Prozessbeschreibungen Datenaustausch mit der Wechselplattform

Datum: 14.11.2012

Version 1.0

Beteiligte Rollen / Komponenten (aus Kommunikationssicht):

- **Sender**: Organisation, die eine ebXML-Nachricht zur Wechselplattform (WP) oder direkt an den Empfänger versendet.
- **Empfänger**: Organisation, die eine ebXML-Nachricht von der WP oder vom Sender empfängt.
- **Teilnehmer** sind entweder Sender oder Empfänger.
- **Wechselplattform** (WP): Die WP wird von der APCS betrieben, um den Lieferanten-Wechselprozess bezüglich der Einhaltung von Fristen zu überprüfen. Die agiert technisch gegenüber einem Sender als Empfänger bzw. gegenüber einem Empfänger als Sender einer ebXML Nachricht.
- **ebXML-Nachricht**: Ein Geschäftsdokument, das bilateral vom Sender zum Empfänger übertragen wird.
- **ebMS** (ebXML Message Service): Hauptkomponente zur Übertragung von ebXML-Nachrichten, welche senderseitig die Validierung, Signierung, Kompression, Verschlüsselung und Versendung von Geschäftsdokumenten durchführt. Empfängerseitig erfolgt dies in entgegengesetzter Reihenfolge. Der êbMS befindet sich in der sicheren Zone des Netzwerkes eines Teilnehmers.
- **Nachricht:** setzt sich zusammen aus dem ebXML Envelope (grundsätzlich nicht verschlüsselt) und der Payload (verschlüsselt). Teile des ebXML-Envelopes können jedoch verschlüsselt sein (WP-spezifisch). Für die Payload wird beim WP-Prozess immer die Verschlüsselung aktiviert.
- **SIA (Single Internet Address):** Eine bei der Kommunikation transparente zentreale Komponente, über die alle Teilnehmer ihren IP-Datenstrom leiten. Die Verbindung zwischen einem Teilnehmer und dem SIA ist SSL-gesichert (verschlüsselt und authentisiert). Im SIA liegt der ebXML Envelope unverschlüsselt vor, die Payload ist für den SIA nicht entschlüsselbar. Hauptzweck des SIA ist, den physischen IP-Verkehr auf eine Adresse zu konzentrieren, so dass Teilnehmer nicht für die Adresse eines jeden Kommunikationspartners die Firewall öffnen müssen.

Generelle IKT-Sicherheitsanforderungen der Teilnehmer

Damit ein Teilnehmer den Lieferantenwechselprozess zukünftig über die Ponton Plattform abwickeln darf, muss er ein Mindestmaß an IKT-Sicherheit erfüllen. Zu diesem Mindestmaß zählen

- Etablierte IKT-Sicherheitsorganisation innerhalb der betroffenen Organisation

- Einsatz von Schutzkomponenten zur Erhöhung der IKT-Sicherheit (Installierte Malwarescanner auf den jeweiligen Arbeitsstation, welche am Datenaustausch beteiligt sind, Firewallkonzept,…)

Diese Anforderungen gelten auch für die WP.

Vorgaben an Datenaustausch für den Lieferantenwechselprozess

Bei der Abwicklung des Lieferantwechselprozesses sollten folgende Kriterien erfüllt sein:

- Verschlüsselung der Inhaltsdaten (enthält eigentliches Dokument, welches über die Wechselplattform ausgetauscht werden soll) sodass nur der Empfänger Zugriff auf die Inhaltsdaten hat.
- Ein Teil der ebXML Headerinformation wird ebenfalls verschlüsselt, auf der zentralen Plattform soll jedoch aus Gründen der Beweisbarkeit Zugriff darauf bestehen.
    - Dies ist der Fall durch eine spezielle Erweiterung der Mappinglogik
- Eine Verschlüsselung der Kommunikationsstrecken Sender – WP und WP – Empfänger erfolgt mit SSL.
- Signierung der Nachrichten durch die Teilnehmer erfolgt durch eine XML Signatur.

Ablauf einer ebXML-Übertragung „End-to-End“

Erfolgreiche Übertragung („Happy Path“)

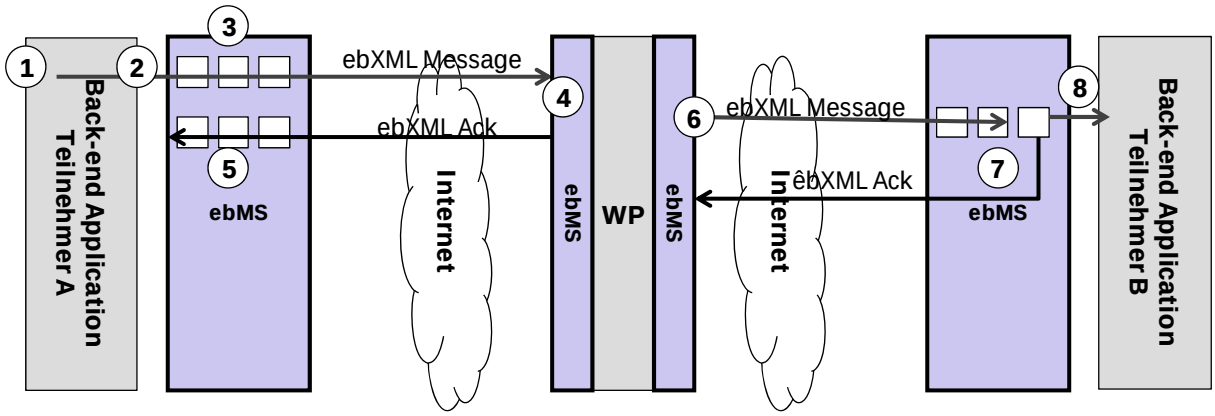
Für die folgende Beschreibung einer EDA-Datenübertragung wird angenommen, dass die beteiligten ebMS des Senders und des Empfängers mit den jeweiligen Back-end-Systemen verbunden sind. Dies kann – je nach Produkt – auf unterschiedliche Weise erfolgen:

- Durch die Verwendung eines „Hot Folders“ für eingehende und ausgehende Dokumente
- Durch die Verwendung einer produktspezifischen API
- Durch die Verwendung einer EDA-spezifischen WebServcie-Schnittstelle. Hierbei hat sowohl der ebMS als auch die Back-end-Anwendung ihren jeweiligen Teil der Services zu implementieren. Diese Schnittstelle ist in einem separaten Dokument spezifiziert.

Die nachfolgende Prozessbeschreibung konzentriert sich auf den Übertragungsweg zwischen Sender und Empfänger. Wo nötig wird auf die Web-Service-Schnittstelle Bezug genommen.

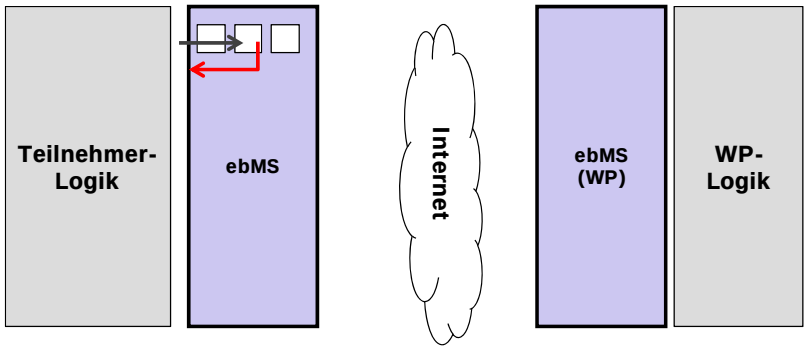
1. Die Back-end-Applikation des Senders erzeugt intern ein ebUtilities-Dokument und führt einen Aufruf der Methode sendMessage() beim WebServcie des ebMS durch. Dabei wird eine XML-Struktur als Parameter des WS-Aufrufs übergeben, die eingebettet das ebUtilities-Geschäftsdokument enthält. In anderen Prozessen als dem Wechselprozess (P2P-Übertragung zwischen EDA-Teilnehmern) muss dies jedoch nicht zwingend ein XML-Dokument sein (es kann sich beispielsweise auch um ein MSCONS- oder Binärdokument handeln).
2. Der ebMS extrahiert die zu übertragende Payload. Weitere Parameter, die evtl. mit dem WSDL-Aufruf erhalten wurden, werden ebenfalls im ebMS-Envelope für die Übertragung gesetzt.
3. Der ebMS führt verschiedene Prozessschritte für den Versand einer ebXML-Nachricht durch und versendet die Nachricht an die WP. Abweichend von der üblichen Verarbeitung erfolgen folgenden Schritte:

   a. Die Verschlüsselung der Payload erfolgt nicht für die WP, sondern für den Empfänger. Hierbei wird der AT-Code aus <Receiver> in <MarketParticipantDirectory> übernommen.
   b. Ein Teil des ebXML Headers wird für die WP verschlüsselt („Transaktionsdaten“). Dieser wird als zusammenhängende XML-Substruktur <MarketParticipantDirectory> aus dem Payload-Dokument kopiert.
   c. Als Empfänger der Übertragung (ebXML-Evelope-Element <To>) wird der AT-Code WP eingetragen.
4. Die WP verarbeitet die Nachricht und antwortet mit einem ebXML Ack, sobald die Nachricht erfolgreich validiert und persistiert werden konnte.
5. Das ebXML Ack wird vom ebMS des Senders verarbeitet und an die back-end-Applikation weiter geleitet.
6. Nach Schritt 4 verarbeitet die WP das Geschäftsdokument fachlich. Insbesondere leitet sie es weiter an den Empfänger. Dabei wird ein neuer ebXML-Envelope erzeugt und neben den Transaktionsdaten (<MarketParticipantDirectory> verschlüsselt für den Empfänger) noch der Empfangszeitstempel der WP eingetragen (als Teil von <HubDirectory>, ebenfalls verschlüsselt für den Empfänger). Diese Daten werden mit dem ebUtilities-Namespace (http://www.ebutilities.at/wechselprozess/1.0/MarketParticipantDirectory) in den ebXML-Envelope eingebettet, das Namespace-Kürzel ist „eda“. Der Zeitstempel der Wechselplattform wird mit dem Namespace http://www.ebutilities.at/wechselprozess/1.0/HubDirectory und dem Kürzel „wp“ eingebettet. Die WP versendet nun eine ebXML-Nachricht an den Empfänger mit beiden Erweiterungen des Envelopes. Diese Nachricht enthält:
   a. Die vom Sender empfangene Payload (verschlüsselt für den Empfänger) wird 1:1 durchgereicht ohne zusätzliches Validieren / Signieren Komprimieren / Verschlüsseln durch die WP.
   b. Die unveränderten Transaktionsdaten des Senders als Einbettung in den ebXML Envelope.
   c. Die Sektion <HubDirectory> mit ihren Subelementen der WP als Einbettung in den ebXML Envelope.
7. Der empfangende ebMS
   a. Validiert die XML-Signatur der WP
   b. entschlüsselt und dekomprimiert die Payload, validiert sie und persistiert das Geschäftsdokument.
   c. Die Payload wird zusammen mit den beiden
   d. Ein ebXML Ack wird an die WP zurück geliefert.
8. Der ebMS des Empfängers liest den Empfangszeitstempel der WP aus und gibt diesen zusammen mit der Payload und den Transaktionsdaten weiter an die Back-end-Applikation des Empfängers – im Falle der WebService-Schnittstelle durch den Aufruf ReceiveMessage().



Folgende **Fehlersituationen** sind denkbar und werden entsprechend behandelt:

1. **Der EbMS des Senders liefert eine Fehlermeldung bei der Vorbereitung für den Versand.**
   - Hierbei liefert die Adapter API eine Fehlermeldung an das Back-end.
   - Nach Fehlerbehebung muss die Back-end-Applikation die Übertragung wiederholen.



Mögliche Gründe für Fehlermeldungen vor Versand sind:

- Dokument konnte nicht Validiert werden
- Dokument konnte nicht signiert werden
- Dokument konnte nicht für den Empfänger verschlüsselt werden
- Sender- oder Empfänger-Codes sind nicht konfiguriert
- Weitere erforderliche Codes sind nicht vorhanden oder falsch

2. **Die Versendung der Nachricht an die WP scheitert.**

Hierbei versucht der EbMS des Senders entsprechend der einheitlich vereinbarten Wiederholversuche, die Nachricht erneut zu übertragen.

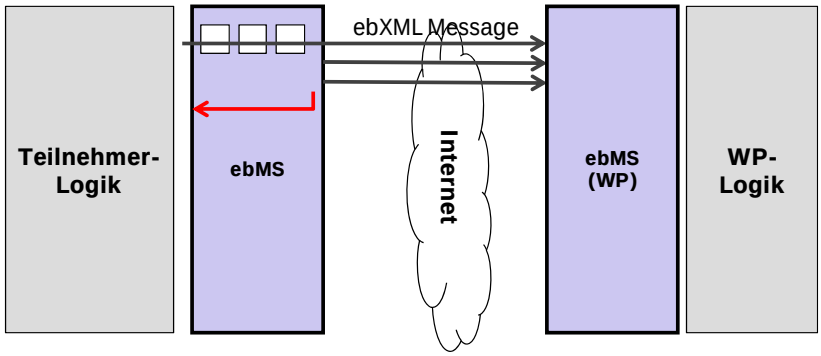
- NumberOfRetries = Wiederholversuche = 2,
- RetryInterval = Timeout-Spanne bis zum nächsten Wiederholversuch.

Dabei sollte das Wiederholmuster den Anforderungen der Wechselverordnung angepasst sein, d.h., die Anzahl der Wiederholversuche ist 2, die TimeToLive ist 15 Minuten:

- Erste Wiederholung: 1 Minute nach dem ersten Übertragungsversuch
- Zweite Wiederholung: 8 Minuten nach dem ersten Übertragungsversuch

- Abbruch der Übertragung: 15 Minuten nach dem ersten Übertragungsversuch.

Scheitern diese Übertragungsversuche, wird der Back-end-Applikation ein Fehler gemeldet. Nach Fehlerbehebung muss die Back-end-Applikation die Übertragung wiederholen.



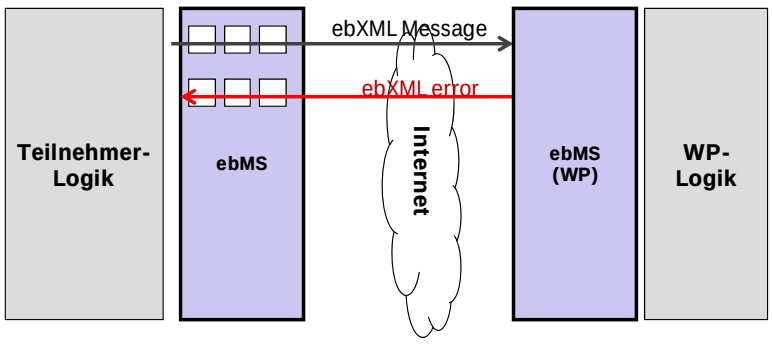
Gründe für einen Übertragungsfehler können sein:

- Der Kommunikationspartner ist nicht erreichbar
- Die eigene oder die gegenseitige Firewall ist falsch konfiguriert
- Probleme beim Aufbau einer SSL-Verbindung (Z.B. wegen abgelaufener Zertifikate)
- Der ebMS des Kommunikationspartners läuft nicht oder auf einer falschen IP-Adesse bzw. Portnummer.

3. Die WP liefert eine Fehlermeldung in Form einer ebXML ErrorMessage.

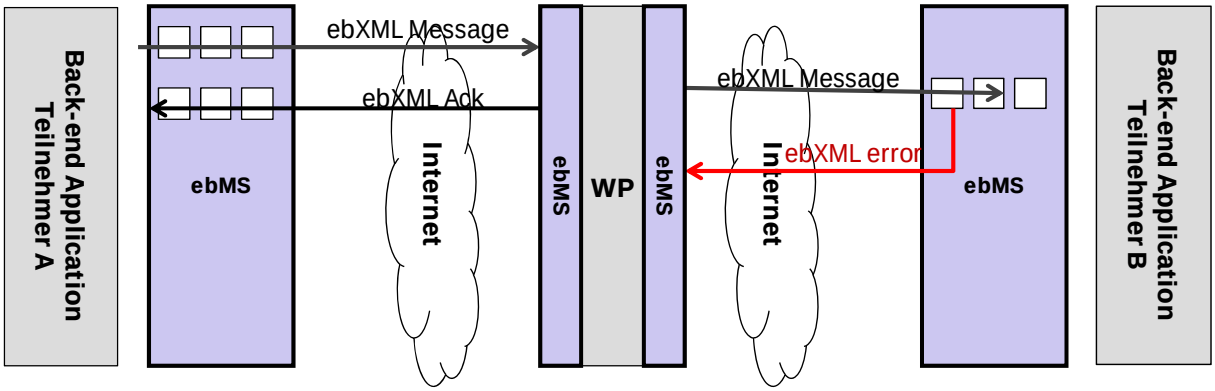
Die Fehlerursache kann hier beispielsweise sein:

- Validierungsfehler der Transaktionsdaten
- Fehler bei Überprüfung der Signatur (Zertifikat abgelaufen?)
- Fehler bei Entschlüsselung (z.B. Falscher Schlüssel bei Sender verwendet?)
- Time to Live abgelaufen
- Partner-ID unbekannt
- Fachliche Fehler, welche die WP überprüft



4. Beim Zustellen der Nachricht zum Endempfänger tritt ein Übertragungsfehler auf.

Ein solcher Übertragungsfehler sollte in der Sphäre der WP und des Empfängers gelöst werden. Ggf. durch Nachsenden der Nachricht von der WP an den Empfänger nach Behebung der Fehlerursache. Bei der Kommunikation zwischen WP und Empfänger können prinzipiell die gleichen technischen Fehler auftreten wie oben in den Schritten 1-3 beschrieben.



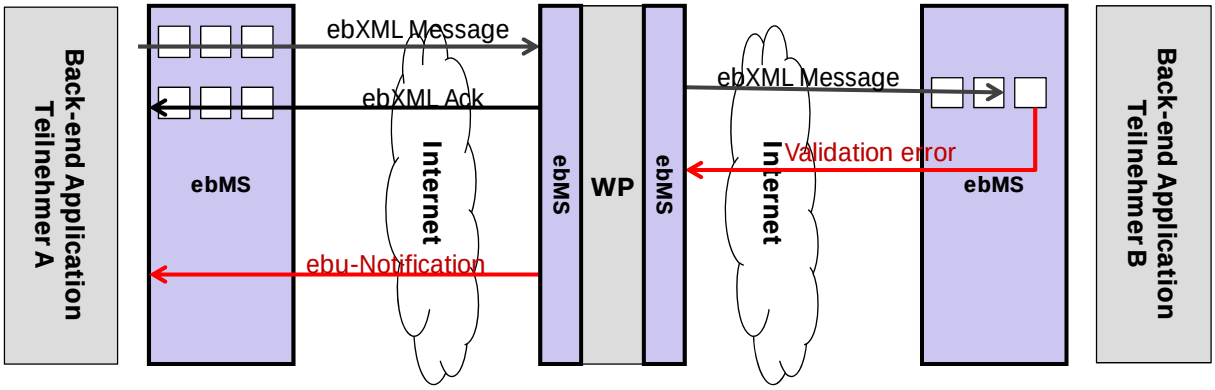
5. Beim Empfänger tritt ein Fehler auf, den er per ebXML ErrorMessage an die WP meldet.

Alle fachlichen Fehler, die sich auf die ebUtilities-Dokumente beziehen, können nicht zwischen WP und Empfänger behoben werden, daher muss auch der Sender über diese Fälle informiert werden.

Hierbei muss die WP anhand der Fehlercodes der ebXML ErrorMessage feststellen, ob es sich um einen fachlichen Fehler handelt.

Die WP schickt in diesem eine fachliche Fehlermeldung an den Sender (unter Verwendung des ebUtility Fehlerdatensatzes, Schema: „Notification“). Wenn diese Nachricht an den ebMS des Empfängers geleitet wird, übermittelt er den Fehlerdatensatz an seiner Back-end-Schnittstelle nicht als fachliches Dokument an die Back-End-Applikation, sondern über die lokale Schnittstelle zur Fehlerbenachrichtigung als eine gescheiterte Übertragung (WS-Aufruf backendResponse()).

Sender und Empfänger haben nun die Diskrepanzen ihrer Konfigurationen zu klären und anschließend das Dokument neu zu übertragen.



Es wird hierbei also in Kauf genommen, dass eine Übertragung zunächst von der WP positiv bestätigt, nachträglich jedoch ein fachlicher Fehler gemeldet wird. Der Grund, dass bei EDA dennoch ein „optimistischer Ansatz“ gewählt wurde ist, dass diese Situation praktisch nur nach Konfigurationsänderungen auftreten kann (z.B. Empfänger validiert gegen eine andere

Schemaversion als der Sender). Insofern wäre eine obligatorische, spätere Empfangsmeldung des Empfängers an den Sender in der überwiegenden Mehrheit aller Übertragungen redundant.

Erforderliche Konfiguration der beteiligten ebXML Message Services:

Im Folgenden wird die Übertragung von Dokumenten im Rahmen des Wechselprozesses beschrieben, bei denen ein Routing über die WP erfolgt.

Schritt 1: Übertragung an WP			**Schritt 2: Weiterleitung an Empfänger**	
Logischer Sender A →	**→ Empfang**	**WP**	**Versand →**	**→ Logischer Empfänger B**
PartyID/From = „A“	PartyID/From = „A“		PartyID/From = „WP“	PartyID/From = „WP“
PartyID/To =„WP“	PartyID/To =„WP“		PartyID/To = „B“	PartyID/To = „B“
MessageType= „SupplierChangeRequest“	MessageType= „SupplierChangeRequest“		MessageType= „SupplierChangeRequest“	MessageType= „SupplierChangeRequest“
XML Validation = Yes	XML Validation = No		XML Validation = No	XML Validation = Yes
Appl-Signatur = No	Appl-Signatur = No		Appl-Signatur = No	Appl-Signatur = No
Compression = Yes	Decompression = No		Compression = No	Decompression = Yes
Encryption = Yes	Decryption = No		Encryption = No	Decryption = Yes
XML Signature = Yes	XML Signature = Yes		XML Signature = Yes	XML Signature = Yes
Envelope-Einbettung: <MarketParticipantDirectory>	Envelope-Einbettung: <MarketParticipantDirectory>		Envelope-Einbettung: <MarketParticipantDirectory>	Envelope-Einbettung: <MarketParticipantDirectory>
			Envelope-Einbettung: <HubDirectory>	Envelope-Einbettung: <HubDirectory>

Falls die Übertragung direkt erfolgt zwischen Sender und Empfänger, dann gilt folgende Parametrisierung:

Logischer Sender A →	**→ Logischer Empfänger B**
PartyID/From = „A“	PartyID/From = „A“
PartyID/To =„B“	PartyID/To = „B“
MessageType= „MoveOutRequest“	MessageType= „MoveOutRequest“
XML Validation = Yes	XML Validation = Yes
Appl-Signatur = Yes	Appl-Signatur = Yes
Compression = Yes	Decompression = Yes
Encryption = Yes	Decryption = Yes
XML Signature = Yes	XML Signature = Yes
Envelope-Einbettung: <MarketParticipantDirectory>	Envelope-Einbettung: <MarketParticipantDirectory>

Konfiguration und standardisierte Einrichtung für OE-Teilnehmer

- **Elemente des ebXML MessageHeader:**
  - <eb:From>/<eb:PartyId>: AT-Nummer des Senders, XML Attribut „type“ = "ATCode" (die WP besitzt dazu eine eigene AT-Nummer)
  - < eb:To>/< eb:PartyId>: AT-Nummer des Empfängers, XML Attribut „type“ = "ATCode"
  - < eb:CPAId>: nicht verwendt, aber gefüllt mit „Dummy“.
  - < eb:ConversationId> nicht verwendt, aber gefüllt mit „Dummy“.

  - < eb:Service>: nicht verwendt, aber gefüllt mit „Dummy“.
  - < eb:Action>: nicht verwendt, aber gefüllt mit „Dummy“.
  - XML Signatur verwenden für MessageHeader, Payload und Attachment(s)
  - < eb:AckRequested>: aktiviert
  - < eb:MessageId>: wird aus Payload-Dokument von MessageNumber übernommen
  - < eb:SyncReply>: aktiviert
  - <NumberOfRetries>: 2
  - RetryInterval: gestaffelt mit 2 Retries:
    - nach 60 Sekunden
    - nach 8 Minuten

    Timeout nach 15 Minuten. Das Ziel ist hierbei, einerseits schnell eine Nachricht nachzusenden, falls dies nicht mit dem ersten Versuch klappte, gleichzeitig aber auch die Frist von 15 Minuten im Falle einer längeren Verzögerung auszunutzen.

  - <eb:DuplicateElimination>: Aktivieren

- Ein Acknowledgement besitzt eine eigene MessageID und eine Referenz auf die Original-MessageID des Senders.
- Manifest Parameter zur Identifizierung von payloads, z.B. ist dies für den Wechseldatensatz (SupplierChangeRequest):
  - eb:location= "http://www.ebutilities.at/wechselprozess/1.0/ContractlessStateRequestV1R00.xsd"
  - Rest bleibt wie die Standardkonfiguration
- Für ebXML genutztes Transportprotokoll: https
- Signatur auf Anwendungsebene
  - bei Peer-to-Peer-Prozessen: SMIME-Signatur
  - bei WP-Prozess: keine Signatur
- Zertifikats-Schlüssellänge: 2048
- Jeder Teilnehmer beantragt zwei Zertifikate
  - eines für die XML Signatur (bzw. die Anwendungssignature im Falle der Peer-to-Peer-Kommunikation)
  - eines für SSL
- hash-Länge: 512 Bit
- Algorithmus zur Dokumentenverschlüsselung: RSA mit AES256
- Algorithmus zur Datenkompression: ZIP
- Einheitliche Verwendung von XML Schema Sets muss sichergestellt werden, z.B. durch ein zentral durchgeführtes Patch-Verfahren.
- Weiterhin sollte der Zeichensatz für übertragene Dateinamen vereinheitlicht werden. Zeichensatz z.B.: [A-Z]|[a-z]|[0-9]|“-„|“_“
- Maximalen Dokumentgröße: 300 KB (dies ist eine Konvention, keine technische Einschränkung.

Ein XML-Dokument muss mit folgendem Header beginnen z.B:

```
<?xml version="1.0" encoding="UTF-8"?>

<ContractlessStateConfirmation xmlns="http://www.ebutilities.at/wechselprozess/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ebutilities.at/wechselprozess/1.0
ContractlessStateConfirmation.xsd">

…
```

Weitere Richtlinien (gilt auch für den Betrieb der WP):

- Ablaufende Zertifikate müssen 4 Wochen vor Ablauf erneuert werden.
- Während dieses Zeitraums sind zwei Zertifikate gültig.

Anforderungen an die WP:

- Validierungs-, Entschlüsselungs-, Dekompressions-Fehlermeldungen (End-to-End-Fehler) des Empfängers werden als ebXML ErrorMessage an die WP geliefert (Error-Codes sind 15, 35 und 31). Verantwortlich für diese Fehlermeldung ist jedoch in den meisten Fällen der Sender. Daher muss die WP in solchen Fällen ein fachliches Fehlerdokument an den Sender schicken (Basis: ebUtilities ErrorNotification).
- Hierbei hat die WP
    - o den Fehlercode des Empfängers zu analysieren
    - o im Falle eines End-to-End-Fehlers ist eine Nachricht mit folgendem Inhalt an den Sender zu schicken:

```
<?xml version = "1.0"?>
<Notification xmlns="http://www.ebutilities.at/wechselprozess/1.0"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation =
“http://www.ebutilities.at/wechselprozess/1.0/NotificationV1R00.xsd“>
      <MarketParticipantDirectory SchemaVersion = "01r00"
                              DocumentMode = "Prod" Duplicate = "false">
            <RoutingHeader>
                  <Sender>AT0000WP</Sender>
                  <Receiver>AT000001</Receiver>
                  <DocumentCreationDateTime>2013-11-07T18:39:09
                  </DocumentCreationDateTime>
            </RoutingHeader>
            <Sector>01</Sector>
            <ProcessStep>TE</ProcessStep>
            <InstallationID>AT1234567890</InstallationID>
            <CaseID>123</CaseID>
      </MarketParticipantDirectory>
      <ProcessDirectory>
            <MessageNumber>1</MessageNumber>
            <ProcessReferenceNumber>3h4g5f3d55hjj5</ProcessReferenceNumber>
            <ResponseData ResponseCodeGroup=”TERR01”>
                  <ResponseCode>31</ResponseCode>
            </ResponseData>
      </ProcessDirectory>
</Notification>
```

Hierbei sollten die Werte durch die WP wie folgt eingesetzt werden:

- DocumentMode, Duplicate, Sector, ProcessStep, InstallationID, CaseID, ProcessReferenceNumber = entsprechend dem Originaldokument
- Sender = WP
- Receiver = Ursprünglicher Sender
- DocumentCreationTime = Erstellungszeitstempel für dieses Fehlerdokument seitens der WP
- MessageNumber = neu vergeben durch die WP
- ResponseData = (TERR01 = Technical Error, Fehlercode wie oben beschrieben).

Die Wechselplattform muss die Zertifikate aller der Teilnehmer (SSL und Anwendungszertifikat) vorhalten, um Signaturen zu überprüfen, und Sitzungsschlüssel zu erlangen. Umgekehrt muss die WP ihre Zertifikate (Anwendungsebene und SSL, einschließlich ggf. der erforderlichen Root-Zertifikate) den Teilnehmern zur Verfügung stellen.