
Integrationsleitfaden zur Anbindung des
OZG-PLUS-Postfachs an bestehende
Fachverfahren

Änderungsnachweise

Version	Freigabe- datum	Autor	Kapitel	Änderungen
1.6	09.08.2024	JUB JWA MIS	4.3, 5.1, 5.3 und 7.1	Anpassung der Versionsnummer des SDKs Umstrukturierung der Kapitelhierarchie in Kapitel 5.3 Entfernen des Unterkapitels 5.3.2 Anpassung von Codebeispielen Entfernen des Unterkapitels 7.1.2 Erweiterung Validierungen und Ausnahmen Tabelle 3 Rückantwort eines bestehenden Bereitstellungsauftrags erweitert Tabelle 4 Statusänderung eines Bereitstellungsauftrags erweitert Verschieben des Abschnitts „Pflichtfelder und optionale Felder einer Nachricht“ in ein neues Unterkapitel Redaktionelle Änderungen
1.5	27.05.2024	MIS	6	Änderung der Integrationsumgebung
1.4	10.11.2023	HT MIS GT TIR	4.3, 5.3 und 7.4	Erweiterung der Dokumentation zur Abfrage von Statusinformationen zu bestehenden Bereitstellungsaufträgen hinsichtlich Status und Zeitpunkten. Ausnahmen bei Vertraulichen Metadaten Hinweis zu XHE-Beispiel eingearbeitet
1.3	01.09.2023	HT TIR MIS GT	2 und 5 Alle	Einfügen einer User-Journey, Erweiterung der Beispielcodes, Erläuterung der Pflichtfelder und optionalen Felder bei Nachrichten und Anpassung Open API ELSTER- Transfer Dateiname, Einfügen eines Abkürzungs-, Abbildungs- und Tabellenverzeichnis Neues Kapitel 6 für Beschreibung der Umgebungen
1.2	16.06.2023	HT	4.3	Ergänzung der Statusabfrage

Version	Freigabe- datum	Autor	Kapitel	Änderungen
1.1	15.06.2023	HT	3 und 4	Absicherung über API-Key im HTTP Header ergänzend beschrieben
1.0	02.06.2023	MIS	Alle	Erste Version des Integrationsleitfadens auf Basis einer ELSTER ähnlichen Schnittstelle für den Nachrichtenversand

Inhaltsverzeichnis

1	Rechtliche Informationen und weitere Hinweise	7
2	Einleitung	8
3	Lookup eines Funktionspostfaches per REST Service	12
3.1	Abfrage aller Funktionspostfächer einer ELSTER Organisation	12
3.2	Abfrage des Verschlüsselungsschlüssels eines Funktionspostfachs.....	12
4	Nachrichtenerstellung per REST Service	13
4.1	Übermittlung von verschlüsselten Nachrichtenbestandteilen.....	13
4.2	Übermittlung des Bereitstellungsauftrages	14
4.3	Abfrage von Statusinformationen zu bestehenden Bereitstellungsaufträgen	14
5	Beschreibung des SDK	16
5.1	Umfang	16
5.2	Dependency.....	16
5.3	Software-Voraussetzungen	16
5.4	Verwendung des SDK zum Erstellen einer Nachricht.....	17
5.4.1	Voraussetzung	17
5.4.2	Ablauf.....	17
5.4.2.1	Setup	17
5.4.2.2	Erstellen einer neuen Nachricht	17
5.4.2.3	LetterBody (Nachrichtentext)	18
5.4.2.4	Attachments.....	18
5.4.2.5	StructuredData.....	18
5.4.2.6	ConfidentialMetaData (Vertrauliche Metadaten).....	19
5.4.3	Pflichtfelder und optionale Felder einer Nachricht.....	21
5.4.4	Ausnahmen bei den Vertraulichen Metadaten	22
5.4.5	Ausnahmen Hierarchie im SDK.....	23
6	Umgebungen	29
7	Beschreibung einer XHE-Nachricht.....	30
7.1	Header.....	30
7.1.1	PartyType und PartyIdentification	31
7.2	Payloads	32
7.2.1	ConfidentialMetaData	33
7.2.1.1	ConfidentialMetaDataType.....	33
7.3	Code Lists.....	34
7.3.1	Public Metadata v1.0.....	35
7.3.2	Confidential Metadata v1.0 Metadata v1.0	36
7.3.3	Payload Attributes v1.0	37
7.4	Beispiel XHE.....	37

Abbildungsverzeichnis

Abbildung 1 Nutzerreise OZG-PLUS-Postfach	8
Abbildung 2 Ablauf zum Versand einer OZGPP-Nachricht	10
Abbildung 3 Auswahl eines Funktionspostfaches für ein Unternehmen (beispielhaft).....	10
Abbildung 4 OZGPP Detailansicht einer empfangenen Nachricht	21
Abbildung 5 XHE Übersicht	30

Tabellenverzeichnis

Tabelle 1 Abkürzungsverzeichnis	6
Tabelle 2 erwartete Felder eines Bereitstellungsauftrags	14
Tabelle 3 Rückantwort eines bestehenden Bereitstellungsauftrags	15
Tabelle 4 Statusänderung eines Bereitstellungsauftrags	15
Tabelle 5 Felderbeschreibung in der OZGPP-Maske	22
Tabelle 6 LeiKa Leistungsgruppen	28
Tabelle 7 Unterscheidung Test- und Produktionsumgebung	29
Tabelle 8 Elemente der XHE-Nachricht	30
Tabelle 9 Elemente des Headers einer XHE-Nachricht	31
Tabelle 10 Party Typ und Identifikation.....	31
Tabelle 11 Attribute einer Payload.....	32
Tabelle 12 Payloadarten.....	33
Tabelle 13 Payload Inhalt mit Metadaten.....	33
Tabelle 14 Typenbeschreibung der Confidential Meta Daten	33
Tabelle 15 Zusammensetzung der Payload Attribute	33
Tabelle 16 Codeliste Shemadarstellung	35
Tabelle 17 Codeliste der Public Metadaten	35
Tabelle 18 Codeliste der Confidential Metadaten	37
Tabelle 19 Codeliste der Payload Attribute.....	37

Abkürzungsverzeichnis

Abkürzung	Beschreibung
OZGPP	OZG-PLUS-Postfach
SDK	Software Development Kit
SSP	Self Service Portal (hier: ELSTER)
API	Application Programming Interface (Programmierschnittstelle)
Mein UK	Mein Unternehmenskonto
CNB	Cloud Native Buildpacks
ECIES	Elliptic Curve Integrated Encryption Scheme
XHE	Exchange Header Envelope
LeiKa	Leistungskatalog
REST	Representational State Transfer
POM	Project Object Model
JAR	Java Archive
URN	Uniform Resource Name

Tabelle 1 Abkürzungsverzeichnis

1 Rechtliche Informationen und weitere 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 orthografische Fehler werden in künftigen Auflagen korrigiert.

Diese Produktinformation sowie sämtliche urheberrechtsfähigen Materialien, die mit dem Produkt vertrieben werden, sind urheberrechtlich geschützt. Alle Rechte sind der Governikus GmbH & Co. KG, im folgenden Governikus KG, vorbehalten. Alle urheberrechtsfähigen Materialien dürfen ohne vorherige Einwilligung der Governikus KG weder ganz noch teilweise kopiert oder auf sonstige Art und Weise reproduziert werden. Für rechtmäßige Nutzer des Produkts gilt diese Einwilligung im Rahmen der vertraglichen Vereinbarungen als erteilt. Jegliche Kopien dieser Produktinformation, bzw. von Teilen daraus, müssen den gleichen Hinweis auf das Urheberrecht enthalten wie das Original.

Governikus ist eine eingetragene Marke der Governikus KG, Bremen. Andere in diesem Produkt aufgeführte Produkt- und/ oder Firmennamen sind möglicherweise Marken weiterer Eigentümer, deren Rechte ebenfalls zu wahren sind.

2 Einleitung

Diese Dokumentation soll die Anbindungspartner befähigen, eine Nachricht für das OZG-PLUS-Postfach (OZGPP) mithilfe des SDK aufzubauen sowie zu verschlüsseln und per REST-Aufrufen zu verschicken.

Um einen genaueren Einblick auf die notwendigen Schritte zu erhalten wird im Folgenden eine Nutzerreise (siehe Abbildung 1 Nutzerreise OZG-PLUS-Postfach) beschrieben.

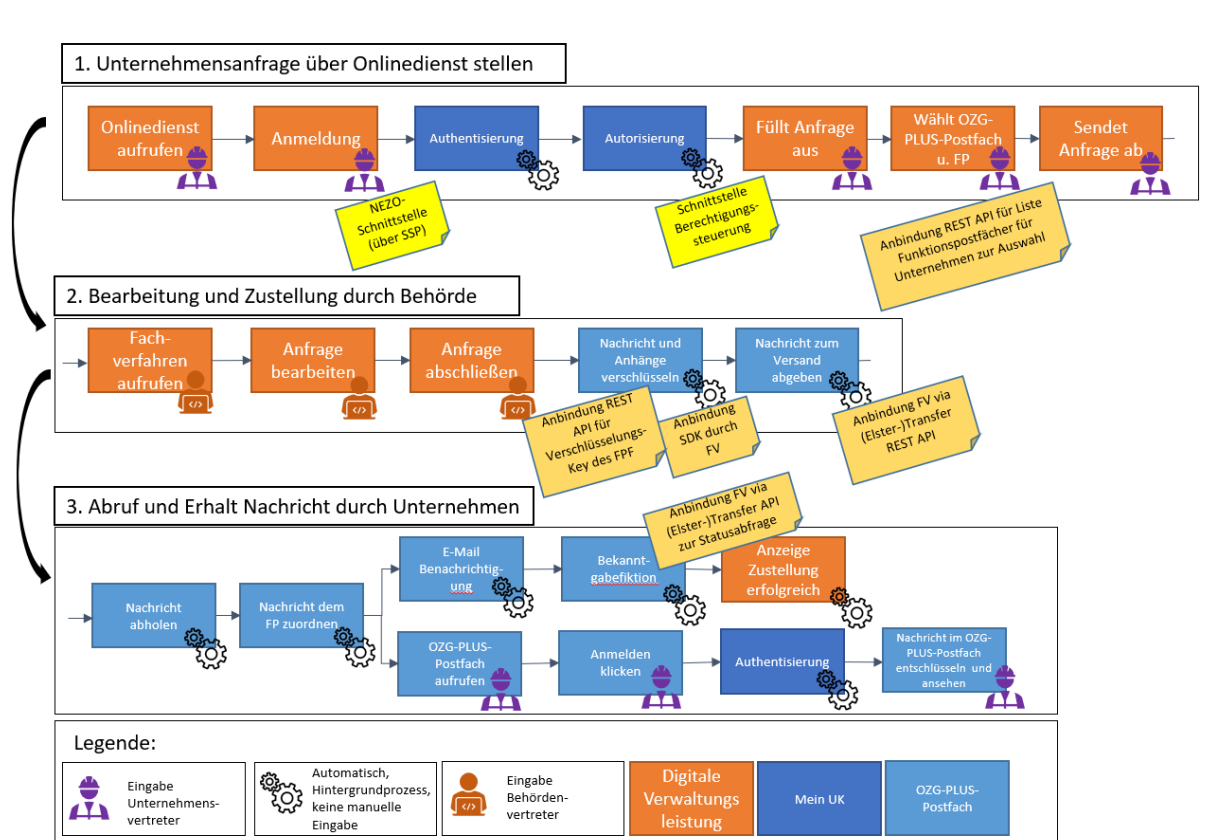


Abbildung 1 Nutzerreise OZG-PLUS-Postfach

Die Nutzerreise teilt sich in drei sequenzielle Abläufe auf:

1. Unternehmensanfrage für eine digitale Verwaltungsleistung über Onlinedienst stellen
2. Bearbeitung der Anfrage und Zustellung der Nachricht durch die Behörde
3. Abruf und Erhalt der Nachricht durch das Unternehmen im OZG-PLUS-Postfach

In der ersten Sequenz möchte eine Person eines Unternehmens einen Onlinedienst für eine digitale Verwaltungsleistung nutzen und meldet sich mit seinem/ihrem ELSTER-Organisationszertifikat bei diesem Onlinedienst an. Im Hintergrund authentifiziert und autorisiert das Mein Unternehmenskonto (Mein UK) das Organisationszertifikat. Bei erfolgreicher Überprüfung folgt der Schritt der Anfragen-/Antragsstellung. Hierbei wird im Rahmen der Einwilligung in die digitale Bereitstellung die Auswahl getroffen an welches Funktionspostfach des OZG-PLUS-Postfachs eine Zustellung der bearbeiteten Anfrage durch die Behörde erfolgen soll. Anschließend sendet der Mitarbeitende die Anfrage über den Onlinedienst ab.

Innerhalb dieser Sequenz sind drei Dinge notwendig, damit die Kommunikation der Schnittstellen funktioniert. Zunächst ist eine Anbindung an die NEZO-Schnittstelle durch den

Onlinedienst notwendig. Dies wird über das Mein UK Self Service Portal (SSP) ermöglicht. Des Weiteren ist die Berechtigungssteuerung (Modul 6 von Mein UK) dem Onlinedienst vorzuschalten, in welcher die Berechtigungen geprüft werden, ob das Verfahren für diese Person und das Unternehmen eingeleitet werden kann. Zuletzt muss der Onlinedienst eine Anbindung an die REST-API des OZG-PLUS-Postfachs für die Liste der Funktionspostfächer vorgenommen haben, um die zur Verfügung stehenden Funktionspostfächer auflisten zu können. Die Anbindung an die REST-API wird im Kapitel 3 beschrieben.

In der zweiten Sequenz erfolgt nach der Abgabe einer Anfrage die Bearbeitung derer im Fachverfahren. Dazu ruft ein Behördenvertreter das zugehörige Fachverfahren auf, bearbeitet die eingegangene Anfrage und schließt diese anschließend ab. Nach dem Abschluss wird die Nachricht für das Postfach des Unternehmens vorbereitet. Hierfür wird über die REST-API des OZG-PLUS-Postfachs der öffentliche Schlüssel für das ausgewählte Funktionspostfach ermittelt (Kapitel 3) und das SDK von Governikus GmbH & Co. KG (Kapitel 5) benötigt. Es verschlüsselt die Nachricht inkl. aller Anhänge mit dem öffentlichen Schlüssel des gewählten Funktionspostfachs des Empfänger-Unternehmens. Anschließend wird die Nachricht zum Versand übergeben. Dies erfolgt durch die Anbindung des Fachverfahrens an die Transfer-Schnittstelle des OZG-PLUS-Postfachs nach ELSTER-Transfer Semantik (Kapitel 4).

Die dritte Sequenz beinhaltet das Abrufen der Nachricht und die anschließende Bekanntgabefiktion. Zunächst erhalten alle zugeordneten Mitarbeitenden des adressierten Funktionspostfachs eine Benachrichtigung via E-Mail, dass sich eine Nachricht im OZG-PLUS-Postfach befindet. Über die Bekanntgabefiktion mit der Statusabfrage (Kapitel 4) erhält die digitale Verwaltungsleistung ebenfalls eine Möglichkeit für die Bestätigung, dass die versendete Nachricht beim Funktionspostfach zugestellt wurde. Damit ist die Sequenz für den Onlinedienst abgeschlossen. Für die Mitglieder im adressierten Funktionspostfach erfolgt in dieser Sequenz noch die Entschlüsselung und das Lesen der Nachricht. Dazu ruft ein Mitglied das OZG-PLUS-Postfach auf, meldet sich mittels des ELSTER-Organisationspostfachs an und wird anschließend an der Schnittstelle zum Mein Unternehmenskonto (Mein UK) authentisiert. Nun können alle Nachrichten im Postfach, welche an die Funktionspostfächer versendet wurden, in denen der Mitarbeitende Mitglied ist, angezeigt werden, sie sind allerdings noch verschlüsselt. Nach Überprüfung der privaten Schlüsseldaten werden die Nachrichten entschlüsselt und können inkl. Anhänge gelesen werden.

Um die in den Sequenzen der Nutzerreise beschriebenen Anbindungen der Onlinedienste bzw. Fachverfahren an das OZG-PLUS-Postfach zu ermöglichen, sind mehrere Schritte erforderlich. Zunächst muss das Fachverfahren mit REST-Aufrufen die ID des Funktionspostfachs, welches die Nachricht erhalten soll, ermitteln. Außerdem muss für die Verschlüsselung der öffentliche Schlüssel für dieses Funktionspostfach vom OZG-PLUS-Postfach abgerufen werden.

Anschließend können die einzelnen Nachrichtenbestandteile mit dem mitgelieferten SDK verschlüsselt werden. Die Nachrichtenbestandteile werden einzeln beim OZG-PLUS-Postfach per REST-Aufrufen hochgeladen und abschließend wird ein Bereitstellungsauftrag übermittelt. Damit ist die Nachricht vollständig im OZG-PLUS-Postfach zugestellt.

Zuletzt kann das Fachverfahren per Statusabfrage prüfen, ob der Bereitstellungsauftrag bereits verarbeitet und gelesen wurde.

Dieser Ablauf ist in Abbildung 2 dargestellt.

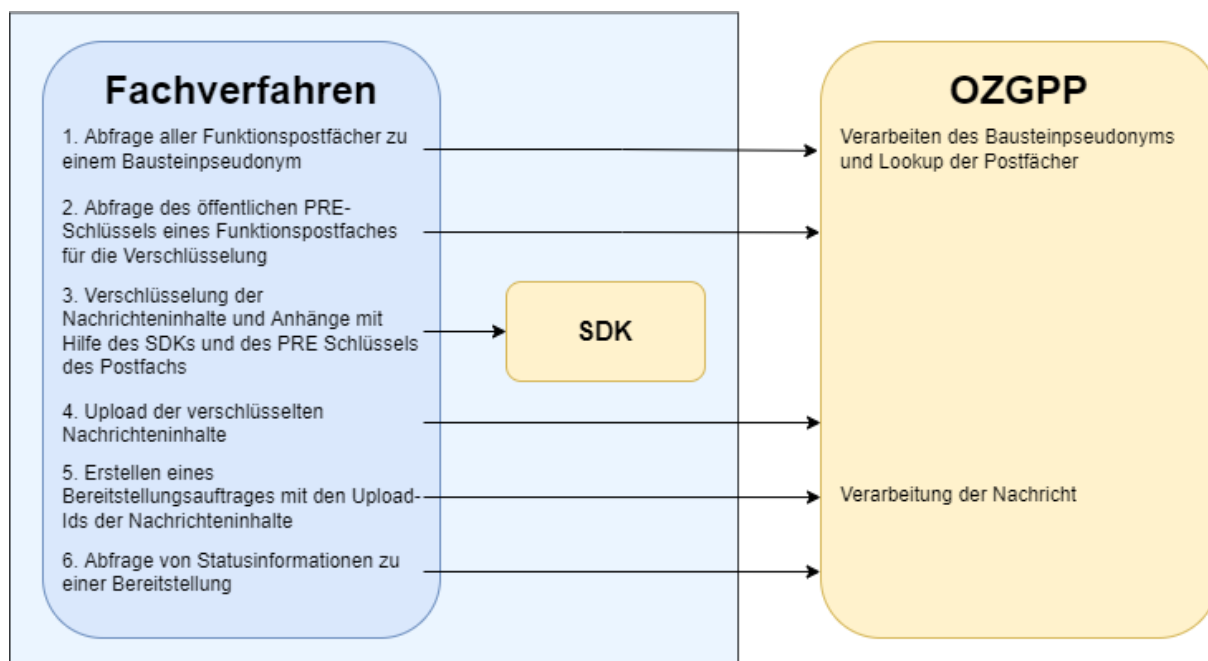


Abbildung 2 Ablauf zum Versand einer OZGPP-Nachricht

Um mit dem OZGPP zu kommunizieren, muss zuerst der Adressat, des aktiven Funktionspostfaches, identifiziert werden. Dies erfolgt mithilfe des ELSTER Bausteinpseudonyms eines Unternehmens. Anschließend können alle Funktionspostfächer eines Unternehmens zur Auswahl angeboten werden, siehe beispielhaft Abbildung 3.

Auswahl der Zustellung:

keine digitale Zustellung

Postfach 2.0

OZG-PLUS-Postfach - Funktionspostfach:

Buchhaltung und Finanzen ▾

Personalwesen

Geschäftsführung

Forschung und Entwicklung

Abbildung 3 Auswahl eines Funktionspostfaches für ein Unternehmen (beispielhaft)

Für das gewählte Funktionspostfach muss dann der öffentliche PRE-Schlüssel zur Verschlüsselung vom OZGPP abgefragt werden.

Sind Identifier und PRE-Schlüssel eines Funktionspostfaches bekannt, kann eine Nachricht an das Funktionspostfach aufgebaut, verschlüsselt und verschickt werden.

Für den Aufbau und die Verschlüsselung einer Nachricht wird ein (Java-)SDK bereitgestellt. Dieses übernimmt die Aufbereitung der Nachricht als XHE-Payload und verschlüsselt die Nachrichtenbestandteile.

Alle vom SDK generierten Nachrichtenbestandteile werden einzeln per File-Upload an das OZGPP per HTTPS gesendet.

Sind alle Nachrichtenbestandteile im OZGPP hinterlegt, kann abschließend ein Bereitstellungsauftrag im OZGPP angelegt werden. Dieser Bereitstellungsauftrag referenziert alle Nachrichtenbestandteile und gibt den Empfänger, der Identifier des aktiven Funktionspostfaches, an.

Nach Erhalt des Bereitstellungsauftrages, validiert das OZGPP die Vollständigkeit und Gültigkeit der Nachricht bestehend aus allen Nachrichtenbestandteilen. Anschließend wird die Nachricht dem Empfänger im OZGPP zugestellt.

Der Status des Bereitstellungsauftrages kann vom Fachverfahren jederzeit per Status-Abfrage abgerufen werden.

3 Lookup eines Funktionspostfaches per REST Service

Für das Versenden einer Nachricht an ein Funktionspostfach muss dieses adressiert werden. Außerdem muss der Nachrichteninhalt mit einem Verschlüsselungsschlüssel für dieses Funktionspostfach verschlüsselt sein.

Zum Abrufen dieser Informationen vom OZGPP liegt ein REST Service vor. Für die aktuellste Version des REST Service sei auf die OpenAPI Spezifikation *lookup-1.0.0-rest-api-docs.yaml* verwiesen. Diese wird zusätzlich zum Integrationsleitfaden bereitgestellt.



Hinweis: Die Verbindungen vom Fachverfahren zum „OZGPP REST Services Lookup“ werden zusätzlich zu HTTPS über einen API-Key im HTTP Header abgesichert (siehe OpenAPI Spezifikation *lookup-1.0.0-rest-api-docs.yaml*). Jedes Fachverfahren erhält seinen individuellen API-Key.

3.1 Abfrage aller Funktionspostfächer einer ELSTER Organisation

Um eine Nachricht an ein Funktionspostfach zu verschicken, muss das Funktionspostfach mit seinem Identifier adressiert werden. Das OZGPP bietet einen Endpunkt an `https://<Host>/rest/mailboxes` (siehe *lookup-1.0.0-rest-api-docs.yaml*), um alle aktiven Funktionspostfächer einer Organisation abzurufen.

Um diese Liste abzufragen, muss die zugehörige Organisation vorab eine Authentisierung mit einem ELSTER Organisationszertifikat an der NEZO Schnittstelle durchgeführt haben. Hierbei erhalten Sie in der SAML Antwort von NEZO neben den ELSTER IDs, die für Ihre Anwendungszwecke einzigartig sind, auch die sogenannten *Bausteinpseudonyme*. (`<saml2:Attribute Name="Bausteinpseudonyme">`) Mit dem Bausteinpseudonym, welches für die Entity ID des OZGPP ausgestellt wurde, kann dann eine Liste aller aktiven Funktionspostfächer bei OZGPP abgefragt werden.

3.2 Abfrage des Verschlüsselungsschlüssels eines Funktionspostfachs

Eine Nachricht an ein Funktionspostfach muss verschlüsselt werden. Die E2E-Verschlüsselung wird vom SDK übernommen. Das SDK benötigt aber den Verschlüsselungsschlüssel, einen sogenannten PRE-Public-Key, des adressierten Funktionspostfachs.

Um diesen Verschlüsselungsschlüssel abzurufen, bietet das OZGPP einen Endpunkt an `https://<Host>/rest/mailboxes/<Postfach-Id>/encryption-key` (siehe *lookup-1.0.0-rest-api-docs.yaml*). Hierbei muss die in Anfrage Kapitel 3.1 erhaltene ID des aktiven Funktionspostfachs mitgeschickt werden.

Für die Verschlüsselung von Nachrichtenbestandteilen siehe Kapitel 5.

4 Nachrichtenerstellung per REST Service

Um eine Nachricht an ein Funktionspostfach zu verschicken, sind mehrere Schritte erforderlich. Dies liegt daran, dass alle Nachrichtenbestandteile einzeln per REST Aufruf übergeben werden.

Zuerst werden alle verschlüsselten Nachrichtenbestandteile einzeln übertragen, abschließend wird ein sogenannter Bereitstellungsauftrag erteilt, der alle vorigen Nachrichtenteile miteinander verknüpft und ein aktives Funktionspostfach adressiert.

Der zugehörige REST Service entspricht der Schnittstellendefinition von ELSTER Transfer in reduzierter Form. Ein wichtiger Unterschied ist jedoch, dass die Schnittstelle nicht in einer lokal betriebenen Anwendung, sondern direkt im OZGPP aufgerufen wird. Für die aktuellste Version des REST Service sei auf die OpenAPI Spezifikation *ozgpp-elster-transfer-3.3.0-rest-api-docs.yaml* verwiesen. Diese wird zusätzlich zum Integrationsleitfaden bereitgestellt.



Hinweis: Die Verbindungen vom Fachverfahren zum „OZGPP REST Services ELSTER Transfer“ werden zusätzlich zu HTTPS über einen API-Key im HTTP Header abgesichert (siehe OpenAPI Spezifikation *ozgpp-elster-transfer-3.3.0-rest-api-docs.yaml*). Jedes Fachverfahren erhält seinen individuellen API-Key.

4.1 Übermittlung von verschlüsselten Nachrichtenbestandteilen

Nach der Generierung und Verschlüsselung der einzelnen Nachrichtenbestandteile mithilfe des SDK werden diese einzeln an das OZGPP an den Endpunkt `https://<Host>/rest/bereitstellungsauftrag/file-upload` (siehe *elster-transfer-3.3.0-rest-api-docs.yaml*) übergeben. Verschlüsselte Nachrichtenbestandteile, die übertragen werden müssen bzw. können, sind

- verpflichtend der Nachrichtentext („LetterBody“),
- optional die Anhänge („Attachments“ und „StructuredData“) sowie
- verpflichtend die schützenswerten Metadaten wie z.B. Betreff („ConfidentialMetadata“).

Hier wird das XHE Nachrichtenformat verwendet. Die Nachrichteninhalte werden mithilfe des SDKs aufbereitet und verschlüsselt. Anschließend liegt eine XHE-Payload im XML-Format vor. Diese XHE-Payload kann dann an das OZGPP übermittelt werden. Wird anderes XML übermittelt, welches keine XHE-Payload darstellt, wird die Anfrage abgelehnt.

Nach erfolgreicher Übermittlung der XHE-Payload wird eine Upload-ID zurückgegeben. Diese muss gespeichert werden, da der abschließende Bereitstellungsauftrag alle Nachrichteninhalte mit ihrer Upload-ID referenziert.

4.2 Übermittlung des Bereitstellungsauftrages

Nachdem alle Nachrichtenbestandteile verschlüsselt und übermittelt wurden, werden sie zusammengeführt. Dies geschieht per Bereitstellungsauftrag am OZGPP Endpunkt `https://<Host>/rest/bereitstellungsauftrag/v3` (siehe *elster-transfer-3.3.0-rest-api-docs.yaml*), welcher alle Upload-IDs der bisherigen Nachrichtenbestandteile enthält. Hier wird erstmals der Empfänger der Nachricht angegeben und die vorab übermittelten Nachrichtenteile werden zusammengeführt und als Ganzes validiert.

Folgende Felder werden dabei verpflichtend erwartet:

Name	Beschreibung
<code>absender</code>	Name des Absenders, wird dem oder der Empfänger:in angezeigt
<code>accountId</code>	ID des Empfänger-Funktionspostfachs, entspricht einer UUID
<code>anhaenge</code>	Liste aller bereits übermittelten Nachrichtenbestandteile mit deren Upload-Ids. Muss genau einen Verweis auf einen <code>LetterBody</code> und eine <code>ConfidentialMetadata</code> enthalten.
<code>datenart</code>	Von ELSTER Transfer vorgegebenes Feld, muss vorhanden sein, der Wert wird nicht ausgewertet.

Tabelle 2 erwartete Felder eines Bereitstellungsauftrags

4.3 Abfrage von Statusinformationen zu bestehenden Bereitstellungsaufträgen

Wenn von einem Fachverfahren ein Bereitstellungsauftrag zum OZGPP versendet wurde, muss dieser dort angenommen werden. Die Annahme eines Auftrags wird in der Rückantwort mit einer Auftrags-ID quittiert.

Die Abfrage der Statusinformation zu einem zugestellten Bereitstellungsauftrag erfolgt mit Hilfe der zuvor erhaltenen Auftrags-ID am OZGPP Endpunkt `https://<Host>/rest/bereitstellungsauftrag/v3/<Auftrags-ID>` (siehe *elster-transfer-3.3.0-rest-api-docs.yaml*). Aus der Rückantwort der Statusabfrage kann der Status der aktuellen Bearbeitung ermittelt werden (siehe *elster-transfer-3.3.0-rest-api-docs.yaml*).

Zu beachten ist hierbei, dass nicht alle Felder des ELSTER Transfers unterstützt werden (siehe *elster-transfer-3.3.0-rest-api-docs.yaml*). In der nachfolgenden Tabelle sind nur die Felder gelistet, die auch befüllt werden.

Name	Beschreibung
<code>abrufzeitpunkt</code>	Zeitpunkt des Dokumentenabrufs durch den Empfänger
<code>absender</code>	Name des Absenders vom Bereitstellungsauftrag
<code>accountId</code>	ID des aktiven Funktionspostfachs vom Bereitstellungsauftrag

anhaenge	Liste aller bereits übermittelten Nachrichtenbestandteile vom Bereitstellungsauftrag
erstellZeitpunkt	Zeitpunkt der angelegten Bereitstellung
id	Id des angelegten Bereitstellungsauftrags
status	Der Bearbeitungsstatus des Auftrags. In der OpenAPI Spezifikation <i>ozgpp-elster-transfer-3.3.0-rest-api-docs.yaml</i> sind die Status definiert. Alle aktuell im OZGPP umgesetzten Status sind in der unteren Tabelle beschrieben.
emailVersandZeitpunkt	Zeitpunkt des Versands der E-Mail-Benachrichtigung

Tabelle 3 Rückantwort eines bestehenden Bereitstellungsauftrags

Status	Beschreibung
EINGESTELLT	Die Nachricht wurde eingestellt, die Benachrichtigung an den Empfänger wurde jedoch noch nicht zugestellt. In der Bereitstellungsinformation wird der Zeitpunkt, zu dem die Nachricht im OZGPP eingestellt wurde, im " "erstellZeitpunkt" gesetzt.
BEREIT_ZUR_ABHOLUNG	Die Nachricht wurde an den Empfängern eines Funktionspostfachs zum Lesen bereitgestellt. Alle Funktionspostfachmitglieder wurden per Email benachrichtigt. In der Bereitstellungsinformation wird der Zeitpunkt des Versands der E-Mail-Benachrichtigung im " "emailVersandZeitpunkt" gesetzt.
VOM_EMPFAENGER_ABGEHOLT	Die Nachricht wurde vom Empfänger abgeholt. In der Bereitstellungsinformation wird der Zeitpunkt, zu dem die Nachricht abgeholt wurde, im " "abrufZeitpunkt" gesetzt.
GELOESCHT	Die Nachricht wurde aus dem Postfach gelöscht.

Tabelle 4 Statusänderung eines Bereitstellungsauftrags

5 Beschreibung des SDK

Zur Erstellung von Nachrichten für das OZGPP wird ein (Java-)SDK bereitgestellt, welches den Aufbau und die Verschlüsselung einer Nachricht auf Basis des XHE Standards übernimmt. Nicht Bestandteil des SDK ist der Abruf des Verschlüsselungsschlüssels des Empfängers vom OZGPP.

5.1 Umfang

Das SDK besteht aus drei Jar-Dateien und den dazugehörigen Maven POM-Dateien:

- *ozgpp-client-sdk-?.?.?.jar* – enthält die Implementierung des SDKs, welche zur Erstellung und Verarbeitung von Nachrichten verwendet werden kann
- *ozgpp-message-library-?.?.?.jar* – wird von *ozgpp-client-sdk-?.?.?.jar* benötigt und enthält Funktionen zur Erstellung und Verschlüsselung von XHE-Nachrichten
- *proxycrypt-?.?.?.jar* – enthält die C-Library für die Verschlüsselung

Die aktuelle Versionsnummer kann den SDK Release Notes unter folgendem Link entnommen werden:

<https://www.governikus.de/loesungen/it-planungsrat/mein-uk/#downloads>

5.2 Dependency

```
<dependency>
  <groupId>de.governikus.ozgpp.library</groupId>
  <artifactId>ozgpp-client-sdk</artifactId>
  <version>?.?.?</version>
</dependency>
```



Hinweis: Diese Abhängigkeit ist nur im Governikus-Repository verfügbar. Fügen Sie <https://nexus.governikus.de/nexus/content/groups/public/> zu Ihren Maven-Spiegeln hinzu.

5.3 Software-Voraussetzungen

Das SDK wird in Form von Jar-Dateien bereitgestellt und kann in Java Anwendungen eingebunden werden. Dabei ist mindestens das JDK 11, sowie Jakarta XML 4.0 Voraussetzung.

Aktuell wird Linux Debian ab Version 11, Ubuntu ab Version 2022.04 und Windows 10 (X64) mit Microsoft Visual C++ Redistributable-Pakete (X64: https://aka.ms/vs/17/release/vc_redist.x64.exe) unterstützt.



Hinweis: Soll das SDK in einer Anwendung mit Cloud Native Buildpacks (CNB) eingesetzt werden, sollte folgendes beachtet werden. Es sollte der Jammy Builder <https://github.com/paketo-buildpacks/builder-jammy-base> verwendet werden, da dieser die

	<p>nötigen C-Library Abhängigkeiten mitbringt.</p> <p>Für Spring-Boot sollten folgende Parameter beim Build-Image Goal https://docs.spring.io/spring-boot/docs/3.1.0/maven-plugin/reference/htmlsingle/#goals-build-image verwendet werden (Maven-Syntax):</p> <pre><spring-boot.build-image.builder>paketobuildpacks/builder-jammy-base</spring-boot.build-image.builder> <spring-boot.build-image.runImage>paketobuildpacks/run-jammy-base</spring-boot.build-image.runImage></pre>
--	--

5.4 Verwendung des SDK zum Erstellen einer Nachricht

Alle Nachrichtenbestandteile werden durch Builder generiert, welche aus der `MessageComposer Factory` erzeugt werden.

Die Nachrichtenbestandteile müssen der im Kapitel 4 beschriebenen ELSTER Transfer REST Schnittstelle übergeben werden. Dies ist nicht Bestandteil des SDK, aber durch die Nutzung der ELSTER Transfer API können bereits existierende Client Implementierungen wiederverwendet werden.

5.4.1 Voraussetzung

Um eine Nachricht zu erstellen, werden das öffentliche Zertifikat (PRE-Public-Key, siehe dazu Kapitel 3.2) und der Identifier des Empfängers benötigt.

Das Abrufen dieser Daten des Empfängers ist nicht Bestandteil des SDK.

5.4.2 Ablauf

5.4.2.1 Setup

Zur Verschlüsselung der Nachrichten setzt das SDK auf BouncyCastle. Um nicht im existierenden Code bereits geladene Security Provider zu überschreiben, verzichtet das SDK darauf, BouncyCastle selbst zu registrieren.

Sollte BouncyCastle noch nicht registriert sein, kann dies wie folgt geschehen:

```
if (Security.getProvider(BouncyCastleProvider.PROVIDER_NAME) == null) {
    Security.insertProviderAt(new BouncyCastleProvider(), 1);
}
```

5.4.2.2 Erstellen einer neuen Nachricht

Für **jede** zu versendende Nachricht **muss ein neuer** `MessageComposer` erstellt werden. Dies geschieht über die integrierte Factory, die zur Erstellung den PRE-Public-Key des Ziel-Postfaches übergeben bekommen muss (siehe dazu Kapitel 3.2):

```
MessageComposer messageComposer = MessageComposer.factory()
    .fpfPrePublicKey(fpfPrePublicKey)
    .encrypt(true)
    .build();
```

Diese `MessageComposer` Instanz stellt nun Builder für die unterschiedlichen Nachrichtenbestandteile bereit. Auf diese kann nach dem erfolgreichen Bauen durch den Builder via `InputStream` zugegriffen werden, um eine Nutzung des SDK zu ermöglichen, ohne jemals die komplette Nachricht im Speicher der Anwendung halten zu müssen.

5.4.2.3 LetterBody (Nachrichtentext)

Den `InputStream` für einen `LetterBody` (Nachrichtentext) wird wie folgt mittels einer `MessageComposer` Instanz aus einem String aufgebaut:

```
LetterBody lb = messageComposer.letterBodyBuilder()
    .content(new ByteArrayInputStream(
        „Text“.getBytes(StandardCharsets.UTF_8)))
    .build();
InputStream lbStream = lb.fileUploadRequestBodyStream();
```

Die Daten des Ergebnis `InputStreams` können nun mittels des im Kapitel 4.1 beschriebenen file-upload Endpunktes der ELSTER Transfer API hochgeladen werden.

Validierung:

Der `content` darf maximal 5000 Zeichen lang sein.

5.4.2.4 Attachments

Attachments können in unbegrenzter Menge einer Nachricht hinzugefügt werden. Der Ablauf für das Hinzufügen eines einzelnen Attachments kann mehrfach durchlaufen werden, um mehrere Attachments der Nachricht hinzuzufügen.

Ein einzelnes Attachment, das als `InputStream` verfügbar ist, wird wie folgt gebaut:

```
Attachment a1 = messageComposer.attachmentBuilder()
    .pdfDocument(attachmentInputStream)
    .name("Attachment1.pdf")
    .description("Eine Beschreibung 1")
    .build();
InputStream a1Stream = a1.fileUploadRequestBodyStream();
AttachmentConfidentialMetaData amd1 = a1.getMetaData();
```

Die `Attachment` Instanz (`a1` in diesem Beispiel) benötigen wir später noch um das Attachment der Nachricht anhängen zu können. Wie beim `LetterBody` müssen auch hier nun die Daten im Ergebnis `InputStreams` in den file-upload Endpunkt hochgeladen werden.

Der unter dem Attribut `name` angegebene Dateiname wird später in der Oberfläche für die Anzeige des Dateinamens in der Liste der Anhänge innerhalb der Nachrichtendarstellung benutzt.

Validierung:

Das Attribut `name` darf maximal 255 Zeichen lang sein, das Attribut `description` 50 Zeichen.

5.4.2.5 StructuredData

Ähnlich wie Attachments können mehrere Payloads vom Typ `StructuredData` hinzugefügt werden.

Dies passiert parallel zu Attachments:

```
StructuredData sd1 = messageComposer.structuredDataBuilder()
    .xmlDocument(xmlInputStream)
    .name("StructuredData1.xml")
    .description("Eine XML-Beschreibung 1")
    .build();
InputStream sd1Stream = sd1.fileUploadRequestBodyStream();
StructuredDataConfidentialMetaData sdmdl = sd1.getMetaData();
```

Die `StructuredData` Instanz wird später noch einmal benötigt und die Daten im Ergebnis `InputStream` müssen hochgeladen werden.

Der unter dem Attribut `name` angegebene Dateiname wird später in der Oberfläche für die Anzeige des Dateinamens in der Liste der Anhänge innerhalb der Nachrichtendarstellung benutzt.

Validierung:

Das Attribut `name` darf maximal 255 Zeichen lang sein, das Attribut `description` 50 Zeichen.

5.4.2.6 ConfidentialMetaData (Vertrauliche Metadaten)

Die vertraulichen Metadaten **können** verschiedene Metadaten enthalten, welche in Kapitel 7.3.2 aufgelistet sind, unter anderem den Betreff der Nachricht. In den vertraulichen Metadaten **müssen** zudem alle der Nachricht angehängten Attachments und `StructuredData` bekannt gegeben werden, wozu wir uns in den vorherigen Abschnitten die jeweiligen Instanzen gemerkt haben:

```
var senderItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.ABSENDER)
    .value("Klaus Mustermann")
    .build();
var abteilungItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.SENDEDEABTEILUNG)
    .value("Abteilung XYZ")
    .build();
var statusItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.STATUS)
    .value("in Bearbeitung")
    .build();
var betreffItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.BETREFF)
    .value("Wichtiges zu Az 4711")
    .build();
var akzItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.AKTENZEICHEN)
    .value("Aktenzeichen 4711")
    .build();
var raItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.RUECKANTWORT)
    .value(RueckantwortValues.JA.value)
    .build();
var rakItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.RUECKANTWORTKANAL)
    .value(RueckantwortKanalValues.MAILTO.value)
```

```
.build();
var raaItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.RUECKANTWORTADRESSE)
    .value("info@behoerde-xyz.de")
    .build();
var knzItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.KNZANTWORTBETREFF)
    .value("Kennzeichnung Demo")
    .build();
var refItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.REFERENZ)
    .value("Referenzierung auf vorherige Nachrichten xy")
    .build();
var leikaItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.LEIKALEISTUNGSGRUPPE)
    .odelistUri("urn:de:gkleika:leika:leistungsgruppierung_20160113")
    .value("025")
    .build();

ConfidentialMetaData cmd = messageComposer.confidentialMetaDataBuilder()
    .attachment(amdl)
    .structuredData(sdmdl)
    .metaDataItem(senderItem)
    .metaDataItem(abteilungItem)
    .metaDataItem(akzItem)
    .metaDataItem(betreffItem)
    .metaDataItem(statusItem)
    .metaDataItem(knzItem)
    .metaDataItem(refItem)
    .metaDataItem(raItem)
    .metaDataItem(rakItem)
    .metaDataItem(raaItem)
    .metaDataItem(leikaItem)
    .build();

InputStream metaDataStream = cmd.fileUploadRequestBodyStream();
```

Auch hier muss nun nur noch der Ergebnis InputStream in den file-upload Endpunkt der ELSTER Transfer API hochgeladen werden.

Validierung:

Folgende Zeichenlimits dürfen für die Werte eines Codes nicht überschritten werden:

Code	Zeichenlimit
SENDEDEABTEILUNG	128
ABSENDER	128
BETREFF	255
AKTENZEICHEN	60

5.4.3 Pflichtfelder und optionale Felder einer Nachricht

Jede empfangene Nachricht im OZGPP besteht aus Pflichtfeldern und optionalen Feldern, die in der Detailansicht im OZGPP dargestellt werden. Bei der Erstellung einer Nachricht für das OZGPP mit dem SDK, welches den Aufbau und die Verschlüsselung einer XHE-Nachricht übernimmt, muss dies berücksichtigt werden. Diese Felder sind in **Fehler! Verweisquelle konnte nicht gefunden werden.** numerisch gekennzeichnet und werden in der **Fehler! Verweisquelle konnte nicht gefunden werden.** den zugehörigen vertraulichen Metadaten im ConfidentialMetaData zugeordnet.

The screenshot shows the 'Detailansicht' of an incoming message in the OZGPP system. The message is titled 'Anfrage zur Firmierung' (9). The sender information (1, 2) is 'Amtsgericht Bremen / Abteilung Recht und Steuern'. The timestamp (3) is '06.07.2023, 11:43 Uhr'. The processing person (4) is 'Christa Müller'. The status (5) is 'Abgelehnt'. The case number (6) is 'AZ AF/429475474/543-S'. The Leika service group (7) is 'Handelsregister'. The message content (8) begins with 'Sehr geehrter Herr Stocker,' and discusses the benefits of entering a company in the Handelsregister.

Abbildung 4 OZGPP Detailansicht einer empfangenen Nachricht

Nr	Feldbezeichnung	Pflicht	Code/Beschreibung
1	Absender	Ja	„absender“ vom Bereitstellungsauftrag (OpenAPI Spezifikation <code>ozgpp-elster-transfer-3.3.0-rest-api-docs.yaml</code>)
2	Absender	Nein	Code „SendendeAbteilung“ vom XHE Confidential Metadata
3	Zeitstempel	Wird generiert	Wird beim Eingang des Bereitstellungsauftrags vom OZGPP-Server generiert
4	Bearbeitende Person	Nein	Code „Absender“ vom XHE Confidential Metadata
5	Status	Nein	Code „Status“ vom XHE Confidential Metadata
6	Aktenzeichen	Nein	Code „AktENZEICHEN“ vom XHE Confidential Metadata

Nr	Feldbezeichnung	Pflicht	Code/Beschreibung
7	LeiKa Leistungsgruppe	Nein	Code „LeiKaLeistungsgruppe“ vom XHE Confidential Metadata. Unterstützt werden die Schlüsselwerte aus dem Leistungskatalog mit der CodelisteURI „urn:de:gkleika:leika:leistungsgruppierung_20160113“
8	Nachricht	Ja	Die Nachricht aus dem Content des XHE-Payload
9	Betreff	Ja	Code „Betreff“ vom XHE Confidential Metadata

Tabelle 5 Felderbeschreibung in der OZGPP-Maske

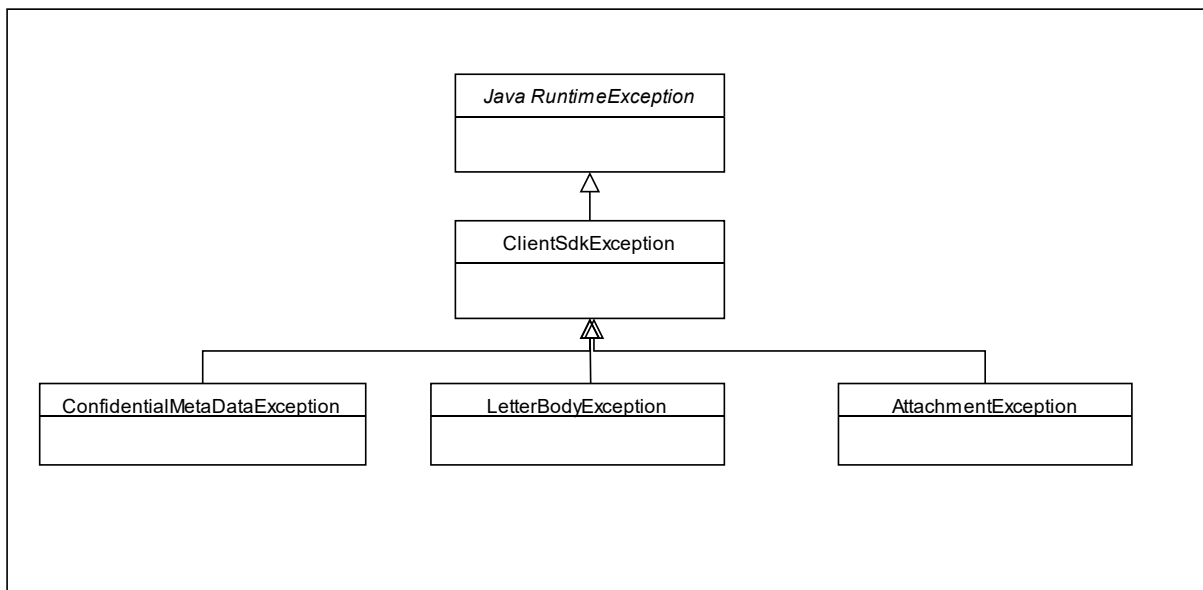
Alle weiteren vertraulichen Metadaten im Confidential Metadata sind optional und werden aktuell in der Verarbeitungskette des OZGPP nicht berücksichtigt.

5.4.4 Ausnahmen bei den Vertraulichen Metadaten

Werden Vertrauliche Metadaten Schlüssel doppelt vergeben, wird eine Laufzeitausnahme geworfen:

```
var azItem = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.AKTENZEICHEN)
    .value("AktENZEICHEN 4711")
    .build();
var azItem1 = ConfidentialMetaDataItem.builder()
    .code(ConfidentialMetaDataCodes.AKTENZEICHEN)
    .value("Wichtiges zu Az 4711")
    .build();
ConfidentialMetaData cmd = messageComposer.confidentialMetaDataBuilder()
    .attachment(a1.getMetaData())
    .structuredData(sd1.getMetaData())
    .metaDataItem(azItem)
    .metaDataItem(azItem1)
    .build(); ← throw ConfidentialMetadataException
```

5.4.5 Ausnahmen Hierarchie im SDK

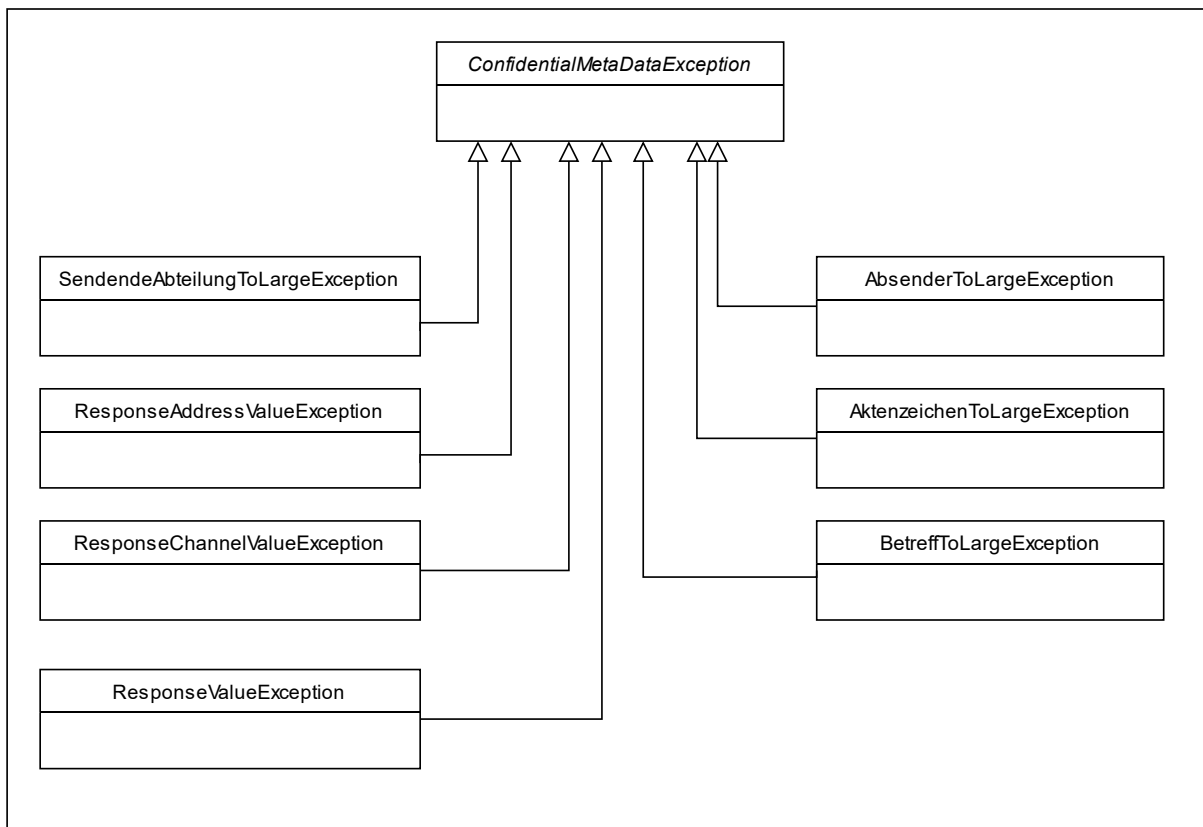


Die Ausnahme `ClientSdkException` wird bei jeder Ausnahme im SDK geworfen und bildet die Basisausnahme im SDK.

Die Ausnahme `ConfidentialMetaDataException` und Kinder wird bei nicht validen Vertraulichen Metadaten geworfen.

Die Ausnahme `LetterBodyException` und Kinder wird bei nicht validem Letter Body geworfen.

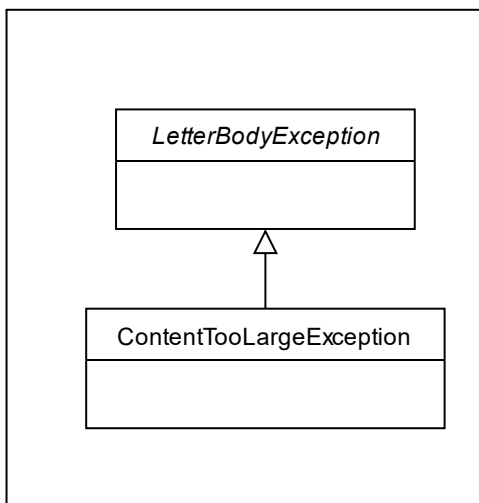
Die Ausnahme `AttachmentException` und Kinder wird bei nicht validem Attachment oder Structured Data geworfen.



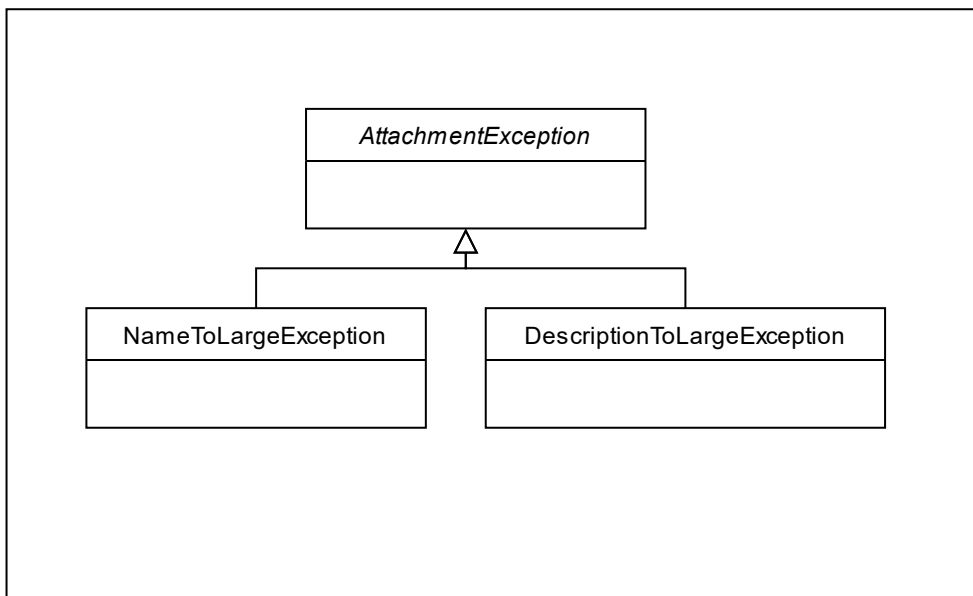
Die Ausnahme `ConfidentialMetaDataException` wird bei nicht Validen Vertraulichen Metadaten geworfen.

Die Ausnahmen `ResponseValueException`, `ResponseChannelValueException` und `ResponseAddressValueException` werden bei folgenden nicht Validen Vertrauliche Metadaten Schlüssel geworfen: `Rueckantwort`, `RueckantwortKanal` und `RueckantwortAdresse`.

Die Ausnahmen `AbsenderToLargeException`, `AktenzeichenToLargeException`, `BetreffToLargeException` und `SendendeAbteilungToLargeException` werden bei entsprechend zu großen Attributwerten geworfen.



Die Ausnahme `ContentTooLargeException` wird bei zu großem Inhalt des Letter Bodys geworfen.



Die Ausnahmen `NameToLargeException` und `DescriptionToLargeException` werden bei entsprechend zu großen Attributwerten geworfen.

Leistungskatalog [urn:de:gk-leika:leika:leistungsgruppierung_20160113](https://nbn-resolving.org/urn:de:gk-leika:leika:leistungsgruppierung_20160113)

Mit diesem Leistungskatalog (LeiKa) wird Deutschlandweit ein einheitliches, vollständiges und umfassendes Verzeichnis der Verwaltungsleistungen über alle Verwaltungsebenen hinweg aufgebaut.

Schlüssel	Bezeichnung
001	Abfall
002	Akademische Grade und Titel
003	Gesundheit
004	Apotheken
005	Arzneimittel
006	Arbeitsschutz
007	Arbeitsförderung
008	Personalausweis
009	Atomare Angelegenheiten
010	Aufenthaltstitel
011	Ausländerangelegenheiten
012	Baurecht
013	Adoption
014	Beglaubigungen und Beurkundungen
015	Behinderte Menschen
016	Beistandschaft
017	berufliche Rehabilitierung
018	Berufsberechtigung
019	Berufsbildung
020	Bodenschutz
021	Börsenangelegenheiten
022	Bundesausbildungsförderung
023	Fahrerlaubnis
024	Fahrerlaubnisregister
025	Gaststätten
026	Fahrzeugangelegenheiten
027	Geburt
028	Gefahrguttransport
029	Fahrzeugregister
030	Bürgerengagement
031	Chemikalien
032	Datenschutz
033	Denkmalschutz
034	Dienstfahrerlaubnis
035	Ehrungen
036	Fahrzeugzulassung
037	Eichrecht
038	Entgeltersatzleistungen
039	Entschädigungsverfahren
040	Existenzgründung
041	Familienförderung

Schlüssel	Bezeichnung
042	Fischerei
043	Grundbuch
044	Genossenschaftsregister
045	Gentechnik
046	Gerichtliche Leistungen
047	Flurstücksangelegenheiten
048	Forst
049	Führungszeugnis
050	Gewerbe
051	Gewaltopferentschädigung
052	Gewerberegister
053	Grünflächen
054	Güterrechtsregister
055	Güterverkehr
056	Häftlingsversorgung
057	Handelsregister
058	Handwerk
059	Heirat
060	Hilfe zur Erziehung
061	Hochschulangelegenheiten
062	Hoheitszeichen
063	Immissionsschutz
064	Informationsfreiheit
065	handwerkliche Berufsbildung
066	Insolvenz
067	Jagd
068	Jugendarbeit
069	Jugendschutz
070	Kartellrecht
071	Kindertagespflege
072	Kindesunterhalt
073	Kirche
074	Kontrollgerätekartensregister
075	Kraftfahreignung
076	Kriegsopferentschädigung
077	Kultur
078	Landwirtschaft
079	Lebenspartnerschaft
080	Luftverkehr
081	Marken
082	Rechtspflege
083	Namen
084	Personenbeförderung
085	Reisepass
086	Schiffsbauregister

Schlüssel	Bezeichnung
087	Schiffsregister
088	Schulangelegenheiten
089	Sicherheit und Ordnung
090	Naturschutz
091	Partnerschaftsregister
092	Patente
093	Pflanzenschutz
094	Rechtsdienstleistungen
095	Scheidung
096	Schifffahrt
097	Spätaussiedler
098	Sport
099	Staatsangehörigkeit
100	Statistik
101	Sterbefall
102	Steuern
103	Stiftungen
104	strafrechtliche Rehabilitierung
105	Straßenpersonenverkehr
106	Pflegeversicherung
107	Sozialleistungen
108	Straßenverkehr
109	Telekommunikation
110	Tierhaltung und Tierschutz
111	Unfallversicherung
112	Unternehmensberatung
113	Unternehmensregister
114	Rentenversicherung
115	Wohnsitz
116	Wohnungswesen
117	Soldaten
118	Verbraucherschutz
119	Vereine
120	Öffentliche Aufträge
121	Zivildienst
122	Zoll
123	Vermessung und Kataster
124	verwaltungsrechtliche Rehabilitierung
125	Visa
126	Vormundschaft
127	Begabtenförderung
128	Wahlen
129	Wasser
130	Grundwehrdienst
131	Weiterbildung

Schlüssel	Bezeichnung
132	Wirtschaftsförderung
133	Anerkennung Vater-/Mutterschaft
134	Krankenversicherung
135	Steuerberatung
136	Hochwasser
137	Katastrophenschutz
138	Erneuerbare Energien
139	Tourismus
140	Architektur
141	Beschaffung
142	Finanzen
143	Innerer Dienst
144	Organisation
145	Personal
146	Sonderleistungen
147	Ingenieurwesen
148	Förderprogramme
150	Anerkennung Ausländischer Berufsqualifikationen

Tabelle 6 LeiKa Leistungsgruppen

6 Umgebungen

Nachfolgend sind die vorhandenen Umgebungen gelistet und beschrieben, wofür diese Umgebungen verwendet werden sollen, wie diese zugreifbar sind und wo Sie notwendige Informationen erhalten.

Umgebung Thema	Integrationsumgebung	Produktionsumgebung
Verwendungszweck	Für die Entwicklung und den Test der Anbindung bzw. Integration des Onlinedienstes / Fachverfahrens vorgesehen	Für die produktive Nutzung des Onlinedienstes / Fachverfahrens
URL Frontend	https://staging.ozgpp.de	https://ozgpp.de
URL Basis für API Fachverfahren	https://staging.ozgpp.de/rest/	https://ozgpp.de/rest/
IP-Absicherung	Nein	Nein
API-Key Absicherung	Ja Den individuellen API Key für die Kommunikation mit den REST Schnittstellen im Rahmen der Anbindung erhalten Sie von Governikus.	Ja Den individuellen produktiven API Key für die Kommunikation mit den REST Schnittstellen erhalten Sie nach erfolgreicher Anbindung von Governikus.
Entity ID für Bausteinpseudonym	https://staging-idp.ozgpp.de/elster-api/sp	https://idp.ozgpp.de/elster/sp

Tabelle 7 Unterscheidung Test- und Produktionsumgebung

7 Beschreibung einer XHE-Nachricht

Nachrichten, die an das OZGPP verschickt werden, müssen ein vorgegebenes Format einhalten. Das Format für solche Nachrichten wurde auf Basis von OASIS XHE 1.0 ausgearbeitet und soll im Folgenden erläutert werden.

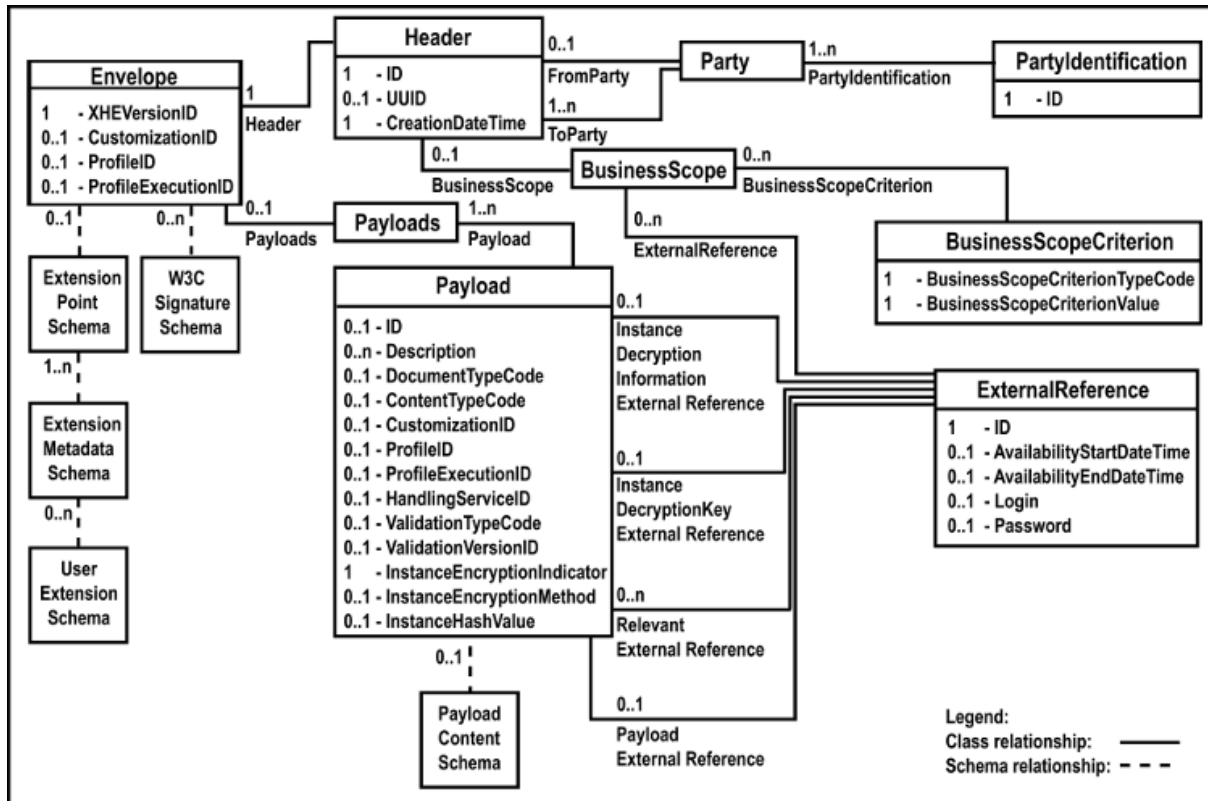


Abbildung 5 XHE Übersicht

Eine OZGPP XHE-Nachricht enthält im Root-Element drei Elemente:

Element/Typ	Anzahl	Beschreibung
XheVersionID xhb:XHEVersionIDType	1	Version Id
Header xha:HeaderType	1	Nicht verschlüsselte Informationen notwendig zur Verarbeitung der Nachricht.
Payloads xha:PayloadsType	1	Die Menge aller Payloads

Tabelle 8 Elemente der XHE-Nachricht

7.1 Header

Element/Typ	Anzahl	Beschreibung
ID xhb:IDType	1	Eindeutiger Identifier zum Tracken von Nachrichten

UUID xhb:UUIDType	1	Zusätzlicher Identifier
CreationDateTime xhb:CreationDateTimeType	1	Erstellungszeitpunkt
BusinessScope xha:BusinessScopeType	1	Öffentliche Meta-Daten über den Anwendungszweck der Nachricht
FromParty xha:PartyType	1	Informationen über den Sender
ToParty xha:PartyType	1	Informationen über den Empfänger

Tabelle 9 Elemente des Headers einer XHE-Nachricht

7.1.1 PartyType und PartyIdentification

Die Elemente FromParty und ToParty enthalten jeweils eine PartyIdentification.

Mit der PartyIdentification werden Informationen über den Sender / den Empfänger mitgegeben.

Element/Typ	Anzahl	Beschreibung
ID xhb:IDType	1	Identifikation der sendenden/empfangenden Behörde bzw. Firma. Für Antworten sollte über diese ID das Leser-Zertifikat ermittelt werden können
attribute:schemeID		Die Identifizierung des Identifizierungsschemas.

Tabelle 10 Party Typ und Identifikation

Beispiel

```
< FromParty>
  < PartyIdentification>
    <ID schemeID="fink-postfach-ref">ab/cd/54321</ID>
  </PartyIdentification>
</FromParty>
<ToParty>
  <PartyIdentification>
    <ID schemeID="ozgp-postfach-ref">cd/ab/12345</ID>
  </PartyIdentification>
</ToParty>
```

7.2 Payloads

Schützenswerte Metadaten der Nachricht, sowie der eigentliche Nachrichteninhalt werden als Payloads in die XHE Nachricht eingebettet. Alle Payloads werden verschlüsselt.

Dabei müssen Payload-Contents in einer XMLENC-EncryptedData-Struktur verschlüsselt sein. Die Algorithmen und Schlüssel (KeyInfo) sollen in allen Payloads einer Nachricht identisch sein. Statt einer asymmetrischen Verschlüsselung des symmetrischen Schlüssels mit einem RSA-Algorithmus soll ein auf elliptischen Kurven basierender hybrider Algorithmus (ECIES) verwendet werden, der effizienter und sicherer ist.

Es ist optional, ob die Payload vor der Verschlüsselung komprimiert werden soll.

Payload

Eine Payload selber besteht aus den folgenden Attributen

Element/Typ	Anzahl	Beschreibung
ID xhb:IDType	1	Eindeutige Payload ID innerhalb einer Nachricht
Description xhb:DescriptionType	0..1	Beschreibung der Payload
DocumentTypeCode xhb:DocumentTypeCodeType	0..1	Archetyp der Payload
ContentTypeCode xhb:ContentTypeCodeType	1	Dateiformat oder Octet Darstellung der Payload text/plain application/pdf
InstanceEncryptionIndicator xhb:InstanceEncryptionIndicatorType	1	Indikator zur Angabe, ob die Payload verschlüsselt ist oder nicht
InstanceEncryptionMethod xhb:InstanceEncryptionMethodType	0..1	Verschlüsselungsmethode
InstanceHashValue xhb:InstanceHashValueType	0..1	SHA-256 Hash der unverschlüsselten Payload
PayloadContent xenc:EncryptedDataType	1	Inhalt der Payload

Tabelle 11 Attribute einer Payload

Payload-Arten

Eine Nachricht kann folgende unterschiedliche Arten einer Payload beinhalten:

ID	ContentType Code	DocumentTypeCode	Anzahl	Inhalt
ConfidentialMetaData	text/xml	Verweis auf Schema	1	XML
LetterBody	text/plain		1	Freitext

StructuredContent_n	text/xml	Verweis auf Schema	0..n	XML
Attachment_n	application/pdf		0..n	Datei

Tabelle 12 Payloadarten

7.2.1 ConfidentialMetaData

Der PayloadContent einer Payload vom Typ ConfidentialMetaData muss aus encodeten ConfidentialMetaData bestehen, welche wiederum mehrere ConfidentialMetaDatum aufweisen.

Element/Typ	Anzahl	Inhalt
ConfidentialMetaDatum muwista5:ConfidentialMetaDatumType	1..n	Key/Value-Paare

Tabelle 13 Payload Inhalt mit Metadaten

7.2.1.1 ConfidentialMetaDatumType

Element/Typ	Anzahl	Inhalt
Code xs:token	1	Code aus der Codeliste (Tabelle 18)
attribute:codelistURI xs:string	0..1	Verweis auf eine für Value verwendete Codeliste. Beispiel für Code = LeikaLeistungsgruppe: urn:de:gkleika:leika:leistungsgruppierung
xs:choice		
Value xs:string	1	Wert
PayloadAttributes muwista5:PayloadAttributes	1	Für Code = StructuredData und Attachment

Tabelle 14 Typenbeschreibung der Confidential Meta Daten

PayloadAttributes setzen sich folgendermaßen zusammen

Element/Typ	Anzahl	Inhalt
PayloadAttribute muwista5:ConfidentialMetaDatumType	1..n	Key/Value-Paare Codes entstammen der Code Liste (Tabelle 19)

Tabelle 15 Zusammensetzung der Payload Attribute

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ConfidentialMetaData xmlns:ns2="http://muwista5.de/schema">
  <confidentialMetaDataItem codelistURI="urn:de:gkleika:leika:leistungsgruppierung">
    <code>LeikaLeistungsgruppe</code>
    <value>1234</value>
  </confidentialMetaDataItem>
  <confidentialMetaDataItem>
    <code>SendendeAbteilung</code>
    <value>Abt. 123</value>
  </confidentialMetaDataItem>
  <confidentialMetaDataItem>
    <code>Aktenzeichen</code>
    <value>K/12345/123</value>
  </confidentialMetaDataItem>
  <confidentialMetaDataItem codelistURI="urn:muwista5.de:codelist:payload-attributes">
    <code>Attachment</code>
    <payloadAttributes>
      <payloadAttribute>
        <code>Id</code>
        <value>Attachment_1</value>
      </payloadAttribute>
      <payloadAttribute>
        <code>Name</code>
        <value>original-filename.pdf</value>
      </payloadAttribute>
      <payloadAttribute>
        <code>Description</code>
        <value>Beschreibung der Datei</value>
      </payloadAttribute>
    </payloadAttributes>
  </confidentialMetaDataItem>
</ns2:ConfidentialMetaData>
```

7.3 Code Lists

Für die Code-Listen wurde ein Schema in Anlehnung an Peppol Edec Code-Listen erstellt. Die Code-Listen werden primär in Excel-Dateien gepflegt. Mit einem von OpenPeppol abgeleiteten Pflege-Tool werden aus den Excel-Dateien vier Formate generiert: xml, html, json, gc. Die Namen der Excel-Dateien und der generierten Listen entsprechen den Peppol-Konventionen.

Element	Typ	Anzahl	Inhalt
Code	xs:token	1	Primärer Schlüssel der Code List

Name	xs:string	1	Anzeigename
Description	xs:string	0..1	Beschreibung
Type	xs:string	0..1	Type des Wertes Default: xs:string
Pattern	xs:string	0..1	Pattern des Wertes (for string types)
State	StateType	1	Zustand des Code List Eintrags (active, deprecated, removed)
initial-release	VersionType	1	Versionsnummer der Code Liste als der Eintrag eingeführt wurde
deprecation-release	VersionType	0..1	Nur vorhanden, wenn der Eintrag deprecated ist. Code List Version seit der der Eintrag deprecated ist.
removal-date	xs:date	0..1	Nur vorhanden, wenn der Eintrag entfernt werden soll. Datumswert an dem der Eintrag entfernt wurde.

Tabelle 16 Codeliste Shemadarstellung

7.3.1 Public Metadata v1.0

Die URN dieser Code List lautet: urn:muwista5.de:codelist:public-metadata

Code	Description
Sender	Absender-Identität, z.B. Name der Behörde oder des Unternehmens
Recipient	Empfänger-Identität, z.B. Name der Behörde oder des Unternehmens
CreationTime	Zeitstempel des Autors (Behörde oder Firmen-MA)
TransferTime	Beginn des Versandvorgangs
EncryptionTime	Beginn der Verschlüsselung
	Weitere Zeitpunkte bzgl. der Übermittlung und dem Empfang werden durch OSCI und XTA protokolliert und bereitgestellt.

Tabelle 17 Codeliste der Public Metadaten

7.3.2 Confidential Metadata v1.0 Metadata v1.0

Die URN dieser Code List lautet: urn:muwista5.de:codelist:confidential-metadata

Code	Beschreibung	Voraussetzungen	Wo verfügbar
LeikaLeistungsgruppe	Leika Leistungsgruppierung	z.B. Behinderte Menschen	Die digitale Verwaltungsleistung kann diese Information mitgeben
SendendeAbteilung	Sendende Abteilung	Name der Abteilung z.B. Abt. für Ordnungswidrigkeiten oder Abteilung des Unternehmens/Funktionspostfach	Die digitale Verwaltungsleistung kann diese Information mitgeben
Aktenzeichen	Aktenzeichen oder ähnliches	Aktenzeichen oder Kennung der eröffneten Verwaltungsleistung	Die digitale Verwaltungsleistung kann diese Information mitgeben
Status	Status	Status der Antragsbearbeitung, z.B. in Bearbeitung	Jedes Bundesland behandelt das individuell im Rahmen der vorhandenen Bürger/Organisation skonten. Wird wohl auch im Rahmen von FINK länderübergreifend harmonisiert.
Absender	Absender Name, Vorname	Name des Sachbearbeiters, bzw. bei Org→Verw. Name des Nutzers	
Betreff	Betreff	Freitextfeld oder automatisches Befüllen durch Aktenzeichen	
Rueckantwort	Rückantwort Ja/Nein	Es gibt Verwaltungsleistungen, bei denen keine Rückantwort durch das Unternehmen erfolgen soll.	Dieses Flag wird von der Verwaltungsleistung gesetzt.

Code	Beschreibung	Voraussetzungen	Wo verfügbar
RueckantwortKanal	Rückantwort Kanal [ozgpp mailto url]	Rückantwort Flag ist auf 'Ja' gesetzt	Die digitale Verwaltungsleistung kann diese Information mitgeben
RueckantwortAdresse	Rückantwort Adresse, z.B. E-Mail Adresse oder URL zu einem Fachverfahren	Rückantwort Flag ist auf 'Ja' gesetzt und Kanal ist 'mailto' oder 'url'	Die digitale Verwaltungsleistung kann diese Information mitgeben
KnzAntwortBetreff	Kennzeichnung von Antwort im Betreff	Wie z.B. bei E-Mail	
Referenz	Referenzierung auf vorherige Nachrichten	Bei 'Dialogen' mit Verwaltung	
StructuredData	Name oder Beschreibung zum Payload		
Attachment	Name oder Beschreibung zum Payload		

Tabelle 18 Codeliste der Confidential Metadaten

7.3.3 Payload Attributes v1.0

Die URN dieser Code List lautet: urn:muwista5.de:codelist:payload-attributes

Code	Description
ID	Payload ID
Name	Name des Dokuments
Description	Beschreibung des Dokuments
	Weitere Attribute sind noch zu bestimmen

Tabelle 19 Codeliste der Payload Attribute

7.4 Beispiel XHE

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns5:XHE xmlns="http://docs.oasis-open.org/bdxc/ns/XHE/1/BasicComponents"
  xmlns:ns6="http://www.w3.org/2001/04/xmlenc#"
  xmlns:ns5="http://muwista5.de/schema"
  xmlns:ns4="http://www.w3.org/2000/09/xmldsig#"
  xmlns:ns3="http://docs.oasis-open.org/bdxc/ns/XHE/1/AggregateComponents">
  <XHEVersionID>1.0</XHEVersionID>
```

```

<ns3:Header>
  <ID>2c6c9ec1-11e1-4863-92bf-df1efcf45afb</ID>
  <CreationDateTime>2023-03-01T15:03:52.959+01:00</CreationDateTime>
  <ns3:BusinessScope>
    <ns3:BusinessScopeCriterion>

<BusinessScopeCriterionTypeCode>CreationTime</BusinessScopeCriterionTypeCode>
  <BusinessScopeCriterionValue>2023-03-01T15:03:52.515+01:00</BusinessScopeCriterionValue>
  </ns3:BusinessScopeCriterion>
</ns3:BusinessScope>
  <ns3:FromParty>
    <ns3:PartyIdentification>
      <ID>by/hh/54321</ID>
    </ns3:PartyIdentification>
  </ns3:FromParty>
  <ns3:ToParty>
    <ns3:PartyIdentification>
      <ID>hh/by/12345</ID>
    </ns3:PartyIdentification>
  </ns3:ToParty>
</ns3:Header>
<ns3:Payloads>
  <ns3:Payload>
    <ID>ConfidentialMetaData</ID>
    <ContentTypeCode>text/xml</ContentTypeCode>
    <InstanceEncryptionIndicator>true</InstanceEncryptionIndicator>
    <ns3:PayloadContent>
      <ns6:EncryptedData Id="ConfidentialMetaData" MimeType="text/xml">
        <ns6:EncryptionMethod
Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm"/>
        <ns4:KeyInfo>...</ns4:KeyInfo>
        <ns6:CipherData>
          <ns6:CipherValue>qQbFnp...A3+64=</ns6:CipherValue>
        </ns6:CipherData>
        <ns6:EncryptionProperties>
          <ns6:EncryptionProperty Target="content" Id="compressed"/>
        </ns6:EncryptionProperties>
      </ns6:EncryptedData>
    </ns3:PayloadContent>
  </ns3:Payload>
  <ns3:Payload>
    <ID>LetterBody</ID>
    <ContentTypeCode>text/plain</ContentTypeCode>
    <InstanceEncryptionIndicator>true</InstanceEncryptionIndicator>
    <ns3:PayloadContent>
      <ns6:EncryptedData Id="LetterBody" MimeType="text/plain">
        <ns6:EncryptionMethod
Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm"/>
        <ns4:KeyInfo>...</ns4:KeyInfo>
        <ns6:CipherData>
          <ns6:CipherValue>TKLpCyA+...93vsYnv</ns6:CipherValue>
        </ns6:CipherData>
        <ns6:EncryptionProperties>
          <ns6:EncryptionProperty Target="content" Id="compressed"/>
        </ns6:EncryptionProperties>
      </ns6:EncryptedData>
    </ns3:PayloadContent>
  </ns3:Payload>
  <ns3:Payload>

```

```
<ID>Attachment_1</ID>
<ContentTypeCode>application/pdf</ContentTypeCode>
<InstanceEncryptionIndicator>true</InstanceEncryptionIndicator>
<ns3:PayloadContent>
  <ns6:EncryptedData Id="Attachment_1" MimeType="text/xml">
    <ns6:EncryptionMethod
Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm"/>
    <ns4:KeyInfo>...</ns4:KeyInfo>
    <ns6:CipherData>
      <ns6:CipherValue>XR8rmAkL...ZS3o8rrWqI=</ns6:CipherValue>
    </ns6:CipherData>
    <ns6:EncryptionProperties>
      <ns6:EncryptionProperty Target="content" Id="compressed"/>
    </ns6:EncryptionProperties>
  </ns6:EncryptedData>
</ns3:PayloadContent>
</ns3:Payload>
<ns3:Payload>
  <ID>StructuredData_1</ID>
  <ContentTypeCode>text/xml</ContentTypeCode>
  <InstanceEncryptionIndicator>true</InstanceEncryptionIndicator>
  <ns3:PayloadContent>
    <ns6:EncryptedData Id="StructuredData_1" MimeType="text/xml">
      <ns6:EncryptionMethod
Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm"/>
      <ns4:KeyInfo>...</ns4:KeyInfo>
      <ns6:CipherData>
        <ns6:CipherValue>thmR5wG...mK4KUjbCY938IOs=</ns6:CipherValue>
      </ns6:CipherData>
      <ns6:EncryptionProperties>
        <ns6:EncryptionProperty Target="content" Id="compressed"/>
      </ns6:EncryptionProperties>
    </ns6:EncryptedData>
  </ns3:PayloadContent>
</ns3:Payload>
</ns3:Payloads>
</ns5:XHE>
```

Hinweis: Das hier beschriebene Beispiel dient zur Veranschaulichung der XHE-Nachricht und stellt nicht den aktuellsten Stand des OZGPP wieder. In der laufenden Anwendung werden die XHE-Payload-Elemente („ConfidentialMetaData“, „LetterBody“, „Attachment“, „StructuredData“) einzeln übermittelt und im Bereitstellungsauftrag referenziert.