Funktionsbeschreibung

OSCI2-StarterKit (Metro)

OSCI2-StarterKit, Release 1.0

Herausgegeben von der Freien Hansestadt Bremen (OSCI-Leitstelle)

Erstellt durch: bremen online services Entwicklungs- und Betriebsgesellschaft mbH & Co. KG

Inhaltsverzeichnis

1 Allgemeine Hinweise

Obwohl diese Produktdokumentation nach bestem Wissen und mit größter Sorgfalt erstellt wurde, können Fehler und Ungenauigkeiten nicht vollständig ausgeschlossen werden. Eine juristische Verantwortung oder Haftung für eventuell verbliebene fehlerhafte Angaben und deren Folgen wird nicht übernommen. Die in dieser Produktdokumentation enthaltenen Angaben spiegeln den aktuellen Entwicklungsstand wider und können ohne Ankündigung geändert werden. Künftige Auflagen können zusätzliche Informationen enthalten. Technische und typografische Fehler werden in künftigen Auflagen korrigiert.

2 Übersicht und Zielsetzung

Die Aufgabe des im Folgenden beschriebenen OSCI2-StarterKit besteht darin, Anwendungsprogrammen und Fachverfahren eine Softwarekomponente zur Verfügung zu stellen, mit deren Hilfe sie das OSCI 2 - Transportprotokoll zum Erzeugen und Empfangen von Nachrichten gemäß der OSCI 2 Spezifikation nutzen können. Das OSCI2-StarterKit soll eine möglichst einfache Integration von OSCI 2 in bestehende Systeme ermöglichen. Im Zusammenwirken mit den später noch näher zu erläuternden externen Modulen umfasst sie alle für Benutzer im Sinne der Spezifikation erforderlichen Funktionalitäten, nämlich Aufbau, Versand, Empfang, Speicherung, Ver- und Entschlüsselung sowie die mathematische Signaturprüfung sämtlicher Nachrichtentypen. Anwendungsentwickler sollen sich lediglich um den Aufbau der Inhaltsdatenstrukturen kümmern nicht aber die OSCI-Strukturen aufbauen müssen.

3 Beschreibung des Leistungsumfangs

Die OSCI-Spezifikation 2.0 ist eine Profilierung verschiedener Webservice-Standards (WS-Standards), die in Abbildung 1 dargestellt werden. Funktionalitäten, die durch die WS-Standards nicht abgebildet werden, sind im Bereich "OSCI-Extensions to WS-Stack" (grüner Kasten) verortet.

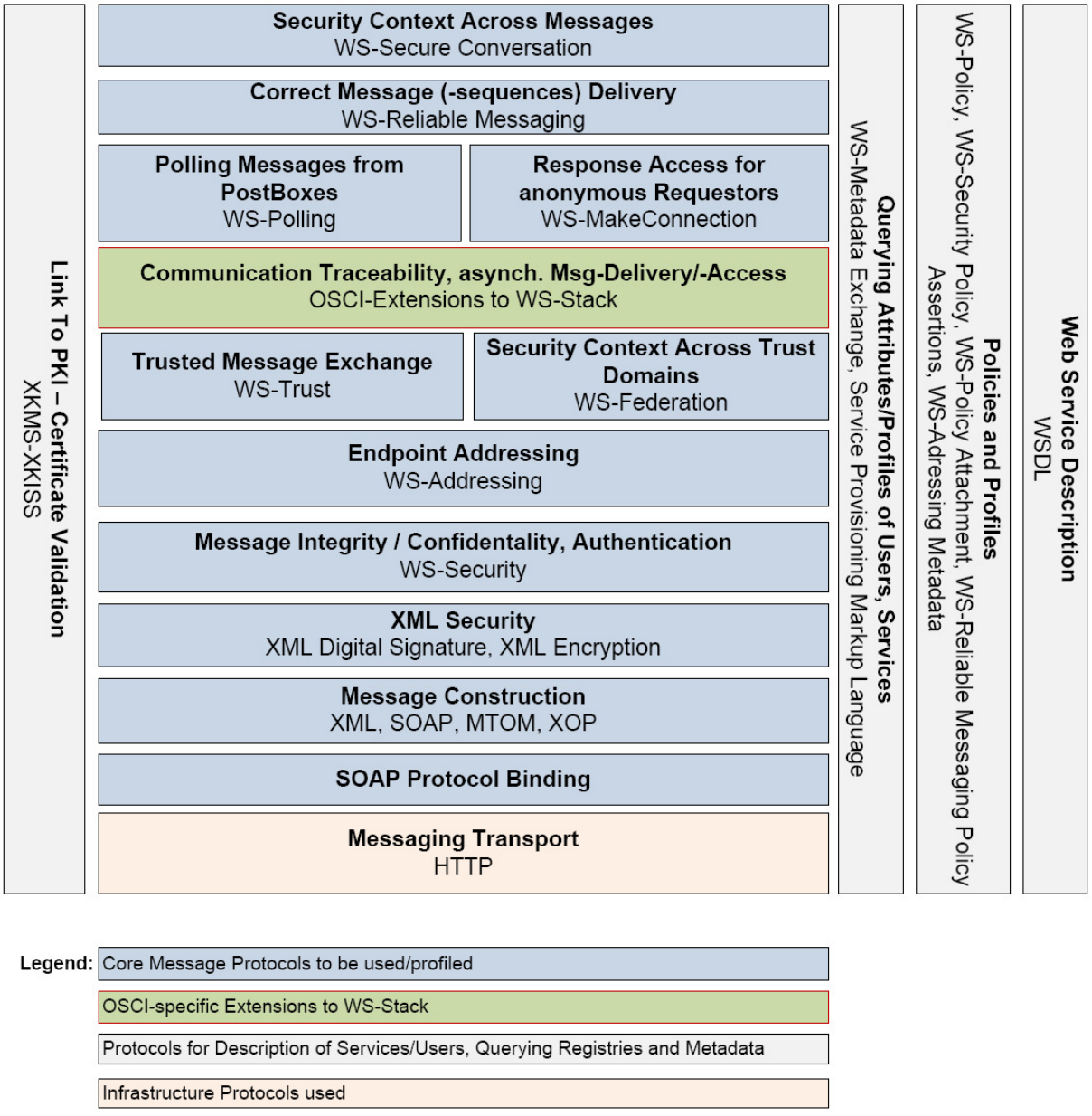

Abbildung 1: Übersicht der WS-Standards, die in OSCI 2 Verwendung finden

Für JAVA und C# gibt es Frameworks, die Entwickler bei der Implementierung von WS-Standards unterstützen. Diese Frameworks decken zum Teil auch Funktionen ab, die Bestandteil der "OSCI-Extensions to WS-Stack" sind. Ergänzungen der Basisklassen-bibliotheken schließen die verbleibende Lücke der Frameworks im Hinblick auf die "OSCI-Extensions to WS-Stack". Diese Ergänzungen der Basisklassenbibliotheken werden als Starter Kit bezeichnet.

Die bos KG bietet die Implementierung und die Bereitstellung solcher Starter Kits für die Frameworks METRO in der Version 1.4 (Java) und .Net in der Version 3.5 (C#) mit Windows Communication Foundation (WCF) an.

Die Starter Kits in Verbindung mit den Frameworks sollen Anwendungshersteller dabei unterstützten, OSCI 2.0 zu implementieren. Zum Funktionsumfang der Starter Kits gehören:

- Aufbauen, Parsen, Senden und Empfangen von spezifikationskonformen Nachrichten (nur Transport),
- Klassen zur Erstellung von Message Objects und OSCI Message Types gemäß. Spezifikation,
- Klassen zur Erstellung von OSCI-Fragmenten und SOAP-Headern,
- Klassen zum Hinzufügen von Signaturen und erzeugen von Verschlüsselungen auf Transportebene gemäß Policy,
- Setter-Methoden zur Übergabe von Inhaltsdaten (Body),
- Setter- und Getter-Methoden für die Nutzung von SAML und OneTimeToken (Authentisierungstoken),
- Schnittstelle für synchrone Nachrichten,
- Methoden zum Hashen und Signieren von Quittungen,
- Methoden, die Policies einer WSDL auswerten können, um zu entscheiden, ob ein SAML-Token eingeholt werden muss oder ob mit einer Local Domain gesprochen wird,
- Getter-Methoden, die eine WSDL auswerten können, um ein TypeOfBusinessSzenario auszuwählen.,
- Samples für synchrone und asynchrone Kommunikationsszenarien.

Nicht zum Lieferumfang gehören:

- Die Implementierung von Postfachfunktionen (eine Implementierung erfolgt nur testweise im Rahmen der Bereitstellung einer Testserverkomponente),
- die Bereitstellung von Methoden zur Zertifikatsvalidierung,
- Methoden zur Inhaltsdatenbearbeitung (Anbringen von Signaturen, Verschlüsselung),
- die Erzeugung von Prüfprotokollen,
- die Bereitstellung einer Kartenansteuerung,
- die Konfiguration über XML-Dateien,
- die Implementierung von CallbackHandlern (Umsetzung der kryptografischen Funktionalitäten, die in den Policies der jeweiligen WSDL der Inhaltsdaten und des Users beschrieben sind)

Die Nutzungs- und Verwertungsrechte einschließlich der Rechte am Quellcode gelten analog zur Erstellung der OSCI-Transport-Bibliothek 1.2. Es gelten die Lizenzbedingungen der Bremer Lizenz für freie Software.

4 Wichtige Änderungen zur OSCI 1.2-Version

Für diejenigen die mit der OSCI 1.2-Version vertraut sind, ist dieses Kapitel sehr wertvoll, da die wichtigsten Änderungen kurz zusammengefasst werden. Leser, die mit der OSCI-Version-1.2 nicht vertraut sind, können dieses Kapitel überspringen.

Einführung des "Home Intermediär"

Durch die Einführung des Home Intermediär können jetzt Nachrichten, Rückantworten und auch Quittungen von einer MsgBox abgeholt werden. Bei OSCI 1.2 musste zum Abholen von Rückantworten und Quittungen meistens ein entfernter Intermediär abgefragt werden. Dies führt zu komplizierten "Sender/Empfänger"-Konfigurationen und der Nutzer musste sehr viele Intermediäre abfragen, um an alle Quittungen und Rückantworten zu gelangen. Das führte dazu, dass Nutzer über E-Mail informiert wurden mussten. Zusätzlich führte das Abfragen vieler Intermediäre zu erheblichen Performanceverlusten. Jetzt hat der Nutzer eine engere Beziehung zu seinem MsgBox-Betreiber, da dieser sein einziger Ansprechpartner für abzuholende Nachrichten ist.

Berücksichtigung aktueller internationaler Standards (WS-Stack)

Die OSCI 1.2-Spezifikation wurde zu einer Zeit geschrieben, in der die Festschreibung der meisten WS-*Spezifikationen noch nicht vollendet war und es noch keine implementierenden Frameworks dafür gab. So wurden vielen Funktionen, die jetzt durch veröffentlichte Spezifikationen erledigt werden, in der OSCI 1.2-Version speziell definiert und auch implementiert. Hier nun eine Gegenüberstellung der Funktionen und der Umsetzung/ Definitionen in den Spezifikationen.

Beschreibung	**OSCI 1.2**	**OSCI 2**
Vertraulichkeit/ Transportsicherheit	XML- Encryption XML-Signature	WS-Security
Authentisierung	Verschlüsselungs-Token	WS-Trust, WS-Federation
Dialogabsicherung	DialogHandler mit Challenge /Response, ConversationId und SequenceNumber	WS-SecureConversation Derived Keys
Absicherung von Nachrichtenfolgen	-	WS-Reliable Message
Adressierung	Verschlüsselungszertifikat MessageID Intermediärs URL	WS-Addressing WS-Make Connection
Beschreibung der Anforderungen	-	WS-Policy WS-Security Policy WSDL

Zwei-Stationen-Kommunikation

OSCI 2 erlaubt die direkte Kommunikation zwischen Sender und Empfänger ohne einen Vermittler (OSCI-Intermediär). Dies erspart den Aufbau der Nachrichten zwischen Intermediär und Backend und vermindert somit die Komplexität und erhöht die Performanz.

Authentisierung über STS

Bei OSCI 1.2 wurde die Authentisierung ausschließlich über das Verschlüsselungszertifikat des Senders sichergestellt. OSCI 2 bietet die Möglichkeit, die Authentisierung über im Webservicebereich übliche SAML Token durchzuführen.

- Username/Password, SAML, Auth. Zertifikate oder Kerberos
- Geschlossene Benutzergruppen (keine Bindung an das Verschlüsselungszertifikat)

Erweiterte Quittungsmechanismen

Die Quittungen werden, sofern sie nicht direkt im Backchannel übermittelt werden, an den "Home Intermediär" übermittelt. Zu den zwei aus OSCI 1.2 bekannten Benachrichtigen Delivery-Receipt (Eingang der Nachricht) und Fetch-Notification-Receipt (erste Abholung im MsgBox-Betrieb) gibt es bei OSCI 2 noch eine weitere Quittung, die direkt vom Empfänger ausgestellt wird (Reception-Receipt). Bei dieser Quittung müssen die Inhaltsdaten entschlüsselt sein, und der Empfänger einer Nachricht quittiert mit einem Timestamp den empfangenen Inhalt.

Erweiterte Sucheigenschaften

In OSCI 2 gibt es durch den veränderten Authentisierungs- und Adressierungsmechanismus ganz andere Möglichkeiten nach Nachrichten zu suchen. Selektiver Zugriff auf Postfächer durch folgende Parameter:

- Empfänger,
- bestimmte Nachrichtentypen (TypeOfBusinessScenario),
- Zeitraum,
- MessageID,
- RelatesTo,
- beliebige Erweiterungen.

Definition von Anforderungen

Bisher gab es keine Unterstützung bei der Definition von Anforderungen an einen Service (Backend oder MsgBox). Diese Anforderungen wurden zumeist verbal durch ein separates Dokument veröffentlicht. Die OSCI 2-Spezifikation ermöglicht es, Anforderungen an einen Service mittels einer WSDL mit Policies zu beschreiben. Dies gilt sowohl für die Transportinformationen, als auch für die Inhaltsdaten. Im Idealfall können diese WSDLs und Policies maschinell ausgewertet werden, so dass Clients automatisiert erstellt werden können, um gültige Anfragen an den Service zu stellen.

5 Funktionsübersicht

Das OSCI2-StarterKit ist eine API für Anwendungsentwickler, die den Zugang zu einer OSCI 2-Infrastruktur ermöglichen soll. OSCI 2 ist eine definierte Folge von konkreten SOAP-Nachrichten. Die OSCI 2-Spezifikation definiert die notwendigen und optionalen SOAP-Header-Elemente sowie den SOAP-Body der OSCI-Nachrichten und beschreibt die Nutzung der integrierten WS-*Spezifikation. Das OSCI2-StarterKit soll genau bei den in Kapitel 3 genannten Aufgaben die Komplexität der OSCI 2-Spezifikation abnehmen und dem Anwendungsentwickler mit nur einigen Zeilen Code die Möglichkeit geben, OSCI-konforme Nachrichten zu erstellen oder abzuholen.

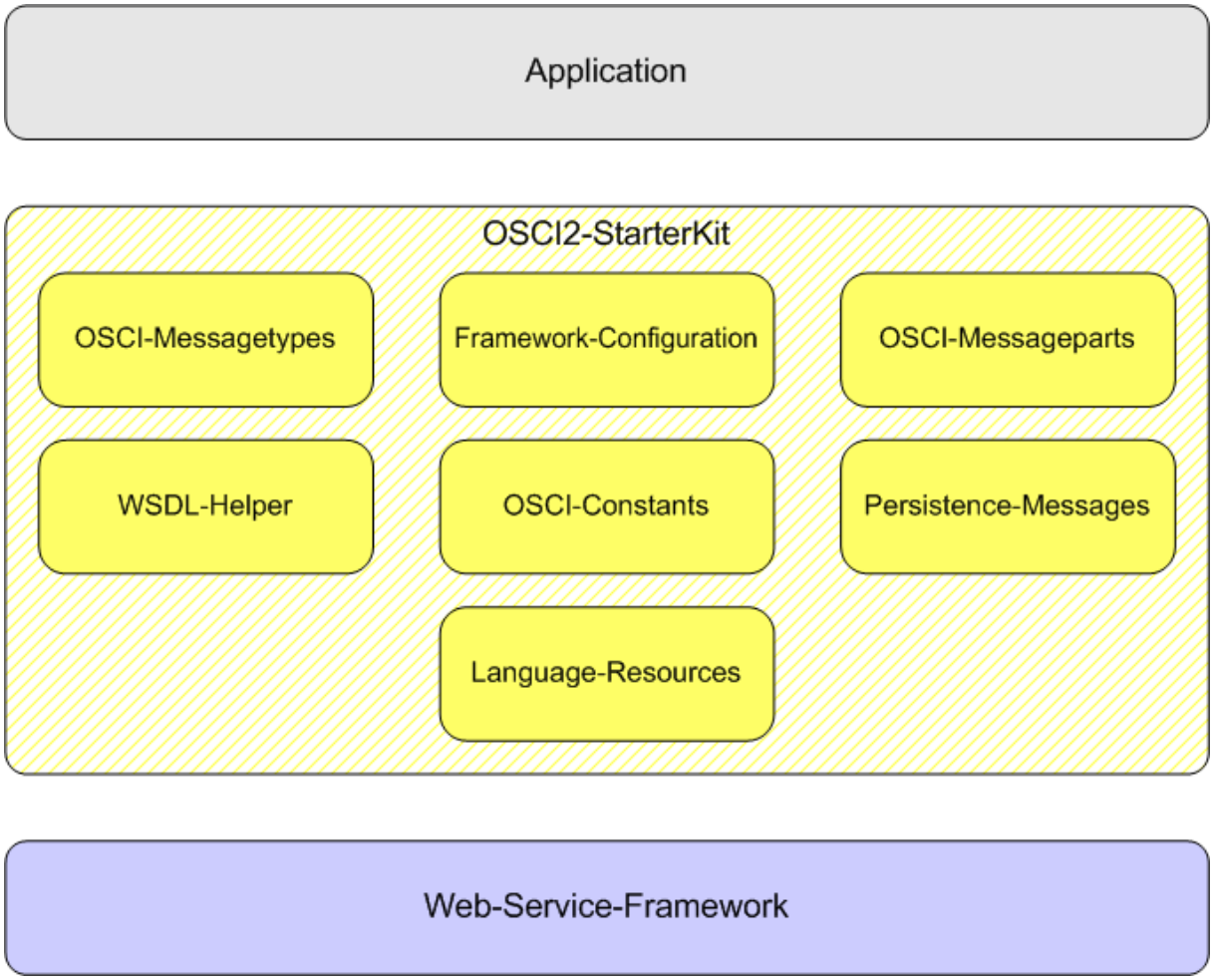

Abbildung 2: Architektur OSCI2-StarterKit

Der Nutzer des OSCI2-StarterKits findet sich in der Abbildung in der Schicht **Application** wieder. Diese Schicht spricht direkt mit dem OSCI2-StarterKit und indirekt über Konfigurationsdateien mit dem Webservice-Framework.

5.1 Application-Schicht

Die Application-Schicht ist für die Zusammenstellung der Inhaltsdaten zuständig. Meistens wird hier ein bestehendes Schema benutzt, um die Inhaltsdatenstruktur sicher zu stellen. Die Inhaltsdatenstruktur kann zumeist mit einer Inhaltsdaten-WSDL auf dem OSCI 2-Server der Empfängerstelle veröffentlicht werden. OSCI 2 definiert keine Struktur der Inhaltsdaten. Der

Nutzer einer OSCI-Infrastruktur kann beliebige Strukturen in den Body der SOAP-Nachricht einhängen. Für unstrukturierte Nachrichtenkommunikation bietet OSCI 2 eine Nachrichtenstruktur mit dem Namen `LetterStyle` an, die jeder Kommunikationspartner im OSCI 2-Umfeld verstehen sollte. OSCI 2 lässt nicht nur die Struktur der Inhaltsdaten offen, ebenfalls werden keine Aussagen über Signaturen und Verschlüsselungen getroffen. Dies ist auch der Grund, warum im Gegensatz zur OSCI 1.2-Version die Verarbeitung der Inhaltsdaten komplett aus der veröffentlichten Bibliothek herausgelöst wurde. Der Anwendungsentwickler stellt also die Struktur der Inhaltsdaten zusammen, bringt die Inhaltsdatensignaturen an und verschlüsselt sie. Der zusammengestellte Inhalt wird an das OSCI2-StarterKit zur Versendung in die OSCI 2-Infrastruktur übergeben.

Beim Empfang von OSCI Nachrichten muss die Application-Schicht die OSCI-Response-Nachricht auswerten und eventuelle Inhaltsdaten zur Weiterverarbeitung entnehmen.

5.2 OSCI2-StarterKit

Das OSCI2-StarterKit wird von der Application-Schicht benutzt, um die zu versendende OSCI-SOAP-Nachricht aufzubauen. Es wird konfiguriert, welche OSCI-Receipts verlangt werden, welche Zertifikate auf dem Transportweg geprüft werden sollen, wo eventuelle Rückantworten empfangen werden können und vieles mehr. Das OSCI2-StarterKit stellt zum Setzen dieser Parameter einfache Methoden auf einem OSCI-Nachrichtenobjekt zur Verfügung. Sobald alles konfiguriert ist, werden dem OSCI2-StarterKit die vorher aufgebauten Inhaltsdaten signiert und/oder verschlüsselt als Datenstrom zur Verfügung gestellt. Den Aufbau einer OSCI-konformen SOAP-Nachricht übernimmt dann das OSCI2-StarterKit. Die notwendigen Header-Elemente werden hinzugefügt, die Inhaltsdaten werden als Body hinzugefügt, und die Nachricht wird mit Zuhilfenahme des Webservice-Framework versendet.

Beim Empfang von Nachrichten analysiert das OSCI2-StarterKit die SOAP-Nachricht und wandelt die Header-Elemente in Objekte um, die von der Anwendungsschicht einfach ausgewertet werden können. Sofern die Response-Nachricht weitere Inhaltsdaten enthält, werden diese ebenfalls der Anwendungsschicht als Datenstrom zur Verfügung gestellt.

5.2.1 OSCI-Messagetypes

Die OSCI-Spezifikation definiert eine Vielzahl von SOAP-Messages mit ihren Optional- und Pflicht-Header-Elementen sowie den Body-Elementen. Für jede dieser SOAP-Nachrichten stellt das OSCI2-StarterKit eine Klasse zur Verfügung, die bei der Erstellung einer gültigen OSCI-Nachricht helfen soll. Es stehen folgende Klassen zur Verfügung.

Requests:

- `OSCIRequest`,
- `OSCIMsgBoxFetchRequest`,
- `OSCIMsgBoxStatusListRequest`,
- `OSCIMsgBoxCloseRequest`,
- `OSCIMsgBoxGetNextRequest`,

Responses:

- `OSCIResponse`,
- `OSCIMsgBoxFetchResponse`,
- `OSCIMsgBoxStatusListResponse`,

Jede Request-Klasse bietet Methoden zum Konfigurieren der OSCI-Nachrichten und "send*"-Methoden zum Versenden der Nachrichten an. Response-Nachrichten bieten zumeist nur Methoden zum Auswerten der Nachricht an.

Relevantes Package:

- `eu.osci.messagetypes`

5.2.2 OSCI-Messageparts

Sobald Nachrichtenbestandteile nicht direkt auf den Nachrichtenobjekten erstellt werden können, unterstützen die folgenden Klassen die Erstellung und Auswertung der Nachrichtenbestandteile:

- `JaxbContextHolder,`
- `OSCIFragmentCreator,`
- `OSCIFragmentParser.`

Zumeist wird die Klasse `OSCIFragmentCreator` benutzt, um Nachrichtenbestandteile auf den Nachrichtenobjekten setzen zu können.

Relevantes Package:

- `eu.osci.messageparts`

5.2.3 WSDL-Helper

Postfächer oder Services im OSCI 2-Kontext werden immer mit WSDLs beschrieben. Der Ablauf einer Kommunikation ist also folgender:

- Abrufen der WSDL der Endpoints,
- Auswerten der Informationen aus der WSDL,
- Aussuchen des Services der angesprochen werden soll (`TypeOfBusinessSzenario`, Port und Service aus der WSDL),
- Konfigurieren und Versenden der Nachricht.

Beim Abrufen, Auswerten und Aussuchen der WSDL und deren Eigenschaften unterstützt die Klasse `EndpointLocator`. Es wird lediglich eine URL zu einer WSDL benötigt. Die Klasse stellt Methoden zur Verfügung, um einfach einen `TypeOfBusinessSzenario` oder Port anzugeben, der für die OSCI 2-Kommunikation benutzt werden soll.

Relevantes Package:

- `eu.osci.helper`

5.2.4 OSCI-Constants

OSCI 2 bietet eine Vielzahl von Konstanten zu Elementnamen, WS-Addressing-Action und Fehlercodes. Hierfür bietet das OSCI2-StarterKit folgende Klassen an:

- `OSCIConstants`
- `OSCIException`
- `OSCIFault`
- `OSCIHeaderTag`

- `OSCIRequestAction`
- `AddressingHeaderTag`
- `ErrorTextTranslation`

Relevantes Package:

- `eu.osci`

5.2.5 Framework-Configuration

Zur Konfiguration des Webservice-Frameworks müssen XML-Dokumente des jeweiligen Frameworks konfiguriert, private und öffentliche Schlüssel sowie Credentials für die Authentisierung am Security-Token-Service bereitgehalten werden. Die Konfiguration dieser Frameworks wird in Beispielen dargestellt und beschrieben, siehe Kapitel 8.

5.2.6 Language-Resources

Zur Ausgabe wichtiger Meldung wie Fehlermeldungen oder Fortschrittsinformationen stellt das OSCI2-StarterKit die Möglichkeit einer Mehrsprachigkeit durch Java-Resource-Files zur Verfügung.

5.3 Webservice-Framework

Das Webservice-Framework bekommt die SOAP-Nachricht mit Header- und Body-Elementen und reichert sie um die benötigten Security-Informationen an. Das heißt, das Webservice-Framework bereitet die SOAP-Nachricht entsprechend der geforderten Policy der angesprochen WSDL des Endpoints auf. Es werden SAML-Token eingeholt, die Transportverschlüsselung vorgenommen und die Transportsignaturen angebracht. Um diese Aufgaben erfüllen zu können, werden weitere Parameter benötigt, wie z.B. Credential für das Einholen von SAML-Token, Token für die Signatur und Verschlüsselung sowie vertrauenswürdige Zertifikate. Wie diese konfiguriert werden, wird weiter unten erklärt. So aufbereitet wird die Nachricht dann an den adressierten Endpoint gesendet.

Beim Empfang der Rückantwort verhält es sich ähnlich. Das Webservice-Framework überprüft entsprechend der Policy aus der WSDL die eingehende Nachricht, löst die Transportverschlüsselung auf und prüft die Transportsignatur und die Struktur der SOAP-Nachricht. Nach erfolgreicher Prüfung der Nachricht wird die Kontrolle wieder an das OSCI2-StarterKit abgegeben.

6 Ablaufdiagramme

Die folgenden Sequenzdiagramme zeigen, wie wesentliche Abläufe mit dem OSCI2-StarterKit durchgeführt werden. Zu einigen Sequenzdiagrammen können Vorbedingungen erforderlich sein, um den entsprechenden Ablauf zu gewährleisten.

6.1 Konfiguration der Webservice-Frameworks

Das OSCI2-StarterKit baut auf Webservice-Frameworks auf. Für die Nutzung des OSCI2-StarterKit wird im Java Umfeld das Metro Framework von SUN und für .Net-Anwender das WCF-Framework von Microsoft benutzt. Um diese Frameworks parametrisieren zu können, bieten beide Frameworks die Möglichkeit, wichtige Parameter mit XML-Dateien zu konfigurieren. Es können z.B. Schlüssel für die Signatur und Verschlüsselung sowie Passwörter konfiguriert werden. Wie dies im Detail geschieht, kann in den Samples nachvollzogen werden.

6.2 Auswählen des korrekten OSCI 2-Services

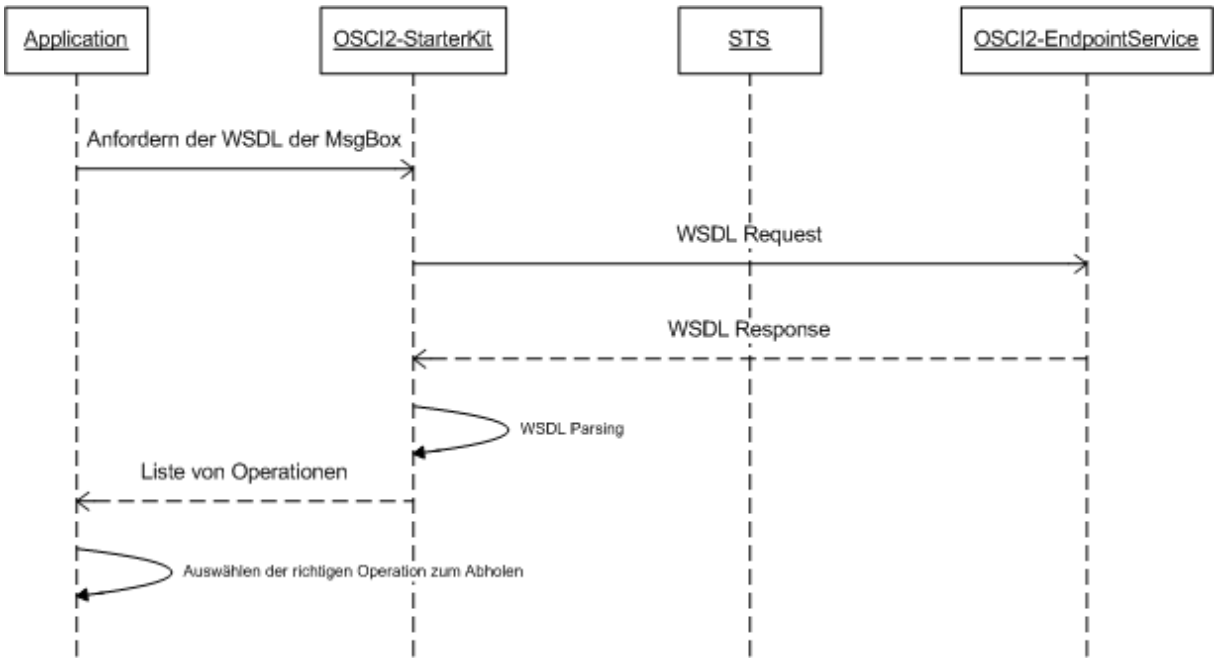

Abbildung 3: Vorbereitung der OSCI-Message

Bevor ein OSCI 2-Service angesprochen werden kann, muss der Benutzer den richtigen Dienst aussuchen. Hierfür wird die WSDL des Empfängers benötigt. Die URL zu der WSDL wird dem OSCI2-StarterKit übergeben. Dieser versucht, die WSDL zu parsen und stellt dem Benutzer Auswahlhilfen zur Verfügung, um den richtigen Dienst oder `TypeOfBusinessSzenario` zu finden.

Konkrete Implementierung dieser Sequenz

- Verzeichnis: `ClientScenarios` (siehe Kapitel 8),
- Klasse: `Send`,

- Methode: `getEndpoint`

Relevante Klassen

- `EndpointLocator`

6.3 Einfaches Versenden

Vorbedingungen

- Sequenzdiagramm 6.2

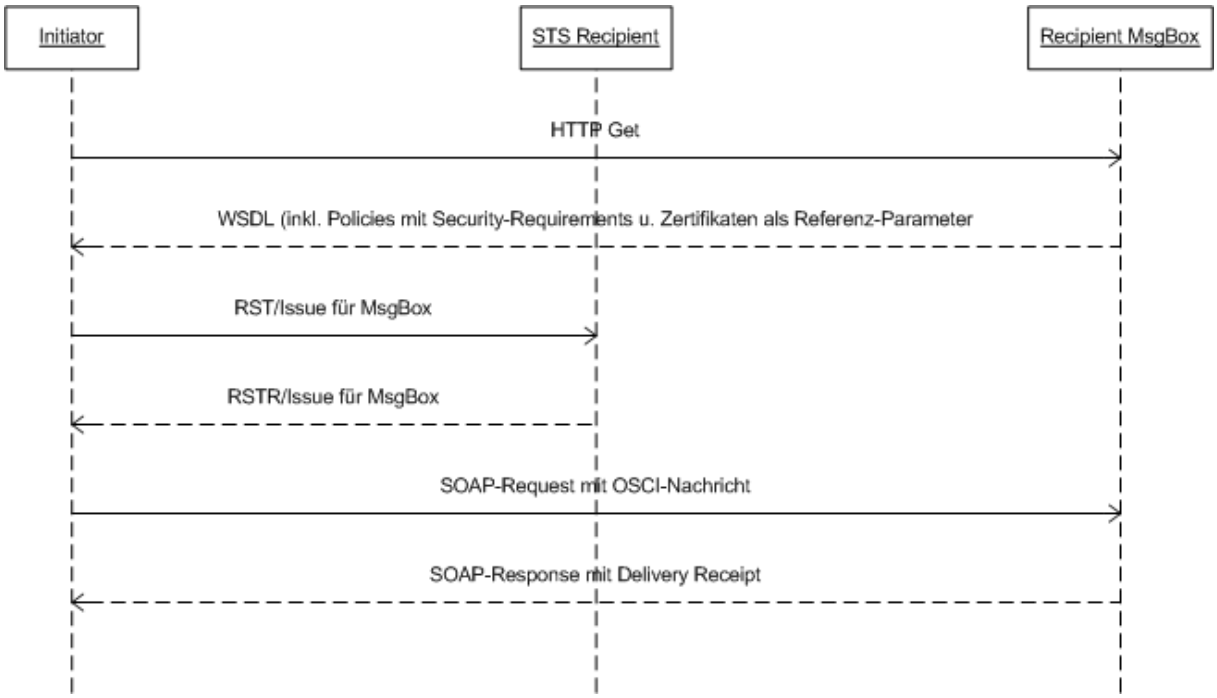

Abbildung 4: Senden an MsgBox

Das Versenden einer OSCI 2-Nachricht an eine MsgBox geschieht mit der Klasse OSCI-Request. Zuvor muss der richtige Service ausgesucht werden (siehe Abschnitt 6.2). Die Inhaltsdaten müssen vorbereitet sein, das heißt, dass die XML-Struktur entsprechend der Anforderungen aufgebaut, signiert und verschlüsselt sein muss. Mit dem Aufruf der `send`-Methode wird das unterliegende Framework vom OSCI2-StarterKit mit der Versendung der OSCI 2-SOAP-Nachricht beauftragt. Entsprechend der Konfiguration des Frameworks (siehe Abschnitt 6.1) und der Anforderungen des Services wird zuerst ein SAML-Token entsprechend den Anforderungen des Services eingeholt. Danach werden die Inhaltsdaten mit den konfigurierten OSCI 2-Header-Elementen und dem SAML-Token an die MsgBox des Empfängers geleitet. Sofern der Benutzer ein Delivery-Receipt beantragt hat, wird dieser sofort im Rückkanal zurückgegeben.

Konkrete Implementierung dieser Sequenz

- Verzeichnis: `ClientScenarios` (siehe Kapitel 8),
- Klasse: `Send`,
- Methode: `sendOSCI2Message`

Relevante Klassen

- `OSCIRequest,`
- `OSCIResponse`

6.4 Empfangen von Nachrichten

Vorbedingungen

- Sequenzdiagramm 6.2

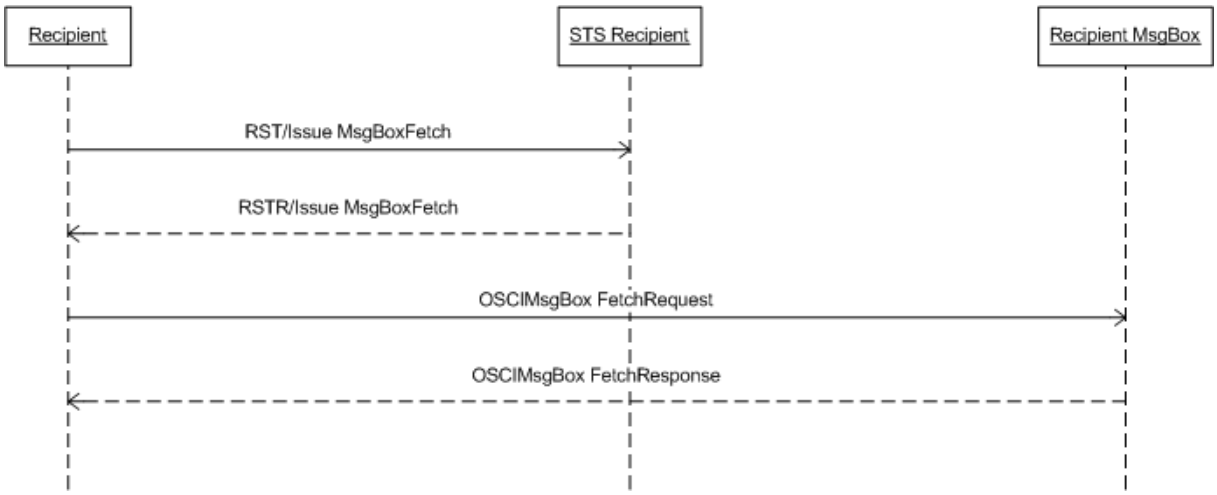

Abbildung 5: Abholen von der MsgBox

Zuvor muss der richtige Service ausgesucht werden (siehe Abschnitt 6.2); anschließend wird ein SAML-Token eingeholt (siehe Abschnitt 6.3). Dies geschieht im Normalfall vom Framework aus, mit vorheriger Abfrage eines Secure-Token-Service. Welcher Secure-Token-Service benutzt wird, kann das Framework der WSDL des Endpunktes entnehmen.

Zum Abholen von Nachrichten wird die Klasse `OSCIMsgBoxFetchRequest` benötigt. Diese Klasse muss zuvor mit den Suchkriterien bestückt werden. Danach wird die `sendRequest`-Methode aufgerufen, um die MsgBox nach Nachrichten abzufragen. In der Rückantwort der MsgBox befindet sich, sofern vorhanden, die erste OSCI 2-Nachricht, die den Suchkriterien entspricht.

Konkrete Implementierung dieser Sequenz

- Verzeichnis: `ClientScenarios` (siehe Kapitel 8)
- Klasse: `SendReceive`
- Methode: `fetchAllOSCI2Messages`

Relevante Klassen

- `OSCIMsgBoxFetchRequest`
- `OSCIMsgBoxFetchResponse`
- `OSCIMsgBoxCloseRequest`
- `OSCIMsgBoxCloseResponse`

6.5 Synchrones Szenario

Vorbedingungen

- Sequenzdiagramm 6.2

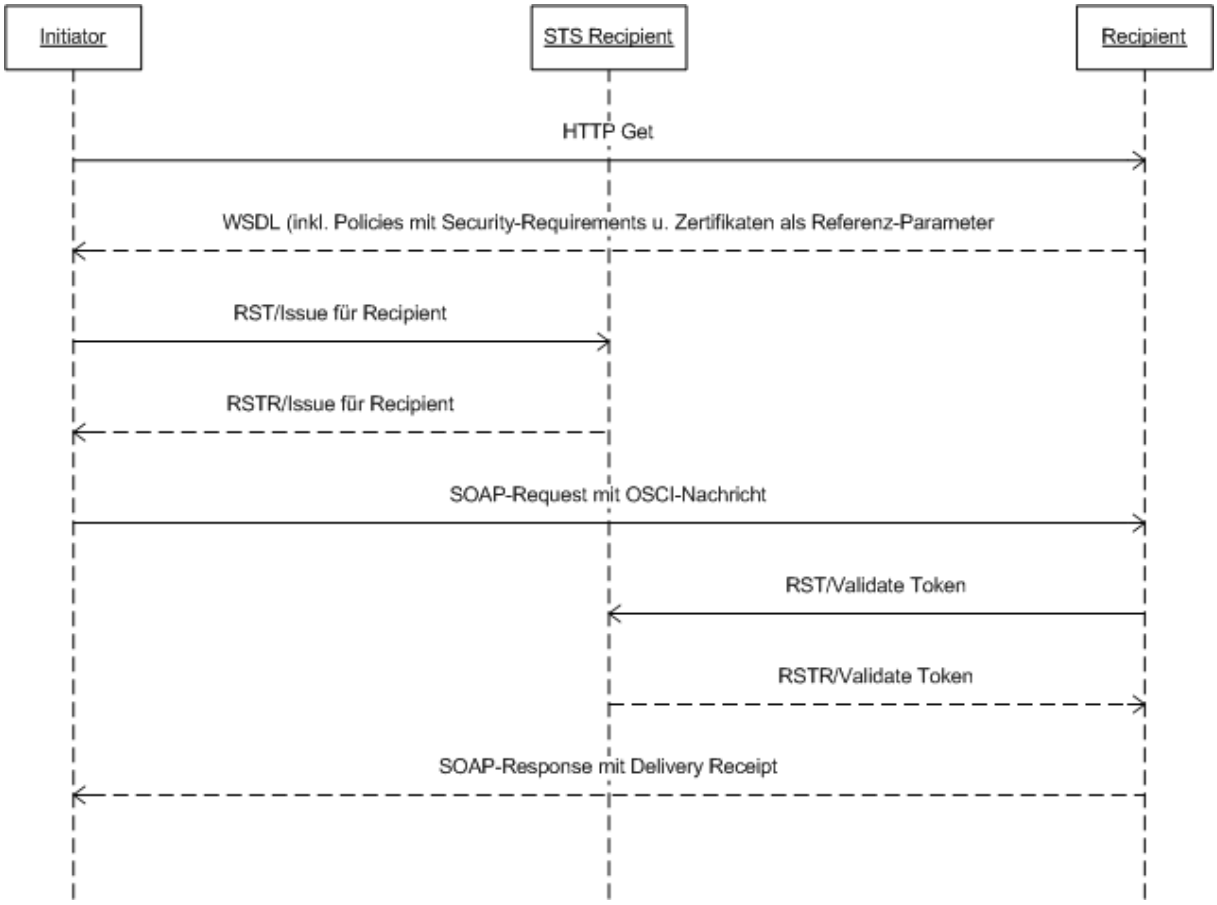

Abbildung 6: Zustellung synchron an Recipient nur mit DeliveryReceipt, geschlossene Benutzergruppe

Der Ablauf bei einem synchronen Szenario entspricht genau dem Szenario zum Versenden zur MsgBox. Der Unterschied ist hier, dass der Kommunikationspartner keine MsgBox, sondern ein synchroner Empfänger ist.

Da lediglich die MsgBox `FetchNotification`-Nachrichten verschickt, ist es nicht sinnvoll `FetchNotificationDemand`-Nachrichten an einen synchronen Empfänger zu senden.

Konkrete Implementierung dieser Sequenz

- Verzeichnis: `ClientScenarios` (siehe Kapitel 8),
- Klasse: `Synchron`,
- Methode: `sendMessage`

Relevante Klassen

- `OSCIRequest`
- `OSCIResponse`

6.6 Receipt

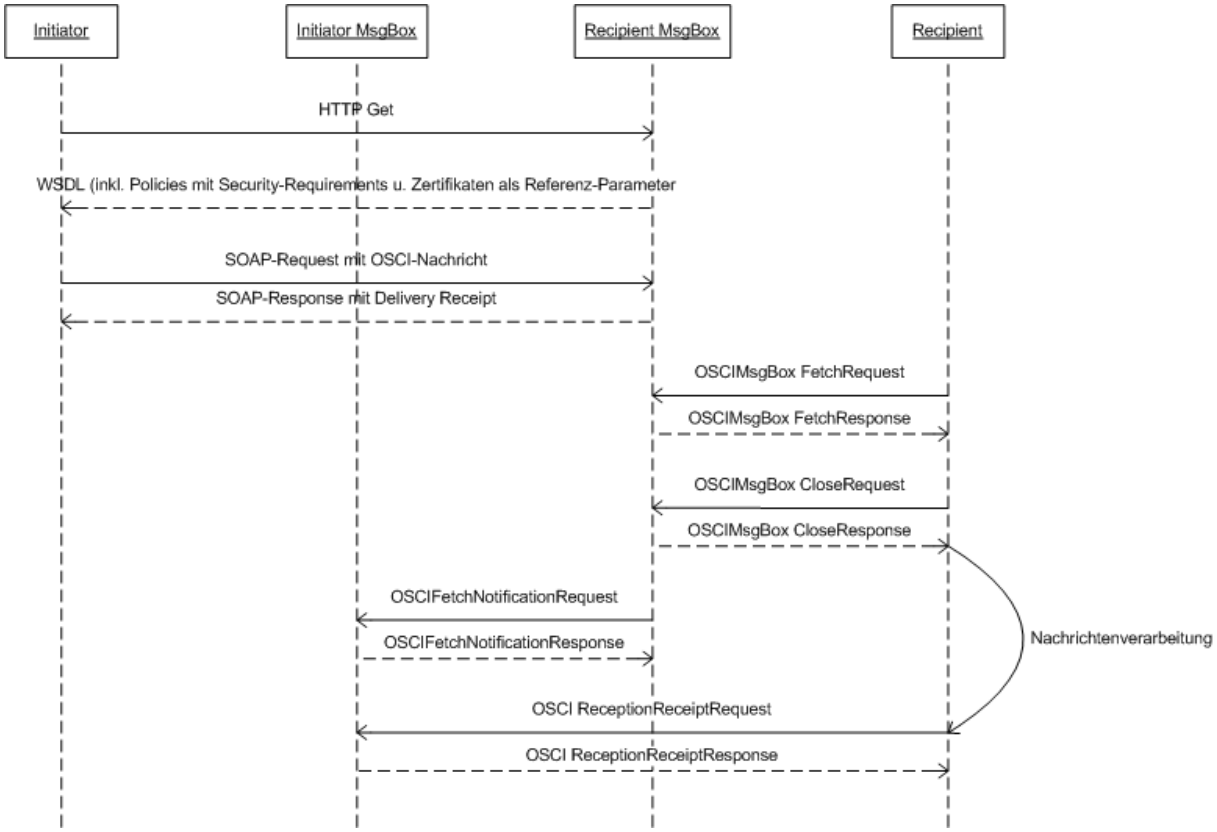

Abbildung 7: ReceptionReceipt

Im Ablaufdiagramm wird das Versenden der Receipts verdeutlicht. Das erste Receipt (Delivery-Receipt) wird im Beispiel direkt nach dem Eingang auf der Recipient-MessageBox erstellt. Das Fetch-Notification-Receipt wird nach erfolgreicher Abholung von der Recipient-MessageBox erstellt und das Reception-Receipt wird vom Empfänger nach erfolgreicher Entschlüsselung und Prüfung versendet.

Relevante Klassen

- `OSCIRequest`
- `OSCIResponse`

6.7 XKMS

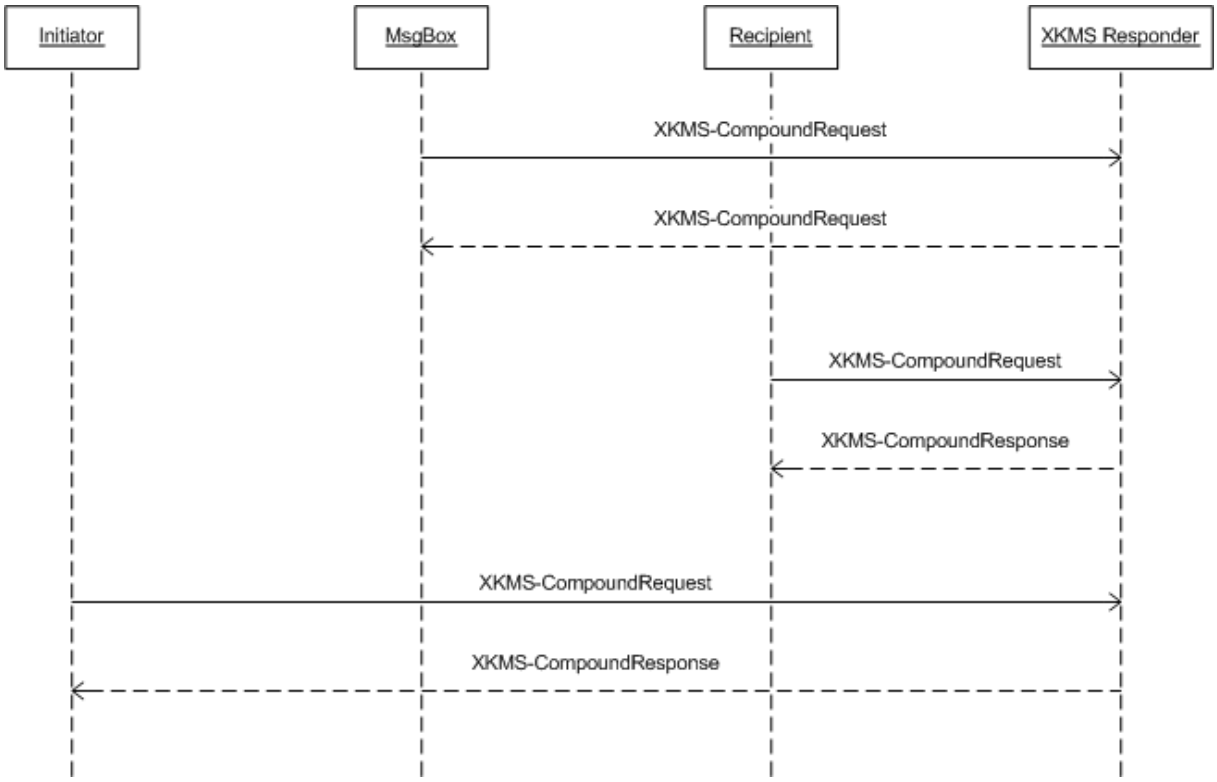

Abbildung 8: Prüfen von Zertifikaten zu jedem Zeitpunkt

Die Prüfung der Zertifikate kann bei OSCI 2 zu jedem Zeitpunkt durchgeführt und in die OSCI 2-Nachricht eingestellt werden. Sowohl der Sender (MsgBox) als auch der Empfänger kann die Zertifikatsprüfung übernehmen. Das OSCI2-StarterKit stellt für das Hinzufügen der XKMS-Prüfergebnisse lediglich die Getter und Setter zur Verfügung. Der Aufruf der XKMS-Abfrage muss vom Benutzer selbst durchgeführt werden.

7 Fehlerbehandlung und Ereigniscodes

Fehler werden bei den OSCI-Nachrichtenobjekten durch die jeweilige, konkrete Exception ausgedrückt, wie z.B. `JAXBException`, `GeneralSecurityException` oder `Datatype-ConfigurationException`. Sollte bei der Kommunikation mit dem Endpoint ein Fehler auftreten, wird eine `OSCIException` ausgelöst, die die Inhalte der SOAP-Fehlermeldung enthält.

Die Klasse `OSCIException` bietet Getter zum Auswerten des SOAP-Fehlers:

- `getCode()`: Liefert den SOAP-Code "Receiver" oder "Sender"
- `getSubcode()`: Liefert die Kurzbeschreibung des Fehlers (z.B. "`Delivery-ReceiptChecksumInvalid`", "`MsgFrequencyLimitExceeded`" usw.)
- `getReason()`: Liefert Detailinformationen zum SubCode in englischer Sprache
- `getCause()`: Liefert die initiale Exception zur detaillierten Auswertung

Die Auflistung der Fehlercodes kann der OSCI 2-Spezifikation entnommen werden.

8 Beispiele

Zur Beschreibung der Nutzung des OSCI2-StarterKit werden Source-Code-Beispiele separat zur OSCI2-StarterKit Bibliothek ausgeliefert (`OSCI2-Samples.zip`).

Dieses Archiv beinhaltet thematisch sortierte Beispiele in folgenden Verzeichnissen:

- `ClientScenarios` einfaches Versenden und Abholen von OSCI-Nachrichten
- `ServerScenarios` Implementierung eines synchronen Empfängers

Des Weiteren befindet sich im Archiv ein Verzeichnis (`environment`), in dem Konfigurationsdateien und Zertifikate abgelegt sind, die zur Ausführung der Beispiele benötigt werden.

Außerdem beinhaltet das Archiv eine Übersichtsdatei (`Overview.txt`), die eine Beschreibung über notwendige Vorbedingungen, Umfang und Konfigurationen der Beispiele beinhaltet.

Die Beispielklassen selbst sind zusätzlich mit kurzen Kommentaren versehen, die entsprechende Methodenaufrufe erklären.

ClientScenarios

Mit diesen Szenarien können einfache Beispiele zum Senden und Empfangen von OSCI-Nachrichten nachvollzogen werden. Es werden Inhaltsdaten im "LetterStyle" zusammengestellt, signiert und verschlüsselt. Die Nachrichten werden dann sowohl an ein synchrones Szenario als auch an eine MsgBox gesendet.

Die Beispiele befinden sich im Verzeichnis:

```
<ZIP-Archiv>/ClientScenarios\src\eu\osci\samples\clientscenarios
```

ServerScenarios

Dieses Szenario ist für Entwickler eines synchronen Backends von Bedeutung. Es wird gezeigt, wie das Webservice-Framework benutzt wird, um OSCI-Nachrichten synchron anzunehmen, zu prüfen und Antworten zu generieren.

Die Beispiele befinden sich im Verzeichnis:

```
<ZIP-Archiv>/ServerScenarios\src\eu\osci\samples\serverscenarios
```

Test-Umgebung

Zum Testen der Beispiele stellt die OSCI-Leitstelle einen Server zur Verfügung, der die notwendigen Services und WSDLs bereitstellt. Die URL kann auf der Webseite für OSCI 2 eingesehen werden.

9 Abbildungsverzeichnis