



ELO XML Importer

Installation & operation



Table of contents

Installation & operation	3
Installation	3
Structure of config.xml	5
XML document import	7
Automatic metadata update	18
Deleting data	23

Installation & operation

Installation

This program is contained in the ELO installation program. An ELO Indexserver must already be installed for the repository, but not necessarily on the same computer/server. Only a rudimentarily configured ELO XML Importer is installed, the configuration of which is located in the *config.xml* file (in the installation directory, e.g. *ELOenterprise\config\im-<Name of repository>*) and not in the database. In contrast to other modules, the ELO XML Importer is not configured in the ELO Administration Console. If multiple import directories or multiple XML importers are required at the same time, you will have to edit the *config.xml* file. If not, you can skip the following chapter.

ELO XML Importer version 20.00.001.001 and higher use *Logback* for logging (older versions use *Log4j*). From this version on, a *logback.xml* file is required for configuring logging (provided by the setup, only has to be created in case of manual installation).

Please note

With ELO XML Importer version 20.00.001.001 and higher, if you install the module manually, you have to provide for a *logback.xml* file.

Example *logback.xml* file:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>c:/temp/logs/im-elo20.log</file>
    <append>true</append>
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} %-60(%thread %X{NDC} %-5level \(%logger{0}.java:%L\))
    </encoder>
  </appender>

  <root level="info">
    <appender-ref ref="FILE"/>
  </root>
</configuration>
```

You can either change the log level as the root level in the *logback.xml* file or on the ELO XML Importer status screen (in Edit mode):

Logging		
Level	INFO	INFO <input type="button" value="v"/>
		DEBUG
System		INFO
Memory (free, total, max)	133.7 MB, 161.5 MB, 2,147.5 MB	WARN
Current time	2020-04-14 16:56:43 +0200	ERROR

Structure of config.xml

Excerpt from the *config.xml* file (example for an ELO XML Importer):

```
<properties>
<comment>parameters for this web application</comment>
<entry key="import1_encryption_FIBUKey">55-219-106-48-149-511-142-114</entry>
<entry key="import1_encryption_GFKey">89-106-23-789-111-654</entry>
<entry key="import1_ixurl">http://localhost:8080/ix-elo/ix</entry>
<entry key="import1_fileextension">ESW</entry>
<entry key="import1_directory1">C:\ELOenterprise\data\im-elo\import1</entry>
<entry key="import1_directory2">C:\ELOenterprise\data\im-elo\import2</entry>
<entry key="import1_movedirectory">C:\ELOenterprise\data\im-elo\processed</entry>
<entry key="import1_username">Administrator</entry>
<entry key="import1_passwd">234-167-21-87-88-80-78-122</entry>
<entry key="import1_sigfile">>false</entry>
<entry key="import1_SigfileExtension">sig</entry>
<entry key="import1_ConfirmDir">C:\import_confirm_files</entry>
<entry key="import1_ConfirmExt">.QIT</entry>
<entry key="import1_ConfirmOk">Document guid:%GUID% imported</entry>
<entry key="import1_ConfirmDelOk">Document guid:%GUID% removed</entry>
<entry key="import1_ConfirmErr">Document Import Error</entry>
<entry key="import1_ConfirmStructure">>true</entry>
<entry key="import1_ConfirmEmpty">>false</entry>
<entry key="import1_sleepTimeUser">120000</entry>
<entry key="import1_keywording_form_content">>true</entry>
</properties>
```

The parameters in the configuration file consist of a combined parameter name (key) and a parameter value. The parameter name consists of the importer name and the corresponding parameter. Both are separated by the first underscore (_). Combined parameters are also separated by underscores, e.g.

"import1_encryption_FIBUKey":

In this case, "import1" is the name of the importer and "encryption_FIBUKey" is the corresponding parameter. If you require several ELO XML Importers at the same time, you can choose another name like import2, import3, and so on.

Entry to config.xml	Explanation
---------------------	-------------

encryption_name	Only when using an ELO Indexserver with a main release older than 12 will you be asked to enter the password for the given encryption key. This entry will no longer be read as of version 12. In the control file, only the encryption key (<i>name</i>) is entered (see chapter Structure of the XML import control file).
-----------------	--

Entry to config.xml	Explanation
ixurl	URL for ELO Indexserver access. If the server is called <i>xmlserver</i> and the repository is named <i>elo</i> : <i>http://xmlserver:8080/ix-elo/ix</i> .
directory1	Exchange directory for document and control files.
directory2	Second exchange directory for document and control files (optional).
directoryX	Any number of additional exchange directories.
fileextension	ESW or XML (default: XML import).
movedirectory	Directory that the import files are moved to after import. If you want the files to be deleted after the import, leave the <entry> tag blank: <entry key="import1_movedirectory">\</entry>.
username	ELO user name (e.g. Administrator) for authentication. Important: This ELO user must have the rights <i>Edit documents</i> and <i>Edit folders</i> .
passwd	Encrypted password for the ELO user name. You can use the ELOenterprise Password Utility to generate a new encrypted password. An unencrypted password will be replaced with the encrypted password in the configuration file by ELO XML Importer.
sigfile	true if a signal file (default: .sig) is required to start the importer. The name of the signal file must be the same as the name of the import file.
SigfileExtension	File extension of the signal file. Default: sig
sleepTimeUser	Idle pause in ms between the processing of the exchange directories (60000 corresponds to one minute). Do not set value 100, 3000, 10000, 60000, or 12000.
keywording_form_content	true if you want to apply the workflow, encryption key, marker (color), and filing definition from the metadata form. This only works if no entries have been made in the XML control file.
Confirm...	For a description, see the chapter Creating receipt files.
locale	Optional setting for a <i>locale</i> which deviates from the <i>default locales</i> (e.g. <i>de</i> for the import of German-language documents using the decimal separator ",", " into an English-language environment).

XML document import

Structure of the XML import control file

When importing new documents, multiple documents can be entered to a single XML import control file. Provided there are no reasons to the contrary, only one document should be provided per file. This method makes it easier to recover data following an error.

The root of the XML tree consists of the `eloobjlist`. It can contain any number of `<obj>` objects. Each object generates an ELO document.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<eloobjlist ver="1.0">
  <obj mode="..." dup="...">
    <desc value="..." />
    <idate value="..." />
    <xdate value="..." />
    <deldate value="..." />
    <type value="..." />
    <marker value="..." />
    <sreg value="..." />
    <path value="..." />
    <dtype value="..." />
    <owner value="..." />
    <filedby value="..." />
    <fulltext value="..." />
    <arcmode value="..." />
    <encryption>...</encryption>
    <acl><access type="..." name="..." subgroup="..." /></acl>
    <memo>...</memo>
    <indexlist><index name="..." value="..." /></indexlist>
    <destlist>
      <destination type="..." value="[...] + L1(1,4) + LA + LD + LK" />
      (LA, LD, LK and L1(1,4) see the note under the following table
      with explanations for the individual tags)
    </destlist>
    <repl>...</repl>
    <map name="..."><data key="k1" value="v1" /><data.. /></map>
    <versioncomment>...</versioncomment>
    <workflowlist><workflow template="..." name="..." /></workflowlist>
    <structurelist><structure type="..." value="..." /></structurelist>
    <docfile name="..." />
    <sigfile name="..." />
    <attfile name="..." />
```

```
</obj>
</eloobjlist>
```

Explanations for the individual tags

Day	ELO field	Explanation (see example above)
obj		<p>The <obj> tag frames a document or folder. Multiple documents within a file are represented by a string of <obj> groups.</p> <p>mode attribute:</p> <p>If folders are only to be created if they do not yet exist, this tag contains an additional mode attribute with a value of check or checkMD5. With check, the target from the first destination entry is checked first. If the object already exists at this location, the editing of this entry is canceled – it is not entered a second time. checkMD5 checks whether the document file (if one has been configured) already exists in ELO by checking the MD5 hash value. In this case, it cancels processing.</p> <p>guid attribute:</p> <pre><obj guid="(B66B68360A-87F4-4AC6-874A-9FE92DF9E6)"></pre> <p>The object is saved with the GUID "(B66B68360A-87F4-4AC6-874A-9FE92DF9E6)" entered to the guid attribute.</p> <p>License attribute: (only in conjunction with guid) <obj guid="(B66B68B60A-87F4-1AC3-874A-9FE92DF9E6)" license="B620BBB2F2B39BCAB6A599D218D5D8"\></p> <p>Normally, the ELO XML importer needs to be unlocked with a serial number to run. However, there is one exception: It can be run without a special license if the source only provides specially labeled XML files. In this case, a guid attribute and a license attribute are entered to the <obj> tag.</p> <p>dup attribute:</p> <pre><obj dup="From"></pre> <p>in connection with an index field <index name="From" value="XXXX">. First, the system searches for a document in the repository with the metadata above. If the document exists, it is downloaded and imported from the XML file, instead of the document.</p>
desc	Short name	The value attribute cannot be empty.

Day	ELO field	Explanation (see example above)
idate, xdate, deldate	Filing date (internal date), document date (external date), expiration date	<p>If the value attribute is empty, the current date at the time of importing is entered. The document date, on the other hand, remains empty. The xdate field may also contain a time in addition to the date in the format YYYYMMDDHHMM or YYYYMMDDHHMMSS. The times will be adjusted for the time zone of the server.</p> <p>Xdate with attribute <mode="current">: The current date is entered to the document date. Complete the date fields with an ISO date (YYYYMMDD), entering a four-digit number for the year.</p> <p>The idate field may contain a time in addition to the date (YYYYMMDDHHMMSS). Note that the seconds must be entered, even though they will not be saved (you can simply enter a fixed number of 00).</p>
type	Metadata form	<p>The value attribute cannot be left empty, as it contains either the number of the metadata form or its name. Default values of the metadata form are applied.</p>
dtype	Document type	<p>You can define an object as various document types by using a value from 254 to 310. You can also create additional structure elements here. These can be assigned values from 1 to 253 and must not have a document entry (from ELO 10, this check is omitted).</p>
access	Key	<p>The name attribute contains the name. If a name cannot be assigned upon import, it is ignored. The type attribute can contain an r (read access), w (write access), d (delete), e (edit document), l (edit lists), p (set permissions, from version 21, 20.00.003.001 and 12.00.004.000, not in ELO 11 and ELO 10), or a combination of these five characters. <i>AND group</i>: AND groups can be defined with the subgroup attribute. Separate multiple groups with a semicolon ";".</p>
acl		<p>All keys (<access>) are located in the <acl> tag. The <acl> tag can additionally contain the attribute mode="new". This attribute shows that all existing document rights (keys) will be deleted from the metadata form and only the new ones from the XML control file will be entered. Assigned permissions apply only to documents, not to folders.</p>
memo	Extra text	<p>If the extra text contains explicit line breaks, these must be specified with \n or \r\n. Line breaks from the XML source are not applied to the ELO entry.</p>
marker	Color marking	<p>This indicates the color marking for the logical document entry. Only the marker number, not its name, can be entered here.</p>

Day	ELO field	Explanation (see example above)
sreg	Version number	Defines which current version number will be displayed in the metadata dialog box with a maximum of eight characters. In update mode, you can set existing whole number versions to be incremented. In this case, enter update in this field.
map	Additional information in the metadata dialog box (with predefined map domain)	<p>In the <map> tag, all name-value pairs are contained within the <data> tag, which in turn contains the key and value attributes. The <map> tag can be given the name attribute, which creates a user-defined map domain (for more details, see chapter 16 <i>Maps (8.0)</i> in the book <i>ELO Indexserver: programming guide</i>). Note: User-defined maps that are generated using the <i>name</i> attribute cannot be viewed or edited in the client.</p> <p>Example:</p> <pre><map> <data key="PAYCOND_DISCOUNTAMOUNT_0" value="345.10"/> <data key="PAYCOND_DISCOUNTPERCENT_0" value="2.00"/> <data key="PAYCOND_DATE_0" value="20170412"/> </map></pre>
owner	Owner	The value attribute contains the name of the owner, who can only be set by the administration.
Filedby	Filed by	Can only be set with administration rights. The user name must already exist in the repository.
Fulltext	Full text	Possible values are ON (document is added to the full text database) or OFF (useful if the full text flag is set in the metadata form, but the document should not be added to the full text database)
arcmode	Document status (filing mode)	Possible values are: FREELYEDITABLE, VERSIONCONTROLLED and REVISION-CONTROLLED

Day	ELO field	Explanation (see example above)
destination	Filing location	<p>The value attribute contains a filing target in the ELO lookup index format. The first filing location automatically becomes the main entry in ELO; all additional locations are references (see Note under the table). If a filing location is defined using the format ¶folder1¶folder2¶etc., the importer creates folders as needed. As the <i>destination value</i> has to be in square brackets, which in this context are syntactic symbols, both have to be masked with a backslash (\) in folder names with square brackets. Optionally, the filing path can also be entered in the index notation of the ELO client in order to automatically generate it from the metadata or via the document ID. In this case, the filing location must already exist. With the type attribute, you can define the metadata form for newly created folders. If this parameter is missing, the <i>Folder</i> metadata form is used. Example: <destination type="Invoice" value=" [...] + LA"/> (for description of placeholders LA, LK, etc., see Note).</p>
index	index field	<p>The name attribute contains either the number of the index field (1 to 51) or the group name of the index field. Since the latter is not necessarily unique (as a group entry can occur multiple times), choose the numerical form in these cases. If the group name is ELO_FNAME, the value is entered to the 51st index field (row for the file name).</p>
structure	Additional folders	<p>The value attribute contains a filing location that is created empty if it does not yet exist. In contrast to the destination tag, no reference is created to the document here; the target remains empty. With this structure tag, you can ensure that when creating a folder, a complete folder structure is created (even those that will not be filled through this import – see Note). You should use this function sparingly, since every <i>structure</i> path must be checked to determine it exists every time a filing process is initiated. Depending on the folder structure, this can have a significant impact on performance. With the type attribute, you can define the metadata form for newly created folders. If this parameter is missing, the <i>Folder</i> metadata form is used. As the <i>structure value</i> has to be in square brackets, which in this context are syntactic symbols, both have to be masked with a backslash (\) in folder names with square brackets.</p>
versions-comment	Comment	Version comment

Day	ELO field	Explanation (see example above)
docfile	Document file	The name attribute contains the file name of the object. It can remain empty, which creates a document without a document file. If a name is entered without a path, this will refer to the directory that contains the XML file. Every document must contain its own document file. If multiple documents exist compressed within a file, first, these must be unpacked in an extra pass before the XML file is transferred to ELO. The file name is saved in the version comment of the DocHistory table.
sigfile	Signature file	With this optional entry, the newly created document is given a signature immediately. The signature file must be created using the ELO signature components. A signature file can only be specified if a document file exists.
attfile	Attachment	Using this optional entry, the new document is created with an attachment. An attachment can only be specified when a document file exists.
path	Filing path	The number of the filing path when filing a new document. If this field is missing, the preset from the form definition will be used. If no value is entered there, the current default filing path is selected. Please note: If an invalid value is specified, the document cannot be filed.
repl	Replication sets	Normally, an XML entry inherits the replication set list from the target folder. However, if you want to file the document to the chaos folder, or set up custom replication sets for the documents, you can use this tag to specify replication sets. The list contains comma-separated 32-bit integer values that result in a bit vector for the desired sets.
encryption	Encryption key	Name of the encryption key. When using an ELO Indexserver with a main release older than 12, the password for the encryption key was entered in the <i>config.xml</i> . Starting with version 12, the concept of <i>extended encryption keys from ELO 12</i> has been implemented. Only encryption keys with standard security which have a user part and a system part with a separate system password are supported (see also the chapter Encryption in the ELO administration documentation). The encryption key information from the configuration file is no longer read.
workflow		Select an existing workflow with the template attribute. The name attribute sets the name of the workflow that is started.

Note

The following placeholders can be used for the filing paths (destination and structure tags):

- LA: The filing date is applied to the filing path (idate). For version IM.16.070.000 and higher, if no filing date has been set in the XML file, the current date is applied.
-

LD: The document date is applied to the filing path.

- LK: The short name is applied to the filing path.
- L2(1,15): The contents of the second index field are applied to the filing path starting with the first character and with a length of 15 characters.

If the XML control file is encoded in UTF-8, enter encoding="UTF-8" to the first line of the XML file. Otherwise, enter encoding="ISO-8859" as specified in the above [example](#).

The following characters must be entered as stated below:

Characters Masking Example

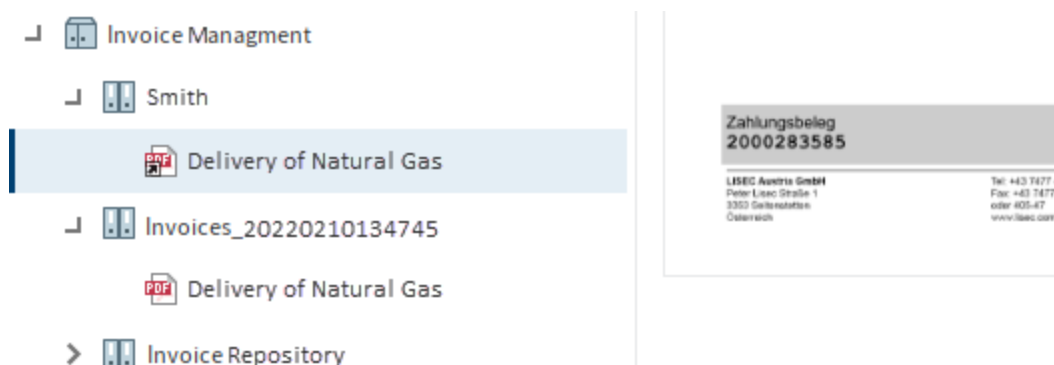
<	<	
>	>	
&	&	<destination value="[\\Monterrey&Cohen]"/>
"	"	
'	'	

Example for document import with reference

A PDF document with the file name *Delivery of Natural Gas* should be filed in ELO as an original in the folder *Invoice Management//Invoices_20220210134745*. Additionally, the code *LD* is added for the document date which is attached to the folder name *Invoices*. Last, the document should be filed as a reference in the folder *Smith*.

```
<destlist>
  <destination value="[\\Invoice Management \\Invoices_]+LD"/>
  <destination value="[\\Invoice Management\\Smith]"/>
</destlist>
```

Result

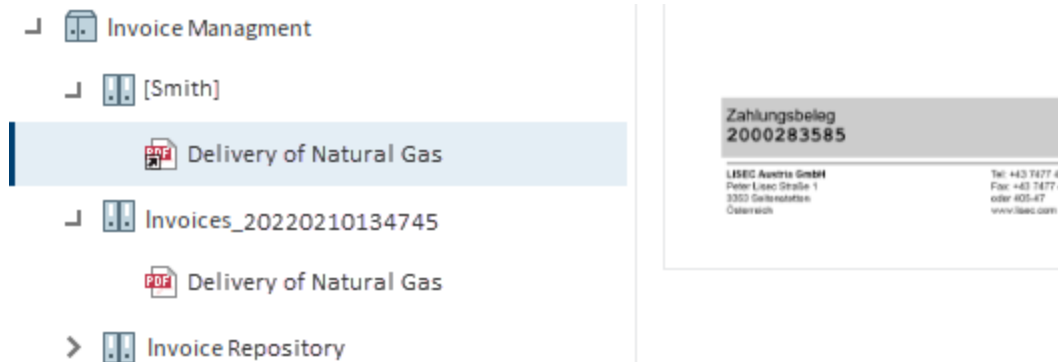


Example of masking square brackets as part of a folder name

As the destination value has to be in square brackets, which in this context are syntactic symbols, both have to be masked with a backslash (\) in folder names with square brackets.

```
<destlist>
  <destination value="[Invoice managementInvoices_]+LD"/>
  <destination value="[Invoice management\[Smith\]]"/>
</destlist>
```

Result



Starting the import process

The ELO XML Importer monitors one or more directories according to its configuration. As soon as a data stream is detected, the XML file is read, the required structural elements are created in the repository, and the documents are filed. The server decides whether a new stream is available based on the exclusive availability of the XML file. A data stream can be created as follows:

The source creates an XML file.

For each document, the document file is copied to the target directory and a corresponding <obj> entry is added to the XML file.

The XML file closes.

After the source has closed the XML file, ELO can start its processing.

Alternatively, the source can construct the XML file using a different extension than *.xml* (such as *.\$\$*). The source application renames the file to *.xml* when the process is complete and the file has been closed. This method is normally the most reliable, but it requires the source to be able to perform renaming.

If the source does not keep the XML file open permanently and renaming is not possible, you can alternatively set ELO to not recognize the completion of the action according to its unrestricted availability, but rather according to an additional control file.

1. The source creates the XML file *xyz.xml*.
2. The documents are written.
3. The XML file closes.
4. The signal file *xyz.sig* is created (empty).

ELO starts processing as soon as the signal file exists.

The HTML status page in the browser presents another option for starting the import process. Click the link *Import now* to start processing manually. This will also refresh the cache of the corresponding server, which is useful if, for example, the configuration has been changed via the ELO client in the meantime (new metadata form, user, etc.).

Column index

In ELO, it is possible to store multiple values to an index field, which is called a *column index*. This format is then useful when a range of equivalent data is available for an entry (such as a list of order numbers for a document named *Invoice*). It is very easy to use this column index entry in the XML control file. If multiple `<index>` entries with the same name attribute exist within the `<indexlist>`, they are automatically combined into a column index.

Example: Index field 1 contains the order numbers for an invoice; the numbers 123, 140, and 210 have been entered. The section of the XML file will then look like this:

```
<indexlist>
<index name="1" value="123"/>
<index name="1" value="140"/>
<index name="1" value="210"/>
</indexlist>
```

The index entries here can be placed in any order you wish; they do not need to be sorted. Note that an entry with the format `<index name="1" value="123,140,219"/>` generates a completely different result. This creates a single large entry instead of a column index. With an entry in this format, it is not possible to efficiently search for order number 140 over the SQL server.

Creating a structure

The required filing structure is automatically created according to the `<destination>` and `<structure>` tags as needed. Here you will only have limited influence on the metadata and the form type of the elements created. However, there is a possibility to explicitly create the structure using custom `<obj>` entries. You will then have complete influence on the object parameters and the metadata assignment. To prevent the folder from being created again each time, and in the process incorrectly creating multiple instances of it, the `<obj>` entry must be marked as *checked*, which is done via the mode attribute pass (`<obj mode="check">`).

An example of creating a structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<eloobjlist ver="1.0">
  <obj mode="check">
    <desc value="Folder99"/>
    <xdate value="20041011"/>
    <type value="Development project"/>
    <dtype value="2"/>
  </obj>
</eloobjlist>
```

```

<indexlist>
  <index name="EWP_TYP" value="Test project 2"/>
</indexlist>
<destlist>
  <destination type="Development" value="[XML Docs]"/>
</destlist>
</obj>
</eloobjlist >

```

This example will first check whether a folder with the name *Folder99* already exists in the *XML Docs* folder. If so, the `<obj>` entry will be ignored. Otherwise, this folder will now be created with the information from the XML control file, where you will have great influence over the basic parameters (e.g. metadata form or date entries) and the metadata. If the XML Docs folder in this example does not yet exist, it will also be created automatically.

Please note

In update mode/when importing ESW files, at least one document has to be imported to create structures using `<structure>` tags. When importing XML files, the `<structure>` tag can also be used without specifying a file for import.

Creating receipt files

Entry to config.xml	Explanation
ConfirmDir	Here, you enter the directory to which receipt files will be written. This entry must not have a \ at the end (as is the case for all directory entries in the XML profiles), such as E:\ELOenterprise\confirm.
ConfirmExt	File extension for the receipt files (.QIT)
ConfirmOk	Text in the receipt file for <i>documents correctly filed</i> . You can use the following placeholders here: %ID%: outputs the object ID of the new entry. %GUID%: outputs the GUID of the new entry. %DESC%: outputs the short name of the new entry. %IX:nn%: outputs the index field <i>nn</i> (starting with 0).
ConfirmDelOk	Text in the receipt file for "documents correctly deleted". You can enter the same placeholders as for ConfirmOk. Example: <entry key="import1_ConfirmDelOK">OK: Document %DESC% with PK: %IX:01%, %IX:01% deleted. guid:%GUID%</entry>
ConfirmErr	Text in the receipt file for <i>Error filing</i> .
ConfirmStructure	If true, receipt file for folders.

Entry to <i>config.xml</i>	Explanation
-------------------------------	-------------

ConfirmEmpty	If true, empty receipt files are also written.
--------------	--

By using the entries ConfirmDir, ConfirmExt, ConfirmOk, ConfirmErr, ConfirmStructure, and ConfirmEmpty, receipt files can be created for the imported data sets, or for the deleted data sets via ConfirmDelOk. The receipt file will be assigned the same name as the XML control file with the file extension defined under ConfirmExt. A line is written to the receipt file for each document entry in the XML control file, containing the text set in either ConfirmOk or ConfirmErr.

EXCEPTION: If the option is selected to check whether the documents exist and a new document or folder is not created because of it, nothing will be written to the receipt file either. Generating an empty receipt file can be set as an option in the *config.xml* file. If the XML structure of the input file is incomplete or damaged, processing may halt before the end of the document. If this occurs, fewer lines will be written to the receipt file than are blocks in the XML control file.

Automatic metadata update

The bulk import service can also be implemented for automatic metadata comparisons. Documents are filed to ELO with minimal metadata (such as a barcode). From an application, an XML file will then be created with the complete metadata. The service automatically adds to the metadata. The format of the XML file here will essentially be identical to that of the import file.

```
<?xml version="1.0" encoding="utf-8" ?>
<eloupdatelist ver="1.0">
  <obj>
    <source value="..." />
    <desc value="..." />
    <idate value="..." />
    <xdate value="..." />
    <acl>
      <access type="..." name="..." />
      ...
    </acl>
    <memo>...</memo>
    <indexlist>
      <index name="..." value="..." />
      ...
    </indexlist>
    <destlist>
      <destination value="..." />
      ...
    </destlist>
    <workflowlist>
      <workflow template="..." name="..." />
    </workflowlist>
    <docfile name="..." />
  </obj>
  ...
</eloupdatelist>
```

Differences between the individual tags

Tag	Name in import data set	Explanation
eloupdatelist	eloobjlist	This difference enables the service to differentiate between both types of lists.

Tag	Name in import data set	Explanation
obj	obj	<p>mode attribute: The value versioning can be entered here. The document will then be created if it does not yet exist.</p> <p>With this field, the existing ELO entry with minimal metadata will be addressed. This can be the internal ELO Objectid (#1234 means <i>Objectid</i> 1234) or a GUID, which also needs to be preceded by a "#", e.g. #(B66B68360A-87F4-4AC6-874A-9FE92DF9E6). However, it can also be a filing path with the format [¶sord1¶sord2...]. It can also be one or more index fields, e.g. "BARCODE=1234567;REGNR=C2Z68", if the entry with the code 1234567 and the entry REGNR with the code C2Z68 are to be selected via the index field BARCODE (you need to specify the group name and not the field name). If the document meant to be updated cannot be uniquely identified, an error document is generated.</p>
source	./ . not available	
desc	desc	<p>If the value attribute of this tag remains empty, the short name will be retained automatically from the existing record.</p>
idate, xdate	idate, xdate	<p>If the value attribute of this tag remains empty, the date information will be retained automatically from the existing record.</p>
index	index	<p>The index tag can also contain a mode attribute. If this is set to the value new, this entry will overwrite all existing entries in this index field. If this is not set, new index values will be added to the existing ones in column index format.</p>
destlist	destlist	<p>The destlist can remain empty, which will keep the entry in its original location in the repository. If the destlist is not empty, the document is moved from its original filing location to the first entry in the list. As the destination value has to be in square brackets, which in this context are syntactic symbols, both have to be masked with a backslash (\) in folder names with square brackets. References will be created for all additional list entries. The first entry can also simply be filled with a "*". In this case, the document will remain at its original location and the additional references will be created (see Examples for metadata update with "destlist").</p>
docfile	docfile	<p>If a docfile is specified in this update data set, it will be attached as a new version to the logical document.</p>
attfile	attfile	<p>If attfile is specified in this update data set, it will be attached as a new version of a document's attachment.</p>
acl		<p>All keys (<access>) are located in the <acl> tag. The <acl> tag can additionally contain the attribute mode="new". This attribute indicates that all existing keys have been deleted and only the new ones will be saved.</p>

Tag	Name in import data set	Explanation
dtype	Document type	Here you can change the document type (values 254 to 310) and the type of folder (values 1 to 253) (from ELO 10, the check to verify whether the values lie in the range 1 to 253 is omitted). Warning: No check as to whether you are changing a document or folder takes place here. For this reason, please take special care to set correct values in the control file.
arcmode	Document status (filing mode)	If the document has been filed with the status <i>Non-modifiable</i> (previously <i>no changes possible, read-only</i>), the document status cannot be changed again.
map		<map> can contain an attribute mode="new". This will delete the old map and create a new one.
Structure	Folder	In update mode, at least one document has to be imported (including when importing ESW files). As the structure value has to be in square brackets, which in this context are syntactic symbols, both have to be masked with a backslash (\) in folder names with square brackets.

All other tags will be used as in the import data set.

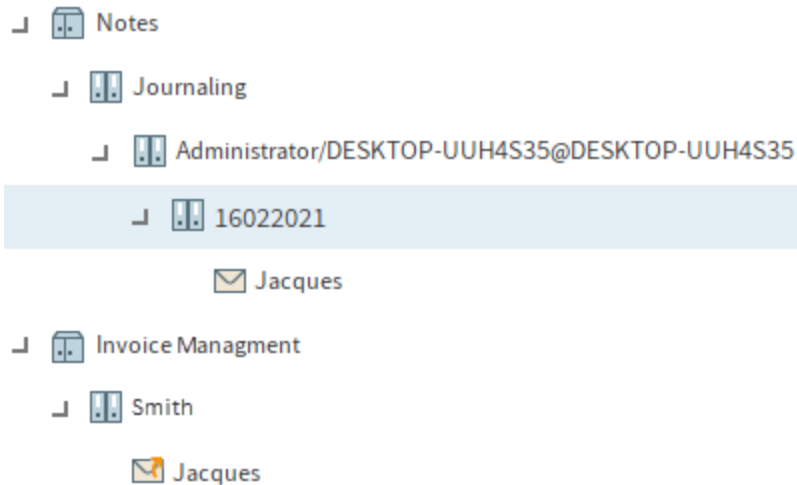
Examples for metadata update with "destlist"

Example 1

A metadata update should be carried out in ELO where additional references for the original document *Jacques* in the folder *Notes//Journaling//Administrator//DESKTOP-UUH4S35@DESKTOP-UUH4S35/16022021* are created in the folder *Invoice Management//Smith*. To ensure the original document remains in its folder, enter "*" as the destination value.

```
<destlist>
  <destination value="*" />
  <destination value="[Invoice Management\Smith]" />
</destlist>
```

Result

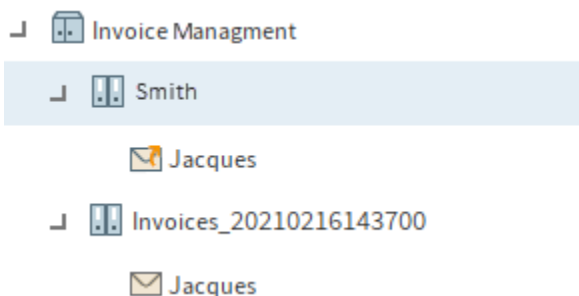


Example 2

A metadata update should be carried out in ELO where the original document *Jacques* is moved to the folder *Invoice Management//Invoices_20210216143700* and a reference is filed to the folder *Invoice Management//Smith*. The original document will be moved to the folder entered in the first destination value. A reference to the original document will be created in the folder entered in the second destination value.

```
<destlist>
  <destination value="[Invoice ManagementInvoices_]+LD"/>
  <destination value="[Invoice ManagementSmith]"/>
</destlist>
```

Result











Example 3

A metadata update should be carried out in ELO where additional references for the original document *Jacques* in the folder *Notes//Journaling//Administrator/DESKTOP-UUH4S35@DESKTOP-UUH4S35/16022021* are created in the folder *Invoice Management//[Smith]*. To ensure the original document remains in its folder, enter "*" as the destination value. As the second destination value has to be in square brackets, which in this context are syntactic symbols, both have to be masked with a backslash (\) in folder names with square brackets.

```
<destlist>
  <destination value="*" />
  <destination value="[Invoice management\[Smith\]]" />
</destlist>
```

Result

- └─  Notes
 - └─  Journaling
 - └─  Administrator/DESKTOP-UUH4S35@DESKTOP-UUH4S35
 - └─  16022021
 -  Jacques
- └─  Invoice Managment
 - └─  [Smith]
 -  Jacques

Deleting data

The ELO XML Importer can also be used to automatically delete data in ELO. The documents and folders are entered to a deletelist and deleted by the ELO XML Importer. The format of the XML file here will essentially be identical to that of the import or update control file, but only the ELO object IDs or GUIDs have to be entered.

Example delete control file:

```
<?xml version="1.0" encoding="utf-8" ?>
<elodeletelist ver="1.0">
  <obj>
    <source value="#1234" />
  </obj>
  ...
</elodeletelist>
```

The objects to be deleted are addressed via an entry in the source field. The entry can be an internal ELO object ID (#1234 means *object ID 1234*) or a GUID, which has to be proceeded with a "#" (e.g. #(B66B68360A-87F4-4AC6-874A-9FE92DF9E6)).

The documents and folders are not deleted permanently; they can still be found in the Recycle bin.

If you want receipt files to be written, configure ConfirmDelOk in the *config.xml* file with the same syntax as for ConfirmOk.