and mechanisms. Services like registries for locating communication endpoints and infrastructure service nodes in an OSCI2 communication network are out of scope of this specification.[3]

Use of X509-Certificates and PKI at least is an ultimate anchor for validity verification, authentication and authorization issues.

Thus, nodes involved in OSCI2 based message exchange must rely on additional service instances as illustrated by general overview in [Figure 2].

2.2.2 Communication Endpoints

OSCI2 defines an **OSCIGateway** for the endpoints of a message **Initiator** and the final message destination **Recipient**[4]; messages named **osci:Request** and **osci:Response** are exchanged between those two endpoints. Optionally needed services can be acquired from intermediaries on the message route according to the underlying particular endpoint policies or scenario specific requirements.

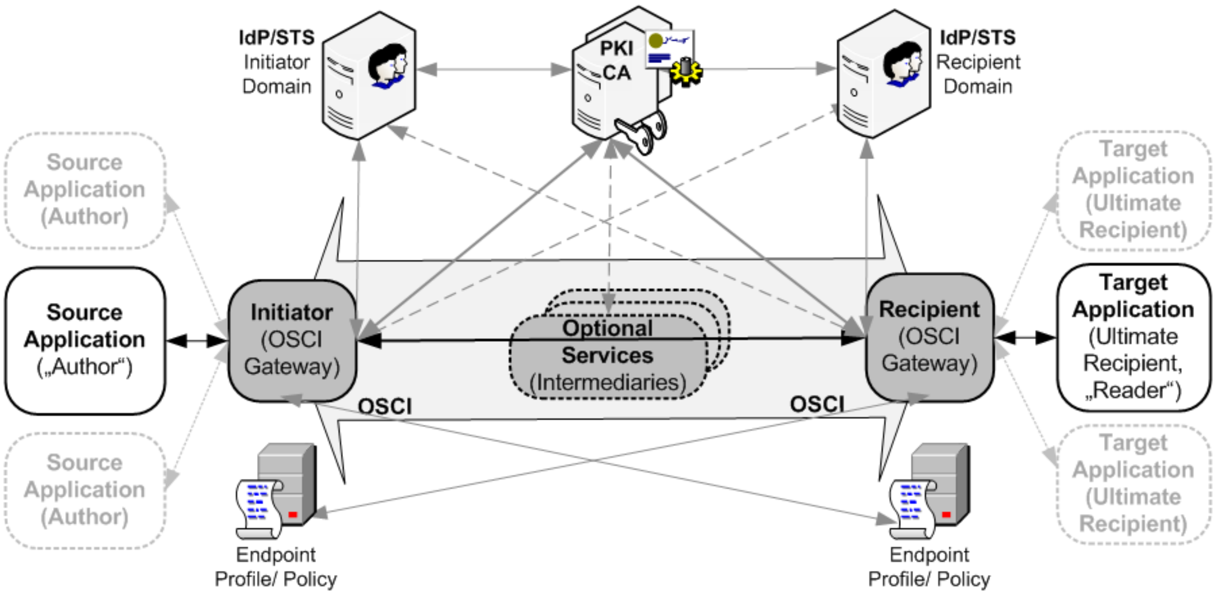
IdP/STS
Initiator Domain
PKI
CA
IdP/STS
Recipient Domain
Source Application (Author)
Target Application (Ultimate Recipient)
Source Application („Author“)
Initiator (OSCI Gateway)
Optional Services (Intermediaries)
Recipient (OSCI Gateway)
Target Application (Ultimate Recipient, „Reader“)
OSCI
OSCI
Source Application (Author)
Target Application (Ultimate Recipient)
Endpoint Profile/ Policy
Endpoint Profile/ Policy

Figure 2: Roles and supplemental Services Overview

An OSCI Gateway is a logical bundle of functionalities for composing/ sending respective receiving/ decomposing/ validating OSCI messages. Concrete implementations should allow decoupling of these functionalities, thus allowing to arrange them in a bus architecture with distinct nodes for stepwise

[3] For a possible solution of these issues as well as attribute services, see the concept of S.A.F.E. (Secure Access to Federated e-Justice/e-Government [SAFE] and the German administration service registry **Fehler! Verweisquelle konnte nicht gefunden werden.**.

[4] With message Initiator and Recipient, we follow the nomenclature choosen in SOAP Message Security 1.1 [WSS]

processing. Like this, an institution operating OSCI Gateways would be able to arrange constituents of an OSCI Gateway according the compliances of their network zones layout.

Source Applications work on content data, designated for **Target Application**s – so far out of scope from OSCI2. But for the possible needs of digital signature, end-to-end encryption and application-level handshake, some functions are needed here, which MUST be made available by an OSCI Gateway implementation[5]:

- Signing of content data, including visualization; optional time stamping
- Signature verification including access to validity verification of X509-Certificates
- Profile/policy information of *Target Application* respective *Source Application* including WSDL for addressing information, content level message exchange, X509-Certificates to be used for end-to-end encryption, digital signature requirements on content data level
- Encryption and decryption functionalities.

Following functions of OSCI Gateways are covered by several implementations of the WS-Stack available today:

The OSCI Gateway in role Initiator composes the communication data and initiates the transport:

- Access to policy of targeted message endpoint
- Acquirement of security token outlined in policy (request to a STS)
- Placement of addressing-data according to targeted endpoint, possible headers and routing information for intermediary service nodes to be passed
- Generation and placement of unique message-id
- If applicable: placement of correlation information to foregoing messages, generally to be triggered by Source- / Target Applications
- Signature, encryption on transport level
- Reliable message delivery
- Receiving and validating response messages including receipts - steps done like by an OSCI Gateway in role Recipient.

The OSCI Gateway in role Recipient:

- Receives (point-to-point) or pulls (inbound request in hold in a message relay, which is explained below) the message
- Decryption and check of communication data, identification and authorization of Initiator of incoming / pulled message
- Delivery of content data and validation information to *Target Application*
- Getting content-response from *Target Application*, composing and initiating response, steps done like by the OSCI Gateway in role Initiator.

Functionalities not at all or only partially addressed by the actual WS-Stack follow. As an anchor for intended interoperability with WS-Stack implementations not able to serve functionalities outlined here, service requests for these extensions may be marked optional on protocol level given by standard SOAP mechanisms.

One of the characteristics of OSCI2 is the support of communication traceability and provableness. An OSCI Gateway must have the possibility to request receipts of message delivery from logical nodes on the message path as well as final message acceptance by the targeted endpoint. For the latter receipt

[5] As far as needed for the secure transport functionality and access to endpoint policy information. Third party products may be used for content data signing, signature verification and encryption/decryption.

– defined as "ReceptionReceipt" later on in this document – a role **Ultimate Recipient** is introduced which corresponds to the Ultimate Receiver of [SOAP12]. This is the endpoint the – eventually end-to-end encrypted - SOAP body is targeted to. An Ultimate Recipient may be positioned behind the transport OSCI Gateway of the Recipient which is only processing the SOAP header blocks.

An OSCI Gateway therefore additionally must

- have the possibility to request receipts of message delivery for outgoing messages from logical nodes on the message path as well as final message acceptance by the targeted endpoint;
- be able to issue such receipts, if requested in incoming messages.

2.2.3 Message Relaying

OSCI2 supports endpoint-to-endpoint communication as well as communication via a relay ("Message-Box Service", short name **MsgBox** in OSCI2). The latter is only applicable for asynchronous message exchange patterns, as the intended Recipient has to pull the message from this service instance. As of today, it’s rather the exception that services of administrations can be reached online; customers of administration are supposed to drive their electronic communication in a mostly sporadic way. This leads to the requirement of MsgBox services for fully asynchronous message exchange. An additional advantage of involving such a relay in the message path is often seen in the fact, that the Recipient must open the network connection and in general is protected from incoming connections.

Hence messages either are sent to the Recipient in a point-to-point manner or to his MsgBox, a MsgBox acts as OSCI/SOAP endpoint, too.

For support of communication traceability and provableness, a MsgBox service must be able to receipt messages passing; as of its relaying behaviour it must be traceable when messages are accepted as well when they are pulled from the Recipient node.

2.2.4 Verifiability of Message Exchange

Different requirements on traceability and provableness of message exchange are defined for messages of type osci:Request and osci:Response. As acknowledgement of technical delivery is always implied on base of the underlying transport protocol http, accurate service of these requirements leads to three types of receipts on message delivery and reception, which can be demanded by Initiator and Recipient instances:

- **DeliveryReceipt –** *what* has been delivered *when* to *which* node; this is a receipt which every logical SOAP/OSCI node on a message path MUST be able to generate at time of acceptance of message pushed to this node. ***If requested, a DeliveryReceipt is always delivered to the foregoing node as header block in the SOAP response delivered in the network backchannel***. Generally spoken, the *"what"* is cryptographic information about the submitted opaque body – which in case of using MsgBox services on the message route must be encrypted data. The DeliveryReceipt has to be signed by its producing instance; regarding the *"what"* a **`ds:Reference`**[6] to the SOAP body block MUST be included in the signature applied by the receipting instance. If present, body block transport encryption has to be stripped off preceding receipt generation. The signature in additions covers delivery time as well as addressing data carried in the SOAP header. Like this, every logical SOAP/OSCI node is able to prove what he has successfully transmitted when to which endpoint. In case the body is end-to-end encrypted for the Ultimate Recipient or a targeted application behind the transport gateway of the Recipient, we strongly recommend the Initiator respective Source Application SHOULD additionally

[6] For definition see [XMLDSIG]

encrypt the SOAP body for themselves and save a copy of the message to be able to decrypt the content in case of possible later verification needs.

- **ReceptionReceipt –** this receipt can only be requested from the Ultimate Recipient instance of a message. Content of this receipt is nearly the same as denoted for the DeliveryReceipt with the difference, that the Recipient instance has to include a **`ds:Reference`** pointing to the decrypted body of the received message in the signature of the ReceptionReceipt. The ReceptionReceipt is delivered in the SOAP body of a separate osci:Request to the endpoint to be exposed in the demand for this ReceptionReceipt – which in general should be the MsgBox of the requesting Initiator (or again, the one of the Recipient node in case of response messages).
- **FetchedNotification** – a MsgBox service can be requested by the Initiator to signal the fact that the intended recipient is pulling the message from his MsgBox instance. This notification gives an additional control to the Initiator, whether there is any activity concerning the once submitted osci:Request.

2.2.5 Message Exchange Patterns

The following sequence diagrams illustrate the different osci:Request/ osci:Response MEPs including respectively possible receipt generation and delivery. For sake of clarity, all additional needed message exchanges with services like STS instances are omitted.

This MEP overview only considers the different possible ways of osci:Request delivery from an Initiator node to the Recipient and the ways back possible for the osci:Response. As communication verifiability must be possible for response messages, too, the sequence diagrams apply accordingly for osci:Response messages.

The diagrams in the following subchapters show every possible asynchronous delivery of receipts as targeted to the MsgBox instance of the Initiator. In fact, this target could be any Initiators specialized node able to accept and consume such type of messages. The same is true for asynchronous delivery of OSCIReponses.

2.2.5.1 Initiator to Recipient synchronous point-to-point

This MEP must be used if the response of the Target Application addressed is to be expected in the network connection backchannel of an osci:Request.

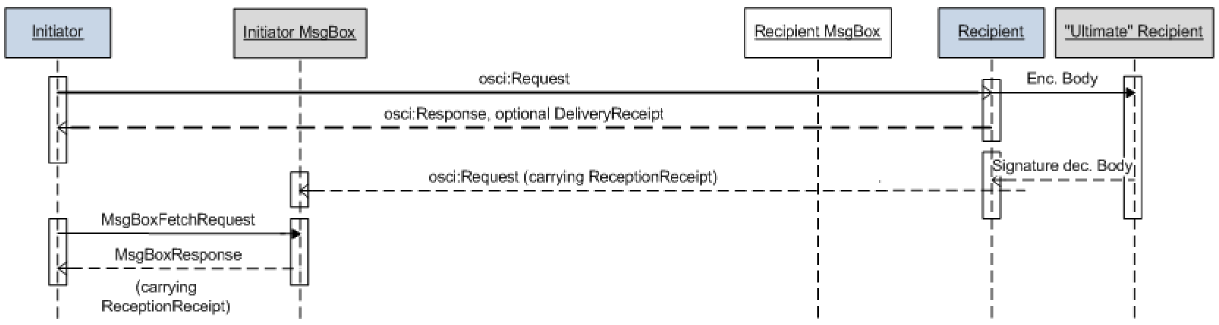

Figure 3: Sequence: Synchronous MEP

If requested by the Initiator of an osci:Request message, a DeliveryReceipt is delivered as SOAP header block of the resulting osci:Response in the network connection backchannel of the osci:Request. A possible response generated immediately by the Target Application is placed in the SOAP body block. If a ReceptionReceipt was demanded, it can not be generated before decryption of the osci:Requests body; the ReceptionReceipt is delivered in the body of new osci:Request message.

Similar is the MEP, where the osci:Request is sent to the Recipient node directly but the response is delivered decoupled. In this case, the osci:Response contains an empty SOAP body (if no error occurred) and the response is sent in the SOAP body block of a separate osci:Request. The target destination of this osci:Request must be ascertained as "ReplyTo"-address by the Initiator in the initial osci:Request.

2.2.5.2 Initiator to Recipients MsgBox, asynchronous Response

The osci:Request is delivered to the Recipients MsgBox. If requested, a DeliveryReceipt generated by the MsgBox endpoint is delivered as SOAP header block in the network connection backchannel of the osci:Request.

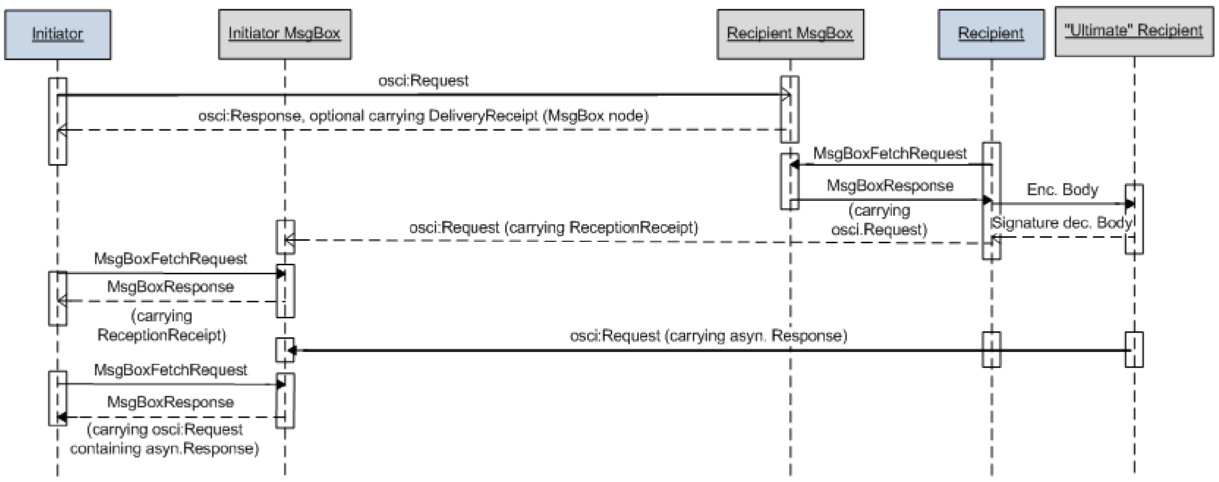

Figure 4: Sequence: MEP Initiator sends request to Recipients MsgBox

The Recipient must pull the osci:Request from his MsgBox (MsgBoxFetchRequest). If a ReceptionReceipt was demanded, it can be generated now after decryption of the osci:Request's body. It is delivered in the body of new SOAP message.

The osci:Response to the request is generated in a totally decoupled way. Once generated, it is delivered to the Initiator MsgBox.

2.2.6 Dedicated Dispatcher Service

Outbound message routing is a functionality which usually should be provided as integrated part of an OSCI Gateway.

If needed in concrete architectural concepts, it is possible to define a further logical node **Dispatcher** A Dispatcher service instance may be used by endpoints Initiator or Recipient as a transfer point, where outbound messages are routed to for delegation of delivery processing in a decoupled mode. Receipts for message dispatching may be gained from those nodes, too.

As this role and related MEPs are not part of the core OSCI Transport specification, see appendices to this document describing concrete application scenarios. Request SOAP header blocks for receipting the dispatching process - **DispatchedReceiptDemand** and resulting **DispatchedReceipt**s - are defined in an own namespace similar to the OSCI constructs for DeliveryReceipts.

2.3 Addressing Messages

The OASIS specification Web Services Addressing [WSA] is incorporated with profilings outlined in this document. Every destination endpoint may define types of business scenarios it is able to accept messages for, which is addressed by `wsa:ReferenceParameters` in a concrete message. In addition a set of metadata information items are defined. The mechanisms described in Web Services Addressing Metadata [WSAM] and Web Services Policy Attachment [WSPA] are used to include WSDL metadata in endpoint references. The SOAP binding for Web Services Addressing specified in [WSASOAP] is incorporated with the restriction to the SOAP version 1.2 binding.

Support of Web Services Reliable Messaging [WSRM] is optional for implementations conformant to OSCI2, as sufficient reliability of delivery is seen to be given by the binding to http(s) and the OSCI2 feature of receipting message delivery and reception.

As an optional feature, implementations may support anonymous endpoints for Service Requestors as described in Web Services Make Connection [WSMC]. It defines a mechanism to uniquely identify non-addressable endpoints, and a mechanism by which messages destined for those endpoints can be delivered. For sake of interoberabilty, a master profiling will be made regarding the mechanisms described there, in addition defining minimum requirements for authentication and authorization suitable for them[7].

2.4 Token for Authentication and Authorization

As already mentioned in chapter [2.2.1], access to nodes in OSCI2 based communication must be secured on base of sufficient authentication and authorization. In general, the token push model is used for authentication purposes, SAML-Token must be carried along with messages, which express authentication and authorization properties. Here, the specifications [SAML2] and [SAMLAC] are incorporated and profiled.

2.5 Securing Messages

Hence OSCI2 messaging is not more than SOAP messaging with certain specific extensions, the mechanisms specified in SOAP Message Security 1.1 [WSS] are the base of secured message exchange. Communication nodes express their security needs in form of policies, as described in WS Security Policy [WSSP]. OSCI2 defines templates for security policies to fulfil minimal security requirements. Based on those templates, more restrictive policies may be defined following the needs of concrete scenarios. On the other hand, it may be possible to drive OSCI2 based communication on base of more or less lax security policies, if security is given by other conditions like message exchange in protected closed networks.

Mechanisms defined by WS Secure Conversation [WSSC] MUST be supported by implementations conformant to OSCI2. The use of WS Secure Conversation is strongly recommende for scenarios where exchange of sequences of request-/responses messages between to endpoints must be expected. I. e., this in general applies to MsgBox access by the Recipient.

2.6 Validation of X509-Certificates

It MUST be possible to carry X509-Certificates used in the SOAP body block of a message on SOAP header block level. Like this – even when the body is encrypted for the Ultimate Recipient of a message – it is possible to check the validity of those X509-Certificates on the message route by nodes offering services to link to CAs (or nodes bridging to such services).

OSCI2 defines a SOAP custom header for carrying X509-Certificates and details of usage in the message body block parts where they where used.

OSCI2 ascertains the format of carrying validation results as defined in the XML Key Management Specification XKMS2/XKISS [XKMS]. Few extensions to the XKMS Validate Results are defined covering following aspects to be included in the certificate validation process:

- requirements of the German Signature Law and –Ordinance
- rankings for algorithm suitability
- rankings of certificate quality / cryptographic service providers ("Trusted Service Lists").

[7] This will be one part of the profilings addendum successively published from mid 2009 on

As this XKMS-Profiling is an ongoing process in the context of EU projects, we refer to specifications produced there.

2.7 Message Types

2.7.1 WS-Trust Messages

For acquisition, validation and other handling of security tokens, OSCI2 conformant implementations must support message types defined by WS-Trust [WST] and the token formats profiled here:

- Request Security Token (RST) – message to be send to STS of IdPs with following bindings defined by WS-Trust:
    - **Issuance** – request to issue a token. Based on the credential provided/proven in the request, a new token is issued, possibly with new proof information.
    - **Cancel** - when a previously issued token is no longer needed, the Cancel binding can be used to cancel the token, terminating its use.
    - **Validate** – The validity of the specified security token is evaluated and a result is returned. The result may be a status, a new token, or both.
- Request Security Token Response (RSTR) – response message of STS to return requested security tokens and/or STS operational results.

Token following the SAML specification version 2.0 [SAML2] must be supported and carried along with OSCI message types described below.

Communication nodes should expose token requirement details and issuer addresses in their WSDL following the specification WS Security Policy [WSSP].

Role-based transient identity mapping is used for cross-domain authentication and authorization. It is up to the Initiator to gain the required security tokens needed for service consumption in a foreign trust domain. To make federation work, IdPs of involved domains must have established a relationship of trust. WS-Federation [WSFED] specifies mechanisms and metadata elements to be able to use security tokens across domain boundaries.

The figure below illustrates the general scenario:

- Requestor (at Initiator OSCIGateway) acquires a SAML-Token from the IdP of his trust domain (STS in TD-I) for authentication at IdP in trust domain of the Recipient (STS TD-R).
- Based on this token for IdP/TD-R, STS of IdP in TD-R issues a SAML-Token for authentication and authorization at Recipient OSCIGateway (entry point of Service Provider).
- This SAML-Token for Recipient Gateway (Rec-GW) is pushed as part of the Initiators osci:Request message.

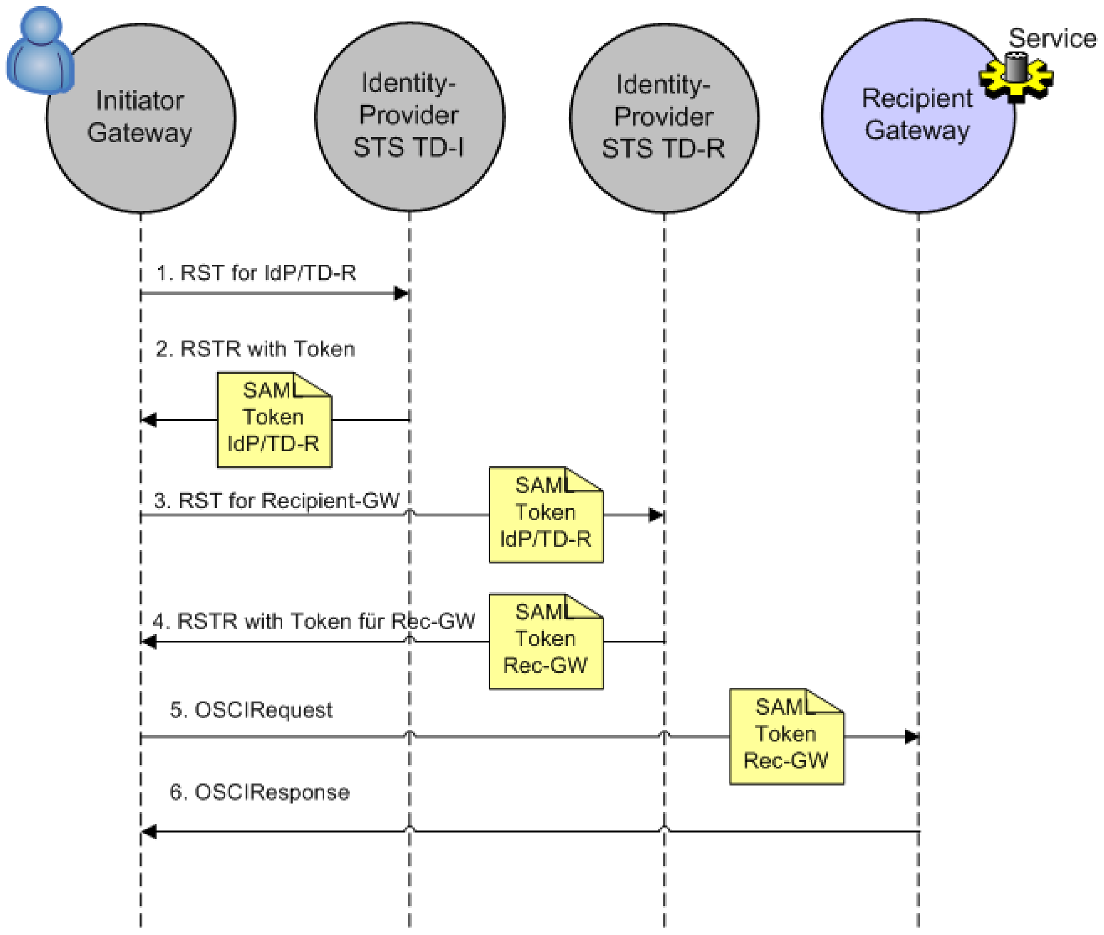

Figure 5: Role-based federation in the push-model

As profilings on WS-Trust and related specifications and for details to SAML-Token are used in the context of OSCI Transport, we incorporate those outlined in the concept S.A.F.E. (Secure Access to Federated e-Justice/e-Government [SAFE]). This is only one possible profiling, other profilings conformant to WS-Trust and SAML may be used.

As WS-Federation is in ongoing OASIS specification process[8] while compiling OSCI2, parts related to means of WS-Federation will be included in OSCI2 at the time WS-Federation in Version 1.2 will be stable[9]. This will only have the impact to extend the capabilities of endpoints concerning acquirement of SAML token; other functionalities described here will not be affected.

2.7.2 OSCI Messages

OSCI messages have to be seen as logical constructs, which differ in constituent blocks of SOAP headers and body. Headers following WS-Addressing, WS-Security including SAML-Token are always implied.

2.7.2.1 osci:Request /-Response

Message exchange with OSCI communication partners is initiated by an **osci:Request**, which always is responded with an **osci:Response** in the network connection backchannel. An osci:Request always carries Content Data in the SOAP body to a Target Application, an osci:Response may carry a SOAP body with the response Content Data produced by the Target Application (respective Source Application in conversational request-/response MEPs).

[8] The actual state of works done by the OASIS Web Services Federation Technical Committee is avaible at http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsfed

[9] Expected during 2009

Asynchronous Content Data responses are delivered as osci:Request messages with the **`wsa:RelatesTo`** header element pointing to the **`wsa:MessageID`**(s) of the related osci:Request message(s).

Both osci:Request and osci:Response may carry custom headers for receipt demands, X509-Certificates to be validated and corresponding XKMS Validate Reponse(s) – the latter header MAY have been inserted on the message route by any node able and willing to validate certificates. Both message types may carry a custom header with a time stamp signalling when a message can be seen as obsolete from point of view of message producer (in regular, only asynchronous MEPs are addressed here). When an osci:Request is targeted to the Recipient using his MsgBox instance, additional time stamps are inserted in this custom header, like time of entry into this node and time when the message was initially pulled out.

An osci:Response carries the **`wsa:RelatesTo`** header element pointing to the **`wsa:MessageID`** of the corresponding osci:Request and – if demanded – the DeliveryReceipt as custom header.

Message of type osci:Request are used to deliver ReceptionReceipts and as well as possible fault messages asynchronously, each of which are placed in the SOAP body block.

2.7.2.2 Message Box Access

Special message types are defined for selecting and pulling out messages of MsgBox instances as well as to gain status lists describing selectable MsgBox contents[10]. Like every OSCI2 based communication, access to MsgBox instances requires the mechanisms of WS-Addressing and WS-Security as well as authentication, whereby for the latter strong mechanisms are needed for MsgBox access.

For both **MsgBoxFetchRequest** (for pulling messages) and **MsgBoxStatusListRequest** (for gaining message status lists) no custom headers are foreseen. Both message types only differ in the URI value of the **`wsa:Action`** header element, which establishes the request type.

For both message types, the same SOAP body block structure is used to define message selection criteria. In this version of OSCI2, selection arguments are restricted to logical expression including message time stamps, **`wsa:MessageID`** and **`wsa:RelatesTo`**. A more generic filtering on SOAP header elements based on XPATH expressions is foreseen for a later version of OSCI2.

Statuses of those requests are delivered in the SOAP header block of the correlating **MsgBoxResponse**, while the requested content - pulled messages respective status lists - are delivered in the SOAP body block of the MsgBoxResponse.

A status list contains major SOAP header element contents as well as the size of selected messages.

A MsgBox search result list may span more items than deliverable in one response: A status list length can be restricted and messages pulled for must be delivered one by one. To be able to iterate on search result lists, a message type **MsgBoxGetNextRequest** is defined. Besides a dedicated URI in the header element **`wsa:Action`** such a request carries relationship information to the iteration initiating request of type MsgBoxFetchRequest respective MsgBoxStatusListRequest.

A pulling node can close an iteration process using the special message type **MsgBoxCloseRequest**. Again, this request carries a dedicated URI in the header element **`wsa:Action`** and in his SOAP body relationships information to the request initiating the iteration.

Successful reception of pulled messages must explicitly be confirmed by the Recipient. This can be done using foreseen elements in the body of subsequent MsgBoxGetNextRequests or by providing a

10 Parts of the contructs defined here resemble to those of WS Reliable Messaging. In contrast to WS Reliable Messaging, here the Initiator of the communication is in the message receiving role and must confirm reception. Thus, WS Reliable Messaging was seen to be not applicable for the purpose of MsgBox message pulling.

list of `wsa:MessageID'`s whose successful reception shall be confirmed in a dedicated body element of a MsgBoxCloseRequest.

Confirmed messages MUST be marked as pulled by a MsgBox instance. There is no special request type designed to delete messages in a MsgBox; message retention periods and rules for deletion of messages are considered to be subject to terms of service conditions of such an instance.

2.8 Supplemental Services for Source/Target Applications

An OSCI Gateway MUST be able to support all functionalities concerning message transport defined in OSCI2. Hence to integrate smooth into application scenarios, some functionalities for Source/Target Applications SHOULD be provided by OSCI Gateway implementations.

OSCI2 defines no interface between OSCI Gateways and Source/Target Applications; this is a matter of concrete implementations.

One bundle of functionalities to be considered here is access to endpoint service descriptions and policies. Handshake on structure of Content Data exchanged between Source/Target-Applications is obvious; further on some of the information exposed in the OSCI2 specific policy are needed as base for end-to-end encryption or triggering valid requests for time stamp quality. As access to endpoint service descriptions is a core functionality needed by OSCI Gateways, endpoint properties needed by Source/Target Applications should be made accessible to them in a comfortable manner.

OSCI2 and/or the related implementations SHOULD support the creation and verification of advanced and qualified electronic signatures on Content Data level according to the Digital Signature Act[11]. In addition, support of End-to-End encryption of Content Data from Source to Target Application is required.

Thus, support for Source- and Target Applications concerning these issues is addressed by OSCI2. For sake of interoperability and usage comfort, conformant OSCI Gateway implementations should provide functionalities based on standards basically defined by CMS/PKCS#7, XML Encryption and XML Digital Signature.

Efforts towards more interoperability and ease of use are pushed in Germany and by running EU "Large Scale Projects" with focus on EU-wide harmonization. One mayor outcome of those activities is the "eCard-API Framework" [eCardAPI] published as Technical Directive by the German Federal Office for Information Security (BSI) in March 2008. This specification is based on international standards mentioned above and concretions produced at ETSI and the OASIS Digital Signature Services Technical Committee [DSS].

The eCard-API Framework specification is planned to be brought in as one cornerstone to the EU Community framework for electronic signatures.

As this framework also addresses functionalities exceeding the needs of interoperable digital signatures and encryption, a suitable subset is identified and formally specified by the German Common PKI Organization as "Part 7, Signature API“ of their specification assembly [COMMPKI].

Functionalities and interfaces defined here are incorporated by OSCI2 as recommended solution for the support of digital signatures and end-to-end encryption.

[11] Must not mandatory be provided as OSCI Gateway functionality. Dedicated products for signature production and verification may be used, too.

3 Specifications and Concepts incorporated by OSCI 2

[COMMPKI] Common PKI Specifications for interoperable Applications, Version 2.0, 20 January 2009; http://www.common-pki.org/uploads/media/Common-PKI_v2.0.pdf

[DSS] Digital Signature Service Core - Protocols, Elements, and Bindings Version 1.0, OASIS Standard, 11 April 2007; http://www.oasis-open.org/specs/index.php#dssv1.0

[eCardAPI] Das eCard-API-Framework (BSI TR-03112). Version 1.0, Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik), March 2008, http://www.bsi.de/literat/tr/tr03112/index.htm

[MTOM] SOAP Message Transmission Optimization Mechanism, W3C Recommendation 25 January 2005, http://www.w3.org/TR/soap12-mtom/

[SAFE] S.A.F.E. (Secure Access to Federated e-Justice/e-Government) / D.I.M. (Distributed Identity Management), Unterarbeitsgruppe SAFE der BLK Arbeitsgruppe ITStandards in der Justiz, April 2008, http://www.justiz.de/ERV/Grob-_und_Feinkonzept/index.php

[SAMLAC] Auhtentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf

[SAML2] Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005; http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[SOAP12] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation 27 April 2007, http://www.w3.org/TR/soap12-part1/

[WSA] Web Services Addressing 1.0 - Core, W3C Recommendation 9 May 2006, http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/

[WSAM] Web Services Addressing 1.0 - Metadata, W3C Proposed Recommendation 31 July 2007, http://www.w3.org/TR/ws-addr-metadata/

[WSASOAP] Web Services Addressing 1.0 – SOAP Binding, W3C Recommendation 9 May 2006, http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/

[WSDL] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation 26 June 2007, http://www.w3.org/TR/wsdl20/

[WSDL11] Web Services Description Language (WSDL) 1.1: W3C Note 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[WSFED] Web Services Federation Language (WS-Federation), public draft release Version 1.1, December 2006, http://specs.xmlsoap.org/ws/2006/12/federation/ws-federation.pdf

[WSI-Basic] WS-I Basic Profile 2.0, Working Group Draft, 2007-10-25, WEB SERVICES INTEROPERABILITY ORGANIZATION, http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html

[WSI-BP11] WS-I Basic Profile 1.1, Final Material, 2006-04-10, WEB SERVICES INTEROPERABILITY ORGANIZATION, http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[WSMC] Web Services Make Connection (WS MakeConnection), Version 1.0, OASIS Standard, 14 June 2007, http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01.pdf

[WSPA] Web Services Policy 1.5 - Attachment, W3C Recommendation, 4 September 2007; http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/

[WSPF] Web Services Policy 1.5 - Framework, W3C Recommendation, 4 September 2007; http://www.w3.org/TR/2007/REC-ws-policy-20070904/

[WSRM] Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1, OASIS Standard Specification incorporating approved Errata, 07 January 2008, http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-spec-os-01-e1.pdf

[WSRMP] Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.1 OASIS Standard Specification incorporating approved Errata, 07 January 2008, http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01-e1.pdf

[WSS] Web Services Security: SOAP Message Security 1.1, OASIS Standard Specification, 1 February 2006, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[WSSC] Web Services Secure Conversation 1.3, OASIS Standard, 1 March 2007, http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf

[WSSP] WS-SecurityPolicy 1.2, OASIS Standard 1 July 2007, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf

[WST] WS-Trust 1.3, OASIS Standard, 19 March 2007, http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf

[XAdES] European Telecommunications Standards Institute. ETSI TS 101 903: XML Advanced Electronic Signatures, V1.3.2 2006-03; http://webapp.etsi.org/action/PU/20060307/ts_101903v010302p.pdf.

[XENC] World Wide Web Consortium. XML Encryption Syntax and Processing, W3C Recommendation, 10.12.2002; http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[XKMS] XML Key Management Specification (XKMS 2.0) v2, W3C Recommendation, 28 June 2005, http://www.w3.org/TR/2005/REC-xkms2-20050628/

[XMLDSIG] World Wide Web Consortium. XML-Signature Syntax and Processing, W3C Recommendation (Second Edition), 10 June 2008; http://www.w3.org/TR/xmldsig-core/