ERP Integration Tools

Guiding Document

Last revision: 22.06.2017

4 pages including the front page

Document No. 155.388.960

Technical documentation

Index

Introduction

ZEDAL is a system started in 2002 used for managing and exchange of documents in electronic waste management processes both nationally in Germany, conforming to the BMU standard, as well as internationally, conforming to the EU, OECD and Basel standards. It was equipped with various interfaces allowing for interaction with third party systems. The ZEDAL ERP Tools provide documentation and samples to integrate your product with the ZEDAL platform. All components are being provided under a free license.

ZEDAL Webservices

Contains documentation for all relevant webservice functions provided by ZEDAL to exchange documents.

→ *Jump to part ZEDAL Webservice*

BmuXmlArtist

BmuXmlArtist is a library encapsulating all the specific requirements for creating both German national as well as international documents and providing an easy to use interface to cope with them. Only minimal XML knowledge is required. A Windows setup is included in the package. A Linux version and free license codes are available on request.

→ *Jump to part BmuXmlArtist*

eTFS Data Model

Explains the structure of the XML documents in detail. The complete rule set incorporated into the BmuXmlArtist to build valid XML documents is described.

→ *Jump to part eTFS Data Model*

ZEDAL Forms Interface

ZEDAL Forms may be used to view, edit, and digitally sign BMU and, in conjunction with ZEDAL TFS plugin, international documents. It is generally used as a plug-in to web browsers or registered as a custom protocol handler, but can also be used within any Windows application. The interface is instantiated via a DLL. After loading, ZEDAL Forms will query various resources and finally provide the edited or signed data using callback methods from your application. A detailed interface description as well as documented example source code for using the interface are provided as part of this package. Windows setups for ZEDAL Forms and the TFS Plugin are available on request.

→ *Jump to part ZEDAL Forms Interface*

Examples

The *examples* directory contains a demonstration project. In order to compile the demo, **cmake 2.6** is needed to generate the build files of choice. Currently only Windows is supported.

The example consists of four files creating one complete XML document for every document type. All fields of the official forms are filled with consecutive unique numbers. Date fields are also filled with unique values. Boolean values are documented to make the meaning of false and true clear.

The four PDF files in *examples\reference_pdf* have been generated from the XML documents output by the example project. Their goal is to make the mapping between fields in the official forms and the path-IDs processed by the BmuXmlArtist obvious.

The remaining files are helping to get the sample project up and running:

- **main.cpp**: main program
- **license.h**: contains two defines providing valid license information.

Contact

Further information as well as license codes and package download links will be provided via e-mail. Please contact:

erp-tools@zedal.com

ERP Integration Tools

ZEDAL Webservice

Last revision: 16.12.2015

73 pages including the front page

Document No. 155.388.956

Technical documentation

ZEDAL Webservice - Index

1 Preface

ZEDAL 2, as its predecessor, offers a webservice interface that can be used by subscribers to access their data in the ZEDAL system with their own applications.

This documentation is based on the state of definition regarding the interface. Extensions and changes in detail are to be expected.

Terminology

Document	A document in ZEDAL 2 correlates to a physical document with its historical changes. A view can be constructed from the historical data that equals the document at that point in time.
Document file	Specifies a continuous block of data that is transferred to the system for processing purposes.
File	A logical grouping of documents. A file also keeps a historical log. For any point in time a view can be created.
File ident	Each file is given a unique ident. This ident is of the form 1234-5
Document ident	Each document is given a unique ident. This ident is of the form 1234-5
Log ident	Each protocol entry is given an ident that contains date, time and additional keys and determines the order within a given file or document. The ident is unique within its file or document.
IKS	The document format of ZEDAL 1
BMU	The document format specified by Bundesministeriums für Umwelt und Naturschutz

1 Changes

2009-05-25

- AssignRefIdToFile
- GetFileByRefId
- SendDocument
- DeliverDocument
- PeekNextTransportDocument
- TransportDocumentReceived
- ImportDocumentToFileAreaFromInbox
- ImportDocumentToFileFromInbox
- ImportToDocumentInFileFromInbox
- ImportNewDocumentToFileFromInbox
- ImportTransportDocumentFromInbox
- ListInbox
- MarkDocumentAsReadInInbox
- RemoveDocumentsFromInbox

2009-06-19

- GetCompletedSubmissions
- GetRejectedSubmissions
- DiscardFinishedSubmission
- GetFileParticipant
- GetDocumentParticipant
- SetDocumentParameter
- GetDocumentParameter
- GetDocumentsByParameter

2009-08-26

- RequestZksAddress
- DBInsert
- DBSynchronize
- DBLookup
- DBSelect
- DBDelete
- DBDescribe
- DBFunction

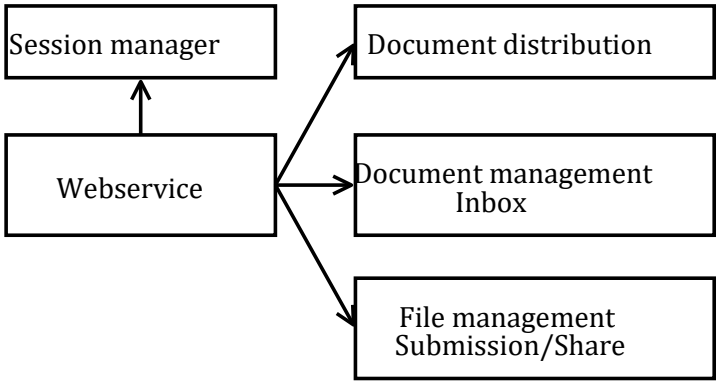
2010-04-15

- SetFileMainDocument
- SetDocumentOutdated
- DeleteDocumentFromFile
- RecoverDeletedDocumentToFile

- MoveDocumentToFile
- ApplyInboxFilter
- CreateTransportTemplate
- DeleteTransportTemplate
- ExportTransportTemplate
- GetTransportTemplateCount
- GetObtainedTPNos
- CreateFileareaTemplate
- DeleteFileareaTemplate
- ExportFileareaTemplate
- GetFileareaTemplateCount

2 Structure and principles

The webservice grants access to certain services of ZEDAL 2 that can be used by external applications.



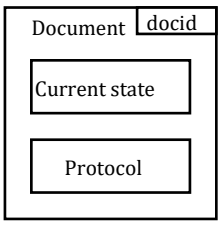
2.1 Session management

Besides very few exceptions the functions require the caller to provide a valid session number. This is obtained from a successful login (OpenSession). The session number is held in the session management for a limited period of time and is deleted afterwards. Activities on the webservice reset the remaining time.

2.2 Document management

Every document is deposited in the document management and given a unique document ident.

A protocol containing every operation is held for each document. The appearance of the document (current view) is determined by the history.



Protocol entries:

- Import of a document file
- Submission of a document
- Sharing a document
- Change of a parameter
- Distribution

A document is automatically created when a document file is to be imported into a new document.

A document can initially only be edited by the subscriber who created it. This subscriber is the owner of the document. Only after further operations of the distribution or file management a document is made available to other subscribers.

2.3 Distribution

The distribution is a means to automatically determine the participating parties in a document and deliver it to them. This behavior can be suppressed through an option parameter in the import function.

By using special options when sharing a file additional participating parties can be involved in the distribution. This way parties that are not explicitly listed in the document (e.g. the agent of the waste generator) can get access to the document.

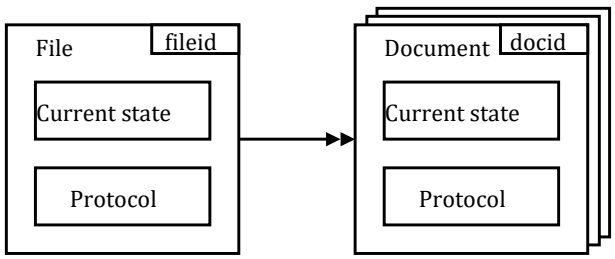
The distribution is the storage for movement documents.

2.4 File management

Apart from the distribution documents can be managed in files. A file can contain multiple documents. This way they can be grouped logically. File operations may apply to single, multiple or all documents in that file.

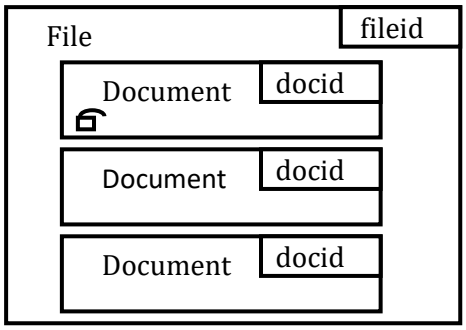
The core functionality for exchanging documents in files are submissions and shares.

The file management keeps a protocol of all operations executed on a file and determines its current state from these.



Protocol entries:

- Creation of the file
- Document added
- Submission of the file
- File shared
- Parameter changed



Initially a file can only be edited by the subscriber who created it. This subscriber is the file owner. Further operations in the file management (e.g. submission and share) are necessary to allow other subscribers to access the file.

The file area is typically used for e.g. notification documents.

2.4.1 Submission

To give a subscriber temporary access to a file and the contained documents a submission has to be issued to this subscriber. The destination subscriber gains access to the file according to the granted permissions and may edit documents contained in the file. Depending on what operations the destination subscriber executed on the file he may keep a permanent permission on the file and contained documents, or just some of them.

If he just rejects the submission the destination subscriber gives up his temporary access.

A submission can be held by the destination subscriber in order to work on the file without the risk of the submission to be revoked.

Rules:

- Before a new submission is issued a previous submission has to be finished.
- A held submission cannot be revoked within the holding period of time.
- The owner, an agent or the submission issuer can revoke the submission (when it is not held).
- Submissions can be issued to external parties (e.g. ZKS) as well. In this case the submission causes all eligible documents to be sent. The issuer, owner or an agent then have to revoke or finish the submission.
- Only a subscriber with the necessary permission may submit a file.
- While he has a submission only the target subscriber may edit the file. When submitted to an external destination no internal subscriber can edit the file. The submission then has to be ended in order to edit the file.
- If a submission is finished the (internal) destination subscriber gains permanent access to the file. If it is revoked or rejected no permanent rights are gained.
- A submission may be limited to a specific period of time by the issuer. It is then automatically revoked when the time limit is reached and no other action has been taken.

Definitions

A,B	Subscribers
A→B	A issued a submission to B
B_H	Submission is held by B
B_Z	Submission was rejected by B
B_B	Submission has been edited by B
B_E	The submission to B has been revoked
B_{ZKS}	B is an external party (e.g. ZKS)
V	Subscriber may submit
E	Subscriber may revoke submission
Z	Subscriber may reject submission
H	Subscriber may hold submission
B	Subscriber may edit submission

Examples for valid operations on a single submission

Submission	**A**	**B**
A	V	-
A→B	E	HZB
A→B_H	-	ZB
A→B_B	V	(V)
A→B_E	V	-
A→B_Z	V	-
A→B_{ZKS}	E	-

2.4.2 Share

The second possibility to allow access to a file is a share. A share allows access to the file until it is revoked.

The system holds one share state for each target subscriber. A new share of the same file for the same destination subscriber can only be issued after the previous share is revoked.

It is possible to share all transport documents that belong to the notification in the file along with the file itself. Only those transport documents that the sharing subscriber himself has access to will be shared.

A share can be limited to a specific period of time.

3 General parameters

Some Parameters are used in a multiple functions. These are described below:

3.1 sessionid

General string that results from a successful login. The string is designed so that it cannot be guessed. It contains the server time and a random portion.

3.2 fileid

External identifier for an electronic file. It consists of two portions: the file number and the numeric provider identifier.

E.g. „322-3“

The numeric provider-ID is encoded in the last two digits of the file number in ZEDAL1.

3.3 docid

External identifier for an electronic document. As the fileid it consists of two portions: the document number and the numeric provider identifier.

E.g. 6538-23

3.4 binarydata

For transferring binary data/files. The content is transferred as a base64-encoded string.

3.5 privacylevel

A document is associated with a privacy level that determines the default behavior of the system when submissions and shares are issued. This behavior can be explicitly overridden.

The following levels are used:

Level	Description	Remark
0	confidential	The document will never be shared by a submission or share.
1	private	The document will not be shared by a submission or share by default.
2	Public for editing	The document will be shared by a submission or share and will be editable by default.
3	Public for viewing	The document will be shared by a submission or share but will not be editable by default.

For all levels > 0 the default behavior can be overridden.

3.6 grantee, receiver

When a submission, share or delivery is issued the target party has to be specified. These two string fields handle this kind of information. The grantee contains the destination subscriber, receiver may contains a sublevel of addressing that is evaluated at the target site.

The following types of addressing are used to specify the destination of a communication within ZEDAL and between ZEDAL and ZKS.

1. Role
2. BEHOERDE@ZKS
3. Role-Number
4. Role-Number@ZKS
5. Role-Number:Role-Number@ZKS
6. Subscriber@Provider
7. Subscriber
8. *
9. Role-Number:BEHOERDE@ZKS

Which type of addressing is acceptable for a specific use depends on the called function.

1.
This type allows addressing the destination by its role in the document. The roles defined by ZKS are allowed (ERZ, BEF, ENT, etc.)

It requires that the document can be uniquely identified and the specified role is defined by the document.

Usable in: SendDocument, DeliverDocument

2.
The predefined constant value BEHOERDE@ZKS selects the inbox of the competent authorities as the destination. The inbox address of the competent authorities is defined by the provider when the ZKS access is configured.

Usable in: SendDocument, DeliverDocument

3.
When addressing by role followed by a hyphen and a registration number (e.g. ERZ-A12345678), ZEDAL checks whether or not this number can be reached via internal ways. If the destination party is no ZEDAL-subscriber the document is sent via ZKS.

Usable in: SendDocument, DeliverDocument

4.
When addressing by role followed by a hyphen and a registration number (z.B. ERZ-A12345678) and @ZKS the user defines that the document has to be sent via ZKS. The destination party's default inbox will be used.

Usable in: SendDocument, DeliverDocument, OpenSubmission

5.
Just as in 4, but the role and registration number of the provider are specified after the colon. This type allows to address a specific ZKS inbox, e.g. when a different than the default inbox is to be used.

Usable in: SendDocument, DeliverDocument, OpenSubmission

6.
A ZEDAL 2 subscriber is addressed by

subscribernumber@providernumber

e.g.

123456@5

Usable in: SendDocument, DeliverDocument, OpenSubmission, GrantAccessToFile

7.
If the ZEDAL-subscriber knows that the destination subscriber is using the same provider, the providerId can be omitted.

Usable in: SendDocument, DeliverDocument, OpenSubmission, GrantAccessToFile

8.
Star * means, that all participating parties are to be determined from a document. The function is to be executed for each of them. It is required that the document can be identified and its participating parties can be determined and are valid for the function in question (e.g. there may be only one submission at a given time).

Usable in: DeliverDocument

9.
This type of addressing is required when documents are to be sent to a specific competent authority, but over the global competent authority inbox at ZKS. In this case both the role and number as well as the intention to address the central competent authority inbox have to be specified.

Usable in: SendDocument, DeliverDocument

3.7 grantoptions

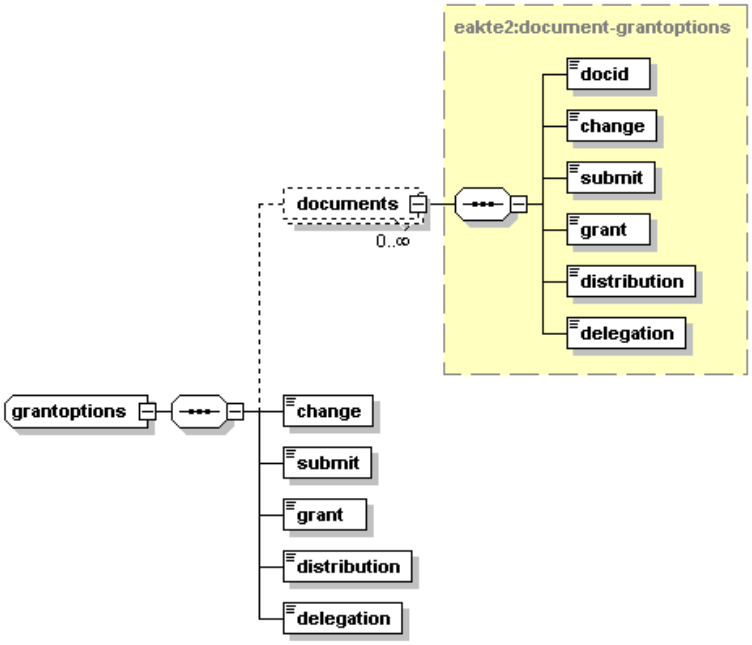
When issuing a submission or share the subscriber may determine what kind of editing options the destination subscriber may use. He may for example determine that the destination subscriber may only view, but not edit the file. He may as well determine that the destination subscriber may not issue further submissions of this file.

Options may be specified for the whole file as well as for individual documents.

When no options are given the system uses default values. If options for individual documents are given only these documents are considered during a submission or share. That means that the

options for individual documents have to be omitted or options for each document of the file have to be given, if the action is to consider all documents.

The individual options and their impact is described below.



3.7.1 Options at file level

change
Destination subscriber may edit the file (e.g. add public documents or edit documents).
If this value is "*false*" then "*false*" is assumed for each document.

If no value is given the system assumes "*true*".

submit
true: The destination subscriber may submit the file to other subscribers. He may only issue a submission for those documents he has submission permissions for.

false: The destination subscriber may not submit the file to other subscribers.
If no value is given the system assumes "*true*".

grant
true: The destination subscriber may share the file with other subscribers.

false: The destination subscriber may not share the file with other subscribers.
If no value is given the system assumes "*false*". This option is not used on submissions.

distribution
This option controls the inclusion of transport documents. The file level value is used as the default for the document level as long as the document level does not specify otherwise.

This option is not used on submissions.

delegation
A warrant is issued to the destination subscriber. The file level value is used as the default for the document level as long as the document level does not specify otherwise.

If this option is set during a submission the destination subscriber becomes an additional owner of the file if he finishes the submission. If he rejects the submission he does not become owner of the file.

3.7.2 Options at document level

Options for a single document are set in the elements “documents” that may occur multiple times. If one or more documents parameters exist only the given documents will be considered, so every relevant document has to be given.

docid
Identifier of the document that the data relates to. The user must have access to the document, and it must be contained in the file.

change
true: the destination subscriber may edit the document. This value is only accepted if editing of the file was enabled as well, and if the issuing user has permission to change the document.
false: the destination subscriber may only view the document.

The default value is defined by the documents privacy level:

privacylevel	**Default value**
0 (confidential)	Not allowed
1 (private)	false
2 (public for editing)	false
3 (public for viewing)	true

submit
true: the destination subscriber may use submit the document. This value is only accepted if submission of the file was allowed as well and the user has permission to submit the document.
false: the destination subscriber may not submit this document.

grant
true: the destination subscriber may share this document. This value is only accepted if sharing of the file was allowed and the user has permission to share the document.
false: the destination subscriber may not share this document.

This option has no function when submitting a file.

distribution
true: Transport documents that relate to this document will be accessible for the destination subscriber if the issuer of the share is a participating party for the transport. This value is only

accepted if the user is owner or received a warrant.
false: this document does not influence the distribution of transport documents.

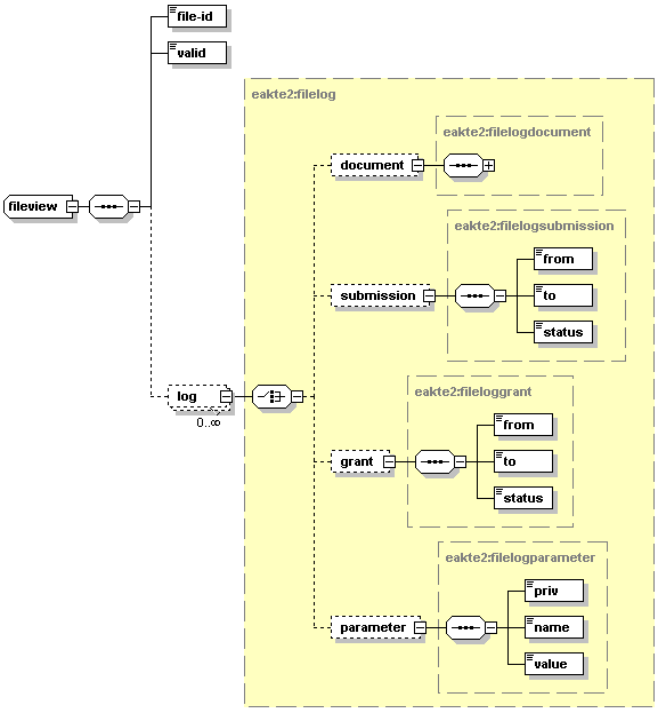
This option has no function when submitting a file.

delegation
true: a warrant is issued for this document.
false: no warrant is issued for this document.

This option has no function when submitting a file.

3.8 fileview

fileview defines the view on a file. This structure is used for both the current (GetCurrentViewOfFiles) as well as the historical view (GetHistoryViewOfFiles). The view determines how many elements are provided and how they are filled



file-id
The identifier of the file for verification purposes.

valid
true, if the file was found and is accessible for the user
false, otherwise

log
A list with different kinds of entries. The caller has to check the type of the entry.

document
View on a document contained in the file. In the current view an entry for every document visible to

the user is created. The order of the entries correlates to the chronological order in which the documents were added to the file.

submission
Submission status. The current state for the subscribers affected by the submission – if there is one – is given here.

The historical view gives an entry for every issued submission.

grant
Share status. In the current view all subscribers with a currently existing share are listed. There may be up to one entry per subscriber.

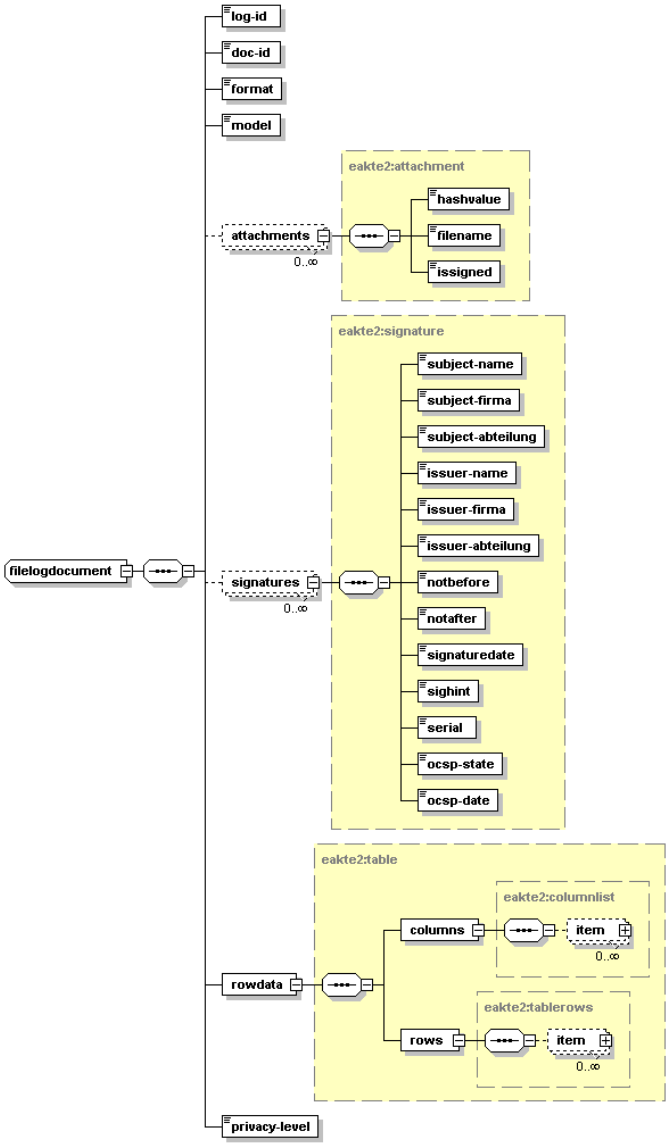
In the historical view all shares are listed.

parameter
State of the parameters. In the current view all currently valid parameter values are listed.

priv
The privacy-level that is defined of this parameter.

name
Parameter name.

value
Parameter value.



log-id

Identification of the protocol entry that caused the addition of the document to the file. The value contains date and time.

doc-id

Document identifier

format

Identifier of the format the document was provided. Currently known formats are:

Format identifier	Description
IKS	ZEDAL 1
BMU	BMU specification

model
Identifier for the usage of the document. For some documents there are different formats, but the purpose is the same.

Currently defined identifiers:

Identifier	Description
BGS	Movement document
UNS	Übernahmeschein
ENS	Notification

attachments
List of attachments of a document, if these are externally referenced by a hash value as in the IKS format.

hashvalue
Hash value of the document attachment calculated using SHA256.

filename
Name of the attachment.

issigned
Flag whether or not the attachment is enveloped by a signature.

signatures
List of all signatures in the document.

subject-name
Name of the card owner.

subject-firma
Company of the card owner, if existing.

subject-abteilung
Department of the card owner, if existing.

issuer-name
Name of the card issuer.

issuer-firma
Company of the card issuer, if existing.

issuer-abteilung
Department of the card issuer.

notbefore
Begin of the validity period of the card's certificate in the format YYYYMMDDHHMISS

notafter
End of the validity period of the card's certificate in the format YYYYMMDDHHMISS

signaturedate
Date and time of the signature in the format YYYYMMDDHHMISS

sighint
Identifier of the role the signature was applied in. The meaning depends on the actual document.

serial
Serial number of the card's certificate.

ocsp-state
State of the signature check according to OSCP.

oscp-date
Date of the last OSCP check.

rowdata
A row with overview data as presented in the ZEDAL portal.

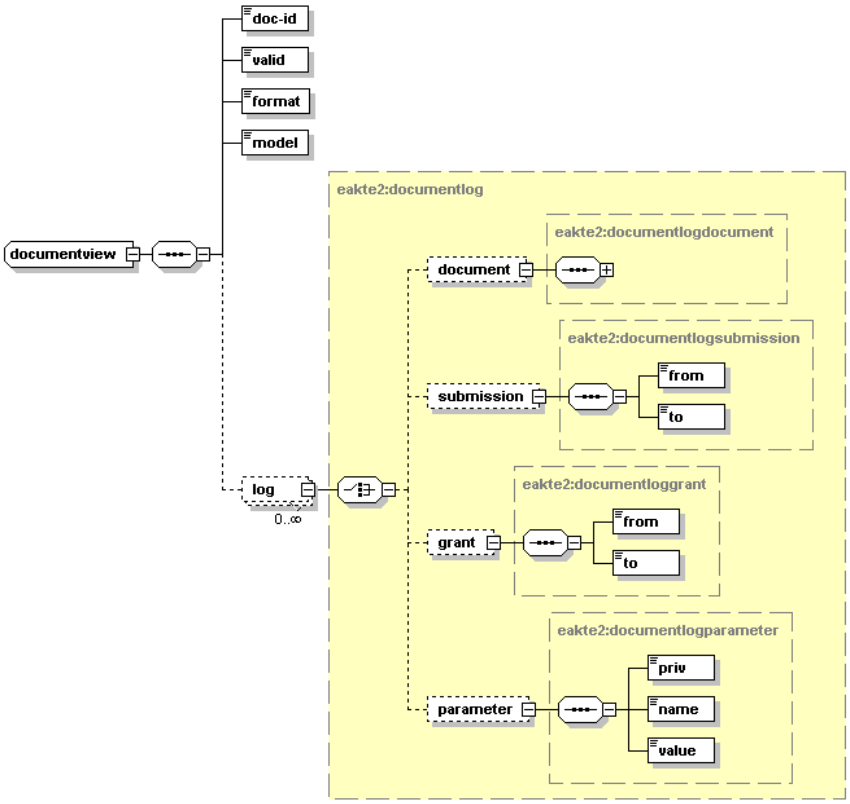
columns
List with column descriptions.

rows
A row of data.

privacy-level
The privacy level of the document.

3.9 documentview

Complex structure containing the view of a document. It is used for both the current and historical view.



doc-id
Document identifier for verification purposes.

valid
true, if the document is found and the user has permission to view it.

format
Identifier of the format the document is stored in.

model
Identifier for the usage of the document.

log
A list with different kinds of entries. The caller has to check the type of the entry.

Document
Entry for a document.

submission
(last) submission, may only be present once in the current view.

grant
(last) share, may only be present once in the current view.

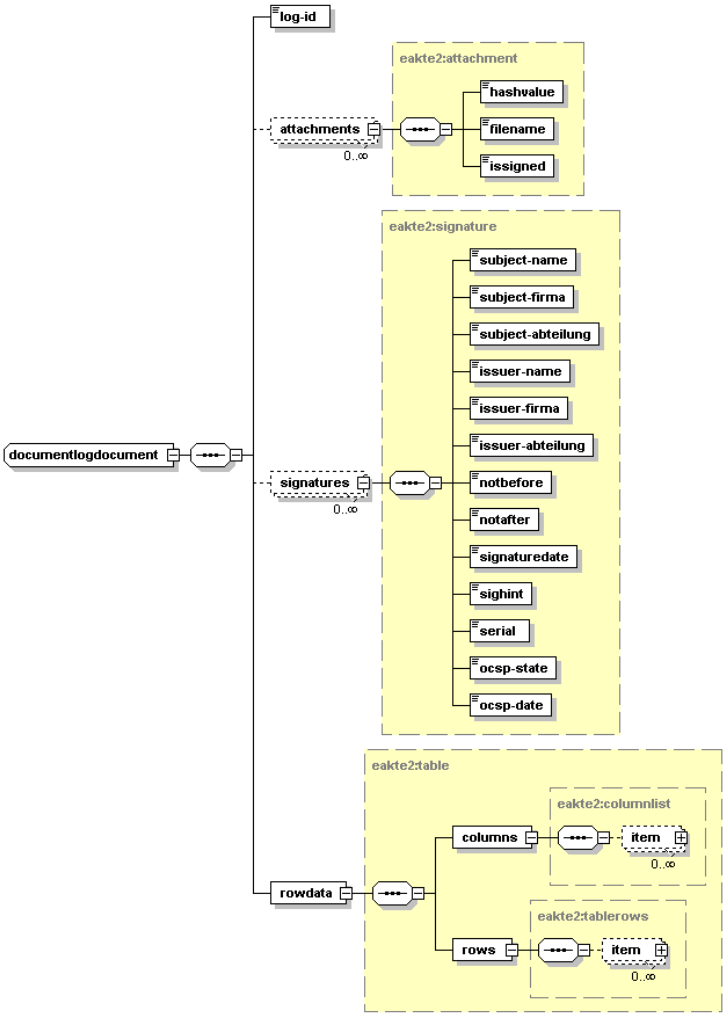
grantee
Identifier of the subscriber that the file has been shared with.

parameter
(last) set parameter.

priv
private value.

name
Name of the parameter.

value
Value of the parameter.



log-id
Identification of the protocol entry that was created when the document file was taken over into the document. The value contains date and time.

attachments
List of attachments of a document, if these are externally referenced by a hash value as in the IKS format.

hashvalue
Hash value of the document attachment calculated using SHA256.

filename
Name of the attachment.

issigned
Flag whether or not the attachment is enveloped by a signature.

signatures
List of all signatures in the document

subject-name
Name of the card owner.

subject-firma
Company of the card owner, if existing.

subject-abteilung
Department of the card owner, if existing.

issuer-name
Name of the card issuer.

issuer-firma
Company of the card issuer, if existing.

issuer-abteilung
Department of the card issuer.

notbefore
Begin of the validity period of the card's certificate in the format YYYYMMDDHHMISS

notafter
End of the validity period of the card's certificate in the format YYYYMMDDHHMISS

signaturedate
Date and time of the signature in the format YYYYMMDDHHMISS

sighint
Identifier of the role the signature was applied in. The meaning depends on the actual document.

serial
Serial number of the card's certificate.

ocsp-state
State of the signature check according to OSCP.

oscp-date
Date of the last OSCP check.

rowdata
A row with overview data as presented in the ZEDAL portal.

columns
List with column descriptions.

rows
A row of data.

4 Functions

The functions and their parameters and return values are described below.

The functions, their call parameters and return values are described by the following notation:

Function (call parameters) → (return values)

The parentheses are empty if no values are expected. After that the function, the parameters, return values and exceptions are described briefly in the following form:

Parameter

Param1	Description of parameter 1
Param2	Description of parameter 2

Return values

Return1	Description of return value 1
Return2	Description of return value 2

Exceptions

(Code1)	Error description 1
(Code2)	Error description 2

After that a more detailed description follows.

The implementation provides just one return value “result” that is structured if necessary. The function descriptions use more descriptive name.

Functions without return value provide an empty value.

Depending on the concrete implementation and development environment the names of parameters and return values may vary.

5 Session management

5.1 OpenSession

OpenSession(subscriberid, accountid, authtype, authparameter) → (sessionid)

Open a session in order to be able to work with the webservice. Most operations require a sessionId.

Parameter

subscriberid	subscriber number
accountid	(optional) Account name. If empty the main account is used
authtype	defines how to interpret the authparameter.
authparameter	authentication data, see below.

Return values

sessionid	result.message contains the sessionId needed for most of the following function calls

Exceptions

Invalid login credentials

All invalid login credentials lead to an abort of function execution. The error code does not distinguish between invalid subscriber / account data and invalid authentication data.

The following authentication types are defined:

authtype	authparameter	description
(empty)	password	Simple authentication by subscriber number / account number and password.

5.2 CloseSession

CloseSession(sessionid) → ()

Ends an existing session.

Parameter

sessionid	(valid) session number

Return values none

Exceptions none

5.3 GetSessionParameter

GetSessionParameter(sessionid, parname) → (parvalue)

This function allows querying session settings. Some parameters may be modified using SetSessionParameter, others are just to query system settings.

Parameter

sessionid	(valid) session number
parname	name of the parameter

Return values

parvalue	value of the parameter

Exceptions

session number is invalid
parameter is not defined

The following parameters are defined:

wait_for_results
This parameter allows setting whether the caller wants to wait for the complete processing of imported data or not.

true: The functions return only after complete processing of an import including all checks and follow-up processing. This parameter should be set if other processing directly follows.

false: The import functions return as soon as the document file has been checked and saved to the database. Follow-up processing, e.g. signature validation may not have finished yet. This option allows for faster handling if that follow-up processing is not required for further functionality.

5.4 SetSessionParameter

SetSessionParameter(sessionid, parname, parvalue) → ()

This function allows for modifying session settings.

Parameter

sessionid	(valid) session number
parname	name of the parameter
parvalue	the value that is to be set

Return values none

Exceptions

session number is invalid
parameter is not defined
parameter is not modifyable
value is invalid

6 File management

6.1 File functions

6.1.1 CreateNewFile

CreateNewFile(sessionid, title) → (fileid)

Create a new, empty file

Parameter

sessionid	(valid) session number
title	Arbitrary description for the file. The title is set as file parameter ‚bezeichnung' and may later be modified by using SetLocalParameter. The text will be shortened if necessary.

Return values

fileid	File identifier in result.message

Exceptions

session number is invalid

The account that created the file will automatically be destination of an internal submission (see: SetLocalSubmission), so that it may see the file regardless of any further restrictions on the account.

6.1.2 GetCurrentViewOfFile

GetCurrentViewOfFile(sessionid, fileids, lastident) → (fileviews)

The function generates a current view for each given file identifier.

Parameter

sessionid	(valid) session number
fileids	List of file identifiers
lastident	Comparative value that is used to limit the used data. All entries from the file with an ident <= lastident will be considered. This parameter may be left empty, all historical data is used then.

Return values

fileviews	List of resulting current views in result.views. There is one entry for each file identifier that was given. Even non-existing and files without access permission will be present, but marked as invalid. Each view entry contains the file identifier for verification. For details see below.

Exceptions

session number is invalid

6.1.3 GetHistoryViewOfFiles

GetHistoryViewOfFiles(sessionid, fileids, lastident) → (fileviews)

The function generates a historical view for each given file identifier.

Parameter

sessionid	(valid) session number
fileids	List of file identifiers
lastident	Comparative value that is used to limit the used data. All entries from the file with an ident <= lastident will be considered. This parameter may be left empty, all historical data is used then.

Return values

fileviews	List of resulting historical views in result.views. There is one entry for each file identifier that was given. Even non-existing and files without access permission will be present, but marked as invalid. Each view entry contains the file identifier for verification. For details see below.

Exceptions

session number is invalid

The return value for historical views uses the same structure that is used for current views (see: GetCurrentViewOfFile). While the current view only has up to one entry for submission, the historical view contains every submission that has been issued.

6.1.4 FindFilesToDocument

FindFilesToDocument(sessionid, docid) → (ids)

Returns the identifiers of the documents that match the functional identifiers of the document specified by docid. The comparison criteria depend on the type of document.

Parameter

sessionid	(valid) session number
docid	Identifier of the reference document

Return values

ids	List of Document-/file identifier-pairs in result.list

Exceptions

session number is invalid
the document does not exists or is inaccessible

6.2 Import functions

6.2.1 ImportDocumentToFileArea

ImportDocumentToFileArea(sessionid, privacylevel, binarydata) → (fileid, docid)

This function imports a document file into a file. It automatically decides which file and document is used. If no matching document is found a new file with a new document will be created.

Parameter

sessionid	(valid) session number
privacylevel	The minimal privacy level that is accepted for an existing document.
binarydata	Binary data containing the document file.

Return values

fileid	File indentifier in result.message
docid	Document identifier in result.message2

Exceptions

session number is invalid
parameters are invalid (invalid privacylevel)
the document could not be identified uniquely
the found document does not satisfiy the required privacy level
the document file could not be processed

The result equals ImportNewDocumentToFile() or ImportToDocument() depending on the situation.

6.2.2 ImportDocumentToFile

ImportDocumentToFile(sessionid, fileid, privacylevel, binarydata) → (docid)

Imports a document file in the specified file. The function automatically decides whether an existing document is to be used or a new document will be created.

Parameter

sessionid	(valid) session number
fileid	File identifier.
privacylevel	The minimal privacy level that is accepted for an existing document.
binarydata	Binary data containing the document file.

Return values

docid	Document identifier of the new or existing document in result.message

Exceptions

session number is invalid
no permission to access the file
Minimum privacy level is not satisfied.
The document can not be identified uniquely.
The document file could not be processed.

The result equals ImportNewDocumentToFile() or ImportToDocument() depending on the situation.

6.2.3 ImportToDocument

ImportToDocument(sessionid, docid, binarydata) → ()

Imports a document file to an existing document. The document has to be in a file or in the distribution.

Parameter

sessionid	(valid) session number
docid	Document identifier.
binarydata	Binary data containing the document file.

Return values none

Exceptions

session number is invalid
The specified document is not accessible.
The document is neither part of a file nor in the distribution.
The document file could not be processed.
The document file does not conform to the existing document.

If the document is contained in a file the importing subscriber gains access to this document and the containing file.

Type and format of the document file have to conform with the existing document. For documents in the distribution the crucial functional identifier (e.g. the waste movement document number) has to match.

In order to specify further options that are specifiy for transport documents the function ImportTransportDocument has to be called instead.

6.2.4 ImportNewDocumentToFile

ImportNewDocumentToFile(sessionid, fileid, privacylevel, binarydata) → (docid)

Imports a new document file into a specific file.

Parameter

sessionid	(valid) session number
fileid	File identifier
privacylevel	The privacy level the new document is created with.
binarydata	Binary data containing the document file.

Return values

docid	Document identifier of the new document

Exceptions

	session number is invalid
	The specified file does not exist.
	The document file could not be processed.

The executing subscriber will be owner of the document and gains editing access to the file.If the document is created with public privacy level it will be accessible for every party that has access to the file.

6.2.5 ImportDocumentToFileAreaFromInbox

ImportDocumentToFileAreaFromInbox(sessionid, minprivacylevel, docidsrc) → (docid, fileid)

This functions works like ImportDocumentToFileArea, but takes a document from the inbox as source document. The source document remains in the inbox.

Parameter

sessionid	(valid) session number
minprivacylevel	see ImportDocumentToFileArea
docidsrc	Document identifier, has to identify a document in the inbox.

Return values

docid	Document identifier of the existing or new document
fileid	File identifier of the existing or new file

Exceptions

	see ImportDocumentToFileArea
	docidsrc is invalid

6.2.6 ImportDocumentToFileFromInbox

ImportDocumentToFileFromInbox(sessionid, fileid, minprivacylevel, docidsrc) → (docid)

This functions works like ImportDocumentToFile, but takes a document from the inbox as source document. The source document remains in the inbox.

Parameter

sessionid	(valid) session number
fileid	see ImportDocumentToFile
minprivacylevel	see ImportDocumentToFile
docidsrc	Document identifier, has to identify a document in the inbox

Return values

docid	Document identifier of the existing or new document

Exceptions

	see ImportDocumentToFile
	docidsrc is invalid

6.2.7 ImportToDocumentInFileFromInbox

ImportToDocumentInFileFromInbox(sessionid, docid, fileid, docidsrd) → ()

This functions works like ImportToDocument, but takes a document from the inbox as source document. The source document remains in the inbox.

Parameter	
sessionid	(valid) session number
docid	The document to edit
fileid	Optional indication of the file to prevent ambiguity. The file must contain the document if specified.
docidsrc	Document identifier, has to identify a document in the inbox

Return values	none
Exceptions	
	see ImportToDocument
	fileid is invalid
	docidsrc is invalid

6.2.8 ImportNewDocumentToFileFromInbox

ImportNewDocumentToFileFromInbox(sessionid, fileid, privacylevel, docidsrc) → (docid)

This functions works like ImportNewDocumentToFile, but takes a document from the inbox as source document. The source document remains in the inbox.

Parameter	
sessionid	(valid) session number
fileid	The file to edit.
privacylevel	The privacy level for the new document.
docidsrc	Document identifier, has to identify a document in the inbox.

Return values	
docid	Identifier of the newly created document.

Exceptions	
	s. ImportNewDocumentToFile
	docidsrc is invalid

6.3 **Submission**

6.3.1 OpenSubmission

OpenSubmission(sessionid, fileid, receiver, grantoptions) → ()

Create a submission to an internal or external participating party.

Parameter	
sessionid	(valid) session number

fileid	File identifier.
receiver	Indication of the receiving party with address and subaddress.
comment	Comment for the submission.
grantoptions	Options regarding file and documents as described in 3.7

Return values none

Exceptions

session number is invalid

Whether or not this function succeeds depends on permissions and the current submission state.

If no grantoptions are given all public documents are submitted with default permissions.

6.3.2 CancelSubmission

CancelSubmission(sessionid, fileid, comment, receiver) → ()

Revoke a submission.

Parameter

sessionid	(valid) session number
fileid	File identifier.
comment	An optional remark may be deposited here.
receiver	Indication of the party the submission is to be revoked from. This argument may be omitted, the function then refers to the submission of the calling subscriber.

Return values none

Exceptions

session number is invalid

The submission is finished after this call.

Temporary permissions gained by the submission will be removed.

Whether or not this function succeeds depends on the current submission state.

6.3.3 HoldSubmission

HoldSubmission(sessionid, fileid, comment) → ()

Hold a submission.

Parameter

sessionid	(valid) session number
fileid	File identifier.
comment	An optional remark may be deposited here.

Return values none

Exceptions

session number is invalid

The submission cannot be revoked for a certain period of time.

Whether or not this function succeeds depends on the current submission state.

6.3.4 RejectSubmission

RejectSubmission(sessionid, fileid, comment) → ()

Reject a submission.

Parameter

sessionid	(valid) session number
fileid	File identifier.
comment	An optional remark may be deposited here.

Return values none

Exceptions

session number is invalid

May only be called by the receiving subscriber.

Temporary permissions gained by the submission will be removed.

Whether or not this function succeeds depends on the current submission state.

6.3.5 CompleteSubmission

CompleteSubmission(sessionid, fileid, comment) → ()

Set a submission into ‚processed' state.

Parameter

sessionid	(valid) session number
fileid	File identifier.
comment	An optional remark may be deposited here.

Return values none

Exceptions

session number is invalid

May only be called by the receiving subscriber.

The destination subscriber permanently gains the permission of this submission.

Whether or not this function succeeds depends on the current submission state.

6.3.6 GetAllSubmissions

GetAllSubmissions(sessionid) → (fileids)

Fetches a list of all currently submitted files.

Parameter

sessionid	(valid) session number

Return values

fileids	A list of file identifiers in result.list

Exceptions

session number is invalid

6.3.7 GetNewSubmissions

GetNewSubmissions(sessionid) → (fileids)

Fetches a list of all newly submitted files.

Parameter

sessionid	(valid) session number

Return values

fileids	A list of file identifiers in result.list

Exceptions

session number is invalid

Newly submitted files are not held yet. By holding the submission the file is removed from this list.

6.3.8 SetLocalSubmission

SetLocalSubmission(sessionid, fileid, account) → ()

Temporary local submission of a file to an account of the same subscriber. This account gains access to the file until the local submission is modified or finished. This function does not create a protocol entry.

Parameter

sessionid	(valid) session number
fileid	Identifier of the file to be locally submitted.
account	The account of the subscriber that is to receive the local submission. If this parameter is empty the local submission will be finished.

Return values none

Exceptions

session number is invalid

This function may e.g. be used for an internal workflow, where the next employee receives a local submission by the previous one.

The local submission may be used by any account with access to the file. This means that an account that received a local submission is able to issue a new local submission on its own.

When a new file is created the creating account automatically receives a local submission.

6.3.9 GetCompletedSubmissions

GetCompletedSubmissions(sessionid) → (file_ids[])

Determines all submission that have been finished (CompleteSubmission) and that have not been been remove from this list by calling DiscardFinishedSubmission.

Parameter

sessionid	(valid) session number

Return values

file_ids	List of file identifiers the subscriber has issued a submission for and that have been completed.

Exceptions

session number is invalid

6.3.10 GetRejectedSubmissions

GetRejectedSubmissions(sessionid) → (file_ids[])

Determines all submission that have been rejected (RejectSubmission) and that have not been been remove from this list by calling DiscardFinishedSubmission.

Parameter

sessionid	(valid) session number

Return values

file_ids	List of file identifiers the subscriber has issued a submission for and a that have been rejected.

Exceptions

session number is invalid

6.3.11 DiscardFinishedSubmission

DiscardFinishedSubmission(sessionid, fileid) → ()

Removes a file from the list of completed or rejected submissions (GetCompletedSubmissions or GetRejectedSubmissions).

Parameter

sessionid	(valid) session number
fileid	File identifier that should be removed from the lists.

Return values none

Exceptions

session number is invalid
file identifier is invalid

6.4 Share

A share applies globally. That means it is independent of the granting party. A share that was issued by subscriber X may be modified or deleted by subscriber Y as long as that subscriber has sufficient permission to do so. A file owner has all permissions.

That also means that in a situation where subscriber A issues a share to subscriber B who in turn issues a share to subscriber C, then C keeps its share if B loses his share.

6.4.1 GrantAccessToFile

GrantAccessToFile(sessionid, fileid, grantee, comment, expirationdate, grantoptions) → ()

Parameter

sessionid	(valid) session number
fileid	File identifier.
grantee	Destination subscriber that is to receive the share.
comment	An optional remark may be deposited here.
expirationdate	Optional date on which the share expires.
grantoptions	Options regarding file and documents as described in 4.7

Return values none

Exceptions

session number is invalid
destination subscriber already has a share

The function may only return successfully if there is not already a share for the destination subscriber. Modifying a share is not directly possible. The old share has to be revoked by calling RevokeAccessFromFile(), then a new share may be issued.

6.4.2 RevokeAccessFromFile

RevokeAccessFromFile(sessionid, fileid, grantee, comment) → ()

Revoke access from a file.

Parameter

sessionid	(valid) session number
fileid	File identifier.

grantee	Destination subscriber whose share is to be removed.
comment	An optional remark may be deposited here.

Return values none

Exceptions

session number is invalid

The function removes the share to that subscriber completely.

6.5 Additional function

6.5.1 GetLocalParameter

GetLocalParameter(sessionid, fileid, name) → (value)

Returns the last parameter value set be the subscriber itself. If the parameter has not been set yet an empty string is returned.

Parameter

sessionid	(valid) session number
fileid	File identifier
name	Name of the parameter

Return values

value	Value of the parameter

Exceptions

session number is invalid

6.5.2 SetLocalParameter

SetLocalParameter(sessionid, fileid, name, value) → ()

Sets a parameter on a file. The parameter will be publically visible.

Parameter

sessionid	(valid) session number
fileid	File identifier
name	Name of the parameter
value	New value of the parameter

Return values none

Exceptions

session number is invalid

6.5.3 GetFilesByLocalParameter

GetFilesByLocalParameter(sessionid, name, value) → (fileids)

Returns the identifiers of all files that have a parameter of the given value. This function allows to implement simple workflows.

Parameter

sessionid	(valid) session number
name	Name of the parameter
value	Comparison value

Return values

fileids	List of file identifiers

Exceptions

session number is invalid

6.5.4 GetPublicLocalParameter

GetLocalParameter(sessionid, fileid, name) → (value)

Returns the last set parameter value. If the parameter has not been set yet an empty string is returned.

Parameter

sessionid	(valid) session number
fileid	File identifier
name	Name of the parameter

Return values

value	Value of the parameter

Exceptions

session number is invalid

As GetLocalParameter(), but may be influenced by other subscribers setting the same parameter.

6.5.5 SetPublicLocalParameter

SetLocalParameter(sessionid, fileid, name, value) → ()

Sets a parameter on a file. The parameter will be publically visible.

Parameter

sessionid	(valid) session number
fileid	File identifier
name	Name of the parameter
value	New value of the parameter

Return values none

Exceptions

session number is invalid

As SetLocalParameter(). There is no difference.

6.5.6 GetFilesByPublicLocalParameter

GetFilesByLocalParameter(sessionid, name, value) → (fileids)

Returns the identifiers of all files that have a parameter of the given value. This function allows to implement simple workflows.

Parameter

sessionid	(valid) session number
name	Name of the parameter
value	Comparison value

Return values

fileids	List of file identifiers

Exceptions

session number is invalid

As GetFilesByLocalParameter(), but other subscribers influenced by other subscribers setting parameters.

6.5.7 AssignRefIdToFile

AssignRefIdToFile(sessionid, fileid, refid) → ()

Assigns a user defined identifier to a file in order to find this file → GetFileByRefId()

Each ident may only be assigned to a single file. If an ident is assigned to a different file the original association is lost.

Parameter

sessionid	(valid) session number
fileid	File identifier.
refid	User defined file identifier.

Return values none

Exceptions

session number is invalid

6.5.8 GetFileByRefId

GetFileByRefId(sessionid, refid) → (fileid)

Determines the file that the given user defined reference is a assigned to (see AssignRefIdToFile()).

Parameter

sessionid	(valid) session number
refid	Assigned user defined file identifier.

Return values

fileid	The associated file identifier or empty.

Exceptions

session number is invalid

6.5.9 GetFileParticipant

GetFileParticipant(sessionid, fileid, logid, fieldname) → (participant)

Determines the data of the participating parties of a file. This data contains the postal address and the numbers assigned by the authorities as far as they are known.

Parameter

sessionid	(valid) session number
fileid	File identifier.
logid	The protocol identifier within the file.
fieldname	Of a protocol entry is associated with multiple participating parties this parameter allows to select the relevant party. If left empty the party that has caused the entry will be returned.

Return values

participant	Structed data containing name, postal address and registration number.

Exceptions

session number is invalid
file ident is invalid
parameter is invalid

6.5.10 SetDocumentOutdated

SetDocumentOutdated(sessionid, docid, outdated) → ()

Defines a document as outdated. This may be the case if a document's validity period expires or it is replaced by a different document not not to be used anymore.

Parameter

sessionid	(valid) session number
docid	Document ident
outdated	Bool value used to set or unset the „outdated“ flag.

Return values none

Exceptions

session number is invalid
document identifier is invalid
document does not exist or no permission

6.5.11 DeleteDocumentFromFile

DeleteDocumentFromFile(sessonid, docid, fileid) → ()

Removes a document from a file and places it into an area for deleted documents.

Parameter

sessionid	(valid) session number
docid	Identifier of the document to delete.
fileid	Identifier of the file that contains the document.

Return values none

Exceptions

session number is invalid
document identifier is invalid
file identifier is invalid
document is not part of that file
document or file do not exist or no permission

6.5.12 RecoverDeletedDocumentToFile

RecoverDeletedDocumentToFile(sessionid, docid, fileid) → ()

Restores a deleted document to a file.

Parameter

sessionid	(valid) session number
docid	Identifier of a previously deleted document.
fileid	File that the document is to placed into.

Return values none

Exceptions

session number is invalid
document identifier is invalid
document is not deleted
file identifier is invalid
document or file do not exist or no permission

6.5.13 MoveDocumentToFile

MoveDocumentToFile(sessionid, docid, fromfileid, tofileid) → ()

Moves a document from one file into another.

Parameter

sessionid	(valid) session number
docid	Identifier of the document to move.
fromfileid	Identifier of the file the document is to be removed from.
tofileid	Identifier of the file the document is to inserted into.

Return values none

Exceptions

session number is invalid
document identifier is invalid
document is not part of the souce file
file identifier is invalid
document or file do not exist or no permission

6.5.14 SetFileMainDocument

SetFileMainDocument(sessionid, fileid, docid)

Declares a specific document to be the main document of its file. This document parameters are shown in the file area overview.

Parameter

sessionid	(valid) session number
fileid	Identifier of the file that contains the document.
docid	Identifier of the document to be the main document.

Return values none

Exceptions

session number is invalid
document identifier is invalid
file identifier is invalid
document is not part of that file
document or file do not exist or no permission

7 Distribution

The transport document distribution uses the document storage as well. Based on the transport document number document files are associated to documents. In order to modify a document a subscriber has to have permissions to that document. A subscriber gains permissions when

- It imports a version of that document
- receives the document by distribution
- a share with distribution for the correlating file document exists

7.1 ImportTransportDocument

ImportTransportDocument (sessionid, options, binarydata) → (docid)

Import a transport document to the distribution. The system automatically searches for a matching document that then updated with the new document file if applicable. Only the transport documents the subscriber has access to are considered. If no matching document is found a new document is created.

Parameter	
sessionid	(valid) session number
options	Additional options to control the distribution and ten-day reporting
binarydata	Binary data containing the document file.

Return values	
docid	Document identifier of a new or existing document

Exceptions	
	session number is invalid

Values for ‘options’:

options.distribution
false: The document version that will be created will not actively be distributed to other subscribers. It will be visible to everybody who already has permission though.

true or empty: the document will be distributed

options.dekadenmeldung
false: The document version that will be created will not take part in the automatic ten-day reporting.

true or empty: The document version will be processed by the ten-day reporting if it meets the requirements.

7.2 ImportTransportDocumentFromInbox

ImportTransportDocumentFromInbox(sessionid, options, docidsrc) → (docid)

This functions works like ImportTransportDocument(), but takes a document from the inbox as source document. The source document remains in the inbox.

Parameter

sessionid	(valid) session number
options	see ImportTransportDocument
docidsrc	Document identifier, has to identify a document in the inbox.

Return values

docid	see ImportTransportDocument

Exceptions

session number is invalid
docidsrc is invalid

7.3 GetNextTransportDocument

GetNextTransportDocument(sessionid) → (docid, logid)

Returns information about the next document that has been distributed to the calling subscriber via transport document distribution and marks this entry as delivered to the subscriber. Further calls of this function will not return the same entry.

Parameter

sessionid	(valid) session number

Return values

docid	A document identifier. Empty if there is no (further) document distribution the subscriber has to be informed of.
logid	Protocol entry identifier that allows to determine the distributed document version. If a document has been delivered multiple times the versions can be distinguished by this value. logid may be used in GetCurrentViewOfDocuments() as lastident parameter. This value is empty if there is no (further) document distribution the subscriber has to be informed of.

Exceptions

session number is invalid
too many consequent calls

This function combined with ImportTransportDocument() control the distribution.

Calling this function acknowledges that the caller now has knowledge of the delivery.

If the function is called too often without any result the system may reject the request with an appropriate error message. The timer will then be reset. Timers are based per account, not per subscriber.

The functions main usage is the automated processing of transport documents via (background) processes.

7.4 GetTransportDocumentByIdent

GetTransportDocumentByIdent(sessionid, idname, idvalue) → (docid)

Determines the document identifier by comparison of the typical functional identifier of a transport document. If a caller knows e.g. the transport document number he can determine wether or not the document is in the distribution. If it is the document identifier is returned for further operations.

Parameter

sessionid	(valid) session number
idname	Determines which functional identifier to look for.
idvalue	Comparison value.

Return values

docid	Document identifier, if the document exists.

Exceptions

	session number is invalid

Not finding a document for the given searching parameters is not an error. An empty value will be returned instead.

idname	Meaning
tpnr	Transport document number

7.5 PeekNextTransportDocument

PeekNextTransportDocument(sessionid) → (docid, logid)

This function returns the same result as GetTransportDocument(), but it does not remove the document from the list. In order to remove it a call to TransportDocumentReceived() is necessary.

Parameter

sessionid	(valid) session number

Return values

docid	see GetNextTransportDocument
logid	see GetNextTransportDocument

Exceptions

	session number is invalid

7.6 TransportDocumentReceived

TransportDocumentReceived(sessionid, docid, logid) → ()

This function confirms that the user has received the document properly. The parameters have been returned by PeekNextTransportDocument() before. Only after calling TransportDocumentReceived() the next document in the list of unreceived documents will be available through PeekNextTransportDocument().

Parameter	
sessionid	(valid) session number
docid	Document identifier, see PeekNextTransportDocument
logid	Protocol identifier, see PeekNextTransportDocument

Return values none

Exceptions

session number is invalid
Document identifier is invalid
Protocol identifier is invalid

8 Document functions

Remark: There are no functions to just create a document. A document always has a relation (e.g. to a file).

8.1 GetCurrentViewOfDocument

GetCurrentViewOfDocument(sessionid, docids, lastident) → (docviews)

Acts the same way as GetCurrentViewOfFile(), but works on documents.

Parameter

sessionid	(valid) session number
docids	List of document identifiers that a view is to be created for.
lastident	Comparative value that is used to limit the used data. All entries from the file with an ident <= lastident will be considered. This parameter may be left empty, all historical data is used then.

Return values

docviews	List of views of documents (see 4.7)

Exceptions

session number is invalid

8.2 GetHistoryViewOfDocuments

GetHistoryViewOfDocuments(sessionid, docids, lastident) → (docviews)

Acts the same way as GetHistoryViewOfFiles(), but works on documents.

Parameter

sessionid	(valid) session number
docids	List of document identifiers that a historical view is to be created for.
lastident	Comparative value that is used to limit the used data. All entries from the file with an ident <= lastident will be considered. This parameter may be left empty, all historical data is used then.

Return values

docviews	List of views of documents (see 3.9)

Exceptions

session number is invalid

8.3 ExportCurrentDocument

ExportCurrentDocument(sessionid, docid, lastident) → (binarydata)

Forms the current view up to lastident from the document protocol and returns the resulting data as a binary block of data. If lastident is left empty all historical data will be used.

Parameter

sessionid	(valid) session number
docid	Document identifier
lastident	Comparative value that is used to limit the used data. All entries from the file with an ident <= lastident will be considered. This parameter may be left empty, all historical data is used then.

Return values

binarydata	Binary representation of the current document. Depending on format and use of the document this may be the latest copy or the result of the processing of multiple document files.

Exceptions

session number is invalid

The export format depends on the format the document is held in. If the document is in IKS format no signatures will be exported.

8.4 ExportHistoryDocument

ExportHistoryDocument(sessionid, docid, logident) → (binarydata)

Exports the document file as it was given during the import process. Each document file is identified by a logident that has to be given as a parameter.

Parameter

sessionid	(valid) session number
docid	Document identifier
logident	The protocol identifier for a specific document version.

Return values

binarydata	The imported document file in binary form.

Exceptions

session number is invalid

8.5 GetDocumentParticipant

GetDocumentParticipant(sessionid, docid, logid, fieldname) → (participant)

Determines the data of the participating parties of a document. This data contains the postal address and the numbers assigned by the authorities as far as they are known.

Parameter

sessionid	(valid) session number
docid	Document identifier
logid	The protocol identifier of the specific document version.
fieldname	Of a protocol entry is associated with multiple participating parties this

parameter allows to select the relevant party. If left empty the party that has caused the entry will be returned.

Return values

participant Structured data containing name, postal address and registration number.

Exceptions

session number is invalid
document ident is invalid
parameter is invalid

8.6 SetDocumentParameter

SetDocumentParameter(sessionid, docid, pub, parname, parvalue) → ()

Sets a parameter for the document the same ways that parameters can be set for files. Specific predefined parameter names allow changing the facility numbers without changing the document itself. Those numbers are part of the user management and influence the access permissions for accounts.

Parameter

sessionid	(valid) session number
docid	Document identifier
pub	Boolean. Determines whether or not the parameter may be seen publically.
parname	Name of the parameter; case-sensitive
parvalue	New value of the parameter

Return values none

Exceptions

session number is invalid
document ident is invalid

The following parameter names are predefined:

- erzeuger_betriebsnr
- befoerderer_betriebsnr
- entsorger_betriebsnr
- befoerderer3_betriebsnr
- befoerderer2_betriebsnr
- behoerde_betriebsnr

8.7 GetDocumentParameter

GetDocumentParameter(sessionid, docid, pub, parname) → (parvalue)

Request a documents parameter value.

Parameter

sessionid	(valid) session number
docid	Document identifier
pub	Determines whether public or only the subscribers own parameters are to be requested.
parname	Name of the parameter; case-sensitive

Return values

parvalue	Value of the parameter or empty if the parameter is not defined.

Exceptions

session number is invalid
document ident is invalid

8.8 GetDocumentsByParameter

GetDocumentsByParameter(sessionid, pub, parname, parvalue) → (docids[])

Determine the documents that have a parameter of a given value.

Parameter

sessionid	(valid) session number
pub	Determines wether public or only the subscribers own parameters are to be respected in the query.
parname	Name of the parameter.
parvalue	The comparison value.

Return values

docids	List of document identifiers.

Exceptions

session number is invalid

8.9 SendDocument

SendDocument(sessionid, docid, logid, receiver, options, comment) → ()

Sends a document to a ZEDAL- or ZKS participating party. The document will be placed into the recipient's inbox. That is especially true for ZEDAL-subscribers.

Parameter

sessionid	(valid) session number
docid	Identifier of the document to be sent.

logid	The protocol identifier (e.g. from the historical view) or empty. If empty the latest document version will be sent.
receiver	Determines the recipient (see general section).
options	Options for sending the document.
comment	An optional comment that will be saves with the resulting protocol entry.

Return values none

Exceptions

session number is invalid
docid does not refer a document or no permission
logid does not refer to a document version
recipient is invalid

8.10 DeliverDocument

DeliverDocument(sessionid, docid, fileid, receiver, options, comment) → ()

This function allows to grant access some recipient access to a document. If the recipient is a ZEDAL-subscriber that subscriber receives permissions on the document. For ZKS-parties the document is sent (→ SendDocument).

Parameter

sessionid	(valid) session number
docid	Identifier of the document to be delivered.
receiver	Determines the recipient (see general section).
options	Options for delivering the document.
comment	An optional comment that will be saves with the resulting protocol entry.

Return values none

Exceptions

session number is invalid
docid does not refer a document or no permission
recipient is invalid

8.11 QueryDeliveryLog

QueryDeliveryLog(sessionid, querydef) → (count)

Initiates a query of the delivery log and returns the number of hits. The result may be fetched using FetchQueryResult().

Parameter

sessionid	(valid) session number
querydef	Definiton of the query criteria (see below)

Return values

count	Number of total rows.

Exceptions

session number is invalid

8.12 GetDeliveryDetails

GetDeliveryDetails(sessionid, docid, logid) → (resultlist)

This function returns detailed data for a specific protocol entry of the delivery history. Depending on the concrete transportation method different details are available (e.g. ZKS acknowledgements only if the document was sent via ZKS).

Parameter

sessionid	(valid) session number
docid	Document identifier.
logid	Protocol identifier.

Return values

resultlist	List of details, each as parameter name and value.

Exceptions

session number is invalid
document not available
protocol entry number is invalid

9 Inbox

9.1 ListInbox

ListInbox(sessionid, condition) → (docids[])

Returns a list of the documents in the inbox. The selection may be limited by the condition parameter. When no condition parameter is given all unread documents without any time based constrains will be listed.

Only documents the user has access to will be listed.

Parameter

sessionid	(valid) session number
condition	Conditions for the selection. May be omitted.

Return values

docids	List of document identifiers for documents in the inbox.

Exceptions

session number is invalid

9.2 MarkDocumentsAsReadInInbox

MarkDocumentsAsReadInInbox(sessionid, docids[]) → ()

Marks the given documents as read so they no longer show up as unread in the inbox. The documents remain in the inbox.

Parameter

sessionid	(valid) session number
docids	List of document identifiers for the documents that are to be set as “read”. Invalid document identifiers will be ignored.

Return values none

Exceptions

session number is invalid

9.3 RemoveDocumentsFromInbox

RemoveDocumentsFromInbox(sessionid, docids[]) → ()

Only after calling this function documents are removed from the inbox.

Parameter

sessionid	(valid) session number
docids	List of document identifiers for the documents that are to be removed. Invalid document identifiers will be ignored.

Return values none

Exceptions

session number is invalid

9.4 ApplyInboxFilter

ApplyInboxFilter(sessionid, onlyUnread) → ()

Apply all automatic filters configured in the ZEDAL portal.

Parameter

sessionid	(valid) session number
onlyUnread	Bool-Value. true, if only unread documents are to be processed. false, if all non-deleted documents are to be processed.

Return values none

Exceptions

session number is invalid

10 Attachments

If file attachments can or should not be saved inside of the document itself they may be referenced by hash values. The hash values are placed inside of the document. Along with that the actual attachment file is transferred to the database and may later be retrieved by the hash value in the document.

10.1 UploadAttachment

UploadAttachment(sessionid, filename, binarydata) → (hashvalue)

This function allows to save attachments in the system that are not to be transferred in the document itself.

Parameter

sessionid	(valid) session number
filename	File name.
binarydata	Binary data content of the attachment.

Return values

hashvalue	Hash value of the attachment content.

Exceptions

session number is invalid

10.2 DownloadAttachment

DownloadAttachment(sessionid, hashvalue) → (filename, binarydata)

Restores an attachment. The attachment is referenced by its hash value. The attachments and their hash values are shown in the current view of a document and may be used here. The download is only permitted if the subscriber has access to the document referring to the attachment.

Parameter

sessionid	(valid) session number
hashvalue	Hash value that identifies the attachment.

Return values

filename	File name.
binarydata	Binary data content of the attachment.

Exceptions

session number is invalid

11 Master data functions

All master data functions require the user to give the name of the respective table. The following table names are valid:

tablename	Bedeutung
SD_ADRESSEN	Address master data
SD_VORSCHLAG	Proposals for address master data
SD_BETRIEB	Facility master data
SD_BETRIEB_VORSCHLAG	Proposals for facility master data
SD_UNTERNEHMEN	Company master data
SD_AK_BASEL	Waste catalogue Basel
SD_AK_EC	Waste catalogue EC
SD_AK_OECD	Waste catalogue OECD

All functions limit access to the area visible to the account identified by the session.

11.1 DBInsert

DBInsert(sessionid, tablename, tabledata) → ()

This function allows to insert one or multiple rows into a table.

Parameter

sessionid	(valid) session number
tablename	Name of the table.
tabledata	The rows to insert.

Return values none

Exceptions

session number is invalid
table is invalid
column is unknown
cell value is invalid
not all required fields transferred
action not allowed

Depending on the table the given primary key may not be respected. Not all tables support insertion.

For tables with an associated proposal table rows are inserted into the proposal tables instead if the four-eyes-principle feature is enabled. Those entries may then accepted using a clearance (see DBFunction).

11.2 DBSynchronize

DBSynchronize(sessionid, tablename, tabledata, options) → ()

This function allows to synchronize complete tables. Whether a row has to be updated or inserted is determined using the primary key of the table. If a row with the same primary key already exists the row is updated, otherwise it is inserted. Depending on the table the given primary key may not be used. Not all tables allow synchronization.

Parameter

sessionid	(valid) session number
tablename	Name of the table.
tabledata	The rows to synchronized.
options	Additional synchronization options.

Return values none

Exceptions

session number is invalid
table is invalid
column is unknown
cell value is invalid
not all required fields transferred
action not allowed

Values for options:

options.deleteSurplus
false (default): After synchronization only those rows are left that have been in the sync request. All other rows are deleted.

true or empty: Rows that are not part of tabledata will be kept.

For tables with an associated proposal table rows are inserted into the proposal tables instead if the four-eyes-principle feature is enabled. Those entries may then accepted using a clearance (see DBFunction).

11.3 DBLookup

DBLookup(sessionid, tablename, key) → (tablelist)

Queries a complete data set including all attached data (changelog, …). There is an entry in tablelist for each table that associated with this row.

The tables key will be given in form of a parameter list. Parameter names represent column names, the values contain the cell values.

Parameter

sessionid	(valid) session number

tablename	Name of the table.
key	Primary key of the row in question.

Return values

tablelist	A list of tables.

Exceptions

session number is invalid
table is invalid
column is unknown
not all required fields transferred

It is no error if no row is found for the given key. The returned tables are empty in that case.

11.4 DBSelect

DBSelect(sessionid, tablename, fieldlist, whereclause, sort) → (table)

Queries a set of rows.

Parameter

sessionid	(valid) session number
tablename	Name of the table.
fieldlist	List of the field names that are to be returned.
whereclause	Restricting conditions in form of a binary tree.
sort	Sort order of the results.

Return values

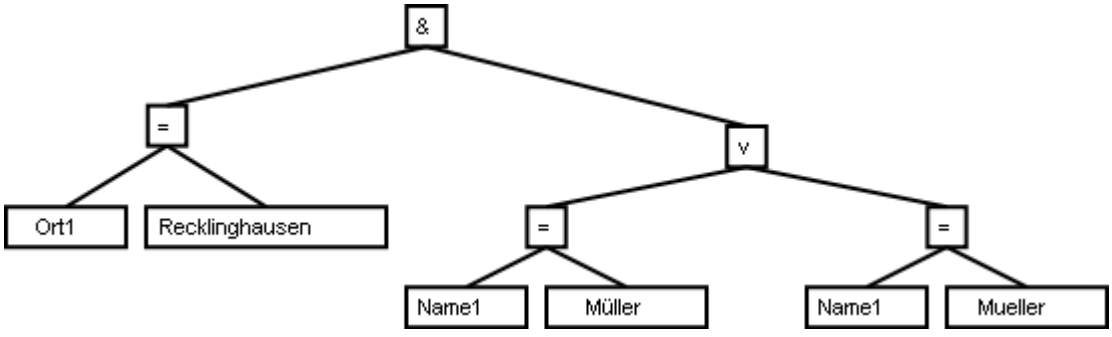
tablelist	A list of tables.

Exceptions

session number is invalid
table is invalid
column is unknown
whereclause is invalid

The where clause allows to restrict the returned result set. It is transferred as a binary tree.

Example: Ort1 = Recklinghausen AND (Name1 = Müller OR Name2 = Mueller)



For each node type specific contents are allowed:

Node type	**Valid content**
Logical	AND OR
Comparison	< <= >= > = <>
Function	LIKE NVL
Field name	All valid field names of the table
Field value	All valid values for this field

Functions are evaluated as below:

LIKE(Column name; Comparison content)
NVL(Column name; Content if null)

Sorting is transferred as a list of parameters. Column names are listed as parameter names, parameter values contain the sorting order (key values *ASC* or *DESC*).

It is not an error if no row is found. The returned table is empty in that case.

11.5 DBDelete

DBDelete(sessionid, tablename, key) → ()

Deletes the row with the given primary key from the table. For details about how to give the key see DBLookup().

Parameter	
sessionid	(valid) session number
tablename	Name of the table.
key	Primary key of the row to be deleted.

Return values none

Exceptions

session number is invalid
table is invalid
column is unknown
not all required fields transferred
action not allowed

It is not an error if the specified row is not found. Deletion is not available for all tables.

11.6 DBDescribe

DBDescribe(sessionid, tablename) → (tableinfo, keyinfo)

Describes a table's structure. A list of all columns and their type and a list of the primary key columns are returned.

Parameter	
sessionid	(valid) session number
tablename	Name of the table.
Return values	Columns with name and type and the key columns.
Exceptions	
	session number is invalid
	table is invalid

11.7 DBFunction

DBFunction(sessionid, tablename, key, functionname, parameterlist) → ()

Executes a function for a row of a table. The available functions differ for each table.

Parameter	
sessionid	(valid) session number
tablename	Name of the table.
key	Primary key of the row the function is to be executed on.
functionname	Name of the function to execute.
parameterlist	The function's parameters.
Return values	none
Exceptions	
	session number is invalid
	table is invalid
	column is unknown
	not all required fields transferred
	function is invalid

For details about how to give the key see DBLookup().

The following functions are available:

Table *SD_VORSCHLAG*:
release – Accepts the proposal with the current account. No parameters are defined.

Table *SD_BETRIEB_VORSCHLAG*:
release – Accepts the proposal with the current account. No parameters are defined.

12 Other functions

12.1 FetchDocumentNumbers

FetchDocumentNumbers(sessionid, range, count) → (list)

Each transport document has to carry a unique number.

For BMU documents this number has to be assigned by the ZKS according to BMU specification. In order to fulfill this requirement ZEDAL 2 fetches ranges of numbers from ZKS in a centralized way for all subscribers and provides these numbers through this function. One has to assume that each subscriber may only use a limited amount of numbers depending on their document volume.

Parameter

sessionid	(valid) session number
range	The range the numbers are to be taken from.
count	Amount of numbers to be taken.

Return values

list	List of transport document numbers.

Exceptions

session number is invalid
range is invalid
out of numbers
requested amount of numbers is invalid (0 <= amount <= 1000)

The range parameter allows to select from the different number ranges.

Only up to 1000 numbers may be fetched in a single call. Values < 0 or > 1000 will be answered with an appropriate error message.

The following ranges are defined:

range	Meaning
BMU.BGS	Numbers for BMU-Begleitschein
BMU.UNS	Numbers for BMU-Übernahmeschein
BMU.NWL	Numbers for BMU-Nachweisliste
TFS.Notification|<country>	Numbers for TFS notifications. See remarks
TFS.WasteMovement|<country>|<notification	Numbers for TFS waste movement documents.

The TFS ranges require further information about the usage. The caller has to replace the parts in right angle brackets (including the brackets).

Examples:

```
TFS.Notification|DE
```

to receive a notification number for Germany.

```
TFS.WasteMovement|DE|1234 DE / 1234567
```

to receive a number for a waste movement document for the notification “1234 DE / 1234567”. The <country> part is required, but currently not used.

Note that there is no central organization that assigns waste movement numbers. The numbers this function delivers are not guaranteed to be unique as they are not reserved for the caller. Two subsequent calls to this function with equal parameters will result in the same return values as long as the resulting transport documents have not been imported.

12.2 GetGlobalParameter

GetGlobalParameter(sessionid, name) → (value)

Requests the value for a subscriber based global parameter.

Parameter	
sessionid	(valid) session number
name	Name of the parameter.
Return values	
value	Value of the parameter.
Exceptions	
	session number is invalid

12.3 SetGlobalParameter

SetGlobalParameter(sessionid, name, value) → ()

Sets a subscriber based global value. May be used in order to e.g. hold counter values independent from single machines.

Parameter	
sessionid	(valid) session number
name	Name of the parameter.
value	New value of the parameter.
Return values	none
Exceptions	
	session number is invalid

12.4 echo

echo(inputstring) → (echo)

Provides a simple means to test connectivity. Returns the same string value that was in the request. String length is limited.

Parameter

inputstring	Any input string, limited length.

Return values

echo	The same value as the input string.

This functions provides a simple means to test basic access to the webservices even without having valid login data.

12.5 RequestZksAddress

RequestZksAddress(sessionid, role, number) → (participant)

Requests data for a participating party from the registration service of ZKS.

Parameter

sessionid	(valid) session number
role	Role of the participating party.
number	Number of the participating party as assigned by the authorities.

Return values

participant	Structure containing name, address and registration number.

Exceptions

	session number is invalid

12.6 FetchQueryResult

FetchQueryResult(sessionid) → (resulttable)

Returns the resulting rows of a previously started query. The resulting lines are only available directly after the query. Upon calling a different function results will be cleared.

Parameter

sessionid	(valid) session number

Return values

resulttable	Resulting table

Exceptions

	session number is invalid
	no active query found

13 Document template functions

13.1 CreateTransportTemplate

CreateTransportTemplate(sessionid, templateno, binarydata, options) → (templateno)

Imports a document template for the transport document distribution. If the template number in the request is empty a new template will be created. The number of the newly created template will then be returned.

Parameter	
sessionid	(valid) session number
templateno	Template number, only when editing an existing template.
binarydata	Binary data containing the template document file.
options	Further options

Return values Template number if a new template was created.

Exceptions

session number is invalid
the format of the data was not recognized
options are invalid
insufficient permissions

The following options are available:

description: A descriptive text for the template.
count: Number of allowed transport document generations from this template.
validfrom: From this date is it allowed to use this template.
validto: To this date is it allowed to use this template.

13.2 DeleteTransportTemplate

DeleteTransportTemplate(sessionid, templateno) → ()

Removes the transport document template with the given number.

Parameter	
sessionid	(valid) session number
templateno	Template number to delete

Return values none

Exceptions

session number is invalid
insufficient permissions

13.3 ExportTransportTemplate

ExportTransportTemplate(sessionid, templateno) → (binarydata)

Exports the transport document template with the given number as binary data.

Parameter

sessionid	(valid) session number
templateno	Template number

Return values Binary data containing the template document file.

Exceptions

session number is invalid
insufficient permissions

13.4 GetTransportTemplateCount

GetTransportTemplateCount(sessionid) → (paramlist)

Returns a list of template numbers and the number of allowed transport document generations.

Parameter

sessionid	(valid) session number

Return values List of all template numbers and the associated number of allowed transport document generations.

Exceptions

session number is invalid
insufficient permissions

13.5 GetObtainedTPNos

GetObtainedTPNos(sessionid) → (stringlist)

Returns a list of transport document numbers that have been created from a transport template from a specific date on. Only available for documents of the transport document distribution.

Parameter

sessionid	(valid) session number
templateno	Template number
createdsince	Begin of the time frame of interest.

Return values List of transport document numbers.

Exceptions

session number is invalid
insufficient permissions

13.6 CreateFileareaTemplate

CreateFileareaTemplate(sessionid, templateno, binarydata, options) → (templateno)

Imports a document template for the file area. If the template number in the request is empty a new template will be created. The number of the newly created template will then be returned.

Parameter	
sessionid	(valid) session number
templateno	Template number, only when editing an existing template.
binarydata	Binary data containing the template document file.
options	Further options.

Return values Template number if a new template was created.

Exceptions

session number is invalid
the format of the data was not recognized
options are invalid
insufficient permissions

The following options are available:

description: A descriptive text for the template.
count: Number of allowed transport document generations from this template.
validfrom: From this date is it allowed to use this template.
validto: To this date is it allowed to use this template.

13.7 DeleteFileareaTemplate

DeleteFileareaTemplate(sessionid, templateno) → ()

Removes the file area template with the given number.

Parameter	
sessionid	(valid) session number
templateno	Template number

Return values none

Exceptions

session number is invalid
insufficient permissions

13.8 ExportFileareaTemplate

ExportFileareaTemplate(sessionid, templateno) → (binarydata)

Exports the file area document template with the given number as binary data.

Parameter

sessionid	(valid) session number
templateno	Template number

Return values Binary data containing the template document file.

Exceptions

session number is invalid
insufficient permissions

13.9 GetFileareaTemplateCount

GetFileareaTemplateCount(sessionid) → (paramlist)

Returns a list of template numbers and the number of allowed file area document generations.

Parameter

sessionid	(valid) session number

Return values Template number

Exceptions

session number is invalid
insufficient permissions

14 Usage examples

As implementations of the interface may vary for different programming languages the following examples are given as pseudo code.

14.1 Creating a file, importing a document, submitting it to a different subscriber

```
session = OpenSession("OP01", "", "", pass);
fileid = CreateNewFile(session, "Akte 1");
binary_data = LoadDocument(filename);
// Remark: binarydata will automatically be encoded as BASE64
// by the soap implementation.
docid = ImportDocumentToFile(session, fileid, 2, binary_data);
OpenSubmission(session, fileid, "", "OP02", "", "For editing", "");
CloseSession (session);
```

14.2 Requesting and editing of a submitted document, finishing the submission

```
session = OpenSession("OP02", "", "", pass);
fileids = GetNewSubmissions(session);
for fileid in fileids
{
    HoldSubmission(session, fileid, "Editing");
    file_view = GetCurrentViewOfFile(session, fileid, "")
    for file_logentry in file_view.log
    {
       if file_logentry.type == document
       {
          docid = file_logentry.docid
          doc_view = GetCurrentViewOfDocument(session, docid, "");
          binary_data = ExportCurrentDocument(session, docid, "");
          StoreDocument(datei, binary_data);
          // ... edit the document ...
          binary_data = LoadDocument(filename);
          ImportToDocument(session, docid, binary_data);
       }
    }
    CompleteSubmission(session, fileid, "Editing completed");
}
CloseSession(session);
```

ERP Integration Tools

BmuXmlArtist (including eTFS)

Last revision: 22.06.2017

64 pages including the front page

Document No. 155.388.957

Technical documentation

BmuXmlArtist - Index

1 Use of the BmuXmlArtist

BmuXmlArtist enables the processing of XML documents in compliance with the BMU and EUDIN specifications without having to engage in the topic of XML itself. Using a field list makes it possible to provide or handle data.

1.1 Range of features

The product covers the following features:

- creates a XML document out of a field list according to the EUDIN specification
- can conversely create a field list out of a XML document
- the following international document types are available:
    - Waste movement document (international) starting at version 1.0.0

      functions with  are available for ZEDAL International
        - Waste notification
        - Waste movement document
        - Annex VII document
- The following national (German) document types are supported:
    - Begleitschein (BGSDokument)
    - Übernahmeschein (UNSDokument)
    - Entsorgungsnachweis (ENSNDokument)
        - Embedded ABANDA DA
    - Freistellungsantrag (FRDokument)
    - Ergänzendes Formblatt (EGFDokument)
    - Landesrechtlicher Bescheid (AGSBescheid)
    - Mitteilung (Mitteilung)
    - Nachweisliste (Nachweisliste)
    - Deklarationsanalyse (DADokument)
- support of signed and unsigned file attachments
- support of free XML structures
- available views on the document:
    - current view
    - historical view
    - layer view
- exchange of data by local codepage or by UTF-8
- logging available for error analysis

1.2 Scope of delivery (changed 0.9.3)

BmuXmlArtist comes as a setup file for Windows that creates the following files and folders upon installation:

- Field lists (one file per document type, for example BGSDocument.csv)
- Docs
  - Manual.pdf
- Include
  - bmu_interface.h
  - bmu_interface_defines.h
- bmudll.dll

Schema files and Xerces are part of the DLL.

1.3 Initialisation (changed 0.9.3)

To initialize the BmuXmlArtist you need to have a customer ID and a valid license key. The following call shows the initialization:

```
rc = BMU_Initialize(licensee, licensekey, id);
```

Id is a reference parameter that is set to a value unequal to 0 upon successful execution. When executing any further functions this has to be specified as the first parameter.

If the license information is wrong the function delivers *BMU_LICENSE_ERROR*. Otherwise, *BMU_OK*.

Recommendation: before using the DLL always check with

```
rc = BMU_GetVersion(buffer, len);
if (strcmp(buffer, "X.Y.Z") != 0)
  throw "wrong version of BmuXmlArtist used!"
```

to make sure that the combination of program and BmuXmlArtist are compatible. Otherwise this can lead to undefined behaviour and crashes.

1.4 Buffer for results

Some functions have to provide memory space to the DLL that will be used for the result of processing. Should the memory space not suffice the function will report *BMU_BUFFERLENGTH_INSUFFICIENT*. The actually needed buffer size can be requested via *BMU_GetBufferSize* (including the #0 at the end of the string!)

This is required because it is impossible to preliminarily determine how large a XML file that was generated by a field list will become.

Functions added since version 0.9 use another method. They provide the memory space for the results by themselves. But those buffers are elusive, meaning the contents will be overwritten at the next call of a DLL function. You can identify those buffers in the function signatures with *char***.*

1.5 Lists as return values

Some functions like *GetLayers* or *GetNextLayer* return lists. Those lists are returned as a string which elements are always separated by a TAB “\t”.

Example:
GetLayer on a BGSDocument could return the following:

```
„BGSBEFLayer\tBGSERZLayer\tBGSVorlageLayer“
```

1.6 Character encoding

Every I/O operation can be done in the local codepage as well as UTF-8 encoded. For the former, please note that there is a conversion of the characters between the local codepage and Unicode. Thus, characters that can’t be shown in the codepage are lost at back conversion. The methods for UTF-8 have a corresponding suffix in the name.

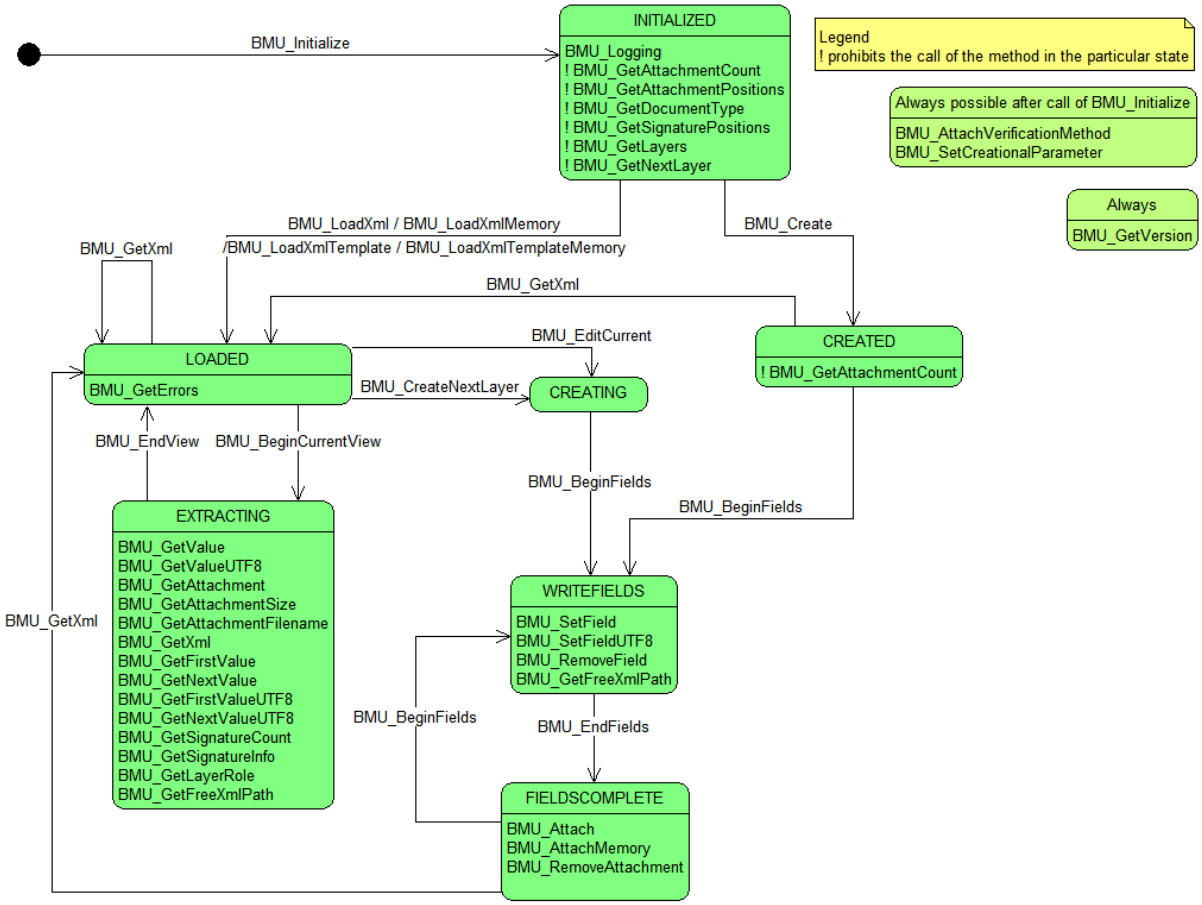
1.7 Layer concept

The national documents support different layers, which contain the information provided by the different participants. The BmuXmlArtist automatically computes the differences between these layers to write only the informationen that have changed from one layer to the next layer.

1.8 Status concept

The DLL internally uses a finite state machine to minimize misuse of the DLL. Some methods are only allowed in specific states. The following finite state diagram clarifies the function of the DLL.

Diagram of the internal states
describes the possible transitions between states and the permitted commands by state



1.8.1 Diagram explanation

The functions within the boxes below the horizontal line can only be called in the corresponding state, otherwise they give the return value *BMU_WRONG_STATE*.

The functions at the arrows lead to the corresponding state transitions.

2 Functions

This chapter describes the functions and their parameters and return values.

All functions return an integer to keep record of the successful call. BMU_OK means that no errors occured.

2.1 Initializing

2.1.1 BMU_Initialize (changed 0.9.3) +ZI

BMU_Initialize(const char* licensee, const char* licensekey,
const char* schemalocation, unsigned int& id)

Initializes BmuXmlArtist and checks whether a valid license key was used. Additionally, an ID is generated that can be used in every subsequent call to identify each instance in multihreading operation. BmuXmlArtist can internally handle an unlimited number of instances. Every instance should be released with BMU_Release after usage.

licensee	Identification of licensee
licensekey	License key
schemalocation	Location of schema files (Unix only)
id	Contains instance ID after call

Return value:

BMU_LICENSE_ERROR	Wrong license
BMU_OK	Successful call
BMU_NULL	Unable to create instance

2.1.2 BMU_Release +ZI

BMU_Release(unsigned int instanceid)

Releases all memory areas connected to this instance.

instanceid	Instance ID

Return value:

BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.2 Creation functions

2.2.1 BMU_LoadXml +ZI

BMU_LoadXml(unsigned int instanceid, const char* filename)

Loads the given file and checks if the schemas adhere to the BMU specification.

instanceid	Instance ID
filename	File name

Return value:

BMU_FILE_NOT_FOUND	Unable to open file
BMU_EXCEPTION_OCCURED	Error processing XML
BMU_ERROR	file does not match the schema
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found
BMU_VERSION_NOT_SUPPORTED	The specification version contained in the file is not supported. This return value is also used for files that are not XML or use a different format.

2.2.2 BMU_LoadXmlMemory +ZI

BMU_LoadXmlMemory(unsigned int instanceid, const char* startaddress, unsigned long bytes)

Analogous to LoadXML, directly process a XML file from memory.

instanceid	Instance ID
startaddress	Start-adress of memory block
bytes	Lenght of the block in bytes

Return value:

BMU_ERROR	File does not match the schema
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found
BMU_VERSION_NOT_SUPPORTED	The specification version contained in the file is not supported. This return value is also used for files that are not XML or use a different format.

2.2.3 BMU_Create +ZI

BMU_Create(unsigned int instanceid, const char* documenttype, bool vorlagelayer)

Creates an empty BMU file.

instanceid	Instance ID
documenttype	one of the supported document types

vorlagelayer	Boolean; create Vorlagenlayer or start with Basislayer. Will be ignored for international documents

Return value:

BMU_UNSUPPORTED_DOCTYPE	given document type not supported.
BMU_ERROR	unable to load schema file
BMU_EXCEPTION_OCCURED	errors occured upon creation
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.2.4 BMU_CreateNextLayer (changed in 0.9.2) +ZI

BMU_CreateNextLayer(unsigned int instanceid, const char* hint = 0);

Creates the next layer in line. This is not unambiguous in some situations and therefore can't be determined automatically. In this case the corresponding layer has to be specified with *hint*.

For WasteMovement documents the value of hint has a different meaning. If it's "true" the new layer will get the next type in sequence. It it's "false", a new layer with the same type as the last one will be created.

instanceid	Instance ID
hint	specifies the next layer (optional)

Return value:

BMU_OUT_OF_RANGE	The follow-up layer can't be clearly determined if no hint was given. If a hint was given then it contains an invalid follow-up layer or you tried to create a 4th Befördererlayer in the BGSDocument.
BMU_ERROR	You tried to add a layer to a newly created document. A new document always contains a Vorlagenlayer or Basislayer.
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.2.5 BMU_EditCurrent (since 0.9.2) +ZI

BMU_EditCurrent(unsigned int instanceid,char **layerid);

Edit existing layer. If using an unlayered document this tries to edit the document. For layered documents this returns the layer ID of the topmost layer. If the pointer is 0 then this value is not set.

instanceid	Instance ID
layerid	contains name of the current layer after the call, if a non-null pointer to a pointer is provided.

Return value:

BMU_CONTENT_SIGNED editing not allowed, a signature protects the contents.

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.2.6 BMU_QueryField (since 0.9.2)

BMU_QueryField(unsigned int instanceid, const char *pfad);

Query the readonly state of an element. It accounts for signatures and elements that are specific to some layers.

instanceid	Instance ID
path	field qualifier

Return value:

BMU_CONTENT_SIGNED	editing not allowed, a signature protects the contents
BMU_ERROR	Field is not known or illegal on the current layer
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.2.7 BMU_GetXml 

BMU_GetXml(unsigned int instanceid, char* buffer, unsigned long length);

Initiates the creation of the XML. All delivered data is integrated into the XML and the result is copied to the passed buffer.

instanceid	Instance ID
buffer	Buffer
length	Buffer length

Return value:

BMU_ERROR	Errors occured upon creation. The log contains further information (see BMU_Logging)
BMU_EXCEPTION_OCCURED	Error occured in XML processing
BMU_BUFFERLENGTH_INSUFFICIENT	Buffer length insufficient
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.3 Using templates to create documents

If a document is used to create further documents remember that it is important to change the IDs of the source document upon creation of the new document. Otherwise, you will run into problems when merging the documents inside a register excerpt (full excerpt only), generating schema errors because of duplicate IDs.

If you try to run this operation on a signed document it will result in an error. It is not allowed to copy signed documents, because the change of a signature ID will break the signature.

2.3.1 BMU_LoadXmlTemplate

BMU_LoadXmlTemplate(unsigned int threadid, const char* filename)

Loads the given file analogous to LoadXml and verifies whether the schema is in accordance to the BMU specification. Additionally the IDs are newly generated. The role abbreviation, which is a part of these IDs, is left unchanged.

threadid	Instance ID
filename	File name

Return values:

BMU_FILE_NOT_FOUND	Unable to open file
BMU_VERSION_NOT_SUPPORTED	The version of the specification contained in the file is not supported. This is also returned for files that are not XML or another format
BMU_EXCEPTION_OCCURED	Error processing XML
BMU_ERROR	The loaded file does not adhere to the schema
BMU_CONTENT_SIGNED	copy impossible due to signatures
BMU_WRONG_STATE	call not allowed (see diagram)
BMU_OK	successful call
BMU_UNKNOWN_ID	no instance with this ID known

2.3.2 BMU_LoadXmlTemplateMemory

BMU_LoadXmlTemplateMemory(unsigned int threadid, const char* startaddress,
unsigned long bytes)

Process an XML-file directly from memory area, analogously to LoadXmlTemplate.

threadid	Instance ID
startaddress	starting address of memory block
bytes	length of block in byte

Return values:

BMU_VERSION_NOT_SUPPORTED	The version of the specification contained in the file is not supported. This is also returned for files that are not XML or another format
BMU_ERROR	The loaded file does not adhere to the schema
BMU_CONTENT_SIGNED	copy impossible due to signatures
BMU_WRONG_STATE	call not allowed (see diagram)
BMU_OK	successful call
BMU_UNKNOWN_ID	no instance with this ID known

2.4 Attachments

2.4.1 BMU_AttachVerificationMethod

BMU_AttachVerificationMethod(unsigned int instanceid, const char* method = 0);

Specifies the checksum calculation method for all BMU_Attach calls.

instanceid	Instance ID
method	Method-ID (max 10 characters) , supported methods: ,' (blank, for no checksum calculation) ,SHA256'

Return value:

BMU_OUT_OF_RANGE	unsupported verification method
BMU_OK	Successful call

2.4.2 BMU_Attach (changed 0.9.2) +ZI

BMU_Attach(unsigned int instanceid, const char* position, int& index, const char* filename,
const char* mimetype)

Adds an attachment at the given position. If a method for checksum calculation was chose it will be applied and returns the value in *index* to make the just added attachment available in future operations.

instanceid	Instance ID
position	Position to add the attachment to
index	contains the logical index of the attachment after call (see chapter 7)
filename	name of the file containing the chosen attachment
mimetype	Mimetype

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	either the name of the position was mistyped or was not defined for the given document type
BMU_EMPTY_FILE_NOT_ALLOWED	empty files are not allowed
BMU_FILE_NOT_FOUND	given file was not found
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.3 BMU_AttachMemory (changed 0.9.2) +ZI

BMU_AttachMemory(unsigned int instanceid, const char* position, int& index,
const char* buffer, unsigned long length,
const char* filename, const char* mimetype)

Adds an attachment with files from a memory block and returns a value in *index* to make the just added attachment available for future operations.

instanceid	Instance ID
position	Position to add the attachment to (see chapter 7)
index	contains the logical index of the attachment after call
buffer	Start-Adress of the memory block
length	Length of the block and at the same time the length of the attachment entered to the XML.
filename	Name of the file. This is just an info added to the XML.
mimetype	Mimetype

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	either the name of the position was mistyped or was not defined for the given document type
BMU_EMPTY_FILE_NOT_ALLOWED	empty files are not allowed
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.4 BMU_GetAttachment +ZI

BMU_GetAttachment(unsigned int instanceid, const char* position, int index,
char* buffer, unsigned long length)

Returns the chosen attachment within a memory block. To determine the actual length of the attachment you can also use BMU_GetBufferSize().

instanceid	Instance ID
position	Position of the attachment (see chapter 7)
index	LfdNr. of the attachment, starting at 1.
buffer	Start adress of the memory block
length	memory block length

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	either the name of the position was mistyped or was not defined for the given document type
BMU_BUFFERLENGTH_INSUFFICIENT	Buffer size insufficient
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found
BMU_OUT_OF_RANGE	index is higher than the number of attachments
BMU_UNSUPPORTED_VERIFYMETHOD	The verification method specified by the XML is unknown. The attachment is still available, but no checksum calculation will be done.
BMU_CHECKSUM_WRONG	The calculated checksum does not match the checksum in the XML.

2.4.5 BMU_GetAttachmentCount +ZI

BMU_GetAttachmentCount(unsigned int instanceid, const char* position,
unsigned int& attachments)

Returns the number of attachments

instanceid	Instance ID
position	Position of the attachment (see chapter 7)
attachments	contains the number of attachments after call

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position was wrong or was it was used on the wrong document type
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.6 BMU_GetAttachmentFilename +ZI

BMU_GetAttachmentFilename(unsigned int instanceid, const char* position,
int attachmentno, char* buffer, unsigned long length)

Returns the file name of the chosen Attachment within a memory block.

instanceid	Instance ID
position	Position of the attachment (see chapter 7)
index	LfdNr. of the attachment
buffer	Memory block start-adress
length	Memory block length

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position was wrong or was it was used on the wrong document type
BMU_BUFFERLENGTH_INSUFFICIENT	Buffer size insufficient
BMU_OUT_OF_RANGE	index is higher than the number of attachments
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.7 BMU_GetAttachmentSize +ZI

BMU_GetAttachmentSize(unsigned int instanceid, const char* position, int attachmentno,
unsigned long& size)

Returns the original size of the chosen attachment

instanceid	Instance ID
position	Position of the attachment (see chapter 7)

index	LfdNr. of the attachment
size	contains the file size after call

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position was wrong or was it was not defined for the given document type
BMU_OUT_OF_RANGE	index is more than the number of attachments
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.8 BMU_GetAttachmentPositions (since 0.9.2) +ZI

BMU_GetAttachmentPositions(unsigned int instanceid, char** positionlist)

Returns a list of all positions that can have an attachment. Since some positions contain umlauts you may run into problems with systems expecting UTF-8. In that case please use BMU_GetAttachmentPositionsUTF8.

instanceid	Instance ID
positionlist	Pointer on a memory area that is filled by the BmuXmlArtist with a list of positions.

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.9 BMU_GetAttachmentPositionsUTF8 (since 0.9.6) +ZI

BMU_GetAttachmentPositions(unsigned int instanceid, char** positionlist)

Returns the list of all positions that may have an attachment.

instanceid	Instance ID
positionlist	Pointer on a memory area that is filled by the BmuXmlArtist with a list of positions.

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.10 BMU_RemoveAttachment (since 0.9.2) +ZI

BMU_RemoveAttachment(unsigned int instanceid, const char* position, int index)

Removes the given attachment.

instanceid	Instance ID
position	Position ID if the attachment
index	Lfdnr of the attachment

Return value:

BMU_OUT_OF_RANGE	*index* is higher than the number of existing attachments or position typed wrong.
BMU_CONTENT_SIGNED	The attachment is signed and cannot be removed
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found
BMU_ERROR	The attachment is not inside the current layer or in the area outside of the layer.

2.4.11 BMU_ReplaceAttachment (since 0.9.2)

BMU_ ReplaceAttachment (unsigned int instanceid, const char* position, int index, const char* filename, const char* mimetype)

Substitutes an unsigned attachment at the given position. If a method for checksum calculation was defined it will be employed.

instanceid	Instance ID
position	Position of the attachment (see chapter 7)
index	Index for the substitution
filename	file name containing the attachment
mimetype	Mimetype

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position was wrong or was it was not defined for the given document type
BMU_FILE_NOT_FOUND	The given file could not be opened
BMU_OUT_OF_RANGE	index is higher than the number of attachments
BMU_CONTENT_SIGNED	The attachment you tried to change is already enclosed by a signature.
BMU_ERROR	The attachment you try to substiute is not inside the current layer and can therefore not be substituted.
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.12 BMU_ReplaceAttachmentMemory (since 0.9.2)

BMU_ReplaceAttachmentMemory (unsigned int instanceid, const char* position, int index, const char* buffer, unsigned long length, const char* filename, const char* mimetype)

Substitutes an unsigned attachment at the given position with data from the given memory block. If a method for checksum calculation was set it will be employed.

instanceid	Instance ID
position	Position of the attachment (see chapter 7)
index	Index for the substitution
buffer	start-adress of the memory block
length	Length of the memory block as well as the attachment length entered in the XML
filename	file name. This is just placed as an information into the XML.
mimetype	Mimetype

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position was wrong or was it was not defined for the given document type
BMU_CONTENT_SIGNED	The attachment you tried to change is already enclosed by a signature.
BMU_ERROR	The attachment you try to substiute is not inside the current layer and can therefore not be substituted.
BMU_OUT_OF_RANGE	index is higher than the number of attachments
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.4.13 BMU_AttachmentInfo (since 0.9.2) +ZI

BMU_AttachmentInfo (unsigned int instanceid, const char* position, int index, char** infos)

Returns information to the given document (field list).The composition of the field list is covered in chapter *attachments.*

instanceid	Instance ID
position	Position of the attachment (see chapter 7)
index	Index for the substitution
infos	the DLL-managed memory area contains the information to this block after call.

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position was wrong or was it was not defined for the given document type
BMU_OUT_OF_RANGE	index is higher than the number of attachments
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)

BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5 Data transport

2.5.1 BMU_BeginFields +ZI

BMU_BeginFields(unsigned int instanceid)

With this command you set the DLL to the *WRITEFIELDS* state. After this the content of the document can be changed.

instanceid	Instance ID

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.2 BMU_EndFields +ZI

BMU_EndFields(unsigned int instanceid)

Stops the editing of the document.

instanceid	Instance ID

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.3 BMU_SetField +ZI

BMU_SetField(unsigned int instanceid, const char* path, const char* value)

Sets a value to a defined field.

instanceid	Instance ID
path	Field qualifier
value	value to set

Return value:

BMU_NULL	(TFS) field does not exist
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.4 BMU_SetFieldUTF8 +ZI

BMU_SetFieldUTF8(unsigned int instanceid, const unsigned char* path,
const unsigned char* value)

Sets a value to a defined field, expecting UTF-8 encoding.

instanceid	Instance ID
path	Field qualifier
value	value to set

Return value:

BMU_ATTACH_POSITION_NOT_FOUND	Either the name of the position is wrong or it was used for the wrong document type
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.5 BMU_GetValue +ZI

BMU_GetValue(unsigned int instanceid, const char* path, char* buffer, unsigned long length)

Returns the value of the given field. Please note that BmuXmlArtist has to be in the „data extraction“-state. If BMU_NULL is returned the corresponding field does not exist and nothing will be copied to *buffer*. This return has to be distinguished from the return BMU_OK with an empty string, that results from an empty overwriting of a xsd:string derived field.

instanceid	Instance ID
path	Field qualifier
buffer	memory block start-adress
length	memory block length

Return value:

BMU_BUFFERLENGTH_INSUFFICIENT	buffer size insufficient
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_NULL	Successful call, value not defined
BMU_UNKNOWN_ID	No instance with this ID found

2.5.6 BMU_GetValueUTF8 +ZI

BMU_GetValueUTF8(unsigned int instanceid, const char* path, char* buffer,
unsigned long length)

Returns the value of the given field in UTF-8 encoding. Please note that BmuXmlArtist has to be in the „data extraction“-state. See BMU_GetValue.

instanceid	Instance ID
path	Field qualifier

buffer	memory block start-adress
length	memory block length

Return value:

BMU_BUFFERLENGTH_INSUFFICIENT	buffer size insufficient
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_NULL	Successful call, value not defined
BMU_UNKNOWN_ID	No instance with this ID found

2.5.7 BMU_BeginCurrentView +ZI

BMU_BeginCurrentView(unsigned int instanceid)

Starts the extraction of data from the current view. For national documents this will merge all the content from the different layers together to produce a final view. International documents have layers too, but every layer is self-contained. It is functional equivalent to BeginLayerView for this use case.

instanceid	Instance ID

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.8 BMU_BeginHistoricalView +ZI

BMU_BeginHistoricalView(unsigned int instanceid, int view)

Starts the extraction of data from the XML based on the historical view.

instanceid	Instance ID
view	0 equals the current view, 1 the former view, etc.

Return value:

BMU_OUT_OF_RANGE	view is higher than the number of attachments.
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.9 BMU_BeginLayerView +ZI

BMU_BeginLayerView(unsigned int instanceid, int layer)

Starts the extraction of data from the XML based on a specific layer. It will return an error if the document has no layers. This characteristic can be queried by this mechanism.

instanceid	Instance ID
layer	0 equals the current view, 1 the former view, etc.

Return value:

BMU_ERROR	The document is not of the layered type. In this case please use BeginCurrentView.
BMU_OUT_OF_RANGE	layer is higher than the number of layers
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.10 BMU_EndView +ZI

BMU_EndView(unsigned int instanceid)

Stops the data extraction process.

instanceid	Instance ID

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.11 BMU_GetDocumentType

BMU_GetDocumentType(unsigned int instanceid, char* buffer, unsigned long length)

Determines the document type residing in the buffer.

instanceid	Instance ID
buffer	Memory block start-adress
length	Memory block length

Return value:

BMU_BUFFERLENGTH_INSUFFICIENT	Buffer size insufficient
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.5.12 BMU_GetFirstValue (since 0.9) +ZI

BMU_GetFirstValue(unsigned int instanceid, char** path, char** buffer)

This function can be used without knowing the field list, and it will return the first value from the field list. The return mechanism is different, too. Pointers to buffers are transferred that provide the contents. This buffer becomes invalid with the call of GetNextValue / GetNextValueUTF8.

Please note that this function won't return metadata (attachments, FreieXmlStruktur or signatures). Those have to be queried as a whole directly using the path or the attachment function.

instanceid	Instance ID
path	Pointer to a Pointer for the buffer of the path
buffer	Pointer to a Pointer for the buffer of the value

Return value:

BMU_OUT_OF_RANGE	end of field list reached
BMU_OK	Successful call
BMU_NULLPTR	path or buffer are null pointers
BMU_NEED_DIFFERENT_BUFFER	Pointers for both buffers are identical
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_UNKNOWN_ID	No instance with this ID found

2.5.13 BMU_GetNextValue (since 0.9) +ZI

BMU_GetNextValue(unsigned int instanceid, char** path, char** buffer)

Returns the next value. See BMU_GetFirstValue().

instanceid	Instance ID
path	Pointer to a Pointer for the buffer of the path
buffer	Pointer to a Pointer for the buffer of the value

Return value:

BMU_OUT_OF_RANGE	end of field list reached
BMU_OK	Successful call
BMU_NULLPTR	path or buffer are null pointers
BMU_NEED_DIFFERENT_BUFFER	Pointers for both buffers are identical
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_UNKNOWN_ID	No instance with this ID found

2.5.14 BMU_GetFirstValueUTF8 (since 0.9) +ZI

BMU_GetFirstValueUTF8(unsigned int instanceid, char** path, char** buffer)

This function can be used without knowing the field list, and it will return the first value from the field list. Contrarily to GetValue the buffer will be valid until the next call of GetNextValue / GetNextValueUTF8. Please note that this function won't return metadata. Those have to be queried using GetAttachment and BMU_GetFreeXmlPath.

instanceid	Instance ID
path	Pointer to a Pointer for the buffer of the path
buffer	Pointer to a Pointer for the buffer of the value

Return value:

BMU_OUT_OF_RANGE	end of field list reached
BMU_OK	Successful call
BMU_NULLPTR	path or buffer are null pointers
BMU_NEED_DIFFERENT_BUFFER	Pointers for both buffers are identical

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_UNKNOWN_ID	No instance with this ID found

2.5.15 BMU_GetNextValueUTF8 (since 0.9) +ZI

BMU_GetNextValueUTF8(unsigned int instanceid, char** path, char** buffer)

Returns the next value. See BMU_GetFirstValue().

instanceid	Instance ID
path	Pointer to a Pointer for the buffer of the path
buffer	Pointer to a Pointer for the buffer of the value

Return value:

BMU_OUT_OF_RANGE	end of field list reached
BMU_OK	Successful call
BMU_NULLPTR	path or buffer are null pointers
BMU_NEED_DIFFERENT_BUFFER	Pointers for both buffers are identical
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_UNKNOWN_ID	No instance with this ID found

2.5.16 BMU_RemoveField (since 0.9.3)

BMU_RemoveField(unsigned int instanceid, char* path)

This method deletes a specific value from the current layer instead of overwriting it empty. All fields defined as string can be overwritten with an empty value by SetValue(path, „“) but the element will stay and only disappear from the resulting view. This method still deletes the node from the current layer, making the former value visible again.

instanceid	Instance ID
path	Pointer to a Pointer for the buffer of the path

Return value:

BMU_OK	Successful call
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_UNKNOWN_ID	No instance with this ID found

2.5.17 BMU_GetFieldType (since 0.9.6)

BMU_GetFieldType(unsigned int instanceid, char* path, char** type)

This method queries the data type of the given field (see data types). The content of the returned buffer will be invalid upon calling a BMU_xxx function.

instanceid	Instance ID
path	Path
type	Pointer to a Pointer for the buffer of the path

Return value:

BMU_OK	Successful call
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_UNKNOWN_ID	No instance with this ID found

2.6 Other functions

2.6.1 BMU_GetBufferSize +ZI

unsigned long BMU_GetBufferSize(unsigned int instanceid)

Returns the minimum needed buffer size to successfully complete the most recently performed action.

instanceid	Instance ID

Return value:
Minimum buffer size needed

2.6.2 BMU_SetCreationalParameter

BMU_SetCreationalParameter(unsigned int threadid, const char* name, const char* value)

A number of steps require additional data for which there is no specific function. The mentioned parameters can be used with these general functions. The following can be used in all documents:

Pruefziffer	generally calculates all check digits in the current layer for national documents. Also deletes the check digit if the corresponding number is deleted.

Further parameters can be found in the descriptions of the document types.

threadid	Instance ID
name	Parameter name (see chapter 7)
value	value

Return value:

BMU_WRONG_STATE	Call not allowed (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.6.3 BMU_GetBmuSpecificationVersion

BMU_GetBmuSpecificationVersion(unsigned int threadid, char* buffer, unsigned int length)

Returns the version of the employed specification. The maximum buffer length is clearly defined.

threadid	Instance ID
buffer	Memory block start-adress
length	Memory block length

Return values:

BMU_BUFFERLENGTH_INSUFFICIENT	Buffer length insufficient
BMU_WRONG_STATE	Call not allowed (see diagram)
BMU_OK	Successful call
BMU_UNKNOWN_ID	No instance with this ID found

2.6.4 BMU_SetBmuSpecificationVersion

BMU_SetBmuSpecificationVersion(unsigned int threadid, const char* version)

Sets the specification version.

threadid	Instance ID
version	version number

Return values:

BMU_VERSION_NOT_SUPPORTED	Desired version not supported
BMU_OK	Successful call

2.6.5 BMU_Logging (changed 0.9.1) +ZI

BMU_Logging(unsigned int instanceid, unsigned int loglevel, const char* logfile, bool append)

Activates the logging of BmuXmlArtist. The log provides valuable information for troubleshooting.

instanceid	Instance ID
loglevel	1 (only errors) – 3 (detailed information)
logfile	Name of the logfile
append	(0.9.1) if true, existing logflies will be extended if false it will always delete the old logfile and create a new one

Return value:

BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OUT_OF_RANGE	Loglevel beyond valid range
BMU_OK	Successful call

2.6.6 BMU_GetVersion (changed 0.9.3) +ZI

BMU_GetVersion(char* buffer, unsigned int length)

Determines the version of the emplyed DLL. The buffer length is fixed (see include/bmu_interface_defines.h).

buffer	Memory block start-adress
length	Memory block length (constant)

2.6.7 BMU_Exists (since 0.9.4)

BMU_Exists(unsigned int instanceid, const char* path)

The data is hierarchically organized, in spite of the “flat” paths. If you choose your path accordingly you can check the existance of multiple fields in one call. Only the layer defined in BeginCurrentView, BeginHistoricalView or BeginLayerView will be factored in.

instanceid	Instance ID
path	path to check

Return value:

BMU_NULL	Successful call, path does not exist
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OUT_OF_RANGE	Loglevel beyond valid range
BMU_OK	Successful call, path exists

Example:
To check if a waste generator is entered on the frontpage of an ENSNDocument you need to do the following queries without BMU_Exists.

```
ENSNDocument.Layer.Deckblatt.Abfallerzeuger.NameUndAdresse.Name.Name1
ENSNDocument.Layer.Deckblatt.Abfallerzeuger.NameUndAdresse.Name.Name2
ENSNDocument.Layer.Deckblatt.Abfallerzeuger.NameUndAdresse.Name.Name3
ENSNDocument.Layer.Deckblatt.Abfallerzeuger.NameUndAdresse.Name.Name4
...
ENSNDocument.Layer.Deckblatt.Abfallerzeuger.Ansprechpartner.Email
```

This means 20 fields would have to be queried.

All fields have a shared path fraction and you can use it with BMU_Exists to simplify the query:

```
int ret = BMU_Exists(„ENSNDocument.Layer.Deckblatt.Abfallerzeuger“);
if (ret == BMU_OK)
{
   // Path is existing
}
else if (ret == BMU_NULL)
{
   // Path not existing
}
else
{
   // error
}
```

Hint: you can also use this query to determine if the document contains a DA:

```
int ret = BMU_Exists(„ENSNDocument.Layer.VE.DADocument“);
```

2.7 Signatures

The signature methods support the allocation and extraction of signature information. Please note that there can be signatures on the layer/document level as well as further signatures inside of a layer. Check the document specific parts for more detailed information.

2.7.1 BMU_CreateSignatureParameters

BMU_CreateSignatureParameters(unsigned int instanceid, const char* position,
bool enveloping, char* buffer, unsigned long length)

Creates the parameter block required by the Infotech Signer to sign the document.

instanceid	Instance ID
position	Position where the signature should be placed (see chapter 7)
enveloping	*true*: the second signature encloses the first signature *false*: the second signature is independend of the first signature
buffer	Memory block start-adress
length	Memory block length (constant)

Return value:

BMU_ERROR	An error occured while processing the XML-data to query the signature position.
BMU_SIGNATURE_POSITION_UNKNOWN	The given position could not be found
BMU_SIG_REFERENCE_NODE_MISSING	The node to attach the signature to does not exist
BMU_BUFFERLENGTH_INSUFFICIENT	given buffer is too small. You can determine the needed size using BMU_GetBufferSize()
BMU_OK	OK

2.7.2 BMU_GetSignatureCount (since 0.9.3) +ZI

BMU_GetSignatureCount(unsigned int instanceid, unsigned int& count, const char* position)

Determines the number of signatures on the given layer of the document. If a position is defined only signatures of the corresponding positions are counted in.

instanceid	Instance ID
count	contains the number of signatures after call
position	(optional) If used only signatures of the given position are taken into account (see chapter 7)

Return value:

BMU_SIGNATURE_POSITION_UNKNOWN	given postion not found
BMU_OK	OK
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OUT_OF_RANGE	Layer-No. beyond allowed range

2.7.3 BMU_GetSignatureInfo (since 0.9.3) +ZI

BMU_GetSignatureInfo(unsigned int instanceid, const char* position, unsigned int index,
char** info)

Returns detailed information to a signature as a field list. For information on the structure of the field list please check chapter 5.

instanceid	Instance ID
position	Position-ID of desired signature
index	1st or 2nd signature on this position
info	Pointer on a Field list contaning signature information. Memory area is only valid until next function call.

Return value:

BMU_SIGNATURE_POSITION_UNKNOWN	given position not found
BMU_OK	OK
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)
BMU_OUT_OF_RANGE	*layerno* or *index* beyond valid range

2.7.4 BMU_GetSignaturePositions (since 0.9.3) +ZI

BMU_GetSignaturePositions(unsigned int instanceid, char** positions)

Returns the possible positions for a signature. *Envelope* is possible, but in some documents there are further signature positions possible. You can find detailed information in the annex *document-specific signatures*.

instanceid	Instance ID
positions	Pointer to a field list containing signature positions. The positions are TAB-separated. Memory area is only valid until next function call.

Return value:

BMU_OK	OK
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)

2.8 Layer

2.8.1 BMU_GetLayers (since 0.9.1) +ZI

BMU_GetLayers(unsigned int instanceid, char** layerlist)

Returns all layers of the just loaded or created document, from the most recent to the oldest layer, as a TAB-separated list. Non-layered documents return an empty list.

instanceid	Instance ID
layerlist	List of layers (see chapter 7)

Return value:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK

2.8.2 BMU_GetNextLayer (since 0.9.1)

BMU_GetNextLayer(unsigned int instanceid, const char* currentlayer, char** layerlist)

Returns all layers that may succeed the given layer. Non-layered documents return an empty value.

instanceid	Instance ID
currentlayer	starting layer
layerlist	list of possible succession layers

Return value:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK

2.8.3 BMU_GetLayerCount (since 0.9.3) +ZI

BMU_GetLayerCount(unsigned int instanceid, int& count)

Determines the number of layers in the document and returns them in *count*. Returns 0 when used on non-layered documents.

instanceid	Instance ID
count	Contains the number of layers after call

Return value:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK

2.8.4 BMU_GetLayerRole (since 0.9.3)

BMU_GetLayerRole(unsigned int instanceid, char** role)

Determines the role of the given layer.

instanceid	Instance ID
role	upon call, the DLL-managed memory area contains the

Return value:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK

2.9 FreieXMLStruktur

2.9.1 BMU_GetFreeXmlPath (since 0.9.3)

BMU_GetFreeXmlPath(unsigned int instanceid, const char* position,
const char* namespaceuri, char** path)

Determines the correct path to a FreieXMLStruktur using the URI and position. If an URI is missing at that position, the path can still be used and points to the first free element at that position. You just need to attach the root with the actual data. Besides the position you can alternatively indicate a path. This ends on the FreieXMLStruktur-element without brackets, otherwise the allocation will not be found.

instanceid	Instance ID
position	the DLL-managed memory area contains the error messages after call
namespaceuri	NamespaceURI of FreieXMLStruktur generated according to the specification.
path	the DLL-managed memory area contains the path after call

Return value:

BMU_OUT_OF_RANGE	Given position not found
BMU_NULL	The returned path point to a new structure
BMU_OK	The returned path point to an existing structure
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)

2.9.2 BMU_GetFreeXmlPathNS (since 0.9.4)

BMU_GetFreeXmlPathNS(unsigned int instanceid, const char* position,
const char* namespaceuri, char** path)

Analogous to BMU_GetFreeXmlPath. The use of this command implicates an intention to write and leads to the creation of the necessary NamespaceURI-element, if not existing already.

Besides the position you can alternatively indicate a path. This ends on the FreieXMLStruktur-element without brackets, otherwise the allocation will not be found.

instanceid	Instance ID
position	the DLL-managed memory area contains the error messages after call
namespaceuri	NamespaceURI of FreieXMLStruktur generated according to the specification.
path	the DLL-managed memory area contains the path after call

Return value:

BMU_OUT_OF_RANGE	Given position not found
BMU_NULL	The returned path point to a new structure
BMU_OK	The returned path point to an existing structure
BMU_WRONG_STATE	Call not allowed at the moment (see diagram)

2.10 Error reporting

2.10.1 BMU_GetErrors (since 0.9.3) +ZI

BMU_GetErrors(unsigned int instanceid, char** errors)

Returns the error message that occured calling GetXML as text. The text is structured like this:

```
Field:'<field name>' Error::'<error message>'<cr><lf>[… next error]
```

Attention: it has been observed that missing mandatory fields were reported twice (nonlayered documents only). It is recommended to check if field names exist if you plan to adopt the messages into your own structures.

instanceid	Instance ID
errors	the DLL-managed memory area contains the error messages after call

Return value:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK

2.11 Document processing

This section describes the methods to merge or separate multiple documents.

2.11.1 BMU_ExtractEGF (ab 0.9.4)

BMU_ExtractEGF(unsigned int threadid, unsigned int threadidnew)

Extracts an EGFDokument from an existing ENSNDokument. The layer chosen by HistoricalView or LayerView is relevant. EGFDokument on temporary layers are not considered.

After the call you can read out the EGFDokument on the 2nd instance using BMU_GetXML.

threadid	Instance ID
threadidneu	Instance ID of the EGFDokument

Return values:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK
BMU_ERROR	The new document could not be created
BMU_INCOMPLETE_CONTENT	There is no EGF inside the document at this position

2.11.2 BMU_ExtractAGS (ab 0.9.6)

BMU_ExtractAGS(unsigned int threadid, unsigned int threadidnew)

Extracts an AGSBescheid from an existing ENSNDokument. The layer chosen by HistoricalView or LayerView is relevant. AGSBescheid on temporary layers are not considered.

After the call you can read out the AGSBescheid on the 2nd instance using BMU_GetXML.

threadid threadid	Instance ID
threadidneu	Instance ID of the AGSBescheid

Return values:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK
BMU_ERROR	The new document could not be created
BMU_INCOMPLETE_CONTENT	There is no AGS inside the document at this position

2.11.3 BMU_ExtractDA (ab 0.9.6)

BMU_ExtractDA(unsigned int threadid, unsigned int threadidnew)

Extracts a DADokument from an existing ENSNDokument. The layer chosen by HistoricalView or LayerView is relevant. DADokument on temporary layers are not considered.

After the call you can read out the DADokument on the 2nd instance using BMU_GetXML.

threadid	Instance ID
threadidneu	Instance ID of the DADokument

Return values:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK
BMU_ERROR	The new document could not be created
BMU_INCOMPLETE_CONTENT	There is no AGS inside the document at this position

2.12 Functions exclusive to international documents

2.12.1 BMU_CreateWM +ZI

BMU_CreateWM(unsigned int instanceid_nf, unsigned int instanceid_wm)

Creates a waste movement document using the corresponding waste notification. The notification has to be in the EXTRACTING state, the instance for the movement document just has to be initialized.

After call the movement document has to be opened to edit, and the mandatory fields *consecutive number of transports*, *actual shipment date*, *actual weight* and **at least** one *Carrier* have to be entered before you can create a valid movement document with GetXml.

instanceid_nf	Instance ID of the waste notification
instanceid_wm	Instance ID of the waste movement document

Return value:

BMU_UNKNOWN_ID	No instance with this ID found
BMU_OK	OK
BMU_ERROR	unable to create new document. Occurs if the source

	document is not a valid document.
BMU_INCOMPLETE_CONTENT	No DA present at this position.

3 Field list

The folder *fieldlists* of the BmuXmlArtist installation contains one CSV-file per document with the following information:

```
Field name, data type, regular expression
```

3.1 Field qualifier

3.1.1 Fields

Every entry in the document has a unique field qualifier.

3.1.2 Arrays / Lists

The specification allows lists of values in some places. You can identify them by squared brackets [ ] in the field qualifier.

In Section 6 of the Notification you can specify different package types. The corresponding entry in the csv file in the *fieldlists* folder contains:

```
".NotificationPackagagingTypes[].Type","C(256)"
```

To transfer a list of package types to the BmuXmlArtist use the following sequence:

```
BMU_SetField(".NotificationPackagagingTypes[1].Type", „1“);
BMU_SetField(".NotificationPackagagingTypes[2].Type", „2“);
```

When reading out please note that the Artist is not able to query the highest index. Thus, the query using GetValue() is complicated because lists with the index element don’t have to allocate them in an orderly fashion, leaving you without an abort criterion.

The recommended course of action is to read out all values using GetFirstValue / GetNextValue and filter all corresponding paths out.

3.1.3 Deleting lines

You have to distinguish several cases. BMU_RemoveField can be used in the topmost layer to erase lines. The index field of the line is adressed for this.

Example: 1st line of the Nebenbedingung (AGS in EN) shall be erased:

```
BMU_RemoveField(instanceid,
"ENSNDocument.Layer.AGSBescheid.Nebenbestimmungen.Nebenbestimmung[1]“);
```

If a layer was applied, the deletion of lines is easily possible with tables without index. The view is completely rewritten in the new layer and is displayed in this exact configuration since no differencing was done.

In tables with index and differencing it is no longer possible to erase lines. You can only overwrite contents with empty or other suitable entries.

In the BGSDocument you can't overwrite waste codes or Übernahmescheinnummern with an empty value. The value has to adhere to the schema.

3.1.4 Data type

The second value in the CSV-file's line is reserved for the data type. Use the following encoding:

Name	**Description**
C(100)	String with a maximum of 100 characters allowed
N(5.3)	Numerical value, 5 digits, 3 of which are decimal fraction, for example *23.345*
D()	Date always in YYYYMMDDHHMISS format. For example *20081010000000*
L()	logical value, either *true* or *false*

3.1.5 Regular Expression

Use this information to carry out checks in advance in the interface. Please note that the notation originates from the schema definition and does not equate to the regular expressions of PERL.

BmuXmlArtist itself always validates the input value.

4 File attachments

The attachments allow for a range of operations, however they don't work immediately. Think batch processing. An attachment can be adressed via a **position** and an **index.** This index has nothing to do with the position in the XML tree, but is purely a logical index to make sure the attachment is explicitly accessed.

These indexes may change at every call of *BMU_LoadXml* and *BMU_GetXml*.

4.1 Case example

Assumption: a BGSDocument has a Vorlagenlayer witch three attachments in the unsigned area (PositionID = „unsigned"). Now the Erzeugerlayer (BGSERZLayer) is added. The attachment is to be deleted and substituted by a new one.

Additionally, a further attachment is added.

First, the number of attachments in the desired area is determined. Now the corresponding index has to be determined, for example by using the file name. Now *BMU_RemoveAttachment* has to be called to erase the attachment.

To substitute an attachment you only have to place a new attachment on an already assigned index.

Remark: it is not possible to assign an attachment to a specific position in the order of attachments.

4.2 Extended informations with BMU_AttachInfo

To get more specific information to an attachment you can use the function

BMU_AttachmentInfo(position, index, char** infos)

The individual value-pairs are seperated by TAB. The values of the pair itself are separated by a = (equality sign).

Example:

```
Filename=deklarationsanalyse.txt\tSize=48\tSignatureId=\tIndex=1\tAttachId=1\t
Layer=0\tPruefmethode=SHA256\tVersion=1
```

The first value of the pair identifies the value.

Ident	**Description**
Filename	file name in the attachment
Size	size of originary file
SignatureId	Signature ID that **directly** encloses the attachment
Index	determines the position of the attachment (i.e. 6th out of 10)
AttachId	This value plus the positionID clearly identifies a specific attachment.
Layer	Layer that houses the attachment: *-1* in a layered document, outside of the layer *0* in the current layer or in the document *>0* in a subsequent layer
Pruefmethode	employed mode of verification
MimeType	Mimetype
ID	Attachment ID (unique)

5 Signatures

The BXA is able to extract signature information out of BMU Documents. But it is unable to apply signatures. For this you need the Infotech Signer.

BXA supports the creation of the parameter block for the Infotech Signer as well as with the extraction of data that it delivers in form of a XML structure.

Notice that when applying a signature that some documents like EGFDocument and BGSDocument may contain several signatures. Those may enclose areas that don't overlap, or contain each other. If the order of signatures is not adhered to then already applied signatures become invalid.

To apply a signature you need the ID of the position. In all signable documents there is the position *envelope* that encloses the whole document.

5.1 GetSignatureInfo

The return of of GetSignatureInfo is a text block that has the following structure:

id=<SignatureID>\t
dname=<subjectname>\t
subjectname=<signee name>\t
signingTime=<system time of signee at the moment of the signature>\t
path=<xpath-expression that defines the signature in the document>\t
role=Signature role(also position if not envelope)

the individual key/value-pairs are separated by TAB (\t). The return is single-line even if the depiction above suggests otherwise.

For the individual fields please note the following length limitations / data types

Field	Length	Format
signingTime	26	[-]CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm] see http://www.w3.org/TR/xmlschema-2/#dateTime
Path	<2143	Path/Path/Signature[1]
Id	40	(ERZ|BEF|ENT|ZWL|BEH|MAK|BVE|PRV|SNT)\-(\d|[A-Fa-f]){8}\-((\d|[A-Fa-f]){4}\-){3}(\d|[A-Fa-f]){12}
Dname	unlimited	distinguished_name in accordance with RFC 2253
subjectname	unlimited	Common Name from dname

The maximum length of the path results from the maximum number of layers (99) multiplied by the longest layer name (ENSNErgaenzungsLayer) as well as the path to the signature inside the AGSBescheid.

ENSNDocument/Layer/ = 19 characters
ENSNErgaenzungsLayer/ = 21 * 99 characters = 2079 characters
AGSBescheid/AGSBescheid/Signature[1] = 36 characters
→ Total = 2143 characters

There are examples for the extraction of signature information at the end of this document.

5.2 Signature hierarchies / inner signatures

Other than the signature that encloses all content (envelope) there are also signatures that only enclose a part of the document. Those enclose other signatures in turn and so a hierarchy of signatures is formed. This hierarchy is displayed in the document-specific information as follows:

envelope	encloses the whole document
→ da	Deklarationsanalyse
→→behoerde	complete EGFDocument
→→→ erzeuger	Waste generator on EGF
→→→bevollmaechtigung	Bevollmächtigter EGF
→→→beauftragung	Beauftragter EGF
→ags	complete AGSBescheid

Signatures on the same level can be applied in a random order, but no more signatures are possible on a subjacent level as those would fall into an area that has already been signed and would thus make the signature invalid. FreieXMLStruktur

5.2.1 International documents

For the international documents only the position "*enveloped*" is supported.

5.2.2 eIDAS support

Beginning with Version 1.0.12, documents containing eIDAS conforming signatures are supported. Because eIDAS is less restrictive on elements like SubjectName and SigningTime, they could possibly be missing. In these cases, BmuXmlArtist cannot provide values for these elements.

The SubjectName can be extracted directly from the signing certificate, but the SigningTime cannot be extracted from the signature und will definitely be missing if not included in the XML structures.

6 FreieXMLStruktur

6.1 Reading a structure

The structure has to be identified with a position and the corresponding NamespaceURI.

Using this data, BMU_GetFreeXmlPath is called and returns the path. The return value informs you if a structure already exists at that place or if a new one is created.

Implementing sketch:

```
char* buf = 0;
int ret = BMU_GetFreeXmlPath(id, „signiertxml“, „someuri“, &buf);
if (ret == BMU_OK)
{
        // Read the value of an existing FreieXMLStruktur „Root.Wert1“
        char buf[2048];
        std::string p1 = buf;
        p1.append(„.Root.Wert1“);
        if (BMU_GetValue(id, p1.c_str(), buf, sizeof(buf)) == BMU_OK)
        {
                // process value
        }
}
else if (ret == BMU_NULL)
{
        // No structure or data available
}
else if (ret == BMU_OUT_OF_RANGE)
{
        // Error, position unknown
}
```

6.2 Writing

You should prefer the command BMU_GetFreeXmlPathNS for writing. It creates the NamespaceURI-element to identify the structure. Without this element further writing processes would lead to duplication of the FreieXMLStruktur because the query for the NamespaceURI would not result in any hits if it was not created.

Since the FreieXMLStruktur needs no schema you have to use the path to define whether an element or an attribute gets used. To write an attribute you have to use "@" as a prefix to the attribute name.

Implementing sketch:

```
char* buf = 0;
int ret = BMU_GetFreeXmlPathNS(id, „signiertxml", „someuri", &buf);
if (ret == BMU_OK) || (ret == BMU_NULL)
{
        // Write „1" to the value of an existing FreieXMLStruktur „Root.Wert1"
        std::string p1 = buf;
        p1.append(„.Root.Wert1");
        BMU_SetField(id, p1.c_str(), "1");
}
else if (ret == BMU_OUT_OF_RANGE)
{
        // Error, position unknown
}
```

Directly below the FreieXMLStruktur you have to define a root element. Only one element is allowed on this level. It might look like this:

```
BGSDocument.Layer.FreieXMLStruktur[1].root.Daten=value1
BGSDocument.Layer.FreieXMLStruktur[1].root.Daten2=value2
BGSDocument.Layer.FreieXMLStruktur[1].root.@version=1.2
```

6.3 ZEDAL

The additional data used by ZEDAL are identified by the following URI:

```
urn:de:bmu:eanv:ext:infotech:zedal:betriebsdaten:1:0
```

When calling GetFirstValue/GetNextValue the contents of the structure are shown in the field list.

Non-ZEDAL FreieXMLStrukturen have to be read out separately.

7 Document specific Information

When usind file attachments you can choose to either enclose them with a signature (signed) or not enclose them with a signature (unsigned).

This is what the attachment remarks refer to. Of course it is also possible to have an unsigned attachment in an area that is enclosed by a signature.

7.1 **BGSDokument**

This section contains additional information to process a BGSDokument.

7.1.1 File attachment

The following position is specified for file attachments:

unsigniert	unsigned file attachment outside of layers

7.1.2 FreieXMLStruktur

The following positions are defined for FreieXMLStrukturen:

unsigniertxml	BGSDokument.FreieXMLStruktur[]
signiertxml	BGSDokument.Layer.FreieXMLStruktur[]
verordnungenxml	BGSDokument.Layer.Daten.AndereVerordnungen.FreieXMLStruktur[]

7.1.3 Signatures

Signatures can be applied to the following positions:

envelope	enclosing the selected layer
→ altoel	In the *andere Verordnungen Altöl* you sign the fields of the *Untersuchungspflichtiger (person to investigate)*.

7.1.4 Layer order

List of layers and their succession layers:

BGSVorlageLayer	BGSERZLayer
BGSERZLayer	BGSBEFLayer
BGSBEFLayer	BGSBEFLayer, BGSZWLLayer, BGSENTLayer
BGSZWLLayer	BGSBEFLayer
BGSENTLayer	BGSBEHLayer
BGSBEHLayer	BGSErgaenzungsLayer
BGSErgaenzungsLayer	BGSBEHLayer, BGSErgaenzungsLayer

7.1.5 Special parameters

7.1.5.1 ATBRolle

You need to additionally defined the ATBRolle when creating BGSVorlageLayer and BGSErgaenzungsLayer. Allowed values are (see page 28 of the BMU specification): *ERZ, BEF1, BEF2, BEF3, ENT, ZWL, BEH, MAK, BEVERZ, PROV, SONST*

7.1.6 Lists

WeitereAbfallschluessel	with Index
UNSNummer	with Index
PCBFraktion	with Index
Sortiment	with Index

7.2 **ENSNDokument**

7.2.1 File attachment

The following positions are defined for file attachments:

unsigniert	unsigned file attachment outside of layers
da	Deklarationsanalyse
agssigniert	signed file attachment in AGSBescheid
gebührenbescheid	signed file attachment in authority (BEH) layer

7.2.2 FreieXMLStruktur

The following position are specified for FreieXMLStrukturen:

unsigniertxml	ENSNDokument.FreieXMLStruktur[]
signiertxml	ENSNDokument.Layer.FreieXMLStruktur[]
daxml	ENSNDokument.Layer.VE.DADokument.DA.FreieXMLStruktur[]
antrag	ENSNDokument.Layer.EGFDokument.Nummer1_Antrag .FreieXMLStruktur[]
bevollmächtigung	ENSNDokument.Layer.EGFDokument.Nummer2_Bevollmaechtigung .FreieXMLStruktur[]
beauftragung	ENSNDokument.Layer.EGFDokument.Nummer3_Beauftragung .FreieXMLStruktur[]
entscheidung	ENSNDokument.Layer.EGFDokument.Entscheidung .FreieXMLStruktur[]
gebührenbescheidxml	ENSNDokument.Layer.Gebuehr.**Gebuehrenbescheid[]**
agssigniertxml	ENSNDokument.Layer.AGSBescheid.FreieXMLStruktur[]
agsgebührenbescheidxml	ENSNDokument.Layer.AGSBescheid.**Gebuehrenbescheid[]**
daschemaxml	ENSNDokument.Layer.VE.DADokument.DA.**Deklarationsanalyse[]**

7.2.3 Signatures

You can apply signatures at the following positions:

envelope	Enclosing the whole document
→da	Deklarationsanalyse
→behoerde	Complete EGFDokument
→→erzeuger	Waste generator signature
→→→bevollmaechtigung	Bevollmächtigter
→→→beauftragung	Beauftragter
→ags	Complete AGSBescheid

7.2.4 Layer order

List of layers and their succession layers:

ENSNVorlageLayer	ENSNERZLayer
ENSNERZLayer	ENSNENTLayer
ENSNENTLayer	ENSNBEHLayer, ENSNErgaenzungsLayer
ENSNBEHLayer	ENSNBEHLayer, ENSNErgaenzungsLayer
ENSNErgaenzungsLayer	ENSNBEHLayer, ENSNErgaenzungsLayer, ENSNENTLayer

7.2.5 Special parameters

7.2.5.1 ATBRolle

You need to additionally define the ATBRolle when creating ENSNVorlageLayer and ENSNErgaenzungsLayer. Allowed values are (see page 28 of the specification): *ERZ, BEF, ENT, BEH, BEVERZ.*

7.2.5.2 Interface

This parameter issues that the AGSBescheid in the ENSNDokument is not constructed from the last AGS plus changes, but exclusively possesses newly written content. This also works for attachments.

7.2.5.3 SchnittstelleBeh

There is a speciality to note when dealing with Behördenlayer in the containers BB, EB, Anordnung, Nachforderung: the creation of the view only considers the values in the corresponding layer, the values from older layers **do not** show through. The authorities wanted it this way and this is also implemented in the BMU-Viewer and the ZKS. To activate this process you need to set this parameter.

7.2.6 Lists

BimSchG	with index
Sammelgebiet	double Index (federal state, district/county/number)
Sonstiges (EGF)	double index (federal state, district/county)
Fehlerbehandlung (EB)	with index
Nebenbestimmung (BB)	with index

Begruendung (BB)	with index
Nebenbestimmung (Anordnung)	with index
Begruendung (Anordnung)	with index
Information (Nachforderung)	
Nutzbarkeit (AGS)	double index (federal state, district/county)

7.3 UNSDokument

7.3.1 File attachment

The following positions are defined for file attachments:

unsigniert	unsigned file attachment outside of layers

7.3.2 FreieXMLStruktur

The following positions are specified for FreieXMLStrukturen:

unsigniertxml	UNSDokument.FreieXMLStruktur[]
signiertxml	UNSDokument.Layer.FreieXMLStruktur[]
verordnungenxml	UNSDokument.Layer.Daten.AndereVerordnungen.FreieXMLStruktur[]

7.3.3 Signatures

You can apply signatures to the following positions:

envelope	enclosing the selected layer
→ altoel	In the *andere Verordnungen Altöl* you sign the fields of the *Untersuchungspflichtiger (person to investigate)*.

7.3.4 Layer order

List of layers and their succession layers:

UNSVorlageLayer	UNSBasisLayer
UNSBasisLayer	UNSErgaenzungsLayer
UNSErgaenzungsLayer	UNSErgaenzungsLayer

7.3.5 Special parameters

7.3.5.1 ATBRolle

You need to additionally define the ATBRolle when creating layers in a UNSDokument. Allowed values are (see page 28 of the specification): *ERZ, BEF1, BEF2, BEF3, ENT, ZWL, BEH, MAK, BEVERZ, PROV, SONST*

7.3.6 Lists

WeitereAbfallschluessel	with index
PCBFraktion	with index
Sortiment	with index

7.4 **EGFDokument**

7.4.1 File attachment

No attachments allowed.

7.4.2 FreieXMLStruktur

The following positions are specified for FreieXMLStrukturen:

antrag	EGFDokument\Nummer1_Antrag.FreieXMLStruktur[]
bevollmächtigung	EGFDokument.Nummer2_Bevollmaechtigung.FreieXMLStruktur[]
beauftragung	EGFDokument.Nummer3_Beauftragung.FreieXMLStruktur[]
entscheidung	EGFDokument.Entscheidung.FreieXMLStruktur[]

7.4.3 Signatures

Signatures may be applied to the following positions:

envelope	enclosing the whole document
→ erzeuger	Waste generators' signature
→→bevollmaechtigung	Bevollmächtigter
→→beauftragung	Beauftragter

7.4.4 Layer order

No layers

7.4.5 Lists

Sonstiges	double index (federal state, county)

7.5 **AGSBescheid**

7.5.1 File attachment

The following positions are defined for file attachments:

signiert	signed file attachment
gebührenbescheid	signed file attachment

7.5.2 FreieXMLStruktur

The following positions are specified for FreieXMLStrukturen:

agssigniert	AGSBescheid.FreieXMLStruktur[]
agsgebührenbescheidxml	AGSBescheid.**Gebuehrenbescheid[]**

7.5.3 Signatures

Signatures may be applied to the following positions:

envelope enclosing the whole document**.**

Important: a valid document has to contain at least one signature, otherwise it will be rejected.

7.5.4 Layer order

No layers

7.5.5 Lists

Nutzbarkeit → double index (federal state, county)

7.6 **Mitteilung**

7.6.1 File attachment

The following positions are defined for file attachments:

signiert signed file attachment

7.6.2 FreieXMLStruktur

The following positions are defined for FreieXMLStrukturen:

Signiertxml Mitteilung.FreieXMLStruktur[]

7.6.3 Signatures

Signatures may be applied to the following positions:

envelope enclosing the whole document

7.6.4 Layer order

No layers

7.7 **Nachweisliste**

7.7.1 File attachments

None.

7.7.2 FreieXMLStruktur

The following positions are defined for FreieXMLStruktur:

signiertxml Nachweisliste.FreieXMLStruktur[]

7.7.3 Signatures

Signatures may be applied to the following positions:

envelope enclosing the whole document

7.7.4 Layer order

No layers

7.8 **FRDokument**

7.8.1 File attachment

The following positions are defined for file attachments:

unsigniert	unsigned file attachment outside of layers

7.8.2 FreieXMLStruktur

The following positions are defined for FreieXMLStruktur:

unsigniertxml	FRDokument.FreieXMLStruktur[]
signiertxml	FRDokument.Layer.FreieXMLStruktur[]
gebührxml	FRDokument.Layer.Gebuehr.**Gebuehrenbescheid[]**
abfallxml	FRDokument.Layer.Deckblatt.Abfall.FreieXMLStruktur**[1]**

Note: *abfallxml* is only valid once per waste

7.8.3 Signatures

Signatures can be applied to the following position:

envelope	enclosing the whole document.

7.8.4 Layer order

List of layers and their succession layers:

FRENTLayer	FRBEHLayer
FRBEHLayer	FRENTLayer

7.9 **Waste notification**

Type: „NotificationDocument“

7.9.1 File attachment

The following positions are defined for file attachments:

signed	signed attachment

7.9.2 FreieXMLStruktur

The following positions are specified for FreieXMLStrukturen:

Signiertxml	Abfallverbringungsdocument.FreieXMLStruktur[]

7.9.3 Signatures

envelope enclosing the whole document.

7.9.4 Layer order

There is only one layer type for the notification that can applied indefinitely:

EUDINNotificationDocument

7.10 **Waste movement document**

Type: „WasteMovementDocument“

7.10.1 File attachment

The following positions are defined for file attachments:

signed	signed attachment

7.10.2 FreieXMLStruktur

The following positions are specified for FreieXMLStrukturen:

Signiertxml	Abfallverbringungsdocument.FreieXMLStruktur[]

7.10.3 Signatures

envelope	enclosing the whole document.

7.10.4 Layer order

EUDINWasteMovementDocument	transport announcement
EUDINCertificateOfWasteReceiptDocument	confirmation of receipt
EUDINCertificateOfWasteRecoveryDisposalDocument	disposal confirmation

7.10.5 Exceptions

BMU_CreateNextLayer can be called with the parameters **true** and **false** for the waste movement document. If **true** this applies the next layer type, if **false** it applies the same layer type.

7.11 **Annex7 document**

Type: „Annex7“

7.11.1 File attachment

The following positions are defined for file attachments:

signed	signed attachment

7.11.2 FreieXMLStruktur

The following positions are specified for FreieXMLStrukturen:

Signiertxml	Abfallverbringungsdocument.FreieXMLStruktur[]

7.11.3 Signatures

envelope enclosing the whole document.

7.11.4 Layer order

EUDINWasteMovementDocument	transport announcement
EUDINCertificateOfWasteReceiptDocument	confirmation of receipt

7.11.5 Exceptions

BMU_CreateNextLayer can be called with the parameters **true** and **false** for the waste movement document. If **true** this applies the next layer type, if **false** it applies the same layer type.

8 History

Hint: red lines denote breaking changes

8.1 Version 1.0.10

- Changing a BGSDokument with ATBBefLfdNr equal to BEF, deletes the role completely

8.2 Version 1.0.11

- TFS
  - GetFirst/GetNext in modes historicalview, layerview wasn't working as expected
  - DateTime elements with timezone offset got corrupted
  - ExtendedSignatureRole was copied to the next layer without applying a signature.
  - Support for document type "Annex VII" was added
- German waste Documents
  - Freistellung: Error in Handling of the Abfall Table
  - EGF: Table Bundesland wasn't handled correctly. Could lead to errors in the Registerauszug/BGSExcerpt

8.3 Version 1.0.12

- Support for handling documents signed conforming the eIDAS specification.
  - Possible loss of SubjectName and SigningTime in function BMU_GetSignatureInfo

9 Example code

9.1 Create a BGSDokument

```
unsigned int id = 0;
if (BMU_Initialize(LIZENZNEHMER, LIZENZSCHLUESSEL, id) == BMU_OK)
{
    if (BMU_SetBmuSpecificationVersion(id, BMU_SPEC_V104) == BMU_OK)
    {
        if (BMU_Create(id, "BGSDokument") == BMU_OK)
        {
            if (BMU_BeginFields(id) == BMU_OK)
            {
                int ret = BMU_OK;
                ret |= BMU_SetField(id, "BGSDokument.Layer.BGSNummer.PaginierNr",
"11234567890123");
                // the following line may be dropped; in this case the checksum
would be calculated internally
                ret |= BMU_SetField(id, "BGSDokument.Layer.BGSNummer.Pruefziffer",
"1");

                ret |= BMU_SetField(id, "BGSDokument.Layer.Daten.Abfallschluessel",
"200301");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.Abfallbezeichnung", "gemischte Siedlungsabfälle");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.Nachweisnummer.NachweisNr", "ENA000000000");
                ret |= BMU_SetField(id, "BGSDokument.Layer.Daten.Menge", "12.44");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Erzeuger.IndicatorQuittungsbeleg", "false");

                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Erzeuger.Nummer.BehoerdlicheNr", "AERZ00000");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Erzeuger.NameUndAdresse.Name.Name1", "Erzeuger");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Lager.DatumUebergabe", "20081010000000");

                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Befoerderer[1].Nummer.BehoerdlicheNr",
"ABEF00000");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Befoerderer[1].NameUndAdresse.Name.Name1",
                                    "Beförderer");

                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Entsorger.Nummer.BehoerdlicheNr", "AENT00000");
                ret |= BMU_SetField(id,
"BGSDokument.Layer.Daten.ATBListe.Entsorger.NameUndAdresse.Name.Name1",
"Entsorger");
```

```
                if (ret == BMU_OK)
                {
                    if (BMU_EndFields(id) == BMU_OK)
                    {
                        BMU_SetCreationalParameter(id, "ATBRolle", "ERZ");

                        // provide memory block for the XML result
                        char buffer[16384];
                        if (BMU_GetXml(id, buffer, sizeof(buffer)) == BMU_OK)
                        {
                            FILE* handle = fopen("beispiel1.xml", "wb");
                            fwrite(buffer, 1, strlen(buffer), handle);
                            fclose(handle);
                        }
                    }
                }
            }
        }
    }
    BMU_Release(id);
}
```

9.2 Load an existing document and read values

```
unsigned int id = 0;
if (BMU_Initialize(LIZENZNEHMER, LIZENZSCHLUESSEL, id) == BMU_OK)
{
    if (BMU_LoadXml(id, "beispiel1.xml") == BMU_OK)
    {
        if (BMU_BeginCurrentView(id) == BMU_OK)
        {
            char buffer[16384];
            int ret = BMU_OK;
            ret |= BMU_GetValue(id, "BGSDokument.Layer.BGSNummer.PaginierNr",
buffer, sizeof(buffer));
            printf("%s\n", buffer);
            ret |= BMU_GetValue(id, "BGSDokument.Layer.BGSNummer.Pruefziffer",
buffer, sizeof(buffer));
            printf("%s\n", buffer);
            ret |= BMU_GetValue(id,
"BGSDokument.Layer.Daten.ATBListe.Befoerderer[1].NameUndAdresse.Name.Name1",
                                buffer, sizeof(buffer));
            printf("%s\n", buffer);
            if (ret == BMU_OK)
            {
                if (BMU_EndView(id) == BMU_OK)
                {
                }
            }
        }
    }
    BMU_Release(id);
}
```

9.3 Extract Attachments

```
unsigned int id = 0;
if (BMU_Initialize(LIZENZNEHMER, LIZENZSCHLUESSEL, id) == BMU_OK)
{
    if (BMU_LoadXml(id, "bmu.xml") == BMU_OK)
    {
        if (BMU_BeginCurrentView(id) == BMU_OK)
        {
            char* positions = 0;
            if (BMU_GetAttachmentPositions(id, &positions) == BMU_OK)
            {
                char* position = strtok(positions, "\t");
                while (position)
                {
                    unsigned int count = 0;
                    if (BMU_GetAttachmentCount(id, position, count) == BMU_OK)
                    {
                        for (int i = 1; i <= count; i++)
                        {
                            if (BMU_GetAttachment(id, position, i, 0, 0) ==
BMU_BUFFERLENGTH_INSUFFICIENT)
                            {
                                long size = BMU_GetBufferSize(id);
                                char* buffer = new char[size];
                                if (BMU_GetAttachment(id, position, i, buffer,
size) == BMU_OK)
                                {
                                    if (BMU_GetAttachmentFilename(id, position, i,
0, 0) == BMU_BUFFERLENGTH_INSUFFICIENT)
                                    {
                                        long namesize = BMU_GetBufferSize(id);
                                        char* filename = new char[namesize+1];
                                        if (BMU_GetAttachmentFilename(id, position,
i, filename, namesize) == BMU_OK)
                                        {
                                            FILE* file = fopen(filename, "wb");
                                            fwrite(buffer, size, 1, file);
                                            fclose(file);
                                        }
                                        delete []filename;
                                    }
                                }
                                delete []buffer;
                            }
                        }
                    }
                    position = strtok(0, "\t");
                }
            }
            if (BMU_EndView(id) == BMU_OK)
            {
                BMU_Release(id);
                return;
            }
        }
    }
}
```

9.4 Creating a notification

```
bool appendtofile = true;
int loglevel = 4; // 1-4
unsigned int id = 0;

if ( BMU_Initialize(LIZENZNEHMER, LIZENZSCHLUESSEL, id) == BMU_OK ) {
    BMU_Logging(id, loglevel, "bmudll_tfs.log", appendtofile);
```

```
    if ( BMU_Create(id, "NotificationDocument", false) == BMU_OK ) {
        BMU_BeginFields(id);

        // set all mandatory fields
        BMU_SetField(id,".DeclarationDateTimeExporter","20130102000000");
        BMU_SetField(id,".DisposalRecoveryIndicator","J");
        BMU_SetField(id,".FirstShipmentDateTime","20130101");
        BMU_SetField(id,".IndicatorSignatureExporter","N");
        BMU_SetField(id,".LastShipmentDateTime","20131231");
        BMU_SetField(id,".Measure","5.00000");
        BMU_SetField(id,".NameExporter","Exporteut");
        BMU_SetField(id,".NotificationCountryStateConcernd[1].CountryIDnr","AD");

BMU_SetField(id,".NotificationCountryStateConcernd[1].CountryRoleIdentificationIDnr
","E");
        BMU_SetField(id,".NotificationCountryStateConcernd[1].zeile","1"); //
consecutive number
        BMU_SetField(id,".NotificationCountrynr","DE");

BMU_SetField(id,".NotificationGenerator[1].DeclarationDateTimeGenerator","201301010
00000");

BMU_SetField(id,".NotificationGenerator[1].IndicatorSignatureGenerator","false");
        BMU_SetField(id,".NotificationGenerator[1].NameGenerator","Erzeuger");
        BMU_SetField(id,".NotificationIdentificationID","DE 2774/079811");
        BMU_SetField(id,".NotificationIntendedCarriers[1].ModeCodenr","A");
        BMU_SetField(id,

".NotificationIntendedCarriers[1].NotificationIdentificationIDOrganizationCarrierPa
rty[1].IdentificationIDnr",
            "Aeuro--01");
        BMU_SetField(id,".NotificationIntendedCarriers[1].zeile","1"); //
consecutive number

BMU_SetField(id,".NotificationPackagagingTypes[1].IndividualPackagesQuantity","5");
        BMU_SetField(id,".NotificationPackagagingTypes[1].TypeCodenr","1");

BMU_SetField(id,".NotificationPhysicalCharacteristicCode[1].PhysicalCharacteristicC
odenr","1");

BMU_SetField(id,".NotificationRecoveryDisposalOperationClassification[1].rdcodenr",
"D01");

BMU_SetField(id,".NotificationWasteGeneratorsProducers[1].AddressCountryIDnr","AD")
;
        BMU_SetField(id,

".NotificationWasteGeneratorsProducers[1].NotificationIdentificationIDOrganizationP
arty[1].IdentificationIDnr",
            "Aeuro--01");
        BMU_SetField(id,".NumberOfAnnexesNumeric","1");
        BMU_SetField(id,".PreconsentedRecoveryFacilityIndicator","false");
        BMU_SetField(id,".SpecialHandlingRequiredIndicator","false");
        BMU_SetField(id,".TotalIntendedNumberOfShipmentsNumeric","1");
        BMU_SetField(id,".WasteDescription","Metallabfälle und Abfälle von
Legierungen”);
        // provide at least one waste classification, e.g. baselcode
        BMU_SetField(id,".baselcodenr","A1010");
        BMU_EndFields(id);

        if ( BMU_GetXml(id, 0, 0) == BMU_BUFFERLENGTH_INSUFFICIENT ) {
            long size = BMU_GetBufferSize(id);
            char* buf = new char[size];
            if ( BMU_GetXml(id, buf, size) == BMU_OK ) {
                // All done!
            }
            delete[] buf;
        }
```

```
        }
}
```

9.5 Creating a waste movement document

```
bool appendtofile = true;
int loglevel = 4; // 1-4
unsigned int id = 0;

if ( BMU_Initialize(LIZENZNEHMER, LIZENZSCHLUESSEL, id) == BMU_OK ) {
    BMU_Logging(id, loglevel, "bmudll_tfs.log", appendtofile);
    if ( BMU_Create(id, "WasteMovementDocument", false) == BMU_OK ) {
        BMU_BeginFields(id);

        // set all mandatory fields
        BMU_SetField(id,".ActualShipmentDateTime","20130701000000");
        BMU_SetField(id,".ActualWeightMeasurementSpecificationMeasure","1.00000");
        BMU_SetField(id,".AddressCountryIDnr","AD");
        BMU_SetField(id,".ConsecutiveTransportNumberNumeric","1");
        BMU_SetField(id,".CorrespondingNotificationIdentificationID","DE
2774/079811");
        BMU_SetField(id,".DisposalFacilityIndicator","true");
        BMU_SetField(id,".RegisteredOfficeContactAddressCountryIDnr","AE");
        BMU_SetField(id,".SpecialHandlingRequiredIndicator","false");
        BMU_SetField(id,".TotalNumberOfShipmentsNumeric","1");

BMU_SetField(id,".WasteCustomsClearance[001].ClearanceDateTime","19700101000000");
        BMU_SetField(id,".WasteCustomsClearance[001].EnteredCountryIDnr","AD");

BMU_SetField(id,".WasteCustomsClearance[001].EnteredCountryRoleIdentificationIDnr",
"E");
        BMU_SetField(id,".WasteDescription","Metallabfälle und Abfälle von
Legierungen");
        BMU_SetField(id,".WastePackage[001].IndividualPackagesQuantity","5");
        BMU_SetField(id,".WastePackage[001].TypeCodenr","1");

BMU_SetField(id,".WastePhysicalCharacteristicCode[001].PhysicalCharacteristicCodenr
","1");

BMU_SetField(id,".WasteRecoveryDisposalOperationClassification[001].rdcodenr","D01"
);
        BMU_SetField(id,".WasteTransportStage[001].ModeCodenr","A");

BMU_SetField(id,".WasteTransportStage[001].WasteIdentificationIDOrganization[001].I
dentificationIDnr","AENT-0011");
        BMU_SetField(id,".WasteTransportStage[001].zeile","1");
        BMU_SetField(id,".WasteWasteOrigin[001].AddressCountryIDnr","AD");

BMU_SetField(id,".WasteWasteOrigin[001].WasteIdentificationIDOrganizationParty[001]
.IdentificationIDnr","Aeuro--01");
        BMU_SetField(id,".baselcodenr","A1010");
        BMU_EndFields(id);
```

```
        if ( BMU_GetXml(id, 0, 0) == BMU_BUFFERLENGTH_INSUFFICIENT ) {
            long size = BMU_GetBufferSize(id);
            char* buf = new char[size];
            if ( BMU_GetXml(id, buf, size) == BMU_OK ) {
                // All done!
            }
            delete[] buf;
        }
    }
}
```

ERP Integration Tools
eTFS Data Model

Last revision: 23.10.2017

39 pages including the front page

Document No. 155.388.958

eTFS is short for an electronic system for the transfrontier shipment of waste. It is used in this sense throughout this document.

Technical documentation

InfoTech
Gesellschaft für Informations- und Datentechnik mbH

1 Introduction

1.1 General

This technical document describes the XML structure used to electronically persist information found on the official paper documents conforming to regulation (EG) No. 1013/2006. Currently the following official paper forms are supported by this specification:

- Notification document
- Movement document
- Consignment Information

1.2 Applied standards

Several specifications are incorporated to define the overall document structure. It will be described in greater detail later in this document. To provide an overview, here is a list of the employed standards:

1.2.1 eTFS[1]

eTFS defines a framework for embedding EUDIN conforming documents and provides facilities for signatures, file attachments, arbitrary data and document layers. Layers provide historical information in a single document. The root element is named *Abfallverbringungsdokument*.

1.2.2 EUDIN[1]

The EUDIN specification defines a superset of fields with regard to the official paper forms. The structures will be used to store the forms data electronically.

1.2.3 TFS schema

In a few cases the EUDIN specification lacks some support for features. The supplemental tfs schema adds these missing features.

1.3 Systems currently using eTFS

This document is part of the documentation “ZEDAL ERP[2] Integration Tools” (http://www.zedal.com/Download/ERP_integration_tools.pdf).

ZEDAL is a document management system for waste disposal in Germany and the transboundary movement of waste (TFS) regulated by BASEL, OECD und EU law. TFS is based on the eTFS specification and is used in production since 2013.

ZEDAL ERP Integration Tools is a set of tools designed to integrate external ERP Systems and software systems into ZEDAL.

[1] See section 4. Standards

[2] Enterprise Resource Planning, Term used for software components that model the whole business process in electronic form.

Infotech provides the ERP Tools and the eTFS documentation under a free license without fee for general use.

2 Data

2.1 General information

2.1.1 XML fundamentals

The documents use **XML 1.0** and are encoded in **UTF-8.**

The use of default-namespaces is prohibited!

It is not possible to employ a validating XML Parser if electronic signatures are to be used. Otherwise the signature will become invalid due to **Whitespace Normalization**.

You can use whitespaces in encoded form, for example: **line feeds (
).**

If you plan to validate documents please keep in mind that elements inside a *FreieXMLStruktur* won't be automatically validated. Depending on the used software, contents inside those nodes have to be schema checked separately.

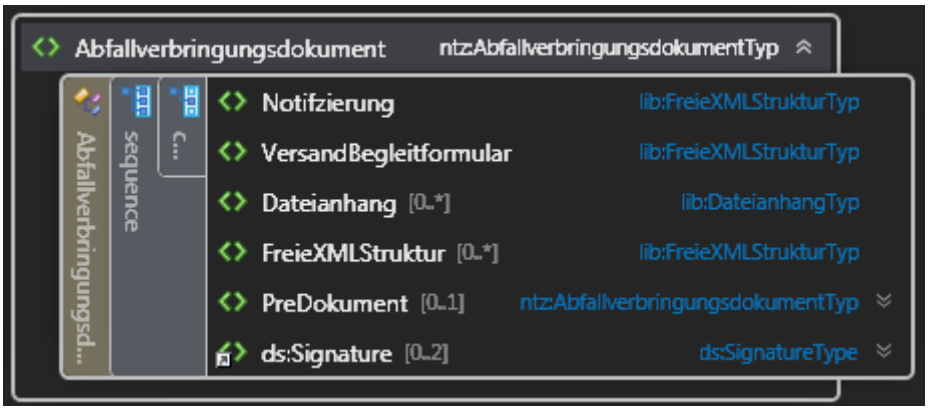
2.1.2 Document types

Three different types are supported:

- Notification document
- Movement document
  - Transport announcement
  - Confirmation receipt
  - Confirmation disposal
- Consignment Information

2.2 Abfallverbringungsdokument

Schema file: **Notifizierung.xsd**



The element *Abfallverbringungsdokument* encloses the EUDIN-specified form data. The EUDIN-structures are embedded either in the element *Notifzierung* or *VersandBegleitformular*.

Attention: the element Notifzierung contains a typing error (an i is missing between f and z). Because of the huge number of existing documents this can't be changed.

The element *Dateianhang* can transport arbitrary BASE64 encoded binary information. Please keep the size in mind to maintain a low load for communication and archiving.

The element *FreieXMLStruktur* allows transmitting specific data that are not defined by the schema. All involved parties have to come to an arrangement since unknown structures usually get ignored. The attribute *NamespaceURI* can distinctly identify such a structure.

Notably, the FreieXMLStruktur is used to incorporate the elements described by the TFS-Schema. It adds elements that are missing in EUDIN to fully represent the official forms.

The *PreDokument* captures the history of a document and is an important element in regard to the signature. If a document is signed you can no longer change the document as the signature would break.

The *PreDokument* can contain a signed document and a further new document encloses the already signed area. Thus, the data of both versions is available and allows for a retrospection.

Important: because of this the PreDokument and the Document have to be of the same type. This restriction is not enforced by the schema.

The up to two *Signature* elements contain electronic signatures adhering to ***XML DSig***. Signing a document prevents the undetected modification of the enclosed data.

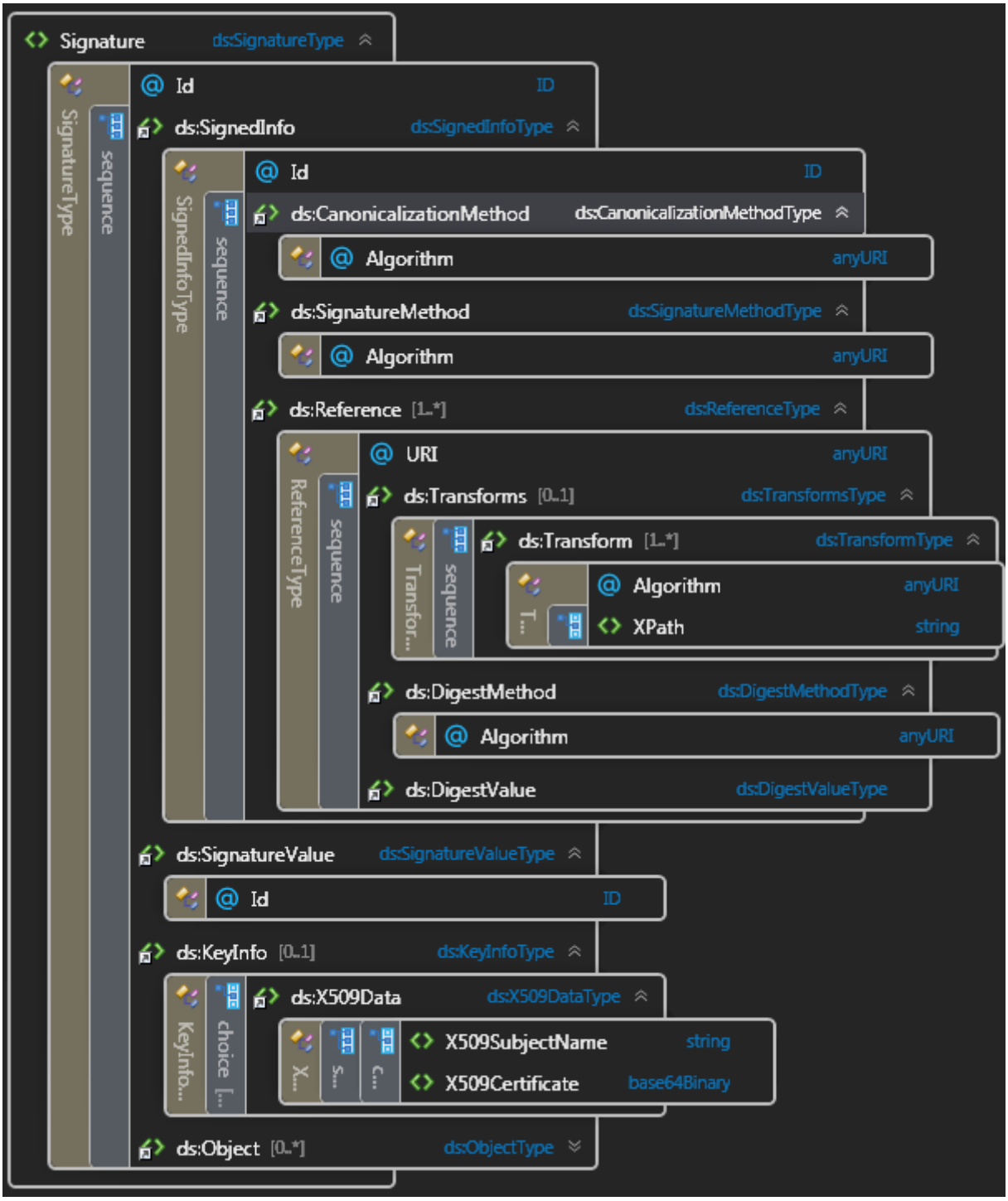
2.2.1 Signatures

Electronic signatures have to be applied adhering to the following standards:

- XML DSig, W3C Recommendation (2008-06)
- ETSI EN 319 132-1 V1.1.1 (2016-04)

This is just a short sketch covering the signature requirements. Find a full description in the ***XML DSig*** standard.

This image shows the elements required to create a valid signature:



The attribute *Id* (element S*ignature*) is formed by the role abbreviation and the UUID. Possible roles are ERZ (waste generator), BEF (Carrier) and ENT (Disposal facility).

Example: ENT-c413e993-f9d6-47d0-9bcc-9cce9dc1dce7

This XML-file excerpt shows how a XML signature is composed. The attribute *Algorithm* is assigned constants that define how signature information was build and how the validation should be done.

The *Reference* element specifies which part of the XML will be enclosed by a signature. Use the following XPATH-filtering 2 expressions for the signature:

Without enclosing a second signature:

```
<XPathFiltering Filter="intersect">here()/ancestor::*[6]<XPathFiltering>
<XPathFiltering Filter="subtract">here()/ancestor::*[6]/*[namespace-
uri()='http://www.w3.org/2000/09/xmldsig#' and local-
name()='Signature']</XPathFiltering>
```

Or with enclosing a second signature

```
<XPathFiltering Filter="intersect">here()/ancestor::*[6]<XPathFiltering>
<XPathFiltering Filter="subtract">here()/ancestor::*[5]/following-
sibling::*[namespace-uri()='http://www.w3.org/2000/09/xmldsig#' and local-
name()='Signature']</XPathFiltering>
```

As you can see there are two *Reference* nodes. The first encloses the entire document (without comments), the second encloses the *QualifyingProperties explained below*, that were created adhering to the ***XAdES*** specification.

X509SubjectName has to contain the distinguished name from the certificate (*X509Certificate)* that was used to create the signature.

The validation of a signature consists of following steps:

1. Formal checks: accumulate all relevant data and judge if requirements are fulfilled.
2. Core validation (see XML DSig): make sure that signed content was not manipulated.
3. Certification path construction and verification: evaluate if the signing certificate can be traced back to a valid trust anchor and if it was valid when the signature was created.
4. OCSP-check: use an online query to ensure that the certificate was known and not revoked at the time of claimed signature creation.

The details of the signature validation process are beyond the scope of this document.

2.2.2 XML Example Signature

```
<ds:Signature Id="ENT-c413e993-f9d6-47d0-9bcc-9cce9dc1dce7">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#WithComments"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"/>
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        <ds:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
          <xpf:XPath xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2"
Filter="intersect">here()/ancestor::*[6]</xpf:XPath>
        </ds:Transform>
        <ds:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
          <xpf:XPath xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2" Filter="subtract">
here()/ancestor::*[5]/following-sibling::*[namespace-
uri()='http://www.w3.org/2000/09/xmldsig#' and local-name()='Signature']]</xpf:XPath>
        </ds:Transform>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#WithComments"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
      <ds:DigestValue>
          aAQ7R0AeQXGLXXjoc0luC6mCPvSiyloCPL8x6WqM2ZiRAqKb6Ckvo+TMg715WnllXmVG83cF
          hGCbFoOPHVeU4A==
      </ds:DigestValue>
    </ds:Reference>
    <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties" URI="#SignedProperties-
20120727110027Z1">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
```

```
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
      <ds:DigestValue>
          … calculated, base64 encoded …
      </ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
    … calculated …
  </ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
          … certificate data, base64 encoded …
      </ds:X509Certificate>
      <ds:X509SubjectName>C=DE,O=Infotech GmbH,OU=Fussballer,CN=Müller\,
Thomas,serialNumber=1</ds:X509SubjectName>
    </ds:X509Data>
    <ds:KeyValue>
      <ds11:ECKeyValue xmlns:ds11="http://www.w3.org/2009/xmldsig11#">
        <ds11:NamedCurve URI="urn:oid:1.3.36.3.3.2.8.1.1.7"/>
        <ds11:PublicKey>
          … certificates public key, base64 encoded …
        </ds11:PublicKey>
      </ds11:ECKeyValue>
    </ds:KeyValue>
  </ds:KeyInfo>
  <ds:Object>
    … XAdES structure, see below …
  </ds:Object>
</ds:Signature>
```

You also need to fill the *KeyValue* element with the signature parameters (RSA, DSA or ECDSA). This is intended to speed up the validation procedure, because its information would have to be extracted from the certificate otherwise. This might potentially cause slowdowns in a mass processing environments.

The following structure, adhering to ***XAdES***, is embedded to save the signature time and to protect the certificate from being exchanged unnoticed. One important attribute is *Target*, which contains a local reference to *Signature* element, the *QualifyingProperties* belong to.

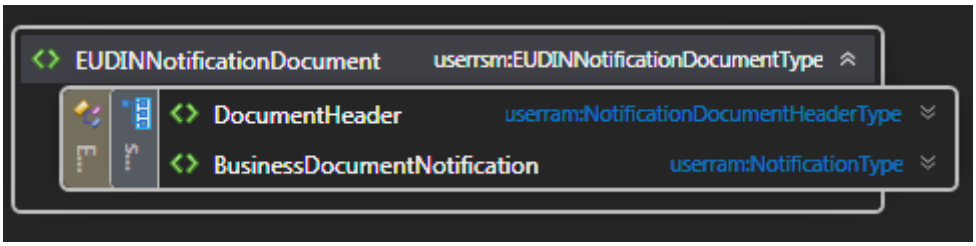
```
<dsx:QualifyingProperties xmlns:dsx="http://uri.etsi.org/01903/v1.3.2#" Target="#ENT-c413e993-
f9d6-47d0-9bcc-9cce9dc1dce7">
  <dsx:SignedProperties Id="SignedProperties-20120727110027Z1">
    <dsx:SignedSignatureProperties>
      <dsx:SigningTime>2012-07-27T11:00:27Z</dsx:SigningTime>
      <dsx:SigningCertificateV2>
        <dsx:Cert>
          <dsx:CertDigest>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
            <ds:DigestValue>
              … hash value of signing certificate, base64 encoded …
            </ds:DigestValue>
          </dsx:CertDigest>
          <dsx:IssuerSerialV2>
            … certificates issuer and serial, base64 encoded …
          </dsx:IssuerSerialV2>
        </dsx:Cert>
      </dsx:SigningCertificateV2>
    </dsx:SignedSignatureProperties>
  </dsx:SignedProperties>
</dsx:QualifyingProperties>
```

2.3 Recurring definitions in EUDIN

The ***EUDIN-specification*** contains basic structures that are used regularly.

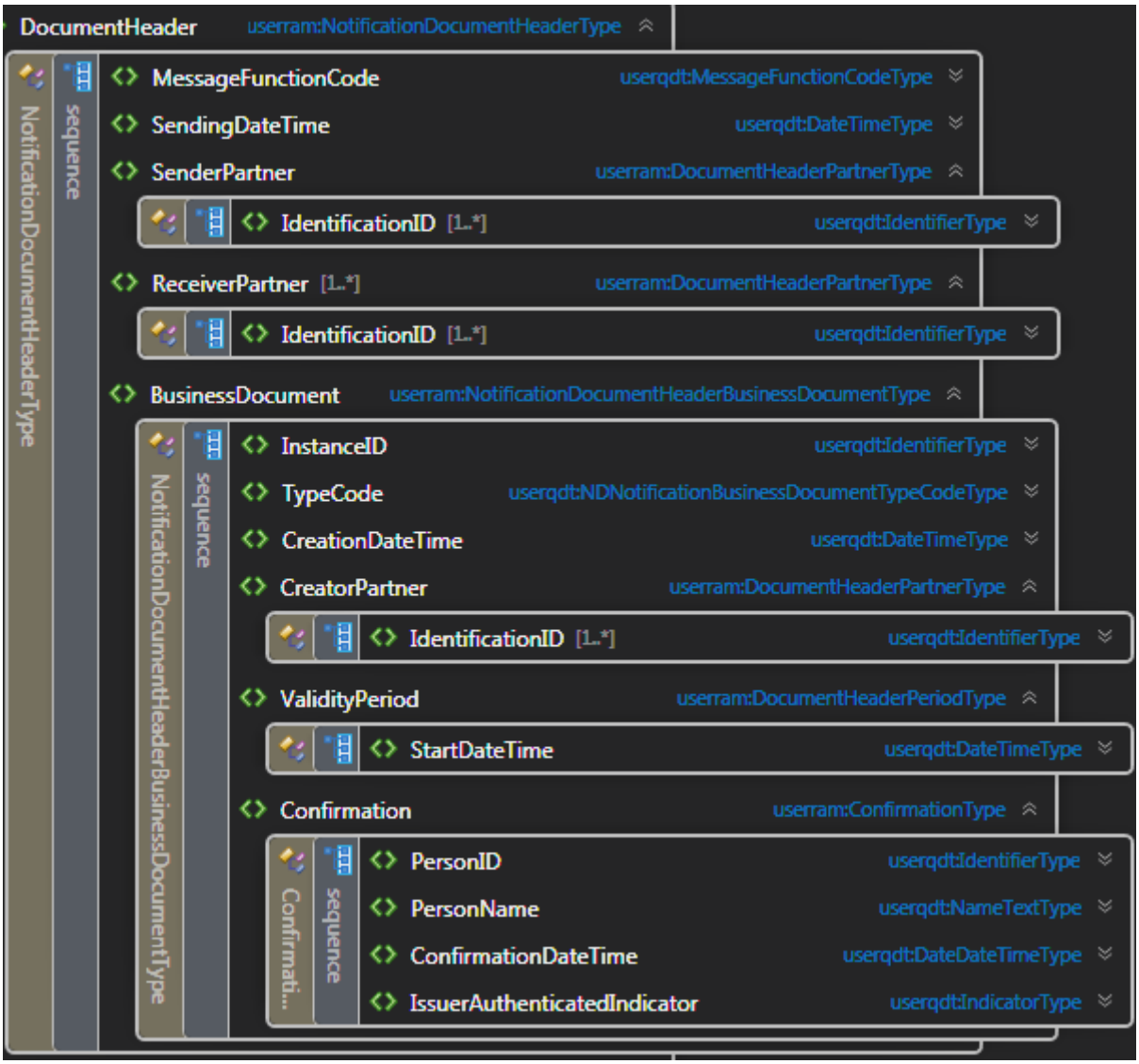
2.3.1 EUDIN-documents

The waste documents generally consist of two parts, illustrated using the example of the notification document:



2.3.2 DocumentHeader

The *DocumentHeader* is currently not processed and should be ignored but it has several mandatory fields:



Proposed standard values to satisfy the schema:

XML path	Value
MessageFunctionCode	ORI
SendingDateTime	<current time>
SenderPartner/IdentificationID	ID_SENDERPARTNER
ReceiverPartner/IdentificationID	ID_RECEIVERPARTNER
BusinessDocument/InstanceID	ID_INSTANCE
BusinessDocument/CreationDateTime	<current time>
BusinessDocument/CreatorPartner/IdentificationID[1]	ID_CREATORPARTNER
BusinessDocument/ValidityPeriod/StartDateTime	<current time>
BusinessDocument/Confirmation/PersonID	CONFIRM_PERSON
BusinessDocument/Confirmation/PersonName	-
BusinessDocument/Confirmation/ ConfirmationDateTime	<current time>
BusinessDocument/Confirmation/IssuerAuthenticatedIndicator	False

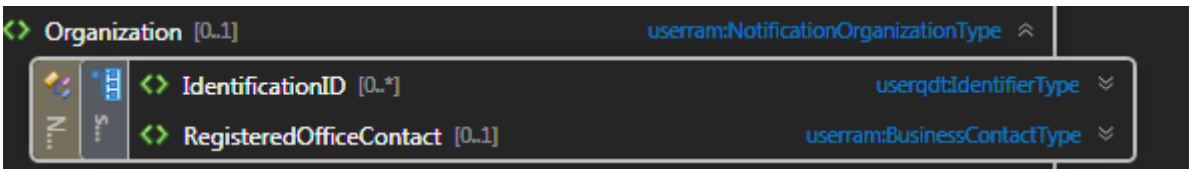
The element *BusinessDocument/TypeCode* is filled with respect to the actual document type.

Type	EUDIN xml element	value
Notification document	EUDINNotificationDocument	ND
Movement document	EUDINWasteMovementDocument	MD
Confirmation receipt	EUDINCertificateOfWasteReceiptDocument	COWR
Confirmation disposal	EUDINCertificateOfWasteRecoveryDisposalDocument	COWD

2.3.3 BusinessDocument

Every party involved in the waste process has a corresponding element in the *BusinessDocument* element. These elements contain always two elements called *Person* and *Organization*. ZEDAL International only use *Organization*, entries in *Person* will be ignored.

2.3.4 Organization



2.3.5 IdentificationID

There can be an arbitrary number of IDs for every involved party with a maximum length of 30 characters. To describe the type of number use the attribute schemeURI, as well as these two URNs:

urn:id:de:bmu:eanv (authority ID adhering to German waste legislation)

urn:id:national:registrationnumber (international numbers)

For documents send to german authorities both types of numbers are required. If only a single number exists it must be duplicated to fulfill this constraint.

2.3.5.1 Communication data

Element: *RegisteredOfficeContact*

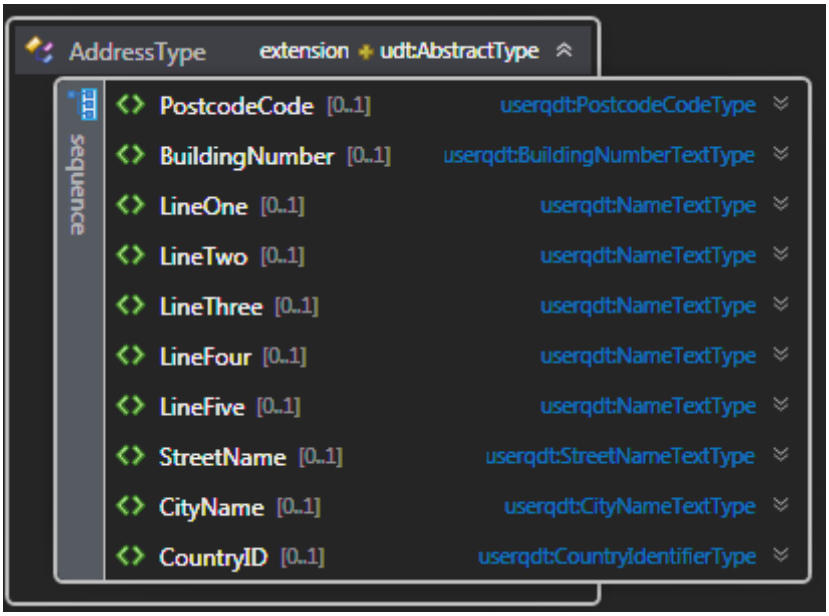
The element *RegisteredOfficeContact* contains the contact person and its communication data, as well as adress data, which will be described in detail below.

For contact person the Email-address (*EmailIdentificationID*) forename and surname of the person (*GivenName*, *FamilyName*) are saved. Furthermore, phone number (*OfficeCommunication*) and fax number (*FaxCommunication*) are used.

2.3.5.2 Address

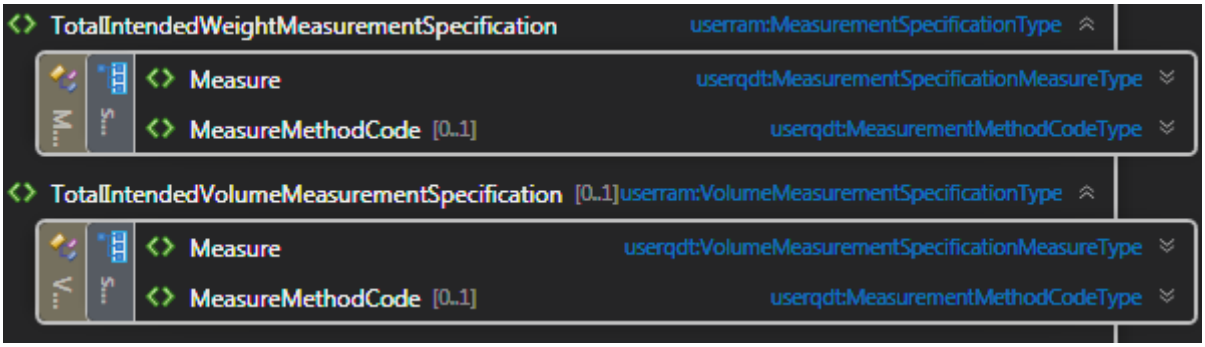
Element: *RegisteredOfficeContact/Address*

These are the fields to capture addresses:



2.3.6 Weight/volume

Weights and volumes are described in the following element, in this example for the notification. Weight gets entered into the *Measure* element. The mandatory element *unitCode* defines the measuring unit. Always use **TNE** for weight and **MTQ** for volume. Please note that weight is a mandatory field while volume is optional. MeasureMethodeCode is optional, too.



2.3.7 Areas not covered by EUDIN

2.3.7.1 Fields missing in EUDIN

Fields from the official forms that aren't covered by the EUDIN-specification are added in a separate definition, the TFS-schema.

2.3.7.2 Definitions in EUDIN that are incompatible

Some fields in the official forms are encoded, but this encoding is incompatible with the EUDIN standard. Examples include the packaging types, physical characteristics and means of transport. There is a lookup table defined for those.

2.3.7.3 Voucher / mixed operation paper/electronic

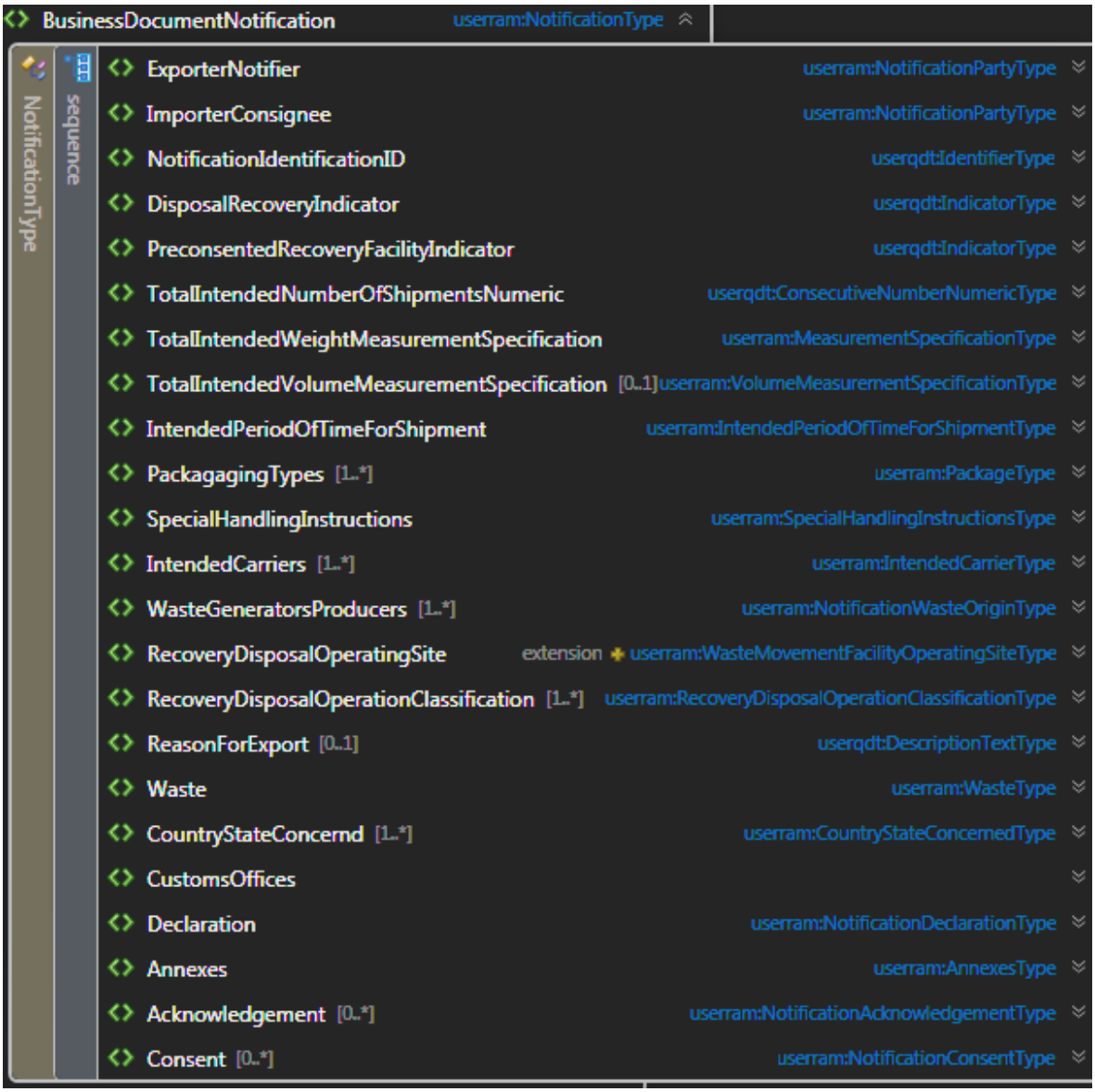
It might be impossible to operate electronically for the whole disposal chain. There are fields in the electronic documents to signal if certain data was recorded on paper.

There are fields to indicate the existence of a hand written signature, the date of the signature, the signee and the stamps on the paper document.

2.4 Notification document

Schema file: EUDINNotificationDocument-2.1.xsd

Element: *BusinessDocumentNotification*



The areas of the official notification are assigned to the XML nodes in the following sections.

2.4.1 Exporter (1)

Element: *ExportNotifier*

This area uses the element *Organization* for address and contact data (described above)

2.4.2 Importer (2)

Element: *ImporterConsignee*

See Exporter

2.4.3 General information (3)

2.4.3.1 Individual/Multiple shipments

This information is computed based on the number of transports. There is no element in the xml storing this information directly.

2.4.3.2 Disposal/Recovery

Element: *DisposalRecoverIndicator*

True means disposal, **false** means recovery.

2.4.3.3 Preconsented Facility

Element: *PreconsentedRecoveryFacilityIndicator*

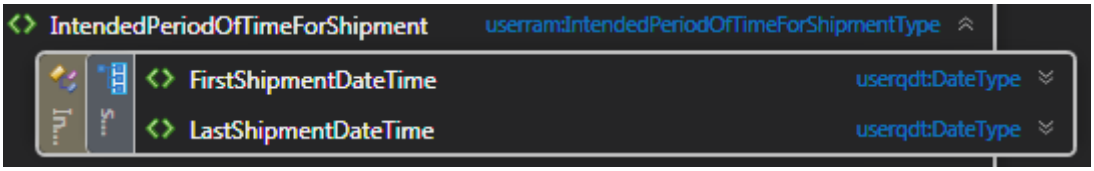
2.4.4 Amount of shipments(4)

Element*: TotalIntendedNumberOfShipmentsNumeric*

2.4.5 Weight/Volume (5)

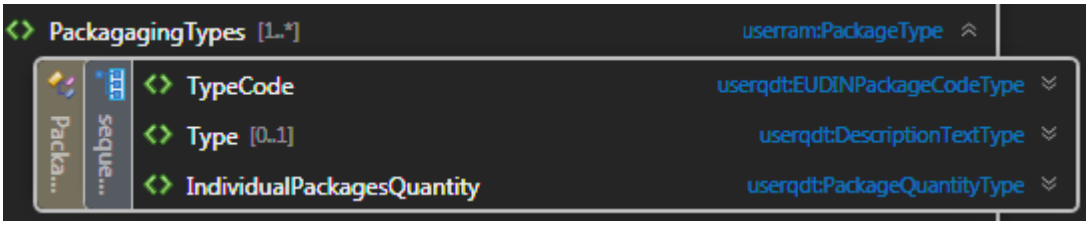
The weight is inserted into the element *TotalIntendedWeightMeasurementSpecification a*nd the Volume in the element *TotalIntendedVolumeMeasurementSpecification*.

2.4.5.1 Intended period of time for shipments (6)



First and last shipment date are to be entered here.

2.4.6 Packaging types (7)



At least one element PackagingTypes has to be defined. The element *TypeCode* contains the packaging type and is encoded like this:

EUDIN code	**official code**	**meaning**
DR	1	Drum/Barrel
W1	2	Wooden barrel
2C	3	Jerrycan
BX	4	Box
BG	5	Bag
W2	6	Composite packaging
W3	7	Pressure receptable
W4	8	Bulk
W5	9	Other (specify)

If type **Other (9)** is chosen you have to add a description for the type of packaging. This description is expected in the element *Type*.

For types 1-8 leave this blank.

IndividualPackagesQuantity covers the amount of packages.

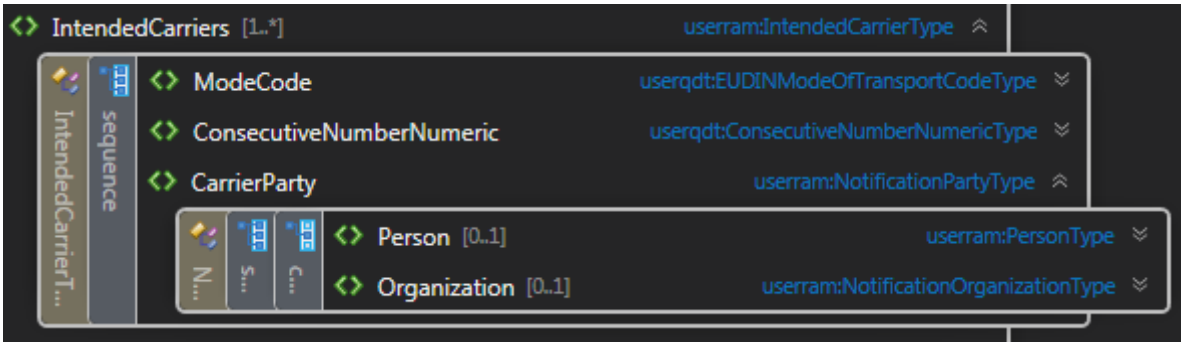
2.4.6.1 Special handling requirements

Element: *SpecialHandlingInstructions/SpecialHandlingRequiredIndicator*

This mandatory element has to be designated either true or false, depending on the existance of special handling instructions. This is not an element of the *PackagingTypes* element, but a sub element of *BussinesDocumentNotification*.

2.4.7 Intended carriers (8)

Element: *IntendedCarriers*



At least one carrier is required. The consecutive number of the carrier is entered in *ConsecutiveNumberNumeric*. The *CarrierParty* and *Organization* elements are used for the registration number, the address and contact data.

The means of transport has to be given:

Element: *IntendedCarriers/ModeCode*

EUDIN code	**official code**	**meaning**
4	A	air
3	R	road
1	S	sea
2	T	rail
8	W	Inland waterways

2.4.8 Waste generator(9)

Element: *WasteGeneratorsProducers*

By default: register number, address and contact data.

2.4.8.1 Site and process of generation

This is expected in *WasteOrigin.Description.*

2.4.9 Recovery/Disposal facility (10)

Element: *RecoveryDisposalOperatingSite*

The disposal facility has the specialty that the *Organization* element is not placed directly in the element for the involved party but under the element of *OperatingParty*.

2.4.9.1 Disposal or recovery facility

Element: *DisposalRecoveryFacilityIndicator*

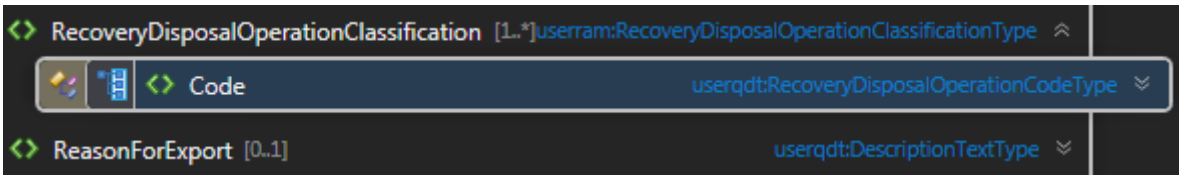
Set this element to **True** for disposal facility or **false** for recovery facility.

2.4.9.2 Place of actual recovery/disposal

Element: *OperatingParty/Comment*

2.4.10 Disposal/Recovery operation(s) (11)

Those can be loosely found in the element *BusinessDocumentNotification*.



2.4.10.1 D-Code / R-Code

Element: *RecoveryDisposalOperationClassification*

At least one code has to be entered, any number is allowed.

2.4.10.2 Reason for export

Element: *ReasonForExport*

2.4.10.3 Technology employed

This field is currently not defined in EUDIN. It is defined in the TFS structure, see *tfs/EUDINNotificationDocument/TechnologyEmployed.*

2.4.11 Designation and composition of the waste (12)

Element: *Waste/Description*

Usually, the BASEL or OECD catalogue text is entered here. EUDIN only allows for 256 characters and is therefore defined too short for some of the longer entries.

2.4.12 Physical characteristics (13)

Element: *Waste/ PhysicalCharacteristicCode*

The EUDIN encoding deviates from the official forms, so please use this conversion table:

EUDIN code	**official code**	**meaning**
POW	1	Powder/powdery
SOLMON	2	solid
SLUVISC	3	Viscous/pasty
SLU	4	sludgy
LIQ	5	liquid
GAS	6	gaseous
OTHER	7	Other (specify)

2.4.13 Waste classification (14)

Element: *Waste/WasteClassification*

At least one *WasteClassification* element has to be defined. The classicification consists of the *Code* and a *CodeListIdentifier*. The following table shows the correlation between the content of the *CodeListIdentifer* attribute and the waste identification code in the official forms:

Official form		**Code**
(i)	Basel Annex VIII	BASEL
(ii)	OECD-Code	OECD
(iii)	EU list of waste	EWC
(iv)	National Code in country of export	NATEXP
(v)	National Code in country of import	NATIMP
(vi)	Other (specify)	OTHER
(vii)	Y-Code	YCODE
(viii)	H-Code	HCODE
(ix)	UN-class	UNCLA
(x)	UN-number	UNNUM
(xi)	UN-shipping name	UNSHIP
(xii)	Customs code	CUST

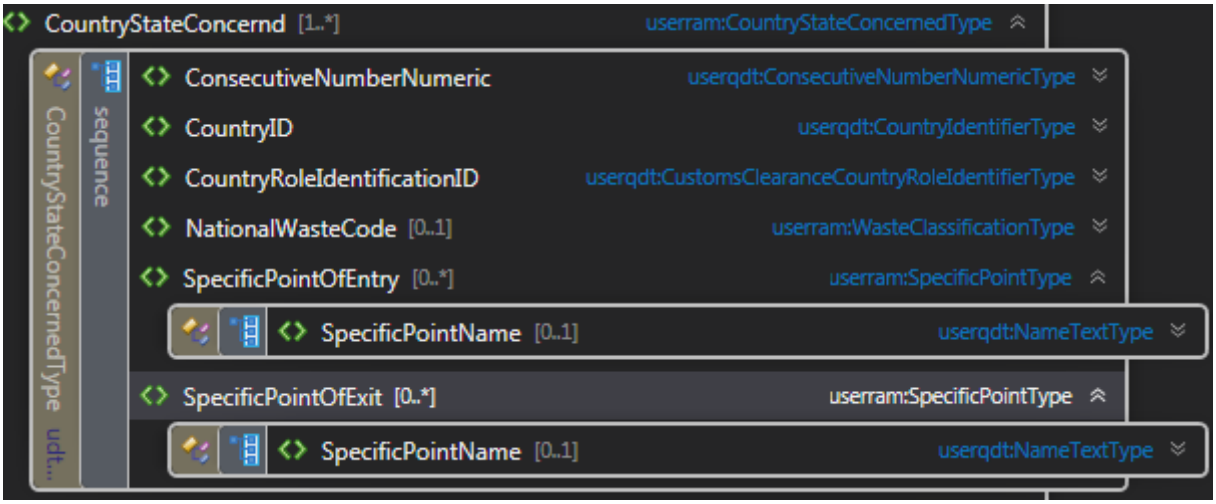
Enter the corresponding code from the waste catalogue into *Code*.

Attention: UN-class 9 has four different definitions. This poses no problem for the form (printout) itself, but electronic processing must be unique. If the UN-class contains the value 9 the content of *comment* decides which class exactly is meant:

H-Code	**Description UN-class 9**	**Content of**
H10	Liberation of toxic gases in contact with air or water	**1**
H11	Toxic (delayed or chronic)	**2**
H12	Ecotoxic	**3**
H13	Waste *capable by any means*, *after disposal*, of	**4**

2.4.14 Countries/states concerned (15)

Element: *CountryStateConcerned*



The element *ConsecutiveNumberNumeric* is used for consecutive numbering.

CountryID contains a country code adhering to ISO-3166 and is entered to line (a) in the official form.

In *CountryRoleIdentificationID*, please enter **T**(ransit), **E**(xport) or **I**(mport).

The *NationWasteCode* contains the national waste code that can be found in line (b) of the official form.

SpecificPointOfEntry and *SpecificPointOfExit* are optional parameters that contain the entry and exit customs office. They are placed in the subordinated element *SpecificPointName* (Line (c) in the official form)

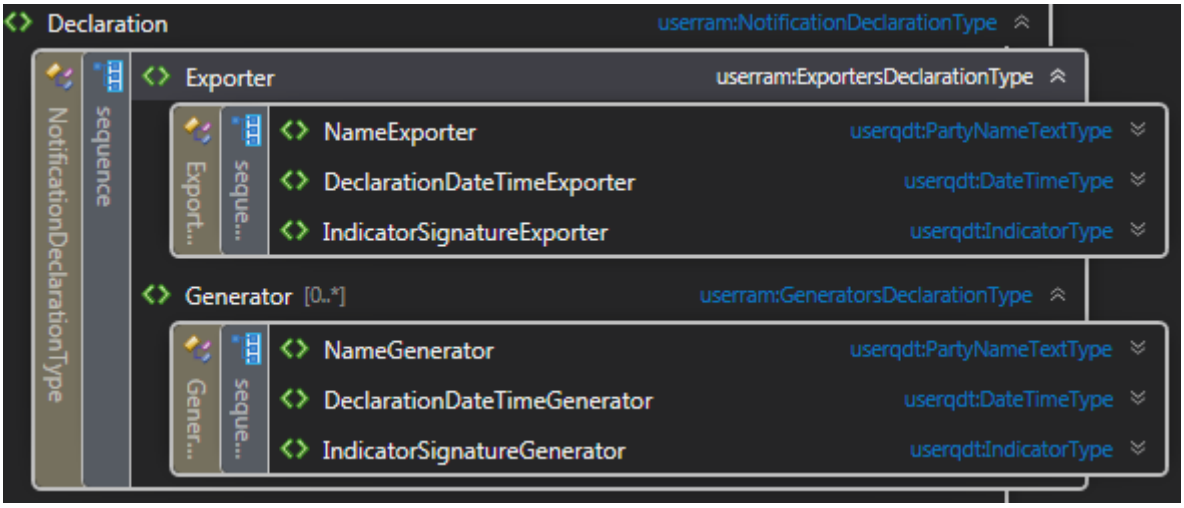
2.4.15 Customs offices entry and/or exit and/or export (European Community) (16)

Element: *CustomOffices*

You can enter involved countries outside of the EU here.

2.4.16 Declaration of the exporter (17)

Element: *Declaration*



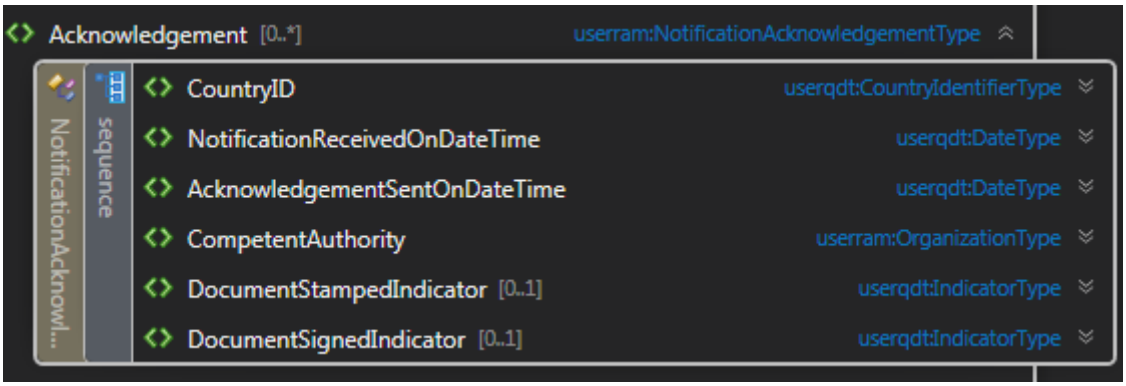
Contains the name (*NameExporter)*, date and signature (*DeclarationDateTimeExporter*) and the indicator if the declaration and the signature are existing. For the exporter this information is mandatory while it is optional for the waste generator. Because of the limited space on the official form there usually is only one waste generator, although an arbitrary number of generators are allowed.

2.4.17 Number of annexes (18)

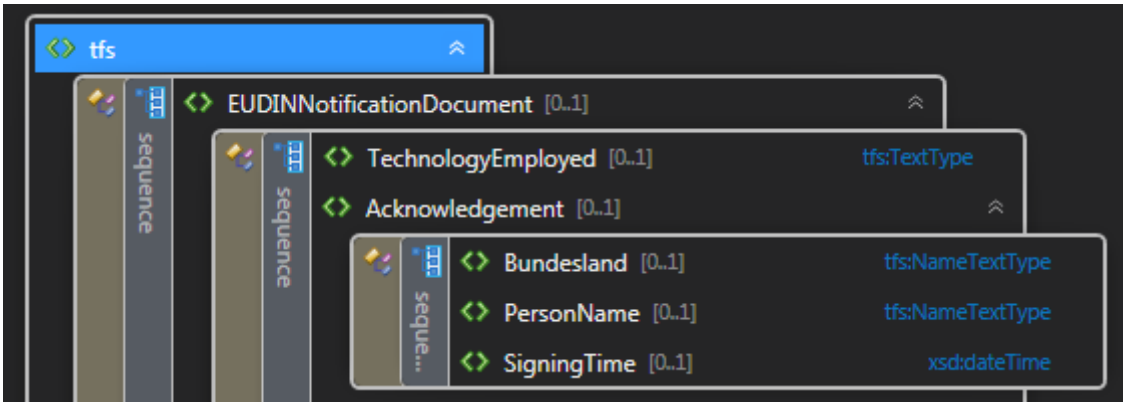
Element: *Annexes/NumerOfAnnexesNumeric*

2.4.18 Acknowledgement(19)

Element: *Acknowledgement*



CountryID is filled with the country in *ImportConsignee*. *NotificationReceivedOnDateTime* and *AcknowledgementSentOnDateTime* just use the time of the receipt of the Notification, as well as the time of acknowledgement sent on. *CompententAuthority* is filled with the name of the competent authority.



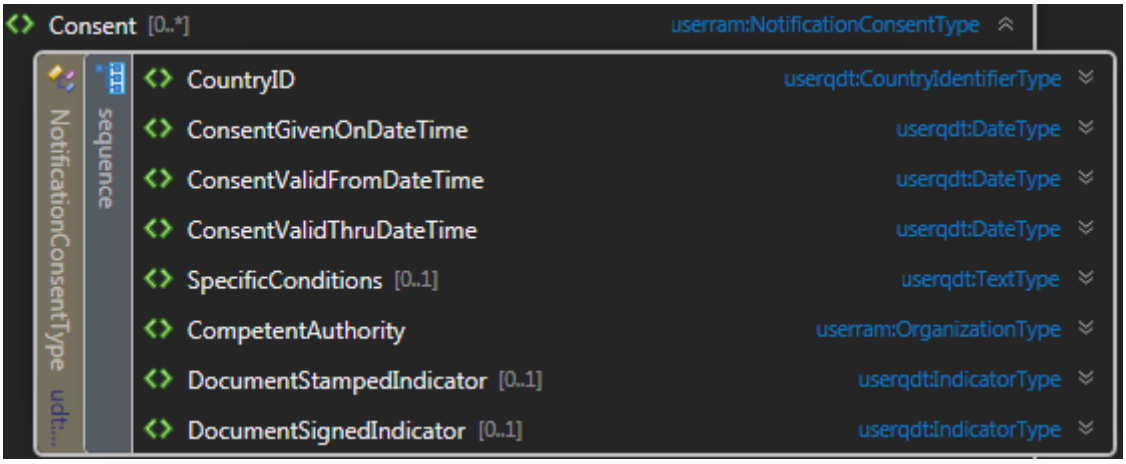
There are no fields for the name of the signee or the date of the signature in EUDIN.

The TFS-schema provides the fields *Acknowledgement/PersonName* and *SigningTime*.

There is one additional element called *Bundesland* to define the federal state.

2.4.19 Consent (20)/Specific conditions (21)

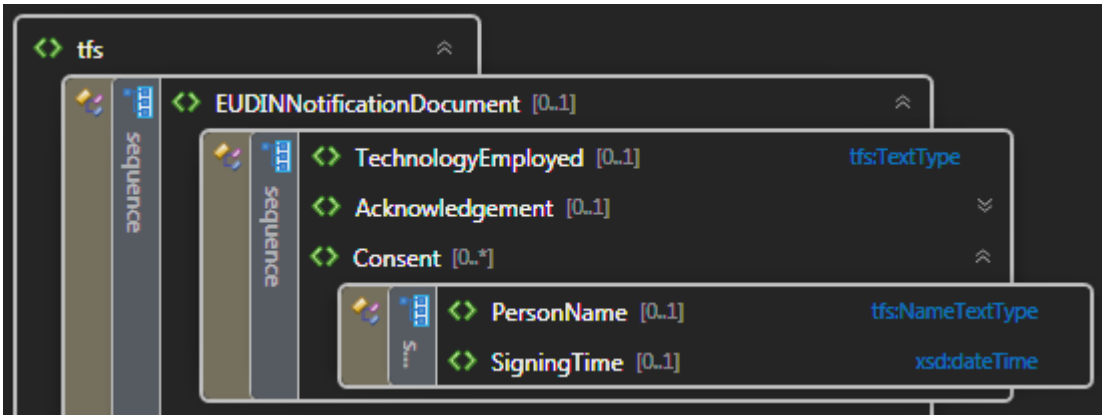
Element: *Consent*



CountryID is filled with the exporter‘s country. The date specifications in *ConsentGivenOnDateTime*, *ConsentValidFromDateTime* and *ConsentValidThruDateTime* contains the information for *consent given*, *consent valid from* and *consent valid until*.

You can indicate specific condition (21) by setting the element *SpecificConditions*.

There are no fields for the signee name and date of signature in EUDIN.



Those have been added in the TFS-structure. The order in the TFS-structure has to match the order in the EUDIN-structure. There is no id to query the mapping between these two places.

2.5 Movement document

The movement document has a special structure. Using the XML, you can create several documents that correspond to the typical workflow of a transport.

- Transport announcement (WasteMovementDocument)
- Confirmation receipt (CertifcateOfWasteReceiptDocument COWR)
- Confirmation disposal (CertificateOfWasteRecoveryDisposalDocument COWD)

Initially, a transport announcment is created. Once the waste reaches the disposal facility the element *PreDokument* is created in the movement document and is filled with the contents of the transport announcement. Then the element *CertifcateOfWasteReceiptDocument* is created in the movement document. It contains the element *CorrespondingWasteMovement* that also gets filled with the data from the transport announcement. This course of action accomplishes two goals:

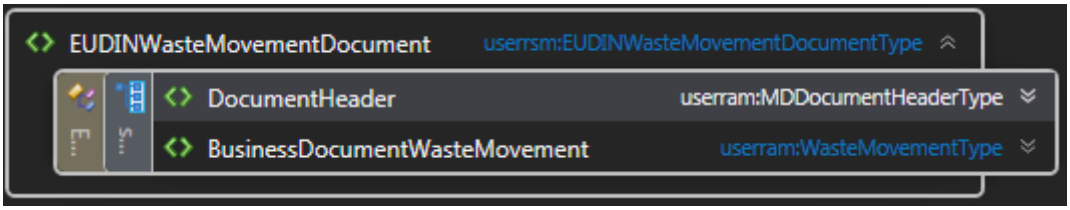
1. The total document contains the complete history of the form.
2. The topmost document always contains all data. Elements below the *PreDokument* don't need to be taken into account for the current status.

The concluding disposal confirmation is dealt with in the same way. A new PreDokument is created and filled with the contents of the disposal confirmation. To finish the operation the element *CertificateOfWasteRecoveryDisposalDocument* is created that also contains a structure to contain the confirmation receipt (CorrespondingCertificateOfWasteReceipt) including the transport announcement.

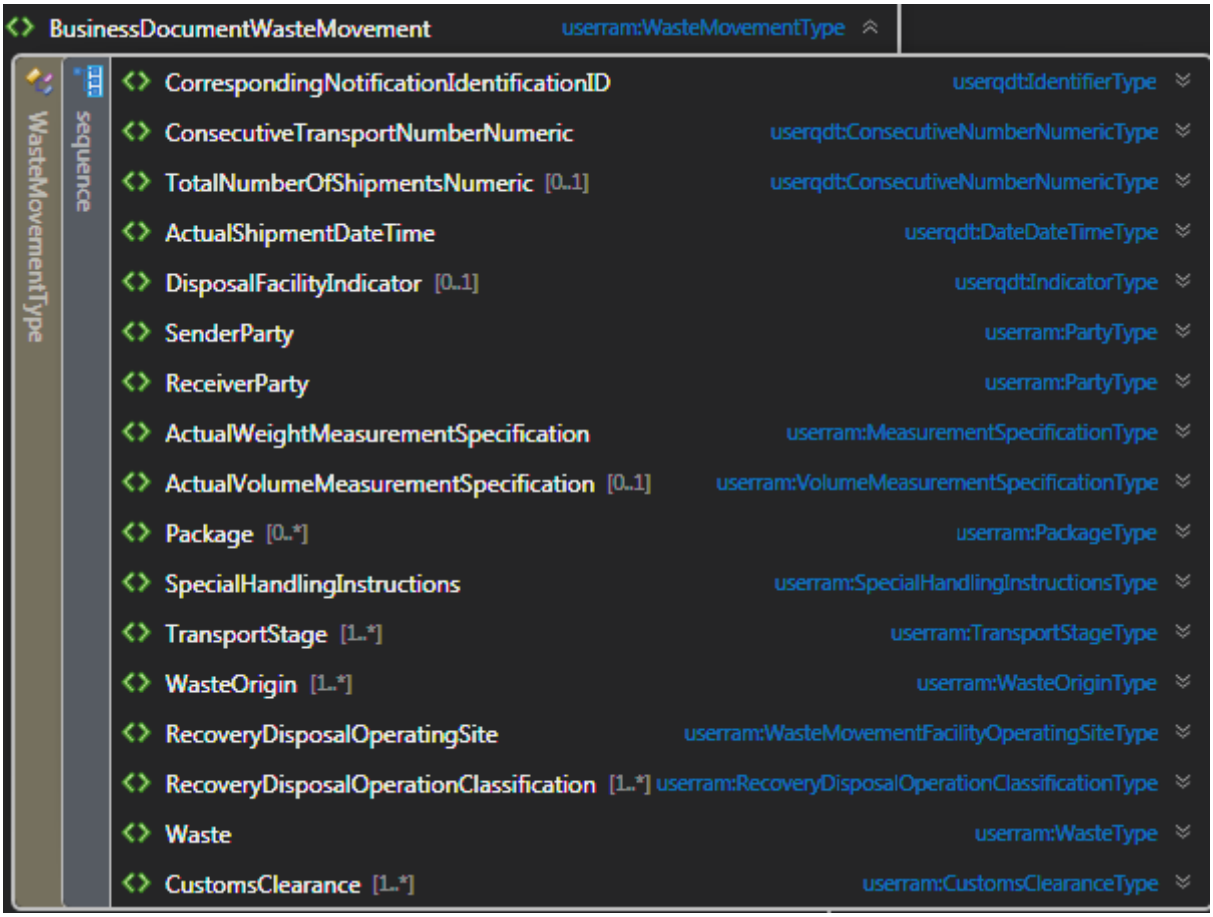
For all three form types fill the *DocumentHeader* analogous to the notification.

2.6 Transport announcement

Schema file: EUDINWasteMovementDocument-2.1.xsd



In element *BusinessDocumentWasteMovement*



2.6.1 Notification number (1)

The element *CorrespondingNotificationIdentificationID* stores the notification number.

2.6.2 Consecutive number / Total number of shipments (2)

The sequential number is entered in *ConsecutiveTransportNumberNumeric* and the total amount of shipments is entered in *TotalNumberOfShipmentsNumeric*.

2.6.3 Exporter (3)

Element: *SenderParty*

This is handled in the same way as the notification, fill the elements of the structure *Organization* here.

2.6.4 Importer (4)

Element: *ReceiverParty*

See Exporter.

2.6.5 Actual quantity(5)

The element *ActualWeightMeasurementSpecification* contains the actual weight. *ActualVolumeMeasurementSpecification* saves the volume. See Weigth/Volume.

2.6.6 Actual date of shipment (6)

Element*: ActualShipmentDateTime*

2.6.7 Packaging types (7)

Element: *Package*

Analogous to notification *PackagingTypes.*

2.6.8 Carriers (8)

Element: *TransportStage*

Largely matches *IntendedCarriers* in the Notification.

Additionally, the date of transfer is placed in *ActualDepartureDateTime*.

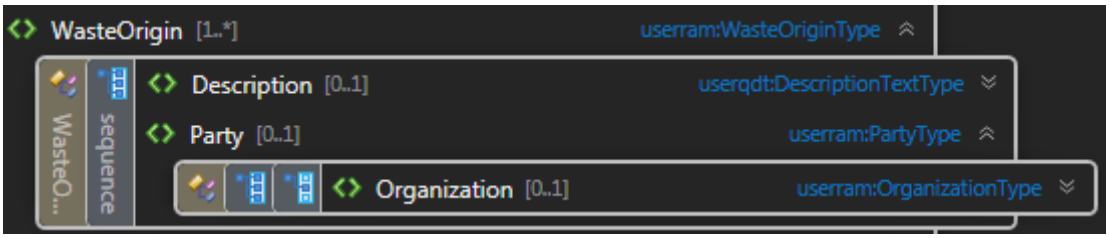
Confirmation contains the information to a potentially existing signature. Although *PersonID* is a mandatory element it isn't needed and usually filled with a dash (-).

PersonName, *ConfirmationDateTime* saves the name and the time of the signature.

IssuerAuthenticatedIndicator is used to indicate the existence of a handwritten signature on paper.

2.6.9 Waste generator/producer (9)

Element: *WasteOrigin*



Organization hides in the *Party* element. The place of the waste generation is placed in the element *Description*.

2.6.10 Recovery/Disposal facility (10)

Element: *RecoveryDisposalOperatingSite*

The element *DisposalFacilityIndicator* is not an element of the *RecoveryDisposalOperatingSite* structure. It is found in *BusinessDocumentWasteMovement* and defines the type of the disposal facility.

T**rue** means disposal facility, **false** indicates a recovery facility.

Address and contact data are placed in the *OperatingParty/Organization* element.

2.6.10.1 Site of generation

Element: *OperatingParty/Comment*

2.6.11 Disposal/recovery operation(s) (11)

2.6.11.1 D-Code / R-Code

Element: RecoveryDisposalOperationClassification

At least one code has to be entered, the maximum number is not limited.

2.6.12 Designation and composition of the waste (12)

2.6.13 Physical characteristics (13)

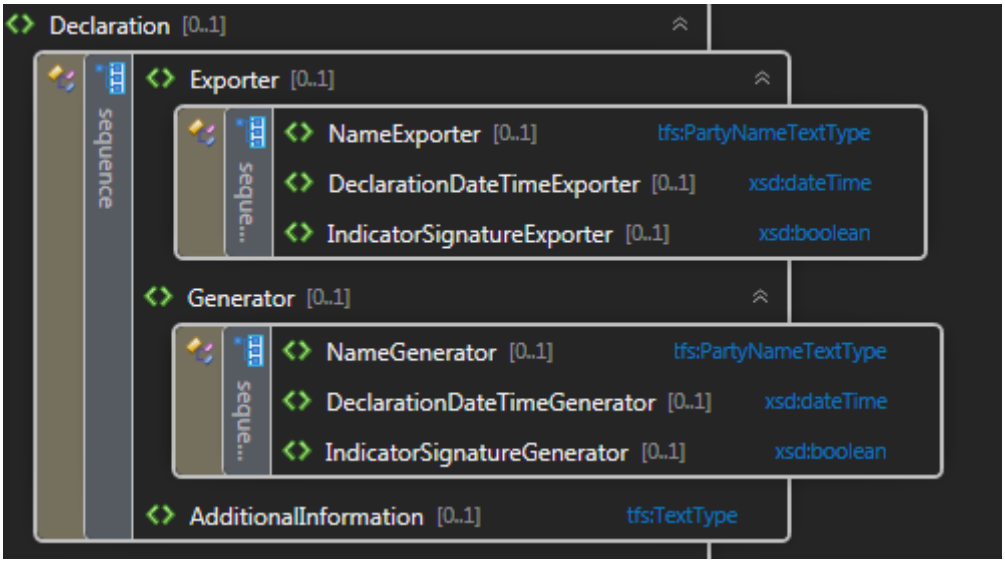
2.6.14 Waste identification (14)

Element: *Waste*

Areas 12-14 are analogous to the Notification.

2.6.15 Declaration of the exporter (15)

Element: *tfs/EUDINWasteMovementDocument/Declaration*



Place the data for Exporter or waste generator here.

2.6.16 For use by any person involved in the transboundary movement in case additional information is required (16)

Element: : *tfs/EUDINWasteMovementDocument/Declaration/AdditionalInformation*

The areas 17-19 are topics of separate documents and will be described later.

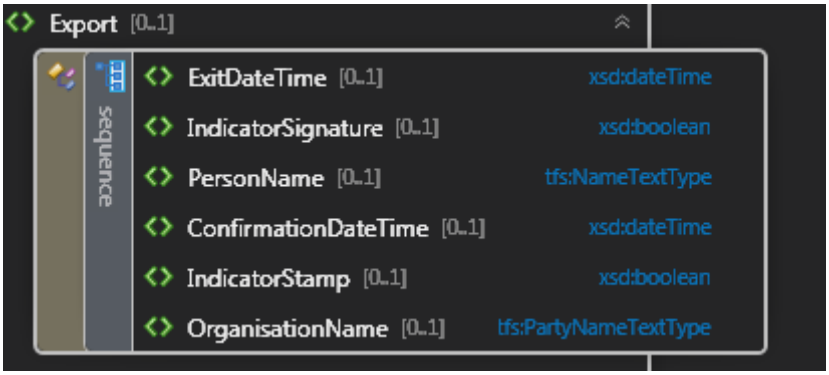
2.6.17 For use by customs offices (page 2)

The second page of the official form is used to capture information about customs offices. They are rarely used inside the EU. EUDIN defines the element *CustomsClearance* for this and it has to be used at least once. This causes contradictions. Because of this the element is not used in EUDIN and is filled with values that satisfy the schema but aren't processed.

Alternatively, the TFS schema defines elements for the areas that can be used instead.

2.6.18 Country of export(20)

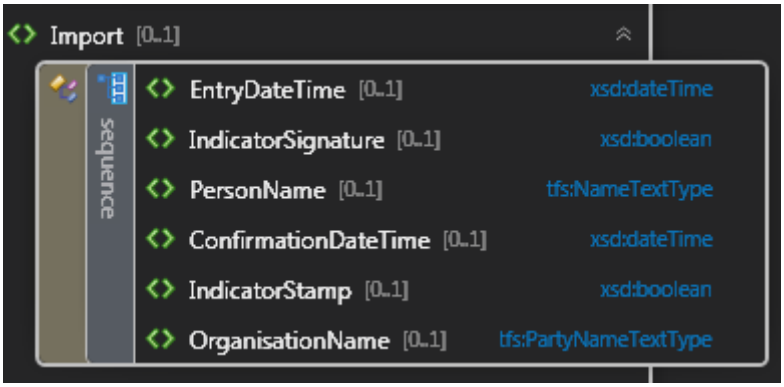
Element: *tfs/EUDINWasteMovementDocument/CustomsClearance/Export*



The element *ExitDateTime* is filled with the export date. *OrganizationName* equals the customs office. *PersonName*, *ConfirmationDateTime*, *IndicatorSignature* and *IndicatorStamp* are used to capture the signee, date of signature and the indicators for an existing signature and/or stamp.

2.6.19 Country of import (21)

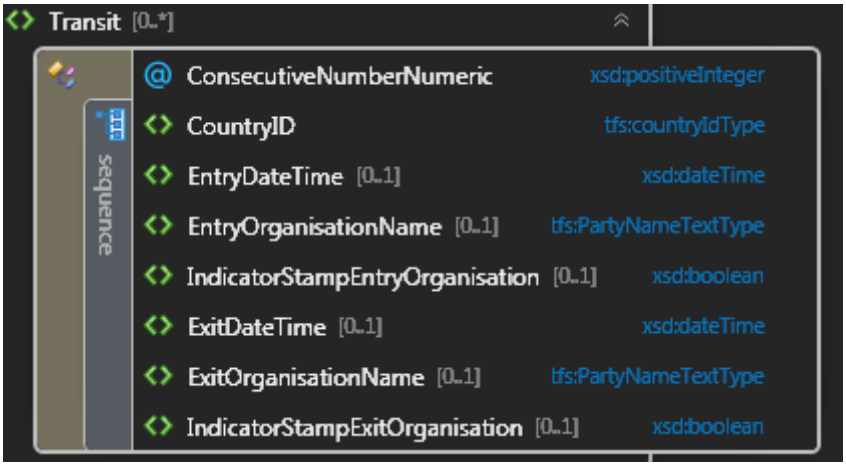
Element: *tfs/EUDINWasteMovementDocument/CustomsClearance/Import*



The import date is written to the element *EntryDateTime*. The other fields are intended for the voucher procedure.

2.6.20 Stamps of customs offices of transit countries (22)

Element: *tfs/EUDINWasteMovementDocument/CustomsClearance/Transit*

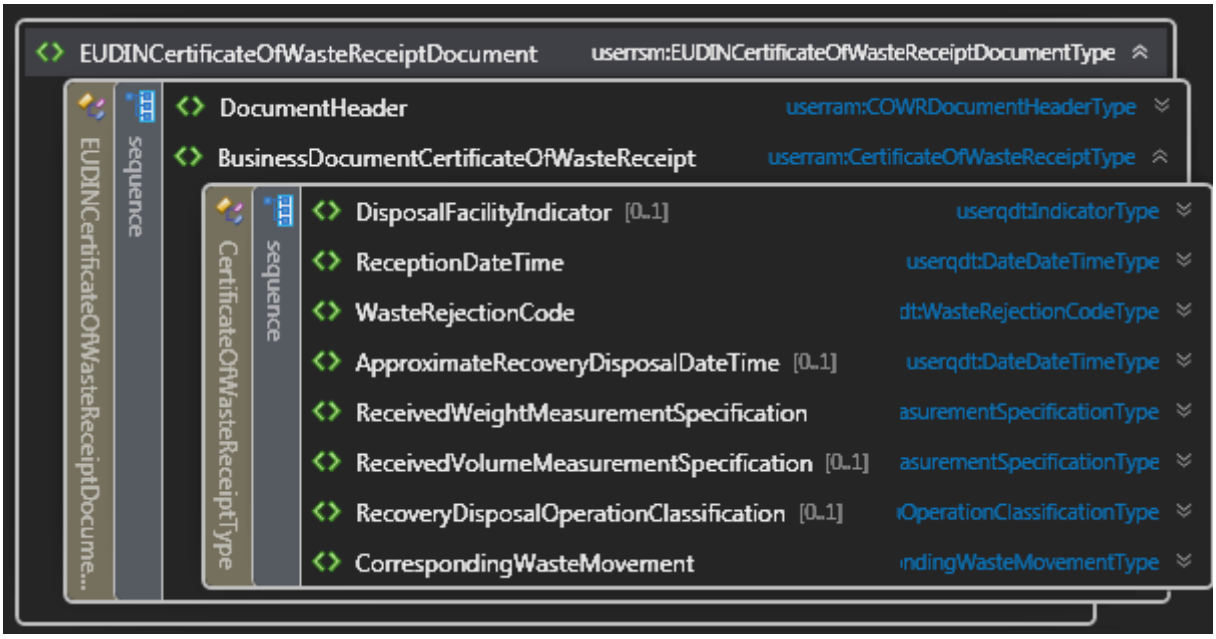


The element *ConsecutiveNumberNumeric* is a sequential number that is used for an allocation with an electronic signature. See extended roles.

CountryID contains the transit country encoded adhering to ISO 3166. Entry and exit dates are written to *EntryDateTime* and *ExitDateTime*. For the names of the entry and exit customs offices please use the elements *EntryOrganizationName* and *ExitOrganizationName*. The two *IndicatorStamp* elements indicate an existing stamp on the paper document.

2.7 Confirmation receipt (17/18)

Schema file: EUDINCertificateOfWasteReceiptDocument-2.1.xsd



Fill the *DocumentHeader* with standard values as usual.

The element *DisposalFacilityIndicator* indicates if the facility is a disposal facility (true) or a recovery facility (false).

The entry date is placed in the element *receptionDateTime* abgelegt.

The element *WasteRejectionCode* indicates if wastes have been accepted or rejected. If it was rejected enter **FR** (full rejection), otherwise **NR** (no rejection). There is a third option: **PR** (partial rejection) that is not used as of yet.

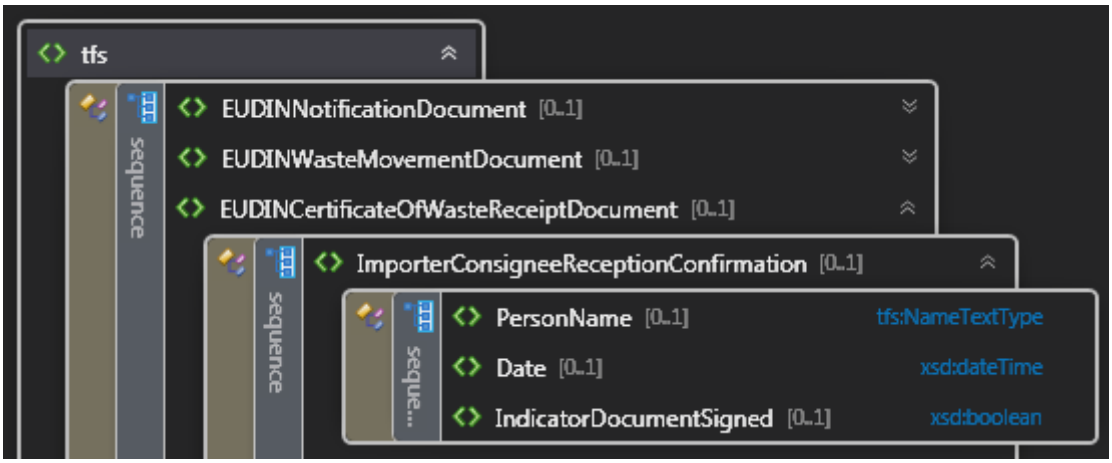
The weight and the optional volume are entered in the elements *ReceivedWeightMeasurementSpecification* and *ReceivedVolumeMeasurementSpecification*. See Weigth/Volume.

RecoveryDisposalOperationClassification stores the R/D code of the recovery/disposal operation.

CorrespondingWasteMovement contains the data of the transport announcement.

2.7.1 Shipment received by importer (17)

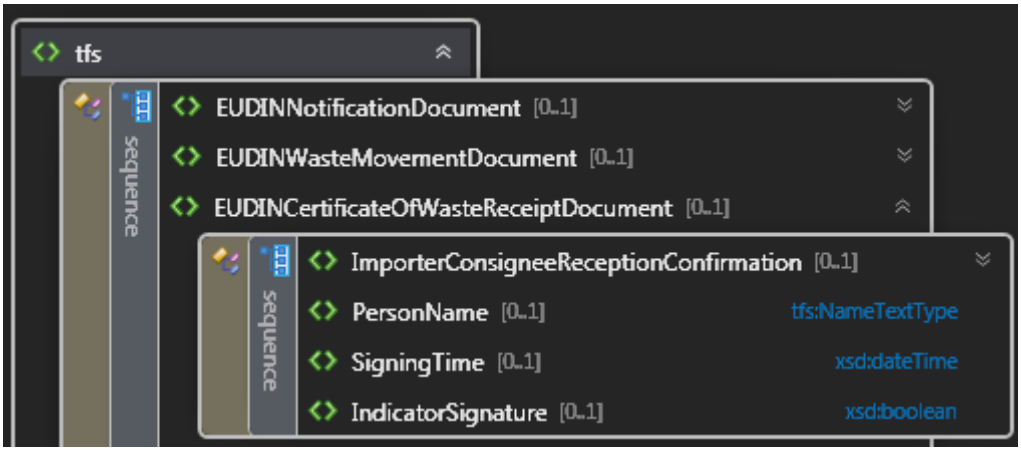
Element: *tfs/EUDINCertificateOfWasteReceiptDocument/ImporterConsigneeREceptionConfirmation*



Place the name of the signee (*PersonName*), the date of the signature (*Date*) and the indicator if the document was signed or not (*IndicatorDocumentSigned*) here.

2.7.2 Shipment received at disposal facility (18)

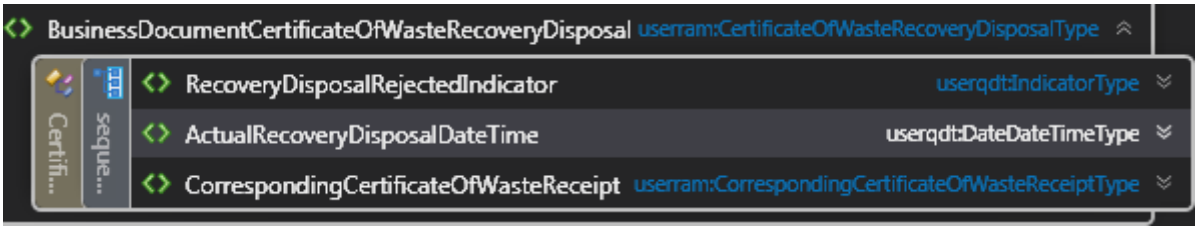
Element: *tfs/EUDINCertificateOfWasteReceiptDocument*



The information about the disposal facility is placed here like the fields under *ImporterConsigneeReceptionConfirmation*.

2.8 Confirmation disposal (19)

Schema file: EUDINCertificateOfWasteRecoveryDisposalDocument-2.1.xsd



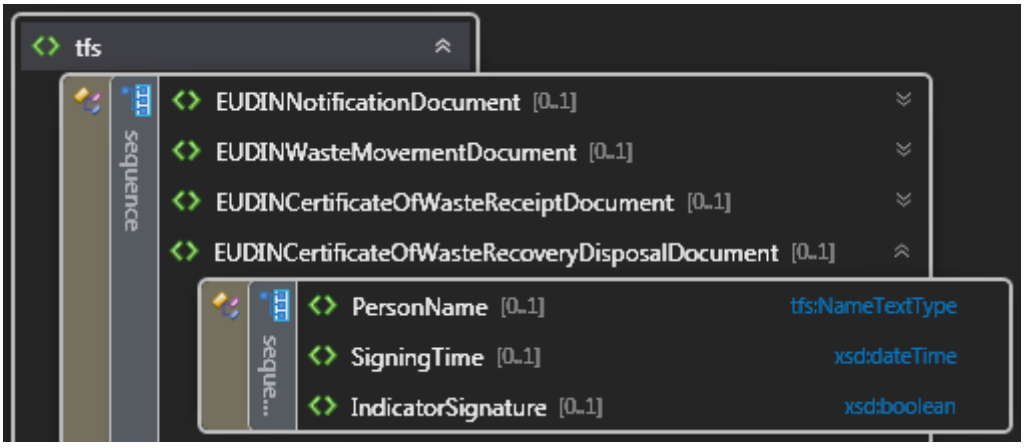
The obligatory element *DocumentHeader* has to be filled like described in ***DocumentHeader***.

The mandatory element *ActualRecoveryDisposalDateTime* has no equivalent on the official forms and is therefore to be filled with the value of *ReceptionDateTime* from the transport announcement.

CorrespondingCertificateOfWasteReceipt contains the confirmation receipt.

Information on the signatures is contained in the TFS schema.

Element: *tfs/EUDINCertificateOfWasteRecoveryDisposalDocument*



2.9 Consignment Information

The annex7 document is identical to a WasteMovementDocument with minor adjustments. The following sections describe these adjustments.

The root element was changed from *Abfallverbringungsdokument* to *Annex7*.

The element DocumentHeader/*BusinessDocument/TypeCode* is always MD.

CorrespondingNotificationIdentificationID and *ConsecutiveTransportNumberNumeric* are mandatory and can contain arbitrary IDs.

Some mandatory fields of the WasteMovementDocument are not applicable for Annex VII. The following table summarizes the replacements.

Mandatory field	Value
SpecialHandlingInstructions	False
Waste/PhysicalCharacteristicCode	OTHER

2.9.1 Section 10 Mapping WasteCodes

Some waste identification categories have no counterpart in the WasteClassificationType. The following table maps them to existing WasteClassificationTypes.

Annex VII Type (Document)	EUDIN WasteClassificationType (XML)
BASEL	BASEL
OECD	OECD
Annex IIIA	NATIMP
Annex IIIB	NATEXP
EU List of wastes	EWC
National code	OTHER

2.9.2 Section 11 (Countries concerned)

Values for the role and country will be written to *CustomsClearance*/*EnteredCountryRoleIdentificationID* and *EnteredCountryID.* The mandatory field *ClearanceDateTime* is unused and will be set constantly to **1970-01-01T00:00:00**.

2.10 **Confirm receipt (CR)**

The *CertifcateOfWasteReceipt* Layer is used in two different roles. The consignee signs the receipt of waste in the first CR layer. Afterwards, the facillity or laboratory signs a second CR layer.

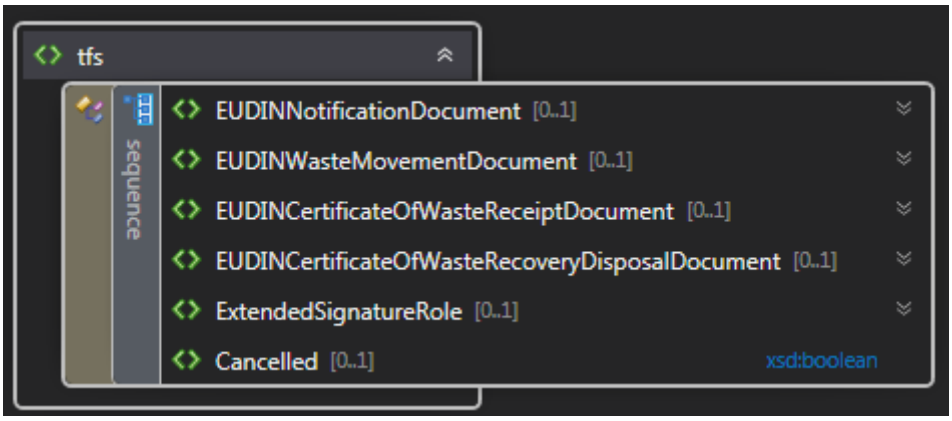
Mandatory field	Value
ReceptionDateTime	Current date and time
WasteRejectionCode	NR
DisposalFacilityIndicator	True=facility, false=laboratory

3 TFS schema

The EUDIN-specification doesn't define all fields of the official forms. The tfs.xsd contains a schema with further element definitions to allow a complete representation.

The *tfs* element is embedded into a *FreieXMLStruktur* element of the *waste movement document*.

The schema possesses the *NamespaceURI* **urn:de:bmu:eanv:ntz:tfs:1:00** for identification purposes.



There are optional elements defined for every document type/state of the movement document to hold contents of the official documents that is not covered in EUDIN.

3.1 Cancellation

Use the element *cancelled* to mark a document as cancelled.

3.2 Extended roles

The element for the extended roles is set prior to each signature to make sure it is enclosed by the signature. Although the role of the signee is obvious due to the document type receipt/disposal confirmation, you still state the extended role in the XML for consistency reasons. Some roles need an additional attribute to allocate them distinctly.

3.2.1 Notification document

Participant	**Role**	**id**	**signature Id prefix**
Waste generator	ERZ		ERZ
Exporter	EXP		SNT
Authority (confirmation)	BEH		BEH
Authority (consent)	CON	ISO-country code	BEH

3.2.2 Transport announcement /Confirmation receipt/Confirmation disposal

participant	**role**	**Id**	**signature Id prefix**
Waste generator	ERZ		ERZ

Exporter	EXP		SNT
Importer	IMP		SNT
Carrier	BEF	Consecutive number as in *TransportStage/ ConsecutiveNumberNumeric*	BEF
Disposal facility (receipt)	ENT	REC	ENT
Disposal facility (disposal)	ENT	DIS	ENT
Customs office export (20)	BEH	EXP	BEH
Customs office import (21)	BEH	IMP	BEH
Customs office transit (22)	BEH	Consecutive number as in *CustomsClearance/Transit/ ConsecutiveNumberNumeric*	

3.2.3 Annex VII / CertificateOfWasteReceipt

participant	**role**	**id**	**signature Id prefix**
Exporter	EXP		SNT
Importer/Consignee	IMP		SNT
Carrier	BEF	Consecutive number as in *TransportStage/ ConsecutiveNumberNumeric*	BEF
Facility/Laboratory	ENT		ENT

4 Standards

4.1 UTF-8

RFC 3629 (2003-11) “UTF-8, a transformation format of ISO 10646”
https://tools.ietf.org/html/rfc3629

4.2 XAdES

ETSI EN 319 132-1 V1.1.1 (2016-04) “Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and XAdES baseline signatures”
http://www.etsi.org/deliver/etsi_en/319100_319199/31913201/01.01.01_60/en_31913201v010101p.pdf

4.3 XML 1.0

W3C Recommendation (2008-11) “Extensible Markup Language (XML) 1.0 (Fifth Edition)”
http://www.w3.org/TR/2008/REC-xml-20081126/

4.4 XML DSig

W3C Recommendation (2008-01) “XML Signature Syntax and Processing (Second Edition)”
http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

4.5 BMU 1.04

http://www.bmub.bund.de/themen/wasser-abfall-boden/abfallwirtschaft/wasser-abfallwirtschaft-download/artikel/datenschnittstelle-zur-nachweisverordnung/

4.6 EUDIN 2.1

http://www.eudin.org

bmu 1.04 contains the xsd schema for eudin 2.1

4.7 eTFS (7.7.2014)

http://www.zks-abfall.de/de/bmu-schnittstelle

ERP Integration Tools

ZEDAL Forms Interface

Last revision: 06.01.2016

20 pages including the front page

Document No. 155.388.961

Technical documentation

ZEDAL Forms Integration - Index

1 ZEDAL Forms

ZEDAL forms presents XML documents in a form view designed in compliance with appendix 3 of the German Nachweisverordnung as of 20. October 2006. Furthermore is the program able to show and edit international notification and waste movement documents as well as to print them as official paper forms.

The user may edit, print or sign one or multiple documents at the same time (including layers), as well as check singed documents.

Creation and validation of signatures is realized via Infotech Signer.

ZEDAL Forms is used in two different environments:

1.1 Webserver / browser based

Webserver and browser provide the necessary infrastructure for communication. The webserver transfers all necessary information to the browser. The browser forwards this information to the browser plugin. The plugin then contacts ZEDAL Forms, which then realizes the document display. This kind of connection is handled in this document.

1.2 Standalone

In this scenario any program can communicate with ZEDAL Forms directly.

An example project demonstrating the program call for national and international documents is located in the directory examples/zedalforms for that case specifically. Please extract all further information from the readme.md as well as the documented sources.

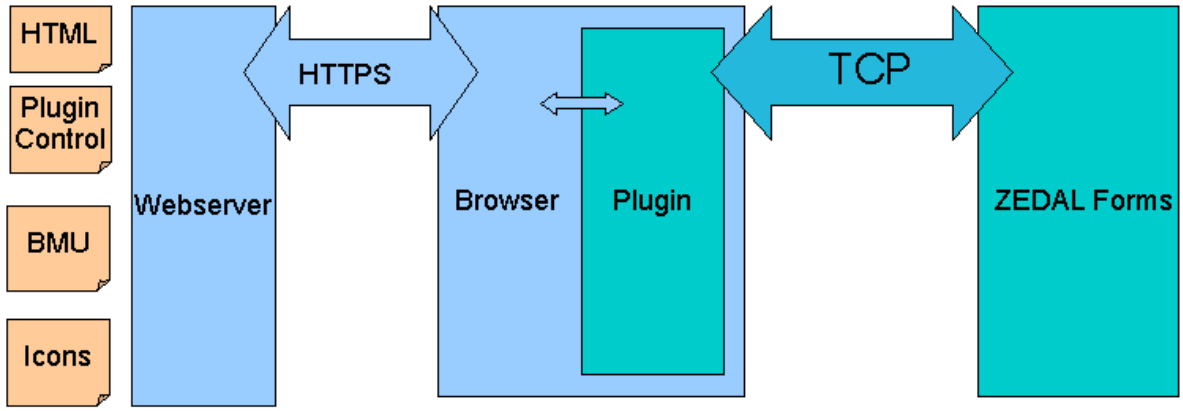
2 The interface

2.1 Embedding via object

To open ZEDAL Forms the webserver sends an HTML page with an embedded object-element to the browser, which in correspondence opens up the browser plugin. The plugin then starts /contacts ZEDAL Forms via TCP and arranges a port for communication (shared memory) on the local host. The call parameters include the URL where ZEDAL Forms can find the PluginControl file. ZEDAL Forms data transfer is completely regulated via the browser and takes up no additional resources to be operational.

When ZEDAL Forms has finished the download of all data it will show the form. The resulting XML file is send to the server when all editing is done and the user activates the send button in ZEDAL Forms. The target URL is extracted from the plugin control file.

2.1.1 Schematics



2.1.2 Object for plugin activation

```
<OBJECT WIDTH="100%" HEIGHT="98%" TYPE="application/x-infotech-libvol">
        <PARAM NAME="GETURL" VALUE="https://www.domain.de/showDataTP.xml>
        <PARAM NAME="PROVIDERID" VALUE="ZEDAL">
        <PARAM NAME="KDNR" VALUE="123456">
</OBJECT>
```

Embedding the object element in an HTML site shown by the browser activates the plugin. If ZEDAL Forms is not running at the time of activation it will be started first and then activated using the window provided by the browser. The MIME type *application/x-infotech-libvol* is linked with ZEDAL Forms during the installation process which results in the browser activating or starting ZEDAL Forms.

2.1.3 Parameters and their meaning

The parameters send with the embedded object element are processed during ZEDAL Forms startup.

- **GETURL** (data source for initial plugin control file)
- **PROVIDERID** (optional) is the abbreviation of the providerinstallation
- **KDNR** (optional) user number, normally the login name.

HINT: *PROVIDERID* and *KDNR* will be shown in title bar of ZEDAL Forms.

2.2 New method via protocol-handler

Because of constantly declining support of NPAPI usage (Google Chrome already cancelled the support completely) an alternative method of use has been implemented. ZEDAL Forms registers a protocol handler during its installation. This protocol handler is linked with *ieframe.exe*. To start ZEDAL Forms from a website a URL of the following format has to be used:

```
zedalforms://server/script?parameter&parameter2
```

The *ieframe.exe* sends a POST request to this URL, receiving a string that ZEDAL Forms then sends to the server, requesting the plugin control file. The URL is constructed as follows:

```
zedalforms://server/script?"GETURL"={documentUrl}&"FORMULARE"=""&
"CLOSEMODE"="EACH"&"PRELOADLIB"="libeay32.dll&"APPLICATION"="InterForm.dll"&
"KDNR"="customerid"&"PROVIDERID"="ZEDAL"&REFRESH"="true"
```

Parameters *GETURL*, *PROVIDERID* and *KDNR* match with the description under 2.1.3. All other parameters have to be included exactly as shown above.

2.3 Startup phase

ZEDAL Forms tries to get a number of information executing a GET-request. One part of this information is the plugin control file. The following resources are also requested:

2.3.1 sakupdate.php

This request is asking for a newer version of the infotech signer and is to be answered with an empty string. This request is transferred once after initial startup.

```
{baseurl}/sakupdate.php?
hash=90f47db794d555b75e5a54be0c96768a068769221f7fa16de136de6f5aad42c8
&version=3main.lv
```

This request contains checkup variables for national and international documents. ZEDAL Forms adds a timestamp to evade the browsers cache. This is contained in the exemplary project.

```
e.g. http://basisurl/main.lv?20160104125200
```

2.3.2 imagepak.lv

This file contains graphic images for the buttons inside ZEDAL Forms. Hand over the file for that purpose.

The *GetBMU* document, located in the *INIT* area is requested after transmission of the plugin control file to the plugin. After successful transmission further elements will be requested (defined in the button elements with *src*). The document will then be displayed.

2.4 Troubleshooting

Little xml files can be send to the plugin to show error messages inside the plugin which will then be shown inside the information bubble of the ZEDAL Forms tray icon.

The xml is structured like this:

```
<?xml version="1.0" encoding="ISO-8859-1"/>
<fehler>
  <text>Error text</text>
</fehler>
```

3 Plugin control file national

The plugin control file defines which functions are available to the user.

3.1 Example of a plugin control file

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<IKS_PlugInControl type="BMU">
 <Init>
  <GetBMU URL="" ATBRolle="ERZ"/>
</Init>
 <Buttons>
  <Button Caption="Bearbeiten" Hint="open document" src="edit.bmp">
   <BeginEdit />
  </Button>
  <Button Caption="Verwerfen" Hint="undo" src="abort.bmp">
    <CancelEdit/>
  </Button>
  <Button Caption="Drucken" Hint="print document" src="print.bmp">
   <PrintXML/>
  </Button>
  <Button Caption="Verwerfen" Hint="discard document" src="cancel.bmp">
   <CancelEdit/>
  </Button>
  <Button Caption="Speichern" Hint="close document" src="speichern.bmp">
   <StoreXML/>
   <EndEdit/>
  </Button>
  <Button Caption="Signieren" Hint="sign document" src="sign.bmp">
   <SignXML/>
  </Button>
  <Button Caption="Senden" Hint="store document" src="send.bmp">
    <PostXML URL="targetl"/>
  </Button>
  <Button Caption="Signaturen prüfen" Hint="verify signatures" src="check.bmp">
   <VerifySignatures/>
  </Button>
  <Button Caption="Update" Hint="update masterdata" src="update.bmp"
          LastChanged="20090724112154" toolbar="extras">
    <Update URL=\"updateurl”/>
  </Button>
  <AttachmentControl Hide="no" ReadOnly="no"/>
 </Buttons>
</IKS_PlugInControl>
```

The plugin control file is an XML file. It contains the following areas:

3.2 Root-element

The root-element is named IKS_PlugInControl and contains two attributes:

- **type**: always has to be “*BMU*” for national (German) documents.
- **docid**: unique ID for a single document. The document will not be loaded again if ZEDAL Forms already has a document loaded with the same given ID. Instead it will just be displayed if editing mode is active.

Optionally the value *maxsize* can be send. This limits the size of attachments to the given value in bytes.

3.3 Initializing

The node *Init* is processed during startup of the document and contains the command *GetBMU* to request and show data from the server.

3.4 Actions

The elements button contained in the element buttons define the buttons available for the usage of ZEDAL Forms as well as the executable batch commands that shall be started upon activation. The order of definition determines the order of buttons in the display.

3.4.1 caption

The caption is equal to an ident of a button which is important for internal processing. Therefore the given names have to be used.

3.4.2 hint

This is the name of the shown tooltip for the targeted button.

3.4.3 src

Src determines how the browser accesses the graphic for the button. The imagepak.lv is searched for graphics (name is important here), if nothing is found, a graphic can be provided:

The graphic has to be the size of 48x48 pixels. Color depth can be chosen arbitrarily. The left quadrant *(0,0,24,24)* contains the graphic for the active button, the other half contains the graphic for the inactive button. Transparent pixel will be coded via the RGB value violet *(255,0,255)*.

3.4.4 toolbar

The attribute toolbar allows the compilation of a number of buttons inside an area of the toolbar. Predetermined areas are:

- **extras** (attachments and master data)
- **tools** (document specific functions)

3.5 Attachment control

Attachment control determines if the paper clip symbol is shown in the toolbar.

- **Hide**=*no* symbol is visible
- **Hide**=*yes* symbol is invisible

ReadOnly furthermore controls if the user is able to delete or attach new attachments or if he is only able to read them.

3.6 Batch command sequences

As aforementioned in actions, XML elements are given inside the elements button which are responsible for executing certain processes.

3.6.1 GetBMU

Fetches the document mentioned inside the URL received by the server and displays that document.

3.6.2 PostXML

This command causes the form to be converted to xml view and transmits it via POST to the in the attribute URL mentioned URL. ZEDAL Forms then waits for an answer, informing it if the transmission was successful. In the event of an successful transmission a file has to be send by the server, containing the string:

```
Mail sent!
```

The transmission will be declared as failed should ZEDAL Forms receive another text. That text is then shown in the tray icons information bubble.

3.6.3 PrintXML

This command prints out the loaded form.

4 Plugin control file international

Prerequisite for display of international documents is the installation of the separate ZEDAL Forms module TFS.

International documents use another format for the plugin control file. The core contains a program written in the programming language Squirrel 2.2.4. (http://www.squirrellang.org) which is embedded in an XML frame.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<PlugInControlContainer>
<IKS_PlugInControl type="eudin" docid="34510-5" maxsize="10485760">
<ScriptInit hash="<hashvalue>">
<![CDATA[function Init(model)
{
  model.GetStream("http://server.de/loaddocument&docid=1”);
}

function Buttons(model)
{
  model.AddButton(Const.ACTION_EDIT, "BeginEdit", 3, "Open document", "base");
  model.AddButton(Const.ACTION_UNDO, "Undo", 1, "Undo", "base");
  model.AddButton(Const.ACTION_ENDEDIT, "Store", 7, "Close document", "base");
  model.AddButton(Const.ACTION_DISCARD, "Discard", 9,
                  "discard document", "base");
  model.AddButton(Const.ACTION_SEND, "Send", 5, "save document", "base");
  model.AddButton(Const.ACTION_ATTACH, "Attach", Const.ICON_NOATTACH,
                  "Attachments", "base");
  model.AddButton(Const.ACTION_PRINT, "Print", Const.ICON_PRINT,
                  "print document ", "base");

  model.AddButton(Const.ACTION_SIGN, "Sign", 6, "sign document", "sign");
  model.AddButton(Const.ACTION_VERIFY, "Verify", 2,
                  "verify signatures", "sign");

  // multi functions (only available if document tree is visible)
  model.AddButton(Const.ACTION_MULTI_SIGN, "MultiSign",
                  Const.ICON_MULTISIGN, "Signieren", "multi");
  model.AddButton(Const.ACTION_MULTI_PRINT, "MultiPrint",
                  Const.ICON_MULTIPRINT, "Drucken", "multi");
  model.AddButton(Const.ACTION_MULTI_SEND, "MultiSend",
                  Const.ICON_MULTISEND, "Speichern", "multi");
}

function Store(model)
{
  // check if all inputs make up a valid document
  if ( model.Validate() )
  {
    // check if notification number is empty
    if ( model.GetValue("ConsecutiveTransportNumberNumeric") == "" )
    {
      // requests a number asynchronously from a url
      // if no server-side numbering scheme exists, this part
      // may be removed
      local url = "http://server.com/number.php";
      // the Const.GET_WASTEMOVEMENT is important, because
      // it inserts the received into the document!
      model.GetURL(url, Const.GET_WASTEMOVEMENT_NUMBER, true);
      // stop script, will be resumed when requests completes.
      ::suspend();
    }
    model.Store();
  }
```

```
}

function Send(model)
{
  // Post XML Document
  model.PostXml("http://server.com/document&docid=1”);
}

function BeginEdit(model) { model.BeginEdit(); }
function Sign(model) { model.Sign(); }
function Print(model) { model.Print(); }
function Attach(model) { model.Attach(!true); }
function Verify(model) { model.Verify(); }
function Discard(model) { model.Discard(); }
function Undo(model) { model.Undo(); }
function StammdatenUpdate(model)
{
  model.Update("http://server.com/masterdata");
}]]></ScriptInit><SigInfo></SigInfo>
</IKS_PlugInControl></PlugInControlContainer>
```

4.1 Security

The element ScriptInit contains the attribute hash which in turn contains the hash value of the script.

The hash value is formed as follows:

- a secret value is written to a buffer
- after that all characters from <![CDATA[to]]> are attached
- the complete buffer is hashed via SHA256

The result is written to the attribute hash as hex-string. ZEDAL Forms checks this hash during startup. The load process will be terminated if a hash result is not correct.

An incorrect hash can be determined by the entry: *script changed!* inside the tfs.log

4.2 AddButton(ActionID, functionname, IconID, toolbar)

Adds a button that executes a specific action.

4.2.1 ActionID

ActionIDs are constants and listed completely in the example

4.2.2 functionname

Gives a name to a script function that is to be executed after pressing the button.

4.2.3 IconID

Defines the icon on the button. Icons are fixed and can be accessed via constants:

- *ICON_ABORT* Undo like symbol, arrow pointing counterclockwise
- *ICON_VERIFY* Magnifing Glass over scribbled text
- *ICON_EDIT* Pen writing on paper
- *ICON_PRINT* A blue printer symbol
- *ICON_SEND* Floppy disc

- *ICON_SIGN* Paper with signature
- *ICON_SAVE* Pen laid down
- *ICON_UPDATE* World globe
- *ICON_CANCEL* Paper with red cross
- *ICON_CHANGE_LANG* German/English flag
- *ICON_ATTACH* Document with paper clip
- *ICON_NOATTACH* Document without paper clip
- *ICON_MULTISIGN* Multiple signed papers
- *ICON_MULTISEND* Multiple floppy discs
- *ICON_MULTIPRINT* Orange printer symbols

4.2.4 Toolbar

Allows grouping of elements, analogous to the national program part.

5 Technicalities of the program call

5.1 Netscape/Mozilla: instantiating of ZEDAL Forms

- *Mime-Type*: application/x-infotech-libvol
- *File extension*: .lv
- *Plugin DLL for Netscape/Mozilla*: npframe.dll

The Mime-Type has to be registered in the Fileinfoblock of the Plugin (tab “version”) in the windows-explorer.

The browser opens up the plugin if a file with the mime-type *application/x-infotech-libvol* should be displayed.

5.2 Internet Explorer: instantiating via ActiveX

- Plugin-DLL for Internet Explorer: *IEFrame.ocx*.

The DLL will be deposited with its properties as an OCX module in the registry path *[HCR]\CLSID* under GUID *{9145A4B0-E4C0-11D5-9C6A-0000E861B1D5}*.

5.3 New method (browsers without NPAPI support)

A new protocol handler that uses the ieframe.exe for communication purposes will be defined.

- *Protocol*: zedalforms
- *Protocol handler*: ieframe.exe

6 Multidocument

6.1 Startup phase

The following tag has to enclose all single IKS_PluginControl tags to transmit multiple documents to ZEDAL Forms:

```
<?xml version="1.0"?>
<PlugInControlContainer>
  <IKS_PlugInControl>
    ...
  </IKS_PlugInControl>
  <IKS_PlugInControl>
    ...
  </IKS_PlugInControl>
</PlugInControlContainer>
```

The xml documents in compliance with the BMU-standard will be requested one by one. The correct use of the URL in GetBMU and PostXML is important here. They should be chosen in a way that the application is able to assign the edited document to the original. One of the problems here is that ZEDAL Forms extends the URL in the PostXML function. The string *&nkdm=true&nver=&stor=true* is added.

6.2 DocID

The DocID is used to mark a document as unique in ZEDAL Forms. The element *<PluginControl>* has been extended by another parameter *docid* to prevent loading up the same document multiple times. ZEDAL Forms will not load the same document twice if it receives a document with an already known number, instead it will just switch to the already opened document. The function of multidocument management is not available if this element is missing.

6.3 Functions

Functions are placed on buttons just like any other command. Besides *toolbar=extra* and *toolbar=tools, toolbar=multidok* is now also available. This places a button in the bar above the document tree (only visible if more than one document is loaded). These commandos do not need any other actions and even ignore those.

6.3.1 Multisign

This command transmits all (or all selected) documents to the IT Signer and can be signed there. After successful signature the documents are transferred back to ZEDAL Forms.

6.3.2 Multisend

All (or all selected) documents will be send to the server

6.3.3 MultiPrint

All (or all selected) documents are printed out.

6.3.4 Plugin control fragment

```
<Button Caption="Multisign" Hint="Dokumente signieren"
        Src="multisign.bmp" toolbar="multidok">
</Button>
<Button Caption="Multisend" Hint="Dokumente senden"
        Src="multisend.bmp" toolbar="multidok">
</Button>
<Button Caption="MultiPrintXML" Hint="Dokumente senden"
        Src="multisend.bmp" toolbar="multidok">
</Button>
```

7 Prerequisites

7.1 Resources on the server

The server has to provide the following data:

- plugin control file (national/international)
- BMU-file or international document
- any further graphics for buttons as .bmp
- imagepak.lv
- main.lv

7.2 Client

- Installed ZEDAL Forms
- Mozilla Firefox, Internet Explorer or a starter (see code example)

7.2.1 Access control

- Plugin needs the right to write to the directory "*%TEMP%\interform2*" to write log information.
- Read/write rights are required to manage master data in the directory "*%APPDATA%\interfom*".
- The registry key *HKEY_CURRENT_USER\Software\InfoTech\Interform* is used to store the options of the plugin.
- ZEDAL web sites have to be added to the trusted sites or access control of the ActiveX component needs to be edited in case of executing the plugin via Internet Explorer.
- The browser plugin and ZEDAL Forms communicate via a TCP connection on IP *127.0.0.1*. The used port is dynamically negotiated via shared memory between them. Therefore, ZEDAL Forms is also viable for terminal server usage

8 Master Data

New Master data is requested in ZEDAL Forms as soon as the information is transmitted that new master data is available. The master data has to be formatted in the correct manner for ZEDAL Forms to be able to process it.

8.1 Recognizing new master data

8.1.1 National

The definition of the button “refresh master data” can contain the attribute “*lastchanged*”. If the herein contained date in the format *YYYYMMDDhhmiss* is younger than the date stored under *HKCU\Software\InfoTech\Interform\update\date* within the registry the URL inside the *<Update>* element is contacted via GET.

8.1.2 International

This is basically identical to the handling of national data, but the way the “last chaged date” is transmitted changes. The function *Init()* has to be edited to contain the following line for international documents:

```
model.SetParameter(„lastchanged “, „YYYYMMDDhhmiss“);
```

8.2 Format (both)

Data is provided as TAB separated file. Every cell is equal to a data block and has to contain exactly 45 entries. Every cell ends with linefeed (unix line endings).

The data contains two full address blocks. The first one regarding the company and the second one regarding the corporate body.

This data is used in the national context as well as in the international one. The order of columns has been determined as follows:

Nr	Description national	Description international (if different)
1	Role (ERZ, BEF, ENT, ERZBEF, BEH)	
2	Registry number (=nr. issued by the authority)	
3	ZKS address(=Role + registry number)	
4	ZEDAL address	
5	Name 1	
6	Name 2	
7	Name 3	
8	Name 4	
9	Street 1	
10	Street 2	
11	House number	
12	ZIP code	

13	City 1	
14	City 2	
15	Country	
16	Post box number (not used)	
17	ZIP code (post box) (not used)	
18	City 1 (post box) (not used)	
19	City 2 (post box) (not used)	
20	Country (post box) (not used)	
21	Contact person	
22	Phone	
23	Fax	
24	E-mail	
25	Name 1 (from here on for **corporation**)	
26	Name 2	
27	Name 3	
28	Name 4	
29	Street 1	
30	Street 2	
31	House number	
32	ZIP code	
33	City 1	
34	City 2	
35	Country	
36	Post box number (not used)	
37	ZIP code (post box) (not used)	
38	City 1 (post box) (not used)	
39	City 2 (post box) (not used)	
40	Country (post box) (not used)	
41	Contact person	
42	Phone	
43	Fax	
44	E-mail	
45	*Not used / undefined*	International registry number

8.2.1 Characteristics

The URL defined in the update element will be extended with the string *&version=<ZEDAL Forms-Version>* by ZEDAL Forms so that the program can react to changes in the master data format.

The first field ("*Role*") contains the role of the waste disposal participant. This restricts usage of the master data in the national area. E.g.: selecting a waste creator will only show the master data entries marked with "*ERZ*".

9 packaging the setup

To support the company-wide rollout of ZEDAL Forms here is a list of registry keys that can be set so that ZEDAL Forms doesn't have to be configured during the first start up. The following keys are defined under *HKCU**Software**Infotech**Interform*:

9.1 Profile

Please enter the following string:

```
Dir=<name of any writeable directory>
```

9.2 Smartcard

Please enter the following string:

```
Default=1
Reader=<name of the cardreader>
```