Transfer Webservice v4

- [1. Unterschied zu Transfer Webservice v3](#)
- [2. Endpoint Adressen](#)
- [3. Methoden](#)
 - [3.1. senden](#)
 - [3.2. ruecksendungenAuflisten](#)
 - [3.3. empfangen](#)
- [4. Request-/Result-Objekte](#)
 - [4.1. SendenRequest](#)
 - [4.2. SendenResult](#)
 - [4.3. RuecksendungenAuflistenRequest](#)
 - [4.4. RuecksendungenAuflistenResult](#)
 - [4.5. EmpfangenRequest](#)
 - [4.6. EmpfangenResult](#)
- [5. Objekte](#)
 - [5.1. SecurityParameters](#)
 - [5.2. ServiceResult](#)
 - [5.3. Datei](#)
- [6. Ruecksendung](#)
- [7. Status Codes](#)
- [8. Beispiel Code](#)
 - [8.1. Request allgemein](#)
 - [8.2. Webservice Aufruf](#)
 - [8.3. Response allgemein](#)
 - [8.4. Webservice Methoden](#)
 - [8.4.1. senden](#)
 - [8.4.1.1. Request](#)
 - [8.4.1.2. Response](#)
 - [8.4.2. rücksendungenAuflisten](#)
 - [8.4.2.1. Request](#)
 - [8.4.2.2. Response](#)
 - [8.4.3. empfangen](#)
 - [8.4.3.1. Request](#)
 - [8.4.3.2. Response](#)

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 verarbeitet 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 (Protokollnummer & 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	<ul style="list-style-type: none">• 000: OK• 401: dateiName zu lange (max 255)• 402: dateiName nicht gesetzt• 403: Datei nicht verarbeitet<ul style="list-style-type: none">◦ ist dieser Code gesetzt befindet sich <code>SendenResult.serviceResult.messages</code> ein String mit dem auslösenden Fehlercode (zB: "fehlerCode: E1")• 404: Datei wird noch verarbeitet<ul style="list-style-type: none">◦ dieser Code wird gemeldet, wenn die Verarbeitung der Datei länger als 40 Sekunden in Anspruch nimmt.• 405: Datei ist Duplikat<ul style="list-style-type: none">◦ ist dieser Code gesetzt befindet sich <code>SendenResult.serviceResult.messages</code> ein String mit der Protokollnummer der Originaldatei (zB: "duplikatVon: 123456789")• 500: Interner Fehler <p>Weitere Codes siehe Abschnitt "Status Codes".</p>
dateiId	Long	Eindeutige, fortlaufende Nummer der empfangenen Datei
eldaZeitstempel	Date	Zeitpunkt des Empfangs in ELDA
protokollnummer	Long	ELDA-Protokollnummer der empfangenen Datei

4.3. RuecksendungenAuflistenRequest

Attribut	Typ	Anmerkungen
keine zusätzlichen Parameter notwendig		

4.4. RuecksendungenAuflistenResult

Attribut	Typ	Anmerkungen
ServiceResult.statusCode	String	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 000: OK• 500: Interner Fehler Weitere Codes siehe Abschnitt "Status Codes".
datei	Datei	

5. Objekte

5.1. SecurityParameters

Attribut	Typ	Anmerkungen
nonce	String	Random String, der pro Request eindeutig sein muss
created	Date	Timestamp zum Zeitpunkt des Requests
apiKey	String	Von ELDA vergebene Client Kennung je Softwarehersteller
seriennummer	String	Seriennummer des Kunden, der den Request abschickt
kundenpasswort	String	Kundenpasswort des Kunden (als SHA512 hex lowercase)

5.2. ServiceResult

Attribut	Typ	Anmerkungen
statusCode	String	
messages	List<String>	

5.3. Datei

Attribut	Typ	Anmerkungen
id	Long	= Datei ID
name	String	Dateiname inkl. Endung
inhalt	String	Base64 kodierte Binärdaten.
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
000	OK
500	Interner Verarbeitungsfehler.
551	Request abgelaufen (created älter als 60 Sekunden).
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 Kundenpasswort falsch.
559	Unerlaubter Content Type.

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/inc
lude", "xop:Include");
includeElement.setAttribute("href", "cid:" + attachmentPart.getContentId());
payload.appendChild(includeElement);
soapMessage.saveChanges();
```

8.4.1.2. 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
*/
```

8.4.2. 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. 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
*/
```

8.4.3. 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. 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(attach -> {
    try {
        String fileName = extractFilename(attach.getMimeHeader("Content-
Disposition")[0]);
        Files.write(Paths.get(receiveDirectory + fileName),
attach.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
*/
```