

ELAK-Transaktionen		Konvention
		elak-trans 1.0.0
		Empfehlung
Kurzbeschreibung	<p>Die Definition einer einheitlichen Struktur zum Austausch von elektronischen Akteninformationen ist ein notwendiger Schritt, um Akten auch in elektronischer Form zwischen Verwaltungsstellen austauschen zu können.</p> <p>Eine darauf aufbauende Definition von ELAK-Funktionen ermöglicht die automatisierte Durchführung von ELAK-Transaktionen über Web-Services und erspart die händische Übermittlung und den Import von Akten.</p> <p>Folgende Use-Cases werden abgedeckt:</p> <ul style="list-style-type: none"> - Akten-, Geschäftsfall-, Geschäftstückübermittlung zwischen Aktensystemen - Speicherung von Geschäftsstücken, welche durch Fachinformationssysteme erstellt wurden - Akteninfo abfragen 	
Autor(en):	Franz Grandits Thomas Rössler	Projektteam / Arbeitsgruppe KommArch (Q-KA)
Beiträge von:	Eduard Breitschuh	

Version 1.0.0 : **19.10.2007**

Fristablauf: **9.11.2007**

ELAK Transaktionen

Inhaltsverzeichnis

1	Einleitung	3
1.1	Beziehungen zu anderen Spezifikationen des E-Governments.....	4
2	Grundlegende Definitionen.....	5
2r.1	Eindeutige ELAK-Objekt-ID.....	5
2.2	Struktur und Anwendung des Schemas.....	6
2.3	Umfang der Umsetzung dieser Schnittstelle	8
2.4	Verwendung von XML-E	8
2.5	Gemeinsame Elemente aller Transaktionen	10
2.5.1	Sequenz-Steuerelemente.....	10
2.5.2	Rollback von Sequenzen.....	13
2.5.3	Sperren von Objekten	15
3	ELAK-Funktionen	19
3.1	Übermittlung zwischen ELAK-Systemen (Service sendElak)	19
3.2	Übermittlung FIS-ELAK	23
3.2.1	Eingang (Service <i>sendEdiaktPayload</i>)	24
3.2.2	Ausgang (Services <i>getGZ, ausfertigung, entfertigung</i>) .	31
3.3	Frage Akteninfo ab (Service searchElak).....	39
4	SOAP-Faults und Message Codes.....	42
5	Referenzen	45

1 Einleitung

Die Definition einer einheitlichen Struktur, die sogenannte EDIAKT-Struktur¹, zum Austausch von elektronischen Akteninformationen (ELAK) ist ein notwendiger Schritt, um Akten auch in elektronischer Form zwischen Systemen der Verwaltung austauschen zu können.

Diese Systeme werden in folgende beide Klassen eingeteilt:

- ELAK-System: generalisiertes elektronisches System zur Verarbeitung von aktenrelevanten Informationen. Ein wesentliches Merkmal ist die Strukturierung aller Informationen und die Gestaltung der Schlüsselsysteme nach Kriterien der Aktenführung. Die Regeln dafür sind in der Regel verfahrensabhängig und für eine Organisation oder Organisationseinheit einheitlich gestaltet.
- Fachinformationssystem (FIS): Auf ein bestimmtes Verfahren bzw. eine bestimmte Verfahrensgruppe hin ausgerichtetes System. Die Strukturierung der Informationen und die Gestaltung der Schlüsselsysteme sind abhängig vom jeweiligen Verfahren und kann organisationsunabhängig gestaltet sein (Führerscheinregister, Identitätsregister etc.).
- Ein sogenannter Fachelak ist eine Mischform, der die Eigenschaften beider Systeme besitzt.

Eine auf die EDIAKT-Struktur aufbauende Definition von ELAK-Funktionen ermöglicht die automatisierte Durchführung von ELAK-Transaktionen über Web-Services und erspart die händische Übermittlung und den Import von Akten, Geschäftsfällen und Geschäftsstücken zwischen unterschiedlichen ELAK-Systemen.

Darüber hinaus können so auch organisationsübergreifende Fachinformationssysteme (wie zum Beispiel das Führerscheinregister) mit den verschiedenen, lokalen ELAK-Systemen über produktunabhängige Schnittstellen gekoppelt werden.

Die im Folgenden spezifizierten Funktionen bestimmen einen Rahmen an Aktionen, welche zwischen den angesprochenen Systemen verwendet werden können. Nach erfolgreicher Umsetzung dieser Funktionen ist gedacht, weitere Services zu definieren

Die Spezifikation erfolgt auf Basis synchroner Web-Services. Das bedeutet keineswegs, dass empfohlen wird, generell Web-Services einzusetzen. Es sollte aber auf die im Folgenden beschriebenen Services zurückgegriffen werden, wenn systemübergreifende synchrone Services für die genannten Use Cases verwendet werden.

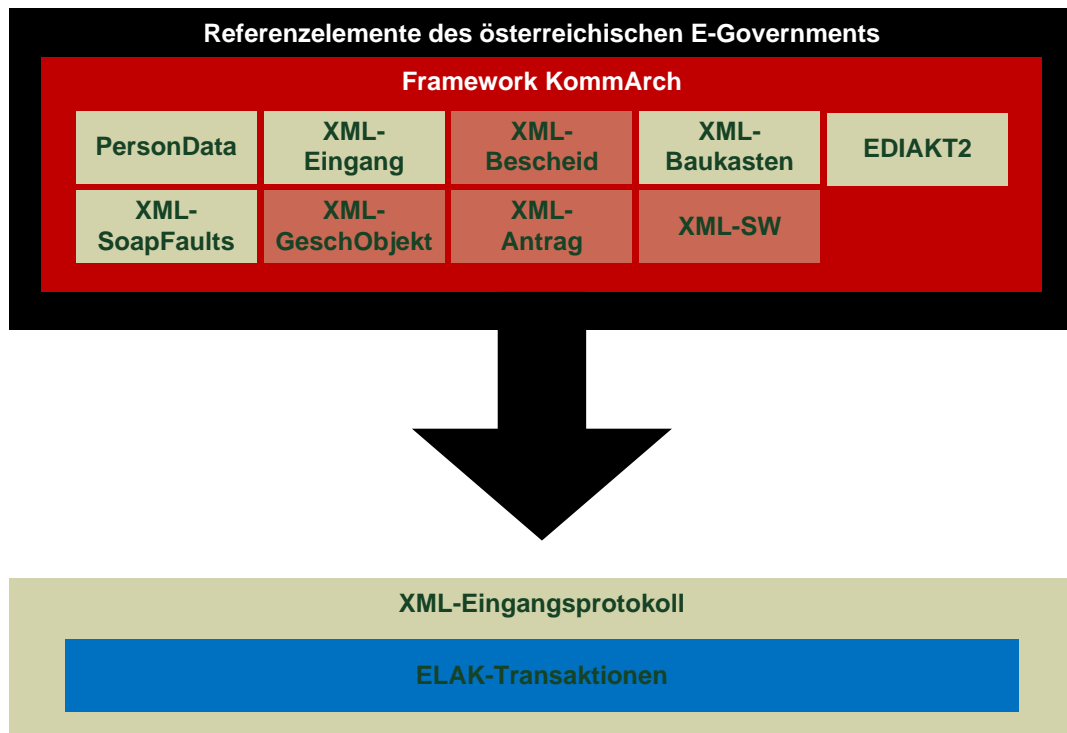
Die Begriffe „Fachanwendung“, „Fachinformationssystem (FIS)“ und ELAK-System stehen synonym für Kommunikationssysteme, die sich dieser Schnittstelle bedienen. Diese Begriffe sollen keinesfalls einschränkend gesehen werden; selbst jene Schnittstellenelemente, die in dieser Spezifikation begrifflich zur

¹ Zum Zeitpunkt der Spezifikation wird die EDIAKT-Struktur der zweiten Generation verwendet, sog. EDIAKTII [ediakt]. Diese Spezifikation referenziert, wo sinnvoll und nötig, EDIAKT-Elemente der EDIAKTII-Spezifikation.

Kommunikation zwischen Fachinformationssystemen und ELAK-Systemen definiert worden sind, z.B. die Funktion sendEdiaktPayload, können selbstverständlich auch zwischen ELAK-Systemen zur Anwendung gebracht werden.

1.1 Beziehungen zu anderen Spezifikationen des E-Governments

Die vorliegende Spezifikation zu ELAK-Transaktionen positioniert sich wie folgt in dem gegebenen Spezifikations-Rahmenwerk des österreichischen E-Governments:



Basis-Spezifikationen, auf die sich die ELAK-Transaktionen stützen bzw. auf die zurückgegriffen wird:

- XML-Baukasten
- XML-Eingangsprotokoll
- Personendaten
- SOAP-Faults

Von einer technischen Betrachtung her bauen die ELAK-Transaktionen auf dem XML-Eingangsprotokoll auf. D.h. einige der darin spezifizierten Transaktionen MÜSSEN in das XML-Eingangsprotokolls eingebettet werden. Manche ELAK-Transaktionen können hingegen, aufgrund deren minimaler Komplexität, auch autonom verwendet werden. Ein hierarchisches Aufbauen auf dem XML-Eingangsprotokoll ist bei ELAK-Transaktionen daher nur bei zwei Funktionen zwingend notwendig; bei den anderen Funktionen hingegen optional möglich.

2 Grundlegende Definitionen

Mit den folgenden Funktionen können Akten, Geschäftsfälle oder Geschäftsstücke übermittelt werden. Der Umfang wird durch das in der Transaktion übermittelte EDIAKT-Paket bestimmt.

Es werden folgende Anwendungsfälle abgebildet:

- Übermittlung zwischen unterschiedlichen ELAK-Systemen: Hier können Geschäftsstücke, Geschäftsfälle und Akten übermittelt werden
- Übermittlung zwischen Fachinformationssystemen und ELAK-Systemen: In diesen Fällen nutzt das FIS den ELAK zur Aktenführung. Ein typischer Fall ist die Kommunikation zentraler FIS mit dezentralen ELAK-Systemen. Die Anwendung ist vorerst auf Geschäftsstücke eingeschränkt, wobei zwischen dem Eingang und dem Ausgang unterschieden wird. Der Ausgang eines Geschäftsstücks läuft in drei Phasen ab:
 - Indizierung (Definition von Metadaten und Vergabe einer Geschäftszahl)
 - Ausfertigung (Ergänzung um die Dokumente)
 - Entfertigung (Versand)

Für die im Folgenden angeführten Übermittlungen werden auch die Strukturen des Eingangsprotokolls [xml-e] verwendet. Für die Fehler werden die Standards der Empfehlung [soap-faults] angewandt.

2.1 Eindeutige ELAK-Objekt-ID

Für eine Verknüpfung zwischen FIS und ELAK bzw. zwischen unterschiedlichen ELAK-Systemen wird die Identifikation (Objekt-ID) des ELAK-Objekts verwendet. Die im Verwaltungsbereich für jedes ELAK-Objekt eindeutige Objekt-ID ist wie folgt aufgebaut:

Objekt-ID:

- ID des Objekts im jeweiligen ELAK
- ID des ELAK-Systems

Die Objekt-ID wird vorzugsweise in Form eines Uniform Resource Identifiers (URI) gebildet; konkret kann man dazu auf Uniform Resource Names (URN) zurückgreifen, bspw. der folgenden Form:

urn:publicid:reference.gv.at:<ELAKSYSTEMID>:<OBJECTIDIMELAK>

Ein konkreter URN für ein ELAK Objekt könnte z.B. wie folgt aussehen:

urn:publicid:reference.gv.at:ELAKBund:O12131234
urn:publicid:reference.gv.at:ELAKLStmk2:a99214321

Voraussetzungen für derartige eindeutige Objekt-Identifizierer:

- 1 die Grundstruktur der URN (d.i. urn:publicid:reference.gv.at) wird registriert und standardisiert. Eine verwaltungsübergreifende Organisation vergibt

zentral eindeutige ELAK-System-IDs (<ELAKSYSTEMID>) an ELAK-System-Betreiber.

- 2 jede ein ELAK-System betreibende Organisation sorgt selbst dafür, dass die im eigenen System (in der eigenen Domäne) vergebenen ELAK-Objekt-IDs (<OBJECTIDIMELAK>) darin eindeutig sind. Die ID des Objektes selbst wird nicht zentral verwaltet.

Da URNs nur einen eindeutigen Bezeichner für Objekte liefern, bedarf es ggf. eines Mechanismus, der ELAK-Objekt-URNs in tatsächlich auflösbare Referenzen (Uniform Resource Location, URL) umformt (zumindest soll über diesen Mechanismus die URL des für das Objekt zuständigen ELAK-Systems, bzw. dessen Web-Service Schnittstelle, eruiert werden können). Ein derartiger Auflösungsmechanismus kann bspw. mittels lokal gehaltener Auflösetabellen realisiert werden. Es ist aber auch ein verwaltungsübergreifenden Auskunftsdienstes (oder mehrere) denkbar, der angelieferte ELAK-Objekt-URNs in auflösbare Referenzen (Uniform Resource Location, URL) umwandelt. Damit erst kann das eigentliche Datenobjekt (Dokument, Datei, etc.) bezogen werden. In einem derartigem Auskunfts-/Verzeichnisdienst können folgende Informationen/Attribute geführt werden:

- ID des ELAK-Systems
- VKZ der zugehörigen Organisation
- Bezeichnung der zugehörigen Stelle
- nähere Bezeichnung des Systems (optional)
- URL des ELAK-Systems, wo unter Anwendung der ID des Objektes dieses auch bezogen werden kann

2.2 Struktur und Anwendung des Schemas

Das durch diese Spezifikation festgelegte XML-Schema referenziert, wo möglich und sinnvoll, bereits bestehende Protokolle und XML-Schemata des E-Governments, wie bspw. XML-E oder das EDIAKT-Schema. Die in dieser Spezifikation definierten Funktionen für Transaktionen zwischen ELAK-Systemen folgen einem einfachen Request-Response Schema und werden in Form von Web-Service Funktionen implementiert.

Für komplexe Funktionen bzw. Funktionsargumente wurde auch die Einbettung der an sich atomaren Requests/Responses in das XML-E Protokoll in Form zusätzlicher Eingangs-/Protokolldaten vorgesehen. Grundsätzlich kann jede atomare Funktion der ELAK-Transaktionen in einem XML-E Container gepackt und so als XML-E Protokoll gehandhabt werden. Der nachfolgende Abschnitt erläutert diese Vorgehensweise. Bei einigen einfachen ELAK-Transaktionen würde die zwingende Verwendung von XML-E Containern jedoch einen Overhead bedeuten und auch keinen Mehrwert bringen. Daher werden die Request-/Responseparameter dieser einfachen Funktionen direkt in den Services verwendet.

Die nachfolgende Tabelle gibt eine Gegenüberstellung von ELAK-Transaktionsfunktionen und Funktionsparametern; es wird auch angegeben, welche der Funktionen nur in Verbindung mit dem XML-E Protokoll verwendet werden sollen und welche zur eigenständigen Verwendung konzipiert wurden. Dennoch kann aber jede Funktion sowohl atomar als auch als Bestandteil des XML-E Protokolls genutzt werden.

Service (Input-/Outputelement)	XML-E	atomar	Anmerkung
sendElak (EingangsDatenSendElak /SendElakResponse)	X	–	Funktionsparameter MUSS in XML-E Protokoll eingebettet verwendet werden.
sendEdiaktPayload (EingangsDatenSend EdiaktPayload /SendEdiaktPayload Response)	X	–	Funktionsparameter MUSS in XML-E Protokoll eingebettet verwendet werden.
getGz (EingangsDatenGetGZ /GetGZResponse)	–	X	Funktionsparameter MUSS eigenständig (atomar) verwendet werden.
ausfertigung (EingangsDatenAusfertigung /AusfertigungsResponse)	–	X	Funktionsparameter MUSS eigenständig (atomar) verwendet werden.
entfertigung (EingangsDatenEntfertigung /EntfertigungResponse)	–	X	Funktionsparameter MUSS eigenständig (atomar) verwendet werden.
searchElak (SearchElak/ SearchElakResponse)	–	X	Funktionsparameter MUSS eigenständig (atomar) verwendet werden. Die Definition der Funktion searchELAK könnte auch auf Basis von XML-SW aufgebaut werden. Die in dieser Spezifikation getroffene Definition ist jedoch eine eigenständige.

In dieser Spezifikation sehen nur die folgenden Services:

- **sendElak** (EingangsDatenSendElak/SendElakResponse)
- **sendEdiaktPayload**
(EingangsDatenSendEdiaktPayload/SendEdiaktPayloadResponse)

die Einbettung in XML-E vor. Alle anderen Funktionsparameter werden atomar verwendet.

2.3 *Umfang der Umsetzung dieser Schnittstelle*

Welche einzelnen Schnittstellenmethoden (Use-Cases) aufgegriffen und tatsächlich umgesetzt werden, ist den einzelnen Systemanbietern selbst überlassen. Dies vor allem da für gewisse Systeme (ELAK, Fachanwendung, etc.) nur eine Auswahl der hier spezifizierten Methoden anwendbar und daher sinnvoll ist.

Diese Spezifikation ist viel mehr als „Baukasten“ standardisierter Schnittstellenmethoden zu sehen. Gelangt eine Schnittstellenmethode zur Realisierung, so ist diese jedoch spezifikationstreu umzusetzen.

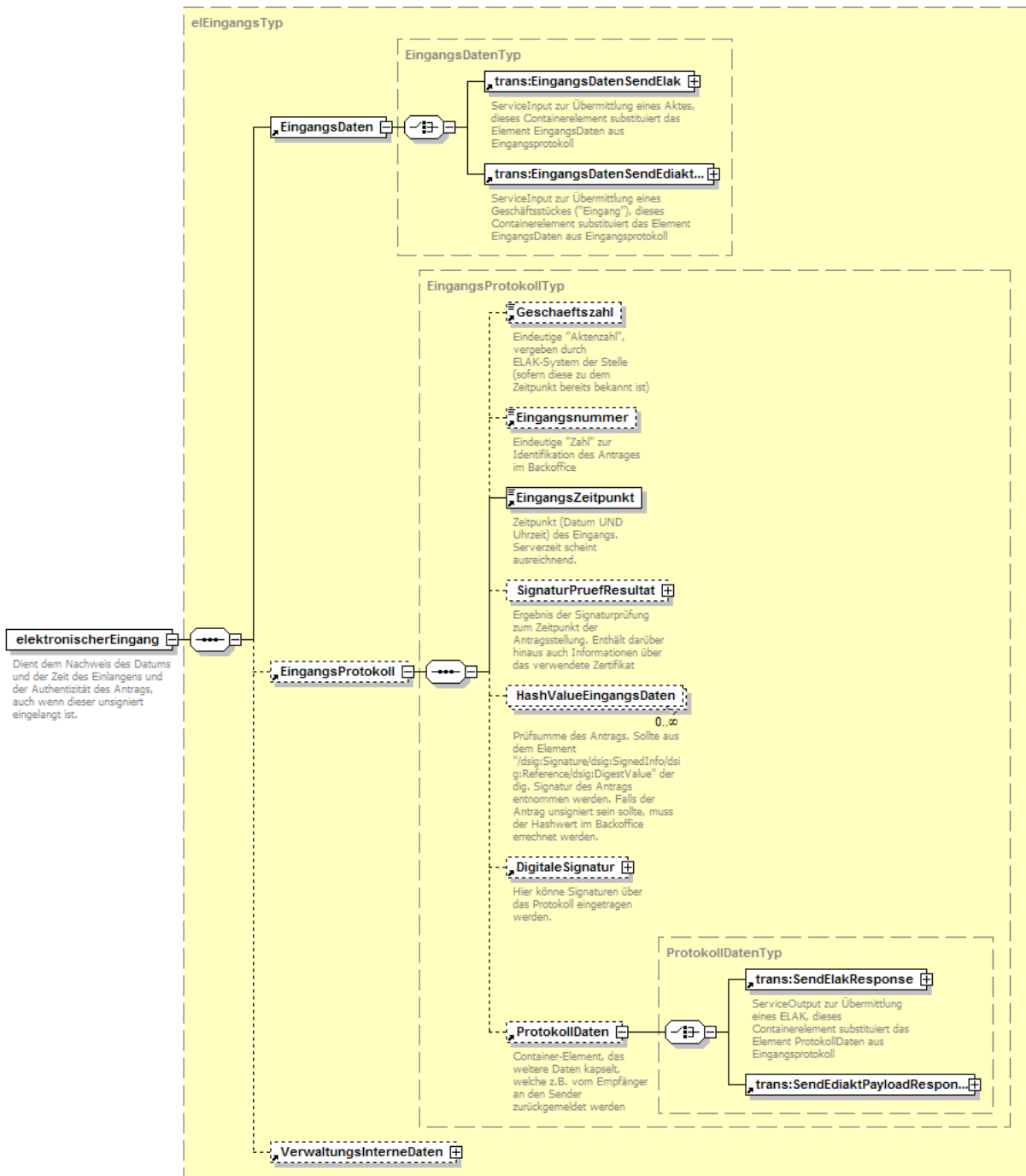
Aus Sicht einer Implementation wird der jeweilige Umsetzungsgrad der Web-Serviceschnittstelle ausreichend über das jeweilige WSDL-File beschrieben, sodass bilaterale Vereinbarungen vor der Nutzung einzelner Schnittstellenbefehle nicht notwendig sind.

2.4 *Verwendung von XML-E*

Einige der im vorliegenden Schema als Web-Services auszugestaltenden Operationen wurden an die XML-Spezifikation für den elektronischen Eingang [xml-e] angelehnt. Dazu werden die spezielleren Elemente der ELAK-Transaktionen als Kindelemente des XML-E Elementes *Eingangsdaten* eingefügt. So wird die generische Verwendung von XML-E als „Transport-Layer“ ermöglicht.

Analog dazu werden spezifische Antwortelemente der ELAK-Transaktionen als konkrete, weitere Kindelemente dem XML-Element *ProtokollDaten* (dies ist ein Unterelement des XML-E Elements *EingangsProtokoll*) angefügt.

Die nachfolgende Abbildung zeigt das um die ELAK-Transaktionselemente konkretisierte XML-E Schema.



2.5 Gemeinsame Elemente aller Transaktionen

In diesem Abschnitt werden Elemente und Möglichkeiten beschrieben, die allen XML-Requests/-Responses der ELAK-Transaktionen gemein sind.

2.5.1 Sequenz-Steurelemente

Durch eine Reihe von Attributen kann im Zuge jedes Requests eine Sequenz von ELAK-Transaktionen eröffnet, fortgesetzt oder beendet werden. So werden semantisch zusammenhängende Transaktionen auch syntaktisch miteinander verknüpft.

Als Motivation, hier exemplarische eine Sequenz von Einzeltransaktionen (Transaktionen jeweils von FIS an ELAK gerichtet), die im engen semantischen Kontext zu einander stehen:

- 1 getGZ()
- 2 sendEdiaktPayload()
- 3 ausfertigung()

Im ersten Schritt wird mit der Transaktion getGZ() eine Geschäftszahl vom ELAK angefordert. Im zweiten Schritt übermittelt das FIS zur gerade bezogenen Geschäftszahl einzelne Dokumente oder ergänzende Geschäftsstücke an den ELAK, unter Bekanntgabe der im ersten Schritt bezogenen Geschäftszahl. Im dritten und letzten Schritt veranlasst das FIS den ELAK das gegenständliche Geschäftsstück auszufertigen, wiederum unter Angabe der Geschäftszahl. Es ist in dieser Abfolge zu erkennen, dass all diese Einzelaktionen semantisch zusammengehören und ein Zwischengriff durch ein anderes FIS, oder gar durch den ELAK selbst, stören würden. Daher ist in diesem Fall nicht nur die Kapselung der Einzelaktionen in eine Transaktionssequenz sinnvoll, sondern wäre auch das Sperren des betreffenden ELAK-Objektes im ELAK ratsam, um zwischenzeitliche Manipulationen zu verhindern.

Die nachfolgend definierten Attribute ermöglichen es, Transaktionen von ELAK-Funktionen zu steuern. Es darf jeweils nur eines der nachgenannten Attribute in einem Request oder einer Response vorkommen.

Öffnen einer Sequenz:

Das Attribut „sequenceOpen“ im XML-Request initiiert dem den Request empfangenden System, dass eine Transaktionssequenz eröffnet werden soll.

Attribut	Datentyp	Beschreibung
sequenceOpen	xs:String	Optionales Attribut in einem Request zur Eröffnung einer Sequenz von Transaktionen. Als Wert dieses Attributes wird jene Dauer (in Sekunden)

		<p>angegeben, innerhalb der das anfragende System weitere Transaktionen der Sequenz garantieren kann. Dieser Wert dient vorallem dazu, um dem Empfangssystem Hinweise bzgl. des zeitlichen Offenhaltens der Transaktionssequenz zu geben. In der Regel wird hier jedes System eine bestimmte, systemabhängige maximale Dauer zulassen. Nach dieser kann die Transaktionssequenz vom empfangenden System aus Zeitgründen beendet werden (Timeout).</p> <p>Ist die im sequenceOpen-Attribut angeführte Dauer kleiner oder gleich jener die das empfangende System akzeptiert, so wird die Transaktionssequenz eröffnet. In diesem Fall wird der Request abgearbeitet und die Response enthält ein sequenceID-Attribut, das die vom empfangenden System generierte Sequence-ID enthält.</p> <p>Andernfalls muss das System den gesamten Request mit einem Fehler 4110 zurückweisen (siehe Abschnitt 4).</p> <p>Als minimaler Wert einer Sequenzdauer, die ein empfangendes System zumindest akzeptieren soll, wird 120 Sekunden empfohlen.</p>
--	--	---

Fortführen einer Sequenz:

Kann eine Transaktionssequenz eröffnet werden, so fügt das empfangende System bei allen Responses der Sequenz die zugehörige Sequence-ID an (diese wird vom empfangenden System nach Eröffnung der Sequenz vergeben). Auch das anfragende System identifiziert in weiterer Folge alle zu einer Sequenz gehörenden Transaktionen anhand dieser Sequenz-ID mittels sequenceID-Attribut.

Attribut	Datentyp	Beschreibung
sequenceID	xs:String	Optionales Attribut bei Requests und Responses, um die

		<p>Zugehörigkeit zu einer bestimmten Sequenz auszudrücken.</p> <p>Das sequenceID-Attribut enthält die vom empfangenden System initial, im Zuge der Sequenzeröffnung, generierte Sequence-ID.</p> <p>Ist die angegebene Sequence-ID unbekannt oder wurde die Sequenz bereits infolge Timeout beendet, so wird die Anfrage mit einem Fehler 4100 (wenn sequenceID im Request enthalten war) bzw. 5100 (wenn sequenceID in Response enthalten war) zurückgewiesen (siehe Abschnitt 4).</p>
--	--	---

Beenden einer Sequenz:

Im Zuge eines Requests kann eine vormals eröffnete Sequenz auch wieder beendet werden. Dazu wird dem Request das Attribut sequenceClose beigefügt.

Attribut	Datentyp	Beschreibung
sequenceClose	xs:String	<p>Optionales Attribut bei Requests, um die Zugehörigkeit zu einer bestimmten Sequenz auszudrücken und um diese Sequenz auch nach Abarbeitung des gegenständlichen Requests zu beenden.</p> <p>Das sequenceClose-Attribut enthält die vom empfangenden System initial, im Zuge der Sequenzeröffnung, generierte Sequence-ID.</p> <p>Das empfangende System beendet die Sequenz nach Abarbeitung des zugehörigen Requests. In der zugehörigen Response wird die Sequenz-ID noch einmal innerhalb des sequenceID-Attributes mitgegeben.</p> <p>Ist die angegebene Sequence-ID</p>

		unbekannt oder wurde die Sequenz bereits infolge Timeout beendet, so wird die Anfrage vom empfangenden System mit einem Fehler 4100 zurückgewiesen (siehe Abschnitt 4).
--	--	---

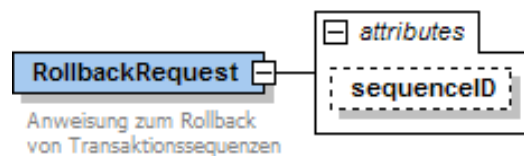
2.5.2 Rollback von Sequenzen

In Verbindung mit Sequenzen erscheint die Möglichkeit eines Rollback von Transaktionssequenzen – beispielsweise im Falle eines Fehlers – sinnvoll. Zu diesem Zweck wird ein einfacher Transaktionsbefehl eingeführt.

Anhand dieser Rollback-Anweisung wird das empfangende System veranlasst, alle innerhalb der in dieser Anweisung identifizierten Transaktionssequenz veranlassten Transaktionen rückgängig zu machen.

2.5.2.1 Rollback-Request:

Dieser Request veranlasst das empfangende System, alle innerhalb der durch diesen Request identifizierten Transaktionssequenz veranlassten Transaktionen rückgängig zu machen. Er enthält nur die betreffende Sequenz-ID.

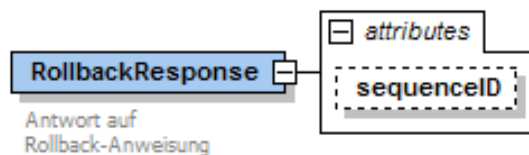


Attribut	Datentyp	Beschreibung
sequenceID	xs:String	<p>Attribut enthält die ID jener Sequenz, deren Transaktionen rückgängig gemacht werden sollen.</p> <p>Das empfangende System muss daraufhin entweder alle Transaktionen einer Sequenz rückgängig machen, oder mit einem entsprechenden SOAP-Fault antworten.</p> <p>Das sequenceID-Attribut enthält die vom empfangenden System initial, im Zuge der Sequenz-eröffnung, generierte Sequence-ID.</p>

		<p>Ist die angegebene Sequence-ID unbekannt oder wurde die Sequenz bereits infolge Timeout beendet, so wird die Anfrage mit einem Fehler 4100 zurückgewiesen (siehe Abschnitt 4).</p> <p>Ist ein Rollback nicht möglich – bedingt durch das System oder einen Fehler – so ist mit Fehlercode 5110 oder 5120 zu antworten (siehe Abschnitt 4).</p>
--	--	---

2.5.2.2 Rollback-Response:

Die Erfolgsantwort auf eine Rollback-Anweisung ist ein leeres Element, das nur die betreffende Sequenz-ID als Attribut enthält. Im Fehlerfall werden alternativ entsprechende SOAP-Fehlercodes retourniert.



Attribut	Datentyp	Beschreibung
sequenceID	xs:String	<p>Attribut enthält die ID jener Sequenz, deren Transaktionen vollständig rückgängig gemacht werden konnten. Nur wenn alle Einzeltransaktionen der Sequenz erfolgreich rückgängig gemacht wurden, darf diese Erfolgsantwort retourniert werden.</p> <p>Die Transaktionssequenz selbst bleibt auch nach dem Rollback bestehen und „geöffnet“, und kann vom Anfragenden ggf. weiter verwendet werden.</p> <p>Das sequenceID-Attribut enthält die vom empfangenden System initial, im Zuge der Sequenzeröffnung, generierte Sequence-ID.</p> <p>Ist die angegebene Sequence-ID unbekannt oder wurde die Sequenz bereits infolge Timeout</p>

		<p>beendet, so wird die Anfrage mit einem Fehler 5100 zurückgewiesen (siehe Abschnitt 4).</p> <p>Ist ein Rollback jedoch nicht möglich – bedingt durch das System oder einen Fehler – so ist anstelle dieser Erfolgsantwort mit Fehlercode 5110 oder 5120 zu antworten.</p>
--	--	---

2.5.3 Sperren von Objekten

Durch ein optionales lockObject-Attribut kann das gegenständliche Objekt eines jeden Requests beim empfangenden System gesperrt werden. Das Sperren von Objekten ist so im Zuge jeder Transaktion möglich. Das Entsperren erfolgt analog anhand desselben Attributes.

Durch das Sperren von Objekten wird vor allem das zeitgleiche Bearbeiten von Objekten verhindert. Nach dem Sperren von Objekten steht es nur mehr dem System zur Verfügung, das dem Objekt die Sperre auferlegt hat. Um dieses System wiedererkennen zu können, ist das Sperren nur innerhalb einer Transaktionssequenz zulässig (siehe Abschnitt 2.5.1), andernfalls wird Fehlercode 4130 retourniert.

Wird ein gesperrtes Objekt von einer anderen Transaktion angesprochen, so wird mit dem Fehlercode 4140 geantwortet und angezeigt, dass das Objekt derzeit gesperrt ist.

Mechanismen und Timeouts zur Aufhebung „vergessener“ Sperren erscheinen sinnvoll und notwendig. Hierzu wird jedoch im Rahmen dieser Spezifikation keine konkrete Ausprägung empfohlen. Eine Verkopplung mit dem Timeout-Verhalten von Sequenzen erscheint jedoch sinnvoll.

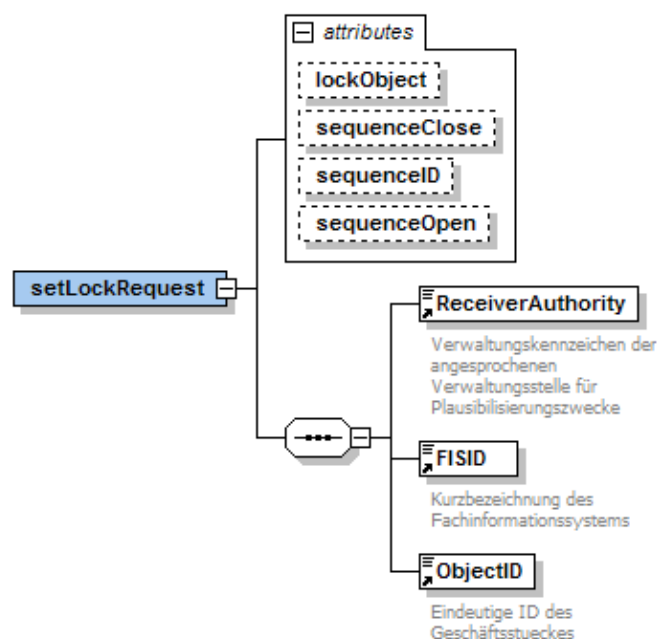
Attribut	Datentyp	Beschreibung
lockObject	xs:boolean	<p>Optionales Attribut bei Requests, um das gegenständliche Objekt des Requests im empfangende System sperren bzw. entsperren zu können.</p> <p>Wert <code>true</code> veranlasst die Sperre des Objektes.</p> <p>Wert <code>false</code> veranlasst das Entsperren des Objektes.</p> <p>Ist das Sperren oder Entsperren nicht möglich, so ist mit Fehler</p>

		4120 bzw. 4130 zu antworten (siehe Abschnitt 4).
--	--	---

2.5.3.1 setLock-Request/Response:

Zusätzlich zur Möglichkeit des Sperrens/Entsperren von Objekten im Zuge von ELAK-Transaktionen erscheint ein einfacher, weiterer Request sinnvoll, der den Sperrstatus eines Objektes setzen oder aufheben kann. Dieser ist vor allem für das Entsperren notwendig.

Anfrage setLockRequest:



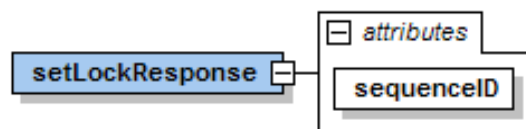
Attribut	Datentyp	Beschreibung
lockObject	xs:boolean	<p>Erforderliches Attribut, um das gegenständliche Objekt des Requests im empfangende System sperren bzw. entsperren zu können.</p> <p>Wert <code>true</code> veranlasst die Sperre des Objektes.</p> <p>Wert <code>false</code> veranlasst das entsperren des Objektes.</p> <p>Ist das Sperrern oder Entsperren nicht möglich, so ist mit Fehler</p>

		4120 bzw. 4130 zu antworten (siehe Abschnitt 4).
sequenceOpen oder sequenceID ... oder sequenceClose	xs:String	Attribut zur Öffnung, Fortführung oder Beendigung einer Transaktionssequenz. Dies ist hier erforderlich, da eine Sperre nur in Zusammenhang mit einer Transaktionssequenz möglich ist.

Element	Datentyp	Beschreibung
ReceiverAuthority	VKZType (string)	Verwaltungskennzeichen der angesprochenen Verwaltungsstelle für Plausibilisierungszwecke
FISID	FISIDType (string)	Kurzbezeichnung des Fachinformationssystems
ObjectID	ObjectIDType	Eindeutige ID des Geschäftstückes, das gesperrt/entsperrt werden soll. Siehe auch 2.1.

Antwort setLockResponse:

Die Antwort ist ein leeres Antwort-Element und signalisiert den Erfolg der Sperre bzw. der Aufhebung einer Sperre. Im Fehlerfall wird ein SOAP-Fault gemäß Abschnitt 4 retourniert.



Attribut	Datentyp	Beschreibung
sequenceID	xs:String	Optionales Attribut bei Requests und Responses, um die Zugehörigkeit zu einer bestimmten Sequenz auszudrücken.

		<p>Das sequenceID-Attribut enthält die vom empfangenden System initial, im Zuge der Sequenzeröffnung, generierte Sequence-ID.</p> <p>Ist die angegebene Sequence-ID unbekannt oder wurde die Sequenz bereits infolge Timeout beendet, so wird die Anfrage mit einem Fehler 4100 (wenn sequenceID im Request enthalten war) bzw. 5100 (wenn sequenceID in Response enthalten war) zurückgewiesen (siehe Abschnitt 4).</p>
--	--	--

3 ELAK-Funktionen

In diesem Abschnitt werden der Reihe nach, die als Web-Service zur Verfügung zu stellenden ELAK-Funktionen definiert. Dabei wird bei jeder Funktion der notwendige Request- und Response-Parameter beschrieben, welche – wie in dem vorhergegangenen Abschnitt erläutert – sowohl atomar als auch im Verbund mit dem XML-E Protokoll genutzt werden können.

Alle neuen Elemente der ELAK-Transaktionen werden im Namespace

```
http://reference.e-government.gv.at/namespace/elaktrans/1#
```

definiert.

3.1 Übermittlung zwischen ELAK-Systemen (Service sendElak)

Mit diesem Service können Geschäftsstücke, Geschäftsfälle und Akten zwischen unterschiedlichen ELAK-Systemen ausgetauscht werden. Übermittelt wird die Struktur des sendenden Systems.

Die Übermittlung kann folgende **Typen (Purpose)** aufweisen:

- Abtretung (AB)
- Archiv (AR)
- Gericht (GER)
- Einsicht (ES)
- sonstige Übermittlung (SO)
(nähere Angabe zum Typ sind in diesem Fall innerhalb der EDIAKT-Struktur zu finden)

Der Übermittlungstyp wird hier und auch in den nachfolgenden Definitionen durch das Sub-Element *Purpose* repräsentiert. Um das Schema nicht im Falle neu aufkommender Übermittlungstypen ändern zu müssen, wurde auf die Definition taxativer Wertevorgaben verzichtet. Bei jeder Funktions-/Parameterdefinition wurde daher textuell festgelegt, welche Werte der Übermittlungstyp im vorliegenden Fall annehmen darf.

Input:

- Typ der Übermittlung
- VKZ des Senders
- Leistungskürzel des Verfahrens
- Objekt-ID des übermittelten Objekt beim Sender (optional)
- Objekt-ID des Bezugs-Objekts beim Empfänger, falls es sich um eine Antwort auf eine Übermittlung handelt (optional)
- EDIAKT-Struktur

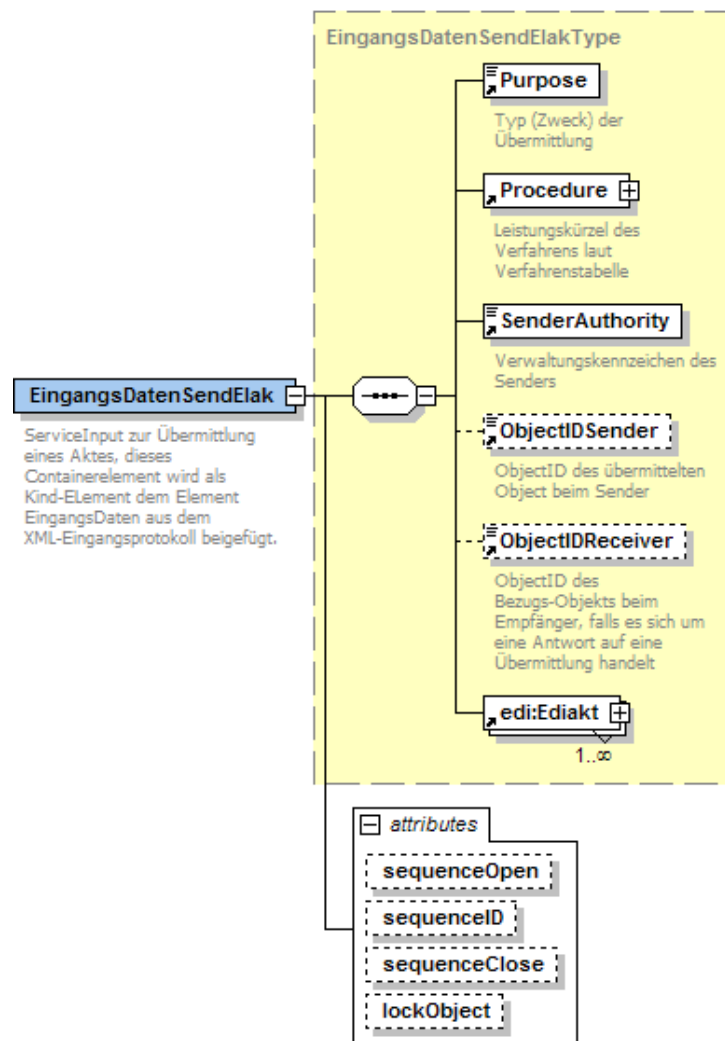
Output:

- Status der Aktion
- durchgeführte Aktion

- Objekt-ID des angelegten Objekts

Spezifikation der Inputelemente: *EingangsDatenSendElak*

Als SOAP-Input dient das Element Eingangsdaten des [xml-e], welches durch das Element EingangsDatenSendElak substituiert wird. Das in der ursprünglichen Struktur vorhandene ##other Element wird nun durch EingangsDatenSendElak ersetzt, welches wiederum das Wurzelement für die Aufnahme der eigentlichen Inputdaten darstellt.



Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).

Element	Datentyp [Kardinalität]	Beschreibung
Purpose	PurposeType [1]	Typ der Übermittlung; hierin ist an dieser Stelle zulässig: AB (Abtretung) AR (Archiv) GER (Gericht) ES (Einsicht) SO (Sonstiges)
Procedure	baukasten: VerfahrenTyp (besteht aus Verfahrenskürzel und Langtext) [1]	Leistungskürzel des Verfahrens laut Verfahrenstabelle
SenderAuthority	VKZType (string) [1]	Verwaltungskennzeichen der angesprochenen Verwaltungsstelle für Plausibilisierungszwecke
ObjectIDSender	ObjectIDType (string) [0..1]	ID des übermittelten Objekt beim Sender (optional). Siehe auch 2.1.
ObjectIDReceiver	ObjectIDType (string) [0..1]	ID des Bezugsobjekts beim Empfänger, falls es sich um eine Antwort auf eine Übermittlung handelt (optional). Siehe auch 2.1.
Ediakt	edi:EdiaktType [1..∞]	EDIAKT-Container der zu übermittelnden Aktenstruktur (Akt, Geschäftsfall, Geschäftsstück). Die Geschäftszahl (MetaData/Identifi

		er) der jeweiligen Payload (Layer1/2/3-Objekt) KANN vom erforderlichen MetaData-Element des einhüllenden EDIAKT-Containers übernommen werden.
--	--	---

Spezifikation der Outputelemente: *SendElakResponse*

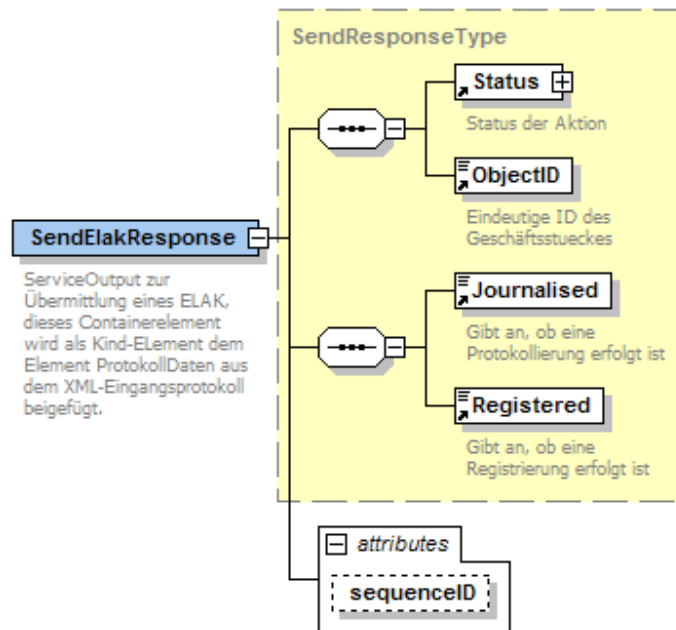
Als SOAP-Output dient das Element Eingangsprotokoll des [xml-e], worin das Sub-Element Protokoll Daten durch das Element SendElakResponse ersetzt wird, welches jene Daten rückmeldet, die in der Struktur Eingangsprotokoll nicht bereits vorgesehen sind.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).

Folgende Daten werden im Element Eingangsprotokoll verwendet:

Element	Datentyp [Kardinalität]	Beschreibung
ReferenceNumber	String (optional) [0..1]	Geschäftszahl des Geschäftsstücks (wenn eine Protokollierung durchgeführt wurde, sonst nicht vorhanden)
Eingangszeitpunkt	DateTime [1]	Zeitpunkt der Übermittlung und der Übernahme

Alle anderen (optionalen) Elemente (insbesondere zur Signatur, welche zwischen öffentlichen Systemen nicht notwendig ist und zu einem Overhead führen würde) werden nicht verwendet. Ergänzt wird die Rückmeldung durch die Daten in *SendElakResponse* :



Element	Datentyp [Kardinalität]	Beschreibung
Status	StatusType (Number + Message) [1]	Status der Aktion; repräsentiert durch einen numerischen Statuscode und einer optionalen Text-Message.
ObjectID	ObjectIDType [1]	Eindeutige ID des Geschäfts- stückes. Siehe auch 2.1.
Journalised	boolean [1]	Gibt an, ob eine Protokollierung erfolgt ist (dann TRUE)
Registered	boolean [1]	Gibt an, ob eine Registrierung erfolgt ist (dann TRUE)

3.2 Übermittlung FIS-ELAK

Die Protokollierung erfolgt nach lokal festlegbaren Strukturen. Auf Grund des Inhalts des Geschäftsstücks kann entschieden werden, ob dafür ein neuer Akt angelegt oder ein Geschäftsstück zu einem bestehenden Akt/Geschäftsfall hinzugefügt wird.

Der **Sender** wird durch einen der folgenden Möglichkeiten klassifiziert:

- Bürger/in (B)

- Unternehmen (U)
- Öffentliche Stelle (O)

Das Attribut "Erstellt von" wird aus dem PVP-Header rekonstruiert.

Übermittlungstyp (Purpose):

- Eingang (Erledigung, Stellungnahme, Information)
- Zustellung (ZU)
- Indizierung (IND)
- Ausfertigung (AUS)
- Entfertigungsstatus setzen (ENT)

Durchgeführte Aktion:

- Registrierung (J/N)
- Protokollierung (J/N)

Folgende Use-Cases werden derzeit abgebildet:

- Eingangsfunktionen:
 - *sendEdiaktPayload*
- Ausgangsfunktionen:
 - *getGZ*
 - *ausfertigung*
 - *entfertigung*

3.2.1 Eingang (Service *sendEdiaktPayload*)

Werden Formulardaten von Partner oder einem Formularserver mit Hilfe des Eingangsprotokolls [xml-e] übermittelt, kann ein Eingangssystem die Umformung übernehmen.

Input:

- Leistungskürzel des Verfahrens
- VKZ der angesprochenen Verwaltungsstelle (zur Plausibilisierung der angesprochenen URL)
- Typ der Übermittlung (=Eingang: Erledigung, Stellungnahme, Information)
- Sender
 - ist ein VKZ des Senders verfügbar, so wird dieses im Request mitgegeben (optional)
 - ist Sender ein Bürger, so wird dessen Identität innerhalb einer Personendaten-Struktur gehalten.
- Typ

- Kurzbezeichnung des Fachinformationssystems
- Betreff
- Übermittelte Dokumente

Output:

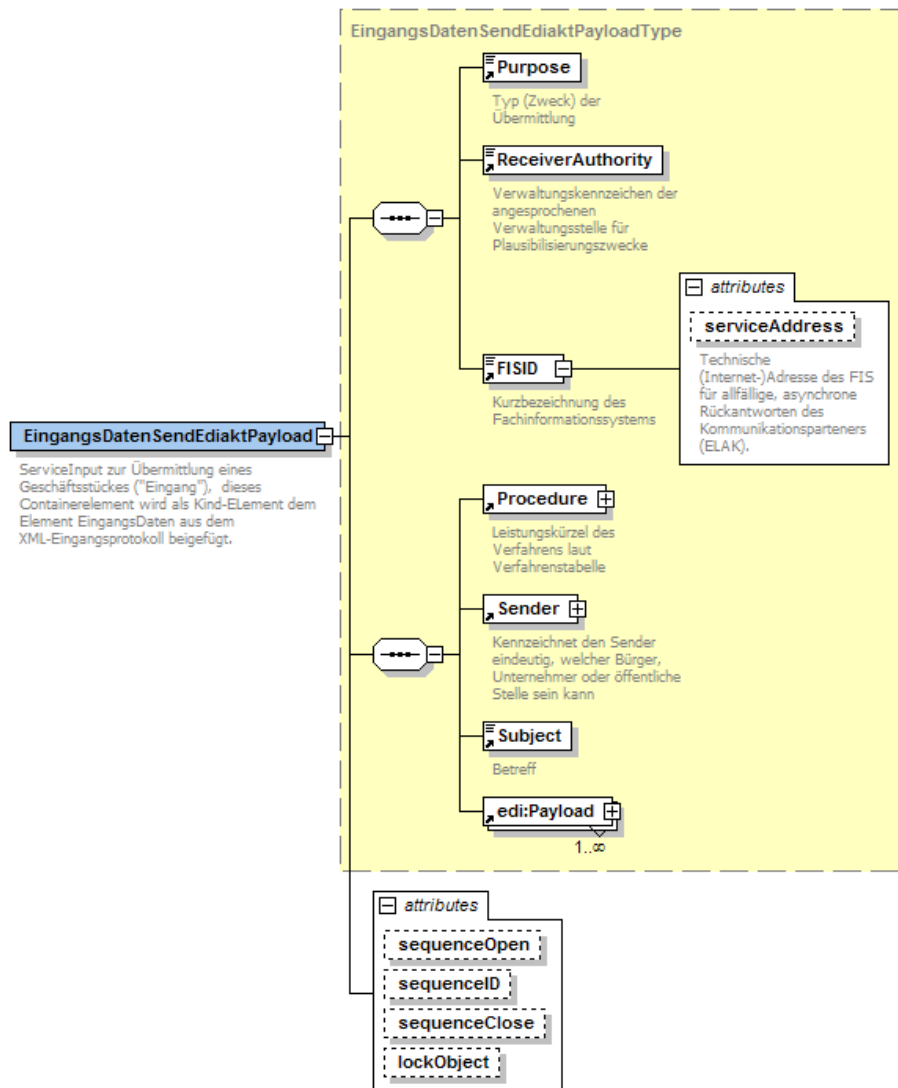
- Status der Aktion
- Durchgeführte Aktion
- GZ des Geschäftsstücks (wenn eine Protokollierung durchgeführt wurde)
- Objekt-ID des Geschäftsstücks

Spezifikation der Inputelemente: *EingangsDatenSendEdiaktPayload*

Als SOAP-Input dient das Element *Eingangsdaten* des [xml-e], welches durch das Element *EingangsDatenSendEdiaktPayload* substituiert wird. Das in der ursprünglichen Struktur vorhandene *any ##other* Element wird nun durch *EingangsDatenSendEdiaktPayload* ersetzt, welches wiederum das Wurzelement für die Aufnahme der eigentlichen Inputdaten darstellt.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).

Hauptelement: *EingangsDatenSendEdiaktPayload*

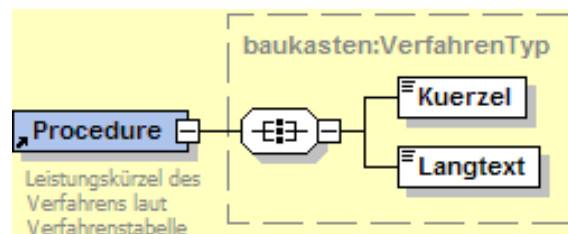


Element	Datentyp [Kardinalität]	Beschreibung
Purpose	PurposeType [1]	Typ der Übermittlung; hier sind an dieser Stelle zulässig: ER (Zur Erledigung), ST (Zur Stellungnahme), INF (Zur Information)
ReceiverAuthority	VKZType (string) [1]	Verwaltungskennzeichen der angesprochenen Verwaltungsstelle für Plausibilisierungszwecke

FISID	FISIDType (string) [1]	Kurzbezeichnung des Fachinformationssystems
FISID @serviceAddress	xs:AnyURI [0..1]	<p>Optionales Attribut zu FISID-Element. Hiermit kann optional ein technische Service-Adresse des FIS mitgegeben werden, über die das FIS allfällige, asynchrone Antworten des Kommunikationspartners (ELAK) erwartet.</p> <p>Hinter dieser Service-Adresse wird in den meisten Fällen ein Web-Service betrieben werden, dem der ELAK bspw. asynchron – dh. zu einem späteren Zeitpunkt – weitere Statusinformationen zu dem gegenständlichen Objekt rückmelden kann (zB. den Abschluss einer Verarbeitung).</p> <p>Zur Identifikation des gegenständlichen Objektes muss im Zuge der Rückmeldung eine geeignete Identifizierung des Objektes stattfinden. Dazu ist die ObjectID heranzuziehen (aus Response zu entnehmen).</p>
Procedure	baukasten: VerfahrenTyp (besteht aus Verfahrenskürzel und Langtext) [1]	Leistungskürzel des Verfahrens laut Verfahrenstabelle
Sender	SenderType (besteht aus Typ des Senders und VKZ des Senders, wenn öffentliche Stelle) Im Falle anderer Personen (natürlich oder nicht-natürlich) kann der Sender als Personendaten-Element eingefügt werden.	Kennzeichnet den Sender eindeutig, welcher Bürger, Unternehmer oder öffentliche Stelle sein kann.

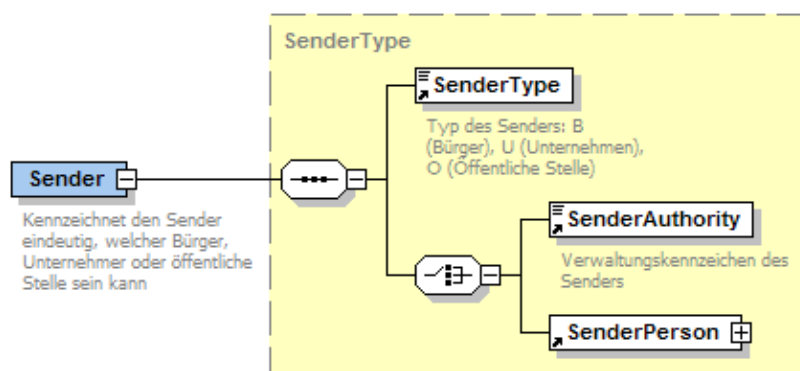
	[1]	
Subject	SubjectType (string) [1]	Betreff
edi:Payload	edi:PayloadType [1..∞]	EDIAKT-Payload; mehrere Payload-Elemente möglich. Soweit befüllt wie für das Verfahren notwendig. GZ jedenfalls noch nicht vergeben.

Sub-Element: Procedure



Element	Datentyp [Kardinalität]	Beschreibung
Kuerzel	string [1]	Verfahrenskürzel
Langtext	string [1]	Verfahrenslangtext

Sub-Element: Sender



Element	Datentyp [Kardinalität]	Beschreibung
SenderType	token (enum: B, U, O) [1]	Typ des Senders: B (Bürger), U (Unternehmen), O (Öffentliche Stelle)
SenderAuthority	VKZType (optional, string) [0..1]	Verwaltungskennzeichen des Senders - kann entfallen, wenn keine öffentliche Stelle
SenderPerson	p:PersonDataType [0..1]	Ist Sender keine Behörde, so werden hier seine Personendaten gefasst (sowohl für natürliche als auch nicht-natürliche Personen).

Spezifikation der Outputelemente:

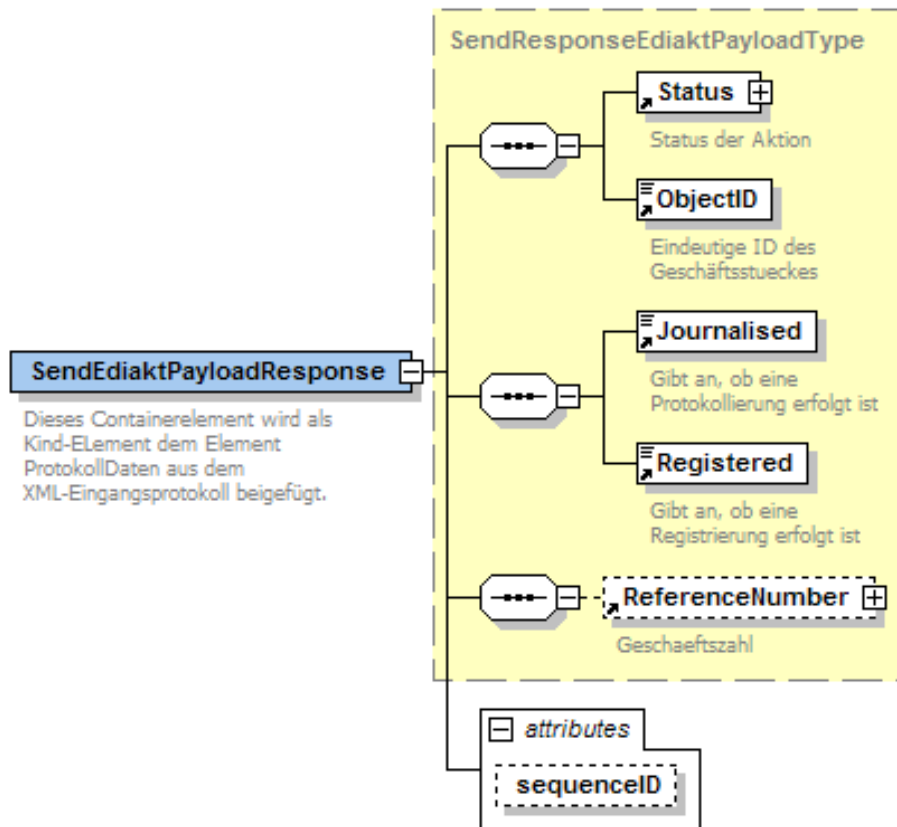
Als SOAP-Output dient das Element *Eingangsprotokoll* des [xml-e], worin das Sub-Element *Protokolldaten* durch das Element *SendEdiaktPayloadResponse* ersetzt wird, welches jene Daten rückmeldet, die in der Struktur *Eingangsprotokoll* nicht bereits vorgesehen sind.

Folgende Daten werden im Element *Eingangsprotokoll* verwendet:

Element	Datentyp	Beschreibung
ReferenceNumber	String (optional)	Geschäftszahl des Geschäftsstücks (wenn eine Protokollierung durchgeführt wurde, sonst nicht vorhanden)
Eingangszeitpunkt	DateTime	Zeitpunkt der Übermittlung und der Übernahme

Alle anderen (optionalen) Elemente (insbesondere zur Signatur, welche zwischen öffentlichen Systemen nicht notwendig ist und zu einem Overhead führen würde) werden nicht verwendet. Ergänzt wird die Rückmeldung durch die Daten in *SendEdiaktPayloadResponse*.

Es sind in der *SendEdiaktPayloadResponse* auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).



Element	Datentyp [Kardinalität]	Beschreibung
Status	StatusType (Number + Message) [1]	Status der Aktion; repräsentiert durch einen numerischen Statuscode und einer optionalen Text- Message.
ObjectID	ObjectIDType [1]	Eindeutige ID des Geschäftsstückes. Siehe auch 2.1.
Journalised	boolean [1]	Gibt an, ob eine Protokollierung erfolgt ist (dann TRUE)
Registered	boolean [1]	Gibt an, ob eine Registrierung erfolgt ist (dann TRUE)
ReferenceNumber	edi:IdentificationType (optional) [0..1]	Geschäftszahl des Geschäftsstücks (wenn eine Protokollierung durchgeführt

		wurde, sonst nicht vorhanden) Dieses Element ist zudem nicht erforderlich, wenn die Response in Verbindung mit einer XML-E Response verwendet wird, da bereits dort die Geschäftszahl als Element vorgesehen ist.
--	--	--

3.2.2 Ausgang

(Services getGZ, ausfertigung, entfertigung)

Mit den folgenden Schnittstellen kann die Entfertigung eines durch ein FIS erstelltes Geschäftsstück vorgenommen werden (z.B. Bescheid).

Folgende Abfolge ist vorgesehen bzw. wäre denkbar:

- Indizierung (GZ-Vergabe)
- Erstellung der Dokumente durch das Fachinformationssystem
- Ausfertigung (Ergänzung um die Dokumente): In diesem Fall kann der ELAK auch die Zustellung übernehmen
- FIS Zustellung/Druck: Dieser Schritt wird durchgeführt, falls der ELAK die Zustellung nicht übernimmt
- Entfertigungsstatus setzen, wenn Zustellung/Druck durch das FIS erfolgreich war

3.2.2.1 Indizierung

(Service *getGz*):

Im Falle der Indizierung werden nur die Metadaten eines Geschäftsstücks ohne Geschäftszahl übergeben. Für den Fall, dass bereits ein (Vor-)Akt besteht und die Fachanwendung die GZ des (Vor-)Aktes/Geschäftsfalls kennt, kann anstelle des Leistungskürzels auch die GZ übergeben werden.

Input:

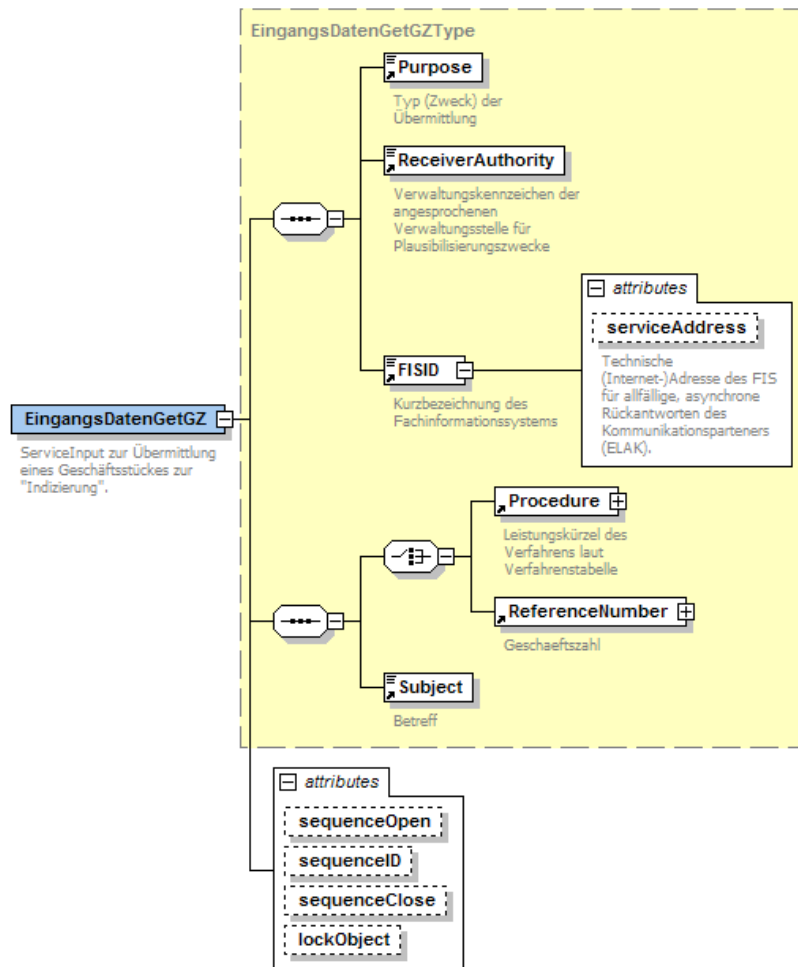
- Leistungskürzel des Verfahrens oder alternativ die GZ des (Vor-)Aktes/Geschäftsfall, dem das neue Geschäftsstück zugeordnet werden soll
- VKZ der angesprochenen Verwaltungsstelle (zur Plausibilisierung der angesprochenen URL)
- Übermittlungstyp (=Indizierung)
- Kurzbezeichnung des Fachinformationssystems
- Betreff

Output:

- Status der Aktion
- GZ des Geschäftsstücks
- Objekt-ID des Geschäftsstücks

Spezifikation der Inputelemente: *EingangsDatenGetGZ*

Als SOAP-Input dient direkt das Element *EingangsDatenGetGZ*; es wird für diese Funktion atomar verwendet.



Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).

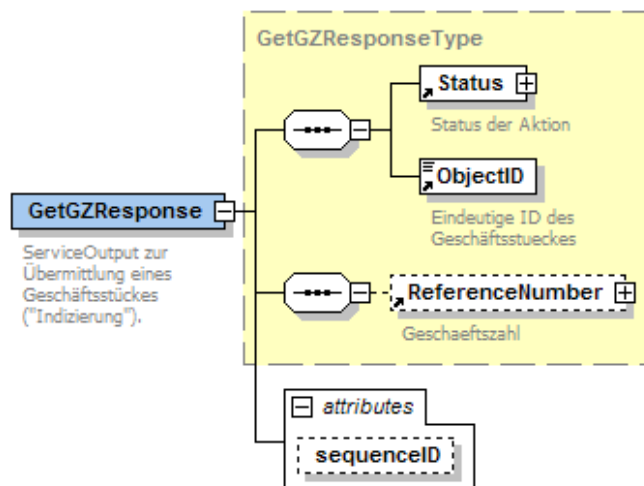
Element	Datentyp [Kardinalität]	Beschreibung
Purpose	PurposeType [1]	Zweck der Übermittlung; an dieser Stelle zulässige Werte: IND (Indizierung)
ReceiverAuthority	VKZType (string) [1]	Verwaltungskennzeichen der angesprochenen Verwaltungsstelle für Plausibilisierungszwecke
FISID	FISIDType (string)	Kurzbezeichnung des Fachinformationssystems

	[1]	
FISID @serviceAddress	xs:AnyURI [0..1]	Optionales Attribut zu FISID-Element. Definition siehe Beschreibung aus Abschnitt 3.2.1.
Procedure	baukasten:VerfahrenTyp (besteht aus Verfahrenskürzel und Langtext, alternativ zu GZ) [1]	Leistungskürzel des Verfahrens laut Verfahrenstabelle
ReferenceNumber	ed:IdentificationType [1]	Geschäftszahl des (Vor-)Aktes, wenn ein solcher besteht.
Subject	SubjectType (string) [1]	Betreff

Spezifikation der Outputelemente: *GetGZResponse*

Das Element *GetGZResponse* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).



Element	Datentyp [Kardinalität]	Beschreibung
Status	StatusType (Number + Message) [1]	Status der Aktion; repräsentiert durch einen numerischen Statuscode und einer optionalen Text-Message.
ObjectID	ObjectIDType [1]	Eindeutige ID des Geschäftsstückes. Siehe auch 2.1.
ReferenceNumber	edi:IdentificationType (optional) [0..1]	Geschäftszahl des Geschäftsstücks (wenn eine Protokollierung durchgeführt wurde, sonst nicht vorhanden) Dieses Element ist zudem nicht erforderlich, wenn die Response in Verbindung mit einer XML-E Response verwendet wird, da bereits dort die Geschäftszahl als Element vorgesehen ist.

3.2.2.2 Ausfertigung (Service *ausfertigung*):

Hier wird ein bereits durch die Indizierung angelegtes Objekt um die Dokumente ergänzt. Bei der Übermittlung können auch die Metadaten mit Ausnahme der GZ korrigiert werden. Weiters kann dem ELAK auch die Zustellung nach Ergänzung aufgetragen werden.

Input:

- Objekt-ID des Geschäftsstücks
- VKZ der angesprochenen Verwaltungsstelle (zur Plausibilisierung der angesprochenen URL)
- Übermittlungstyp
(=Ausfertigung (AUS) bzw. Ausfertigung und Zustellung (AUZ))
- Kurzbezeichnung des Fachinformationssystems
- Dokumente

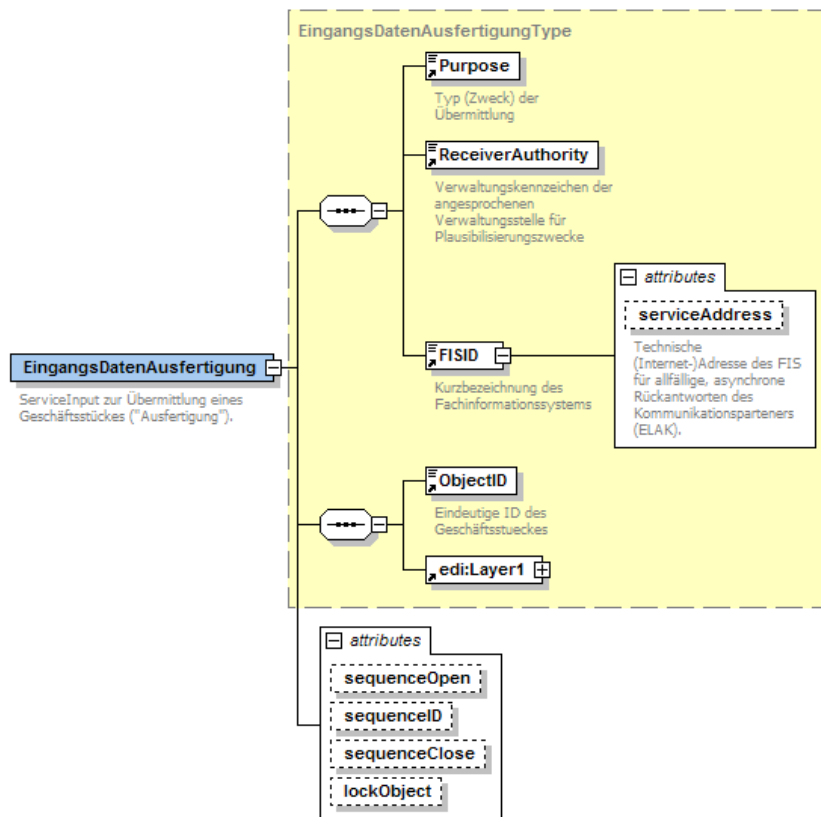
Output:

- Status der Aktion

Spezifikation Inputelemente: *EingangsDatenAusfertigung*

Das Element *EingangsDatenAusfertigung* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).



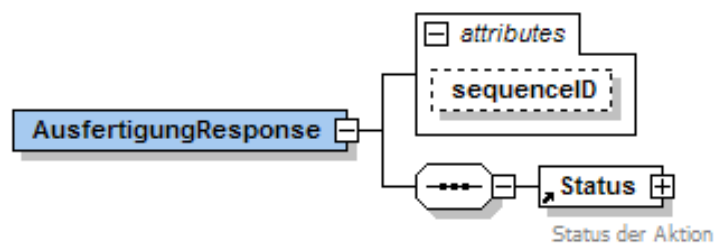
Element	Datentyp [Kardinalität]	Beschreibung
Purpose	PurposeType [1]	Typ der Übermittlung; hierin ist an dieser Stelle zulässig: AUS (Ausfertigung) AUZ (Ausfertigung und Zustellung)
ReceiverAuthority	VKZType (string) [1]	Verwaltungskennzeichen der angesprochenen Verwaltungsstelle für Plausibilisierungszwecke
FISID	FISIDType (string) [1]	Kurzbezeichnung des Fachinformationssystems
FISID	xs:AnyURI	Optionales Attribut zu FISID-

@serviceAddress	[0..1]	Element. Definition siehe Beschreibung aus Abschnitt 3.2.1.
ObjectID	ObjectIDType (string) [1]	ID des Geschäftsstückes. Siehe auch 2.1.
Layer1	edi:Layer1Type [1]	Container für das zu übermittelnde Geschäftsstück (Layer 1 Objekt).

Spezifikation der Outputelemente: *AusfertigungsResponse*

Das Element *AusfertigungsResponse* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).



Element	Datentyp [Kardinalität]	Beschreibung
Status	StatusType (Number + Message) [1]	Status der Aktion; repräsentiert durch einen numerischen Statuscode und einer optionalen Text- Message.

3.2.2.3 Entfertigungsstatus setzen (*Service entfertigung*):

Input:

- Objekt-ID des Geschäftsstücks
- VKZ der angesprochenen Verwaltungsstelle (zur Plausibilisierung der angesprochenen URL)
- Übermittlungstyp (Entfertigung ENT)
- Kurzbezeichnung des Fachinformationssystems

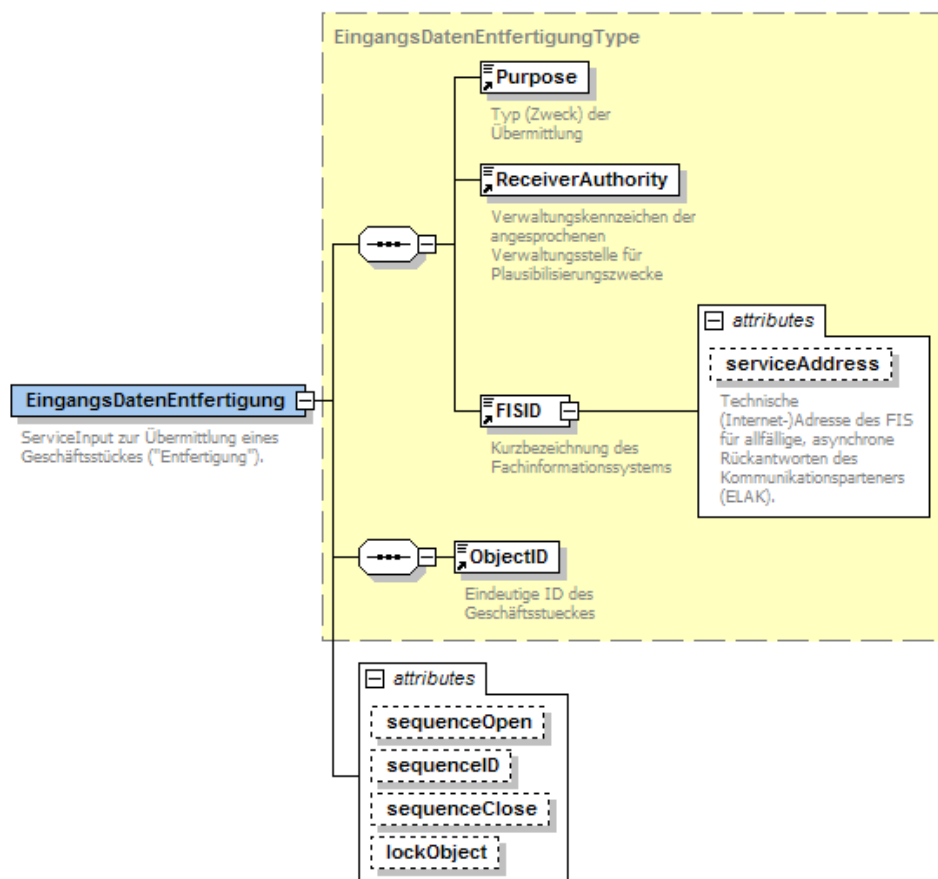
Output:

- Status der Aktion

Spezifikation der Inputelemente: *EingangsdatenEntfertigung*

Das Element *EingangsdatenEntfertigung* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).

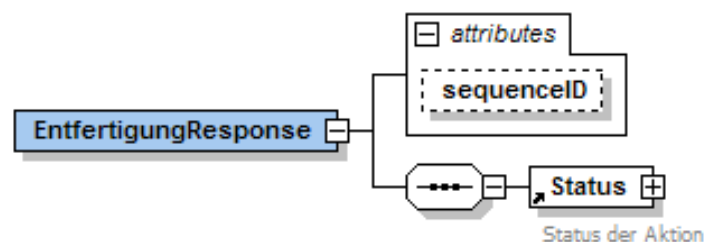


Element	Datentyp [Kardinalität]	Beschreibung
Purpose	PurposeType [1]	Typ der Übermittlung; an dieser Stelle zulässige Werte: ENT (Entfertigung)
ReceiverAuthority	VKZType (string) [1]	Verwaltungskennzeichen der angesprochenen Verwaltungsstelle für Plausibilisierungszwecke
FISID	FISIDType (string) [1]	Kurzbezeichnung des Fachinformationssystems
FISID @serviceAddress	xs:AnyURI [0..1]	Optionales Attribut zu FISID-Element. Definition siehe Beschreibung aus Abschnitt 3.2.1.
ObjectID	ObjectIDType [1]	Eindeutige ID des Geschäftstückes. Siehe auch 2.1.

Spezifikation der Outputelemente: *EntfertigungResponse*

Das Element *EntfertigungResponse* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).



Element	Datentyp [Kardinalität]	Beschreibung
Status	StatusType (Number + Message) [1]	Status der Aktion; repräsentiert durch einen numerischen Statuscode und einer optionalen Text- Message.

3.3 **Frage Akteninfo ab (Service searchElak)**

Mit dieser Funktion kann ein Akt, ein Geschäftsfall oder ein Geschäftsstück abgefragt werden

Input:

- Objekt-ID
- Typ des Response:
 - v = Version des gesuchten Objektes (ggf. inkl. letztem Änderungsdatum)
 - m = Version + Metadaten,
 - d = Version + Metadaten + Dokumente,

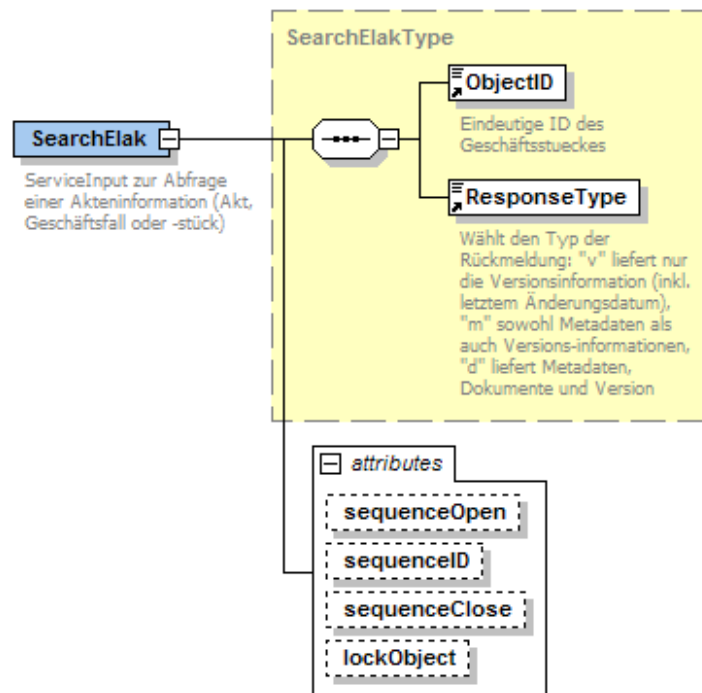
Output:

- Status der Aktion
- EDIAKT-Struktur

Spezifikation der Inputelemente: *SearchElak*

Das Element *SearchElak* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).



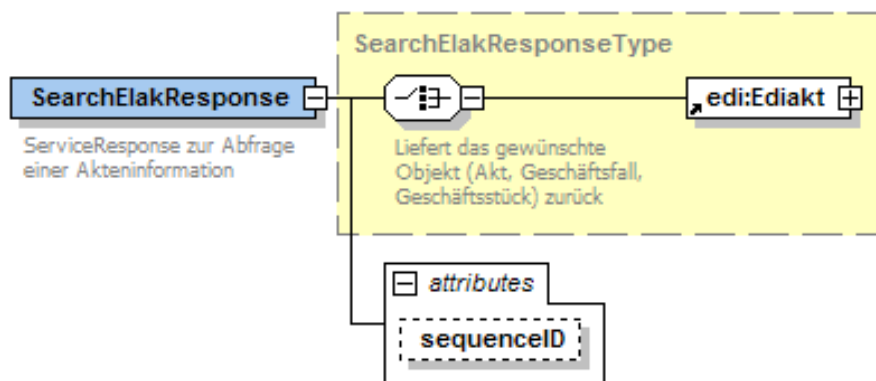
Element	Datentyp [Kardinalität]	Beschreibung
ObjectID	ObjectIDType (string) [1]	ID des abzufragenden Objektes (Akt, Geschäftsfall, oder Geschäftsstück). Siehe auch 2.1. <i>Anmerkung:</i> obschon die ID gem. Abschnitt 2.1 auch die Identifikation des ELAK-Systems enthält, muss zum Zeitpunkt der Abfrage schon das richtige, für das betreffende Objekt zuständige ELAK-System bekannt und angesprochen werden.
ResponseType	Token (enum: m, d, v) [1]	Wählt den Typ der Rückmeldung: "v" liefert nur die Versionsinformation (inkl. letztem Änderungsdatum), "m" sowohl Metadaten als auch Versionsinformationen, "d" liefert Metadaten, Dokumente und Version

Spezifikation der Outputelemente: *SearchElakResponse*

Das Element *SearchElakResponse* wird atomar verwendet.

Es sind hier auch alle in Abschnitt 2.5 definierten gemeinsamen Elemente/Attribute möglich (Definition siehe Abschnitt 2.5).

Als Ergebnis der Suche wird ein EDIAKT-Container zurückgegeben, der je nach geforderten Detaillierungsgrad der Antwort (v,m,d) unterschiedlich granular zu befüllen ist. Unabhängig vom geforderten Detaillierungsgrad der Antwort sind verschachtelt vorkommenden Sub-Elemente des EDIAKT-Containers (zB. Layer1-Elemente, etc.) je nach Vorkommnis zu retournieren.



Element	Datentyp [Kardinalität]	Beschreibung
Ediakt	edi:EdiaktType	<p>EDIAKT-Container der gefundenen und retournierten Aktenstruktur (Akt, Geschäftsfall, Geschäftsstück). Die Struktur wird je nach gefordertem Detaillierungsgrad der Antwort (v,m,d) entsprechend befüllt.</p> <p>Die Geschäftszahl (MetaData/Identifier) der jeweiligen Payload (Layer1/2/3-Objekt) KANN vom erforderlichen MetaData-Element des einhüllenden EDIAKT-Containers übernommen werden.</p>

4 SOAP-Faults und Message Codes

Die Fehlerbehandlung von ELAK-Transaktionen entspricht der in [soap-faults] vorgeschlagenen. Für allgemeine Server-Fehler sind die dort definierten Message-Codes zu verwenden. Für ELAK-Transaktions-spezifische Fehler sind die hier vordefinierten Fehlercodes zu verwenden oder gegebenenfalls zu erweitern.

Message-Codes sind vierstellig und in Klassen eingeteilt (gemäß [soap-faults]). Alle Message-Codes sind mit Standard-Texten verknüpft. Einige Messages werden in dieser Spezifikation vordefiniert, innerhalb der Klassen kann jedoch jede Implementierung eigene Message-Codes und korrespondierende Texte definieren. Dies sollte ausschließlich in der Message-Klasse 6 geschehen.

Message-Klassen

Die Klasse eines vierstelligen Message-Codes wird durch die erste Ziffer bestimmt. Folgende Message-Klassen sind in Anlehnung an die Status-Code-Definitionen des HTTP-1.0-Protokolls vordefiniert:

Message-Klasse	Bedeutung
2	Aktion erfolgreich ausgeführt
3	Zur erfolgreichen Ausführung sind weitere Aktionen notwendig
4	Client-Fehler
5	Server-Fehler
6	Von Implementierungen definierte Fehler

Message-Codes

Die im Folgenden vorgeschlagenen Fehlercodes sind numerisch und kommen in einem <faultcode>-Element eines SOAP-Faults zur Anwendung. Laut SOAP-Schema sind dort aber nur Qualified Names zulässig, d.h. Zeichenketten, die nicht mit einer Ziffer beginnen dürfen. Aus diesem Grund müssen die Fehlercodes in SOAP-Faults mit einem führenden „F“ versehen werden (z.B. „F4010“, ...).

Nr.	Text	Bedeutung
4010	Error in Input Data Structure	Die übermittelte XML-Struktur ist fehlerhaft oder wird in dieser Form nicht unterstützt.
4020	Error in Input Data Element	Ein Element der übermittelten Struktur ist fehlerhaft oder sein Inhalt bzw. dessen Typ ist für die Operation nicht gültig.
4030	Object not found	Ein angefragtes Objekt konnte aufgrund der Abfragekriterien nicht gefunden werden oder existiert nicht.
4040	PurposeType not supported	Das Service unterstützt den angeführten PurposeType nicht.
4050	Verfahren not supported	Das Service unterstützt das angeforderte Verfahren nicht.
4060	VKZ mismatch	Die Plausibilisierung des VKZ führte zu einem Fehler.
4070	FISID not found	Das über die FISID angesprochene FIS kann vom Service nicht adressiert werden.
4080	ELAKID not found	Das über die ELAKID angesprochene ELAK-System kann vom Service nicht adressiert werden.
4100	Sequence-ID not found	Sequenz unbekannt oder bereits beendet (infolge Time-Out).
4110	Sequence Timeout not accpeted	Die im Zuge der Sequenzeröffnung geforderte max. Wartedauer zwischen den einzelnen Transaktionen der Sequenze kann vom empfangenden System nicht garantiert werden. Die Sequenz wird nicht geöffnet.
4120	Locking impossible.	Sperrung bzw. Entsperrung des Objektes ist nicht möglich.

4130	Locking impossible without a sequence	Sperre kann nur innerhalb einer Sequenz von Transaktionen durchgeführt werden.
4140	Object Locked	Das gegenständliche Objekt ist derzeit gesperrt.
5100	Sequence-ID not found	Sequenz unbekannt oder bereits beendet (infolge Time-Out).
5110	Rollback not supported.	Das Rollback in Bezug auf die angegebene Transaktionssequenz wird nicht unterstützt (System bedingt).
5120	Rollback impossible.	Das Rollback in Bezug auf die angegebene Transaktionssequenz ist in Folge eines Verarbeitungsfehlers (ist in Antwort ggf. näher zu beschreiben) nicht möglich.

5 Referenzen

[ediakt]

Freitter, Gradwohl, Denner: „XML-Schema EDIAKT II“ in der Version vom 14.12.2005, <http://reference.e-government.gv.at>

[xml-e]

DI Herbert Pacnik, DI Michael Liehmann: „Erläuterung der XML-Spezifikation für den elektronischen Eingang“ in der Version vom 11.11.2005 ergänzt am 24.2.2006, <http://reference.e-government.gv.at>

[xml-baukasten]

Herpers, Pacnik, Liehmann, Wimmer: „XML-Baukasten für Behördenkommunikation, Basistypen“ in der Version vom 11.11.2005, <http://reference.e-government.gv.at>

[soap-faults]

Michael Liehmann, Franz-Josef Herpers: „SOAP-Faults und deren Behandlung“ in der Version vom 9.5.2005, <http://reference.e-government.gv.at>

[vkz]

Franz Grandits, Rainer Hörbe, Harald Wiesner: „Kennzeichen für Organisationseinheiten von Gebietskörperschaften bzw. von Körperschaften öffentlichen Rechts (Verwaltungskennzeichen)“ in der Version 1.1.0 vom 15.5.2003, <http://reference.e-gouvernement.gv.at>

[pvp]

Rainer Hörbe: „Spezifikation Portal Verbund Protokoll“ in der Version 1.8.9 vom 1.2.2005, <http://reference.e-governemnt.gv.at>