Transfer Webservice v4 (public)

Webservice Schnittstelle, die verschiedene Methoden zum Senden und Empfangen von Dateien bereitstellt. Diese Schnittstelle stellt eine Alternative zur Webtrans- und FTPS-Schnittstelle dar.

Abgesichert ist dieses Webservice über SecurityParameters im Request Objekt (Seriennummer, Kundenpasswort, API Key).

1. Unterschied zu Transfer Webservice v3

- SendenResult liefert nun zusätzlich
  - die Protokollnummer
  - Zusätzliche Statuscode, die über den Verarbeitungsstatus der Datei Auskunft geben. Dafür wurde das **Feld verarbeitungsStatus aus SendenResult entfernt**:
    - konnte die Datei nicht verabeitet werden wird Statuscode 403 retourniert.
    - handelt es sich bei der Datei um ein Duplikat wird Statuscode 405 retourniert.
    - dauert die Verarbeitung länger als 40 Sekunden und ist noch im Gange wird Statuscode 404 retourniert.
    - Statuscode 000 bedeutet, dass die Datei grundsätzlich verarbeitet werden kann. Fehlgeschlagene Inhaltsprüfungen haben aber keine Auswirkung auf den Wert des Statuscodes - auch in diesem Fall wird Statuscode 000 zurückgeliefert.
  - Fehler und Warnungen, die bei der Verarbeitung auftreten, im messages-Block

2. Endpoint Adressen

- Kundenintegration: https://online-test.elda.at/eldaws/transfer/v4/TransferService
- Systemintegrationstest: https://online-itu5test.elda.at/eldaws/transfer/v4/TransferService
- Produktion: https://online.elda.at/eldaws/transfer/v4/TransferService

3. Methoden

3.1. senden

Methode zum Übertragen von Dateien

3.2. ruecksendungenAuflisten

Liste von Ruecksende-Objekten (Protkollnummer & Dateiname) der abholbereiten Rücksendungen. Welche Datei/Protokollnummer ursächlich für die Rücksendung ist, steht in Rücksendung.dateiName: dort ist die Protokollnummer der entsprechenden Datei (siehe SendenResult.protokollnummer) enthalten. Mit Ruecksendung.protokollnummer und der Methode empfangen kann die Rücksendedatei abgeholt werden.

3.3. empfangen

Anhand von einer Protokollnummer (aus dem Result von 'ruecksendungenAuflisten' bzw. Ruecksendung.protokollnummer) kann eine Rücksendung abgerufen werden.

4. Request-/Result-Objekte

4.1. SendenRequest

Attribut	Typ	Anmerkungen
payload	base64Binary	Base64 kodierte Binärdaten
dateiName	String	

4.2. SendenResult

Attribut	Typ	Anmerkungen

ServiceResult.
statusCode
	String	• 000: OK
		• 401: dateiName zu lange (max 255)
		• 402: dateiName nicht gesetzt
		• 403: Datei nicht verarbeitet
		  ◦ ist dieser Code gesetzt befindet sich SendenResult.serviceResult.messages ein String mit dem auslösenden Fehlercode (zB: "fehlerCode: E1")
		• 404: Datei wird noch verarbeitet
		  ◦ dieser Code wird gemeldet, wenn die Verarbeitung der Datei länger als 40 Sekunden in Anspruch nimmt.
		• 405: Datei ist Duplikat
		  ◦ ist dieser Code gesetzt befindet sich SendenResult.serviceResult.messages ein String mit der Protokollnummer der Originaldatei (zB: "duplikatVon: 123456789")
		• 500: Interner Fehler
		
		Weitere Codes siehe Abschnitt "Status Codes".

dateiId	Long	Eindeutige, fortlaufende Nummer der empfangenen Datei
eldaZeitstempel	Date	Zeitpunkt des Empfangs in ELDA
protokollnummer	Long	ELDA-Protkollnummer der empfangenen Datei

4.3. RuecksendungenAuflistenRequest

Attribut	Typ	Anmerkungen
keine zusätzlichen Parameter notwendig		

4.4. RuecksendungenAuflistenResult

Attribut	Typ	Anmerkungen

ServiceResult.statusCode	String	• 000: OK • 500: Interner Fehler Weitere Codes siehe Abschnitt "Status Codes".
ruecksendungen	List<Ruecksendung>	

4.5. EmpfangenRequest

Attribut	Typ	Anmerkungen
protokollnummer	Long	

4.6. EmpfangenResult

Attribut	Typ	Anmerkungen
ServiceResult. statusCode	String	• 000: OK • 406: Datei mit Protokollnummer nicht vorhanden • 407: Keine Berechtigung um Datei zu empfangen (Seriennummer stimmt nicht mit der ursprünglich übermittelten Datei überein) • 500: Interner Fehler Weitere Codes siehe Abschnitt "Status Codes".
datei	Datei	

---

5. Objekte

EldaWebserviceRequest

5.1. SecurityParameters

Attribut	Typ	obligatorisch	Anmerkung
nonce	String	Ja	Zufällige Zeichenkette, die vom Client gesetzt werden muss
created	Date	Ja	Erstellzeitpunkt des Request Objekts
seriennummer	String	Ja	Seriennummer des ELDA Kunden
kundenpasswort	String	Ja	Kundenpasswort des ELDA Kunden im Format SHA512 hex lowercase
apiKey	String	Ja	Von ELDA vergebener Wert zur Identifizierung des verwendeten Clients

Durch das Verwenden von nonce und created sollen Replay Attacken verhindert werden. Nonces müssen eindeutig sein

- nonce: Vom Client zu erstellen. Müssen eindeutig sein. Leer (Fehlercode 554) und Duplikate (Fehlercode 552) nicht zulässig . Serverseitig wird ein Nonce Cache verwaltet (registrierte Nonces werden nach einer Zeit gelöscht).
- created: Requests sind nur eine gewisse Zeit gültig. Sind sie abgelaufen bekommt der Client den Fehlercode 551 bzw. Fehlercode 555 falls created nicht gesetzt.

5.2. Datei

Attribut	Typ	Anmerkungen
id	Long	= Datei ID
name	String	Dateiname inkl. Endung
payload	application/octet-stream	Binärdaten als Attachment
dateiTyp	Integer	

md5	String	Wert aus datei.md5

6. Ruecksendung

Attribut	Typ	Anmerkungen
protokollnummer	Long	Protokollnummer der Datei
dateiName	String	Dateiname inkl. Endung

7. Status Codes

Code	Beschreibung	SendenResult	RuecksendungenAuflistenResult	EmpfangenResult
000	OK	✓	✓	✓
500	Interner Verarbeitungsfehler.	✓	✓	✓
551	Request abgelaufen (created älter als 60 Sekungen).	✓	✓	✓
552	Nonce wurde bereits verwendet.	✓	✓	✓
553	Seriennummer für dieses Service nicht berechtigt.	✓	✓	✓
554	Nonce nicht gesetzt.	✓	✓	✓
555	Created nicht gesetzt.	✓	✓	✓
557	API Key ungültig.	✓	✓	✓
558	Seriennummer und/oder Kundepasswort falsch.	✓	✓	✓
559	Unerlaubter Content Type.	✓	✓	✓
401	dateiName zu lange (max 255)	✓	✗	✗
402	dateiName nicht gesetzt	✓	✗	✗
403	Datei nicht verarbeitet	✓	✗	✗
404	Datei wird noch verarbeitet dieser Code wird gemeldet, wenn die Verarbeitung der Datei länger als 40 Sekunden in Anspruch nimmt.	✓	✗	✗
405	Datei ist Duplikat ist dieser Code gesetzt befindet sich SendenResult.serviceResult.messages ein String mit der Protokollnummer der Originaldatei (zB: "duplikatVon: 123456789")	✓	✗	✗
406	Datei mit Protokollnummer nicht vorhanden	✗	✗	✓
407	Keine Berechtigung um Datei zu empfangen (Seriennummer stimmt nicht mit der ursprünglich übermittelten Datei überein)	✗	✗	✓

8. Beispiel Code

Um einen Soap Request in Java zu erstellen, werden folgende **javax.xml.soap** Factory Klassen benötigt.

javax.xml.soap Factories

```
private SOAPConnectionFactory soapConnectionFactory;
private SOAPConnection soapConnection;
private MessageFactory messageFactory;

// Initialisierung
soapConnectionFactory = SOAPConnectionFactory.newInstance();
soapConnection = soapConnectionFactory.createConnection();
messageFactory = MessageFactory.newInstance();
```

8.1. Request allgemein

Als Request Objekt dient eine **javax.xml.soap.SoapMessage**. Alle Request-Objekte müssen mit den Security Parameters (Seriennummer, Kundenpasswort, apiKey, etc.) befüllt werden. Dieses Request Objekt dient als Ausgangsbasis für alle Webservice Methoden.

Request Objekt befüllen

```
String method = "senden"; // oder "empfangen", "ruecksendungenAuflisten"
String version = "v4";
String namespace = "http://v4.transfer.ws.elda.at/";

SOAPMessage soapMessage = messageFactory.createMessage();
SOAPBody soapBody = soapMessage.getSOAPPart().getEnvelope().getBody();
SOAPElement methodElement = soapBody.addChildElement(method, version, namespace);        //Hier werden die
Parameter Methode, Version und Namespace im Soap Envelope gesetzt
SOAPElement arg0Element = methodElement.addChildElement("arg0");
SOAPElement securityElement = arg0Element.addChildElement("securityParameters");
SOAPElement createdElement = securityElement.addChildElement("created");
createdElement.addTextNode(createCreate());
SOAPElement nonceElement = securityElement.addChildElement("nonce");
nonceElement.addTextNode(createNonce());
SOAPElement apiKeyElement = securityElement.addChildElement("apiKey");
apiKeyElement.addTextNode(apiKey);
SOAPElement seriennummerElement = securityElement.addChildElement("seriennummer");
seriennummerElement.addTextNode(username);
SOAPElement passwortElement = securityElement.addChildElement("kundenpasswort");
passwortElement.addTextNode(password);
```

Hilfsmethoden

```
private String createNonce() {
        return UUID.randomUUID().toString();
}

private String createCreate() {
        return new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'").format(new Date());
}
```

8.2. Webservice Aufruf

Der Aufruf des Webservices ist für alle Methoden gleich. Der Call wird mit dem Request Objekt (der oben erstellten **javax.xml.soap.SoapMessage**) und der **javax.xml.soap.SoapConnection** abgesetzt.

Webservice Aufruf

```
SoapMessage request = messageFactory.createMessage();
...
//Security Parameters und Methoden-spezifische Parameter (zB.: Protokollnummer bei empfangen Request) setzen
...
//Endpoint an den der Request gesendet wird (hier im Beispiel die Prod-Url des Webservices)
String endpointUrl = "https://online.elda.at/eldaws/transfer/v4/TransferService";
//Webservice Call
SOAPMessage soapResponse = soapConnection.call(request, endpointUrl);
```

8.3. Response allgemein

Alle Methoden liefern zumindest einen StatusCode mit (000 = OK), der wie hier aus dem Response ausgelesen werden kann.

Response Verarbeitung

```
SOAPMessage soapResponse = soapConnection.call(request, endpointUrl);
NodeList nodeResult = soapResponseBody.getElementsByTagName("serviceResult");
for (int i = 0; i < nodeResult.getLength(); i++) {
        Node node = nodeResult.item(i);
        String statusCode = node.getTextContent();
        LOGGER.info("StatusCode " + statusCode);
}
```

8.4. Webservice Methoden

Als Ausgangsbasis für den Request dient immer das Request Objekt mit gesetzten Security Parameters. Die Methoden-spezifischen Parameter werden an das arg0-Node des Requests angehängt.

8.4.1. senden

8.4.1.1. Request

Mit dieser Methode kann eine Datei an ELDA übermittelt werden.

Die Senden-Methode benötigt **Dateiname** und **Inhalt** zusätzlich zum Ausgangsbasis Request Objekt, wobei der Inhalt als Soap Attachment (**javax.xml.soap.AttachmentPart**) übergeben wird.

Dateiname zu arg0 hinzufügen

```
SOAPElement element = arg0Element.addChildElement("dateiName");        //arg0 Element aus Ausgangsbasis Request
Objekt
element.addTextNode("meineDatei.txt");
```

Payload (= zu übermittelnde Datei) zu arg0 als Attachment hinzufügen

```
File file = new File("D:/path/to/file.txt");
byte[] fileBytes = Files.readAllBytes(file.toPath());
AttachmentPart attachmentPart = soapMessage
                                .createAttachmentPart(new DataHandler(fileBytes, "application/octet-stream"));
attachmentPart.setContentId("payload");
soapMessage.addAttachmentPart(attachmentPart);
SOAPElement payload = arg0Element.addChildElement("payload");        //arg0 Element aus Ausgangsbasis Request
Objekt
Element includeElement = payload.getOwnerDocument().createElementNS("http://www.w3.org/2004/08/xop/include",
"xop:Include");
includeElement.setAttribute("href", "cid:" + attachmentPart.getContentId());
payload.appendChild(includeElement);
soapMessage.saveChanges();
```

8.4.1.2. Senden Request Beispiel aus SoapUI

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v4="http://v4.transfer.ws.
elda.at/">
   <soapenv:Header/>
   <soapenv:Body>
      <v4:senden>
         <arg0>
            <securityParameters>
               <apiKey>_apiKey_</apiKey>
               <created>${=  String.format('%tF', new Date() + 10) }</created>
               <kundenpasswort>_passwort_</kundenpasswort>
               <nonce>${=java.util.UUID.randomUUID().toString()}</nonce>
               <seriennummer>_seriennummer_</seriennummer>
            </securityParameters>
            <dateiName>meineDatei.txt</dateiName>
            <payload>cid:1526066113758</payload>
         </arg0>
      </v4:senden>
   </soapenv:Body>
</soapenv:Envelope>
```

8.4.1.3. Response

Der Response der Senden-Methode beinhaltet (zusätzlich zum StatusCode) die Datei-Id der gesendeten Datei, den Verarbeitungsstatus und einen Zeitstempel.

Die Werte können wie folgendermaßen aus dem Response ausgelesen werden.

Response Verarbeitung

```
SOAPMessage soapResponse = soapConnection.call(request, endpointUrl);
SOAPBody soapResponseBody = soapResponse.getSOAPBody();
NodeList nodeId = soapResponseBody.getElementsByTagName("dateiId");               //statt der "dateiId" kann
hier auch "protokollnummer" oder "eldaZeitstempel" ausgelesen werden
for (int i = 0; i < nodeId.getLength(); i++) {
        Node node = nodeId.item(i);
        String dateiId = node.getTextContent();
        LOGGER.info("Datei gesendet, id " + dateiId);
}

/* Output
Datei gesendet, id 199565708
*/
```

8.4.1.4. Senden Response Beispiel aus SoapUI

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:sendenResponse xmlns:ns2="http://v4.transfer.ws.elda.at/">
         <return>
            <serviceResult>
               <messages>fehlerCode: E1</messages>
               <statusCode>403</statusCode>
            </serviceResult>
            <dateiId>155764331</dateiId>
            <eldaZeitstempel>2025-11-25T13:43:55.057+01:00</eldaZeitstempel>
            <protokollnummer>155764331</protokollnummer>
         </return>
      </ns2:sendenResponse>
   </soap:Body>
</soap:Envelope>
```

8.4.2. rücksendungenAuflisten

8.4.2.1. Request

Mit dieser Methode können die Protokollnummern und Dateinamen der Rücksendungen aus ELDA aufgelistet werden. Diese Protokollnummern können dann in der empfangen-Methode verwendet werden, um diese Rücksendungen abzufragen.

Diese Methode benötigt **keine zusätzlichen Parameter** zur Ausgangsbasis des Request Objekts.

8.4.2.2. Request Beispiel aus SoapUI

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v4="http://v4.transfer.ws.
elda.at/">
   <soapenv:Header/>
   <soapenv:Body>
      <v4:ruecksendungenAuflisten>
         <arg0>
            <securityParameters>
               <apiKey>_apiKey_</apiKey>
               <created>${=  String.format('%tF', new Date() + 10) }</created>
               <kundenpasswort>_passwort_</kundenpasswort>
               <nonce>${=java.util.UUID.randomUUID().toString()}</nonce>
               <seriennummer>_seriennummer_</seriennummer>
            </securityParameters>
         </arg0>
      </v4:ruecksendungenAuflisten>
   </soapenv:Body>
</soapenv:Envelope>
```

8.4.2.3. Response

Der Response der RücksendungenAuflisten Methode beinhaltet (zusätzlich zum StatusCode) eine Liste der Rücksendungen bestehend aus dem Objekt "Ruecksendung", welche die Protokollnummer und den Dateinamen beinhalten.

Die Werte können wie folgendermaßen aus dem Response ausgelesen werden.

Response Verarbeitung

```
SOAPMessage soapResponse = soapConnection.call(request, endpointUrl);
Map<String, String> resultList = new TreeMap<>();
SOAPBody soapResponseBody = soapResponse.getSOAPBody();
NodeList nodeList = soapResponseBody.getElementsByTagName("ruecksendungen");
for (int i = 0; i < nodeList.getLength(); i++) {
    Element element = (Element) nodeList.item(i);
        String protokollnummer = element.getElementsByTagName("protokollnummer").item(0).getTextContent();
        String dateiname = element.getElementsByTagName("dateiName").item(0).getTextContent();
        resultList.put(protokollnummer, dateiname);
}
LOGGER.info(resultList.size() + " einträge gefunden");

/* Output
12  einträge gefunden
*/
```

8.4.2.4. Response Beispiel aus SoapUi

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:ruecksendungenAuflistenResponse xmlns:ns2="http://v4.transfer.ws.elda.at/">
         <return>
            <serviceResult>
               <messages>OK</messages>
               <statusCode>000</statusCode>
            </serviceResult>
            <ruecksendungen>
               <dateiName>fehler</dateiName>
               <protokollnummer>155764332</protokollnummer>
            </ruecksendungen>
         </return>
      </ns2:ruecksendungenAuflistenResponse>
   </soap:Body>
</soap:Envelope>
```

8.4.3. empfangen

8.4.3.1. Request

Mit der Methode empfangen kann eine Rücksendung über die Protokollnummer empfangen werden.

Diese Methode benötigt die **Protokollnummer** der Rücksendung zusätzlich zum Ausgangsbasis Request Objekt.

Protokollnummer zu arg0 hinzufügen

```
SOAPElement element = arg0Element.addChildElement("protokollnummer"); //arg0 Element aus Ausgangsbasis Request
Objekt
element.addTextNode("123456");
```

8.4.3.2. Request Beispiel aus SoapUI

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v4="http://v4.transfer.ws.
elda.at/">
   <soapenv:Header/>
   <soapenv:Body>
      <v4:empfangen>
         <arg0>
            <securityParameters>
               <apiKey>_apiKey_</apiKey>
               <created>${=  String.format('%tF', new Date() + 10) }</created>
               <kundenpasswort>_passwort_</kundenpasswort>
               <nonce>${=java.util.UUID.randomUUID().toString()}</nonce>
               <seriennummer>_seriennummer_</seriennummer>
            </securityParameters>
            <protokollnummer>1</protokollnummer>
         </arg0>
      </v4:empfangen>
   </soapenv:Body>
</soapenv:Envelope>
```

8.4.3.3. Response

Der Response der Empfangen Methode beinhaltet (zusätzlich zum StatusCode) die abgefragte Datei als Attachment sowie weitere Infos über die Datei (typ, name, md5, etc).

Die Werte können wie folgendermaßen aus dem Response ausgelesen werden.

Response Verarbeitung

```
SOAPMessage soapResponse = soapConnection.call(request, endpointUrl);
//Auslesen des Response Attachments
String receiveDirectory = "D:/path/to/receiveDirectory/";                                    //hier hin wird die
empfangene Datei geschrieben
soapResponse.getAttachments().forEachRemaining(att -> {
        try {
                String fileName = extractFilename(att.getMimeHeader("Content-Disposition")[0]);
                Files.write(Paths.get(receiveDirectory + fileName), att.getDataHandler().getInputStream().
readAllBytes());
        } catch (Exception e) {
                throw new RuntimeException(e);
        }
});

SOAPBody soapResponseBody = soapResponse.getSOAPBody();
NodeList nodeList = soapResponseBody.getElementsByTagName("datei");               //Element Datei wird
ausgewählt
for (int i = 0; i < nodeList.getLength(); i++) {
        Node node = nodeList.item(i);
        NodeList children = node.getChildNodes();
        for (int j = 0; j < children.getLength(); j++) {                                                    //Datei
hat Children-Nodes, die weitere Infos beinhalten (siehe unten Output)
                Node child = children.item(j);
                String nodeName = child.getNodeName();
                String nodeValue = child.getTextContent();
                LOGGER.info("Node "  + nodeName + " with value "+ nodeValue);
        }
}


/* Output
Node dateiTyp with value XML
Node id with value 199565708
Node md5 with value be2de85640149ca44c18b873d52c6d8f
Node name with value mitteilung_1452548.xml
*/
```

8.4.3.4. Response Beispiel aus SoapUI

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:empfangenResponse xmlns:ns2="http://v4.transfer.ws.elda.at/">
         <return>
            <serviceResult>
               <messages>Keine Rücksendung mit Protokollnummer 1 vorhanden.</messages>
               <statusCode>406</statusCode>
            </serviceResult>
         </return>
      </ns2:empfangenResponse>
   </soap:Body>
</soap:Envelope>
```