ELO XC

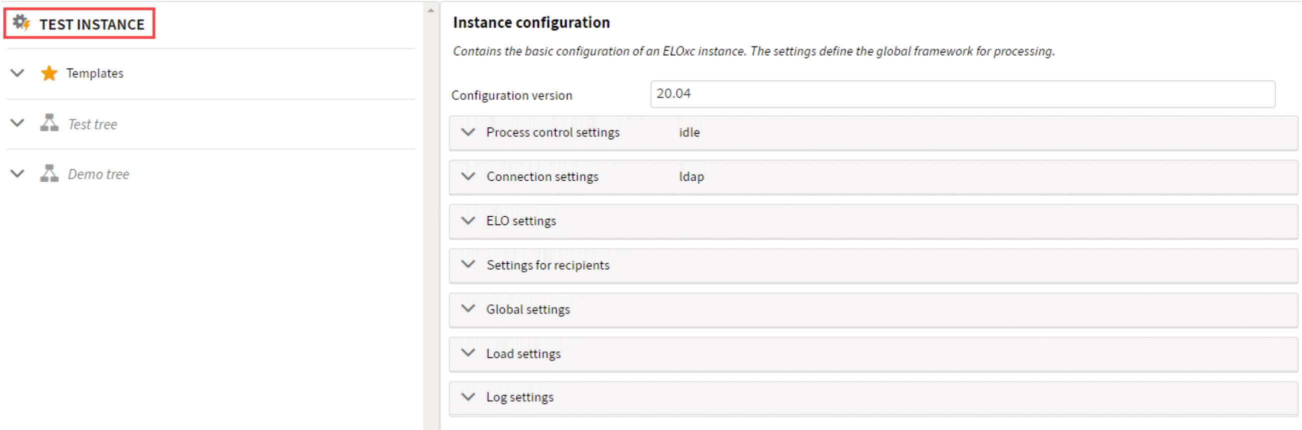
Configuration

Table of contents

Configuration

Instance

By clicking on the instance node, you can edit the instance parameters in the form view.



TEST INSTANCE

Templates

Test tree

Demo tree

Instance configuration

Contains the basic configuration of an ELOxc instance. The settings define the global framework for processing.

Configuration version: 20.04

Process control settings: idle

Connection settings: ldap

ELO settings

Settings for recipients

Global settings

Load settings

Log settings

The configuration version is always identical to the program version. If ELO XC detects an outdated configuration version while loading an instance, an update will occur automatically. If you want an update to run again, you can enter an older version under *Configuration version* so that the internal update runs again the next time the configuration is validated.

The instance configuration contains the following parameter settings:

- Processing settings: Processing activity and frequency
- Connection data: Access settings for LDAP, m365, and EWS
- ELO settings: Connection to the ELO Indexserver and ELO repository
- Scaling: Load behavior during processing, EWS settings
- Recipients: Supported mailbox types, use and recognition of SMTP addresses
- Additional settings: Various parameters of no specific type

Processing settings

When the program starts, all instance configurations are loaded, validated, and executed according to the defined processing settings. A configuration is executed as a job.

The time specifications of the processing settings are all in the format *dd:hh:mm:ss*:

- d: Day
- h: Hour
- m: Minute
- s: Second

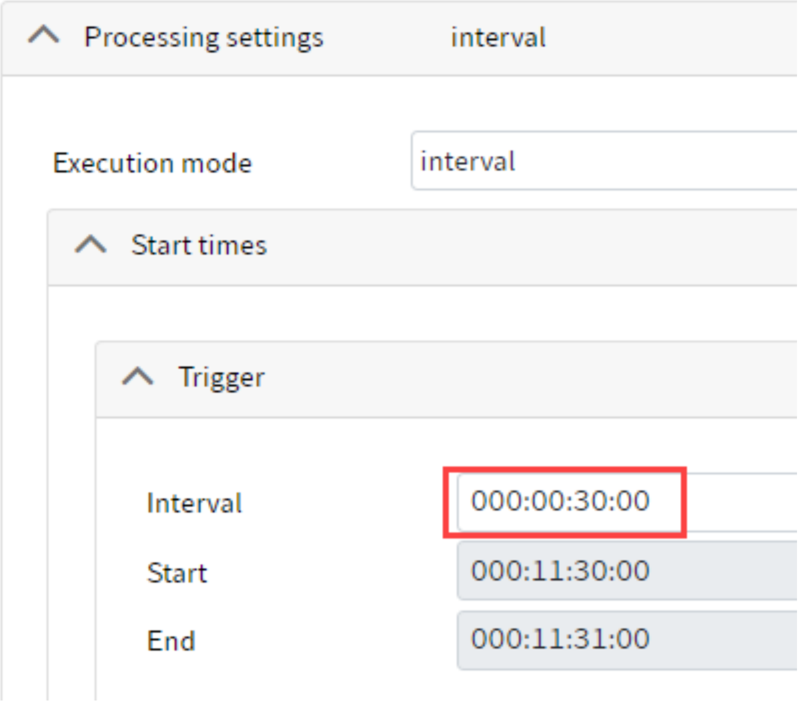
Processing depends on the respective parameter and the execution mode.

You can choose between the following execution modes:

-

idle: After loading and validating the configuration, the instance is in an idle state. Jobs are only executed manually.

- once: Exactly one job is executed. Afterwards, the instance is idle.
- interval: Only one trigger is fired in this mode. Any number of jobs are executed. The trigger interval specifies the delay between jobs.



Processing settings interval

Execution mode: interval

Start times

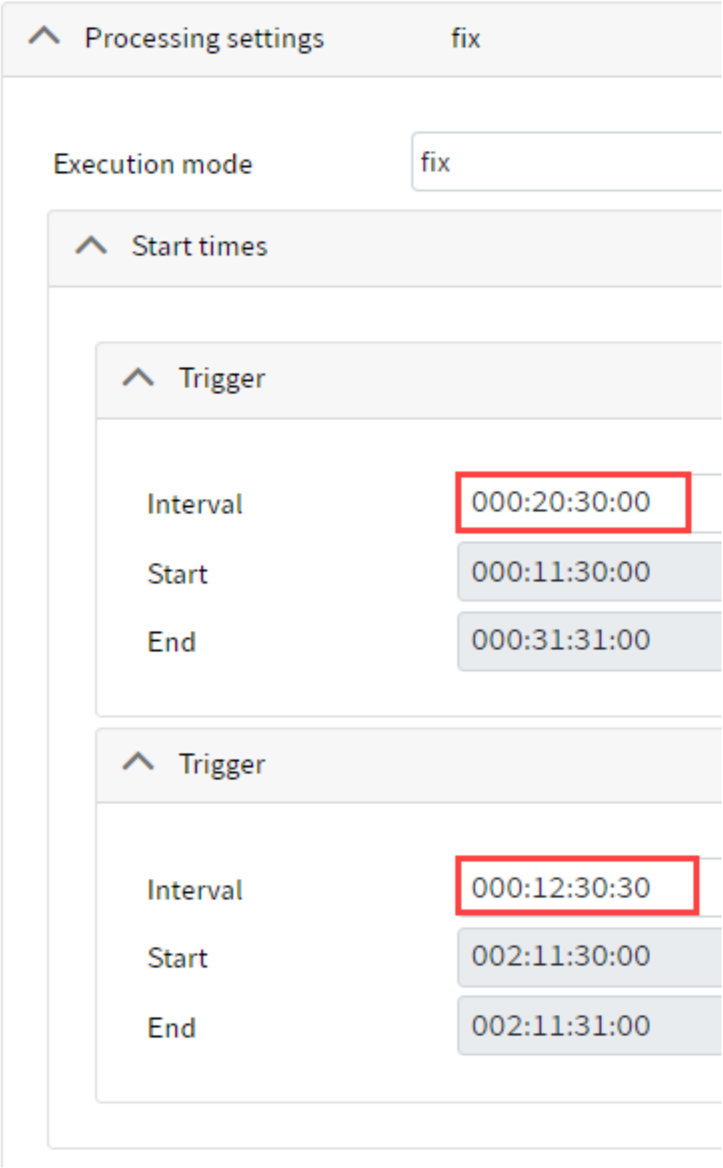
Trigger

Interval: 000:00:30:00

Start: 000:11:30:00

End: 000:11:31:00

- fix: You can use as many triggers as you like. The trigger interval is a fixed time at which the job starts.



Processing settings fix

Execution mode: fix

Start times

Trigger

Interval: 000:20:30:00

Start: 000:11:30:00

End: 000:31:31:00

Trigger

Interval: 000:12:30:30

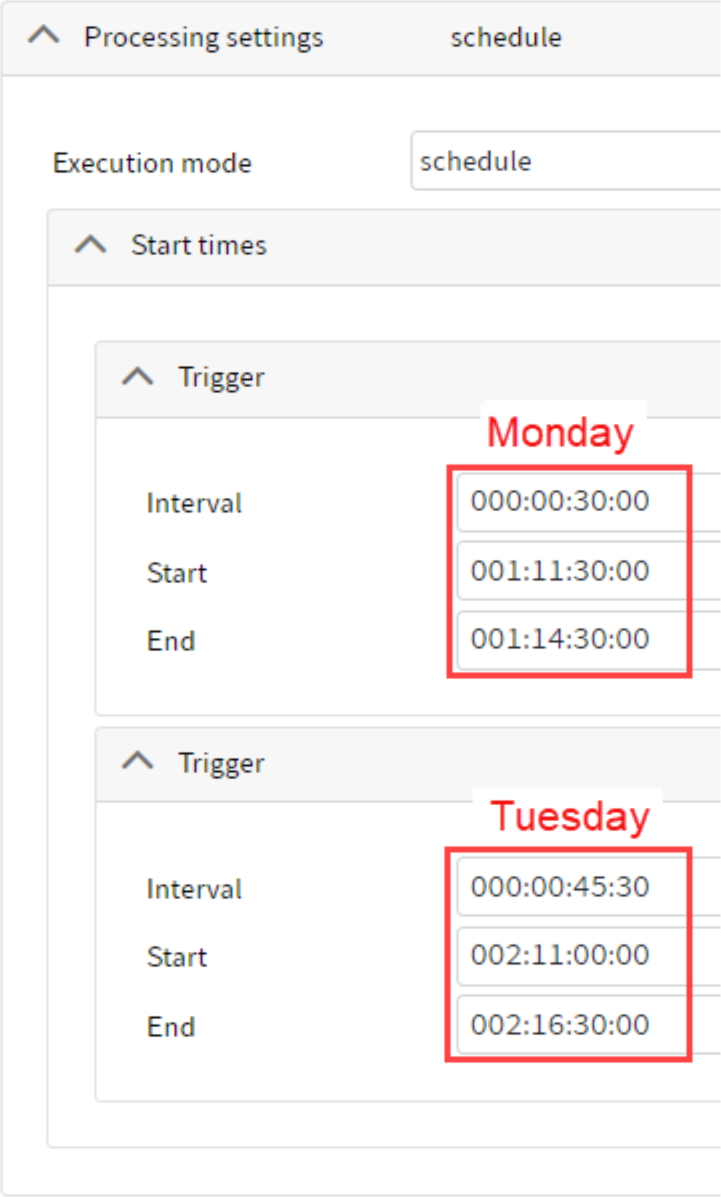
Start: 002:11:30:00

End: 002:11:31:00

In the figure above, jobs are executed daily at 12:30 and 20:30. The first date component is ignored.

- schedule: You can use as many triggers as you like. Each trigger has a defined start and end on a specific day as well as a corresponding period of activity. The trigger interval specifies the delay between processing within the period of activity,

  The first date component is the day of the week, with `0` being Sunday and `6` being Saturday. The remaining components specify the period of activity. The interval specifies the delay between jobs, although the first component is ignored.



Processing settings schedule

Execution mode schedule

Start times

Trigger

Monday

Interval 000:00:30:00

Start 001:11:30:00

End 001:14:30:00

Trigger

Tuesday

Interval 000:00:45:30

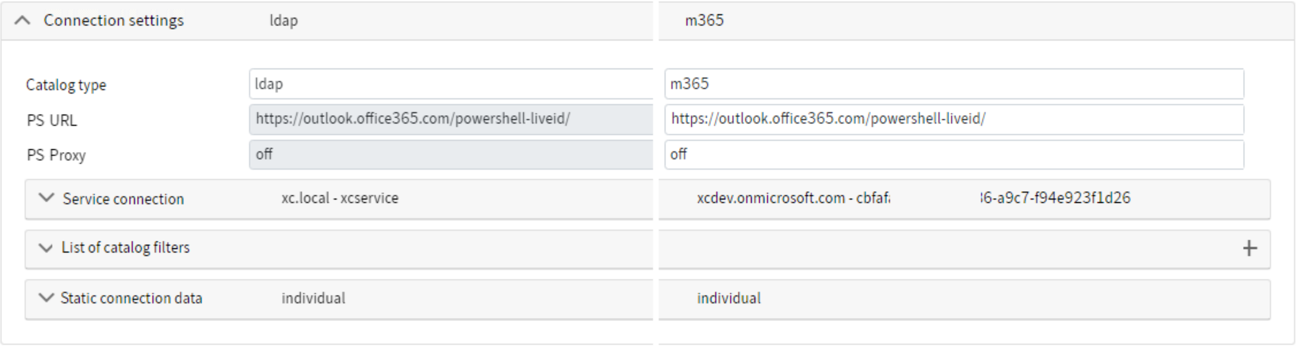
Start 002:11:00:00

End 002:16:30:00

In the figure above, jobs are executed on Monday between 11:30 and 14:30 with a delay of 30 minutes between each run. On Tuesdays, jobs are executed between 11:00 and 16:30 with a delay of 45 minutes and 30 seconds.

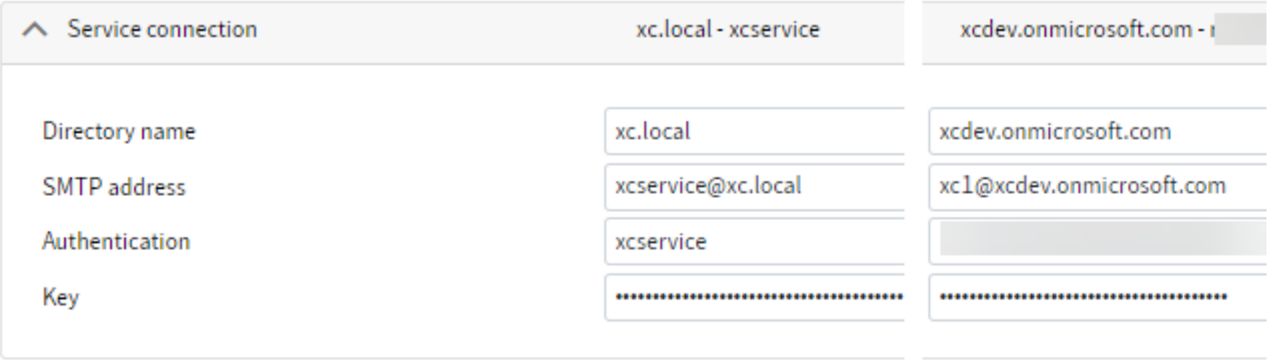
- ondemand: Internal use; is not supported
- creator: Internal use; is not supported

Connection data



Connection settings	ldap	m365
Catalog type	ldap	m365
PS URL	https://outlook.office365.com/powershell-liveid/	https://outlook.office365.com/powershell-liveid/
PS Proxy	off	off
Service connection	xc.local - xcservice	xcdev.onmicrosoft.com - cbfaf[redacted]6-a9c7-f94e923f1d26
List of catalog filters		+
Static connection data	individual	individual

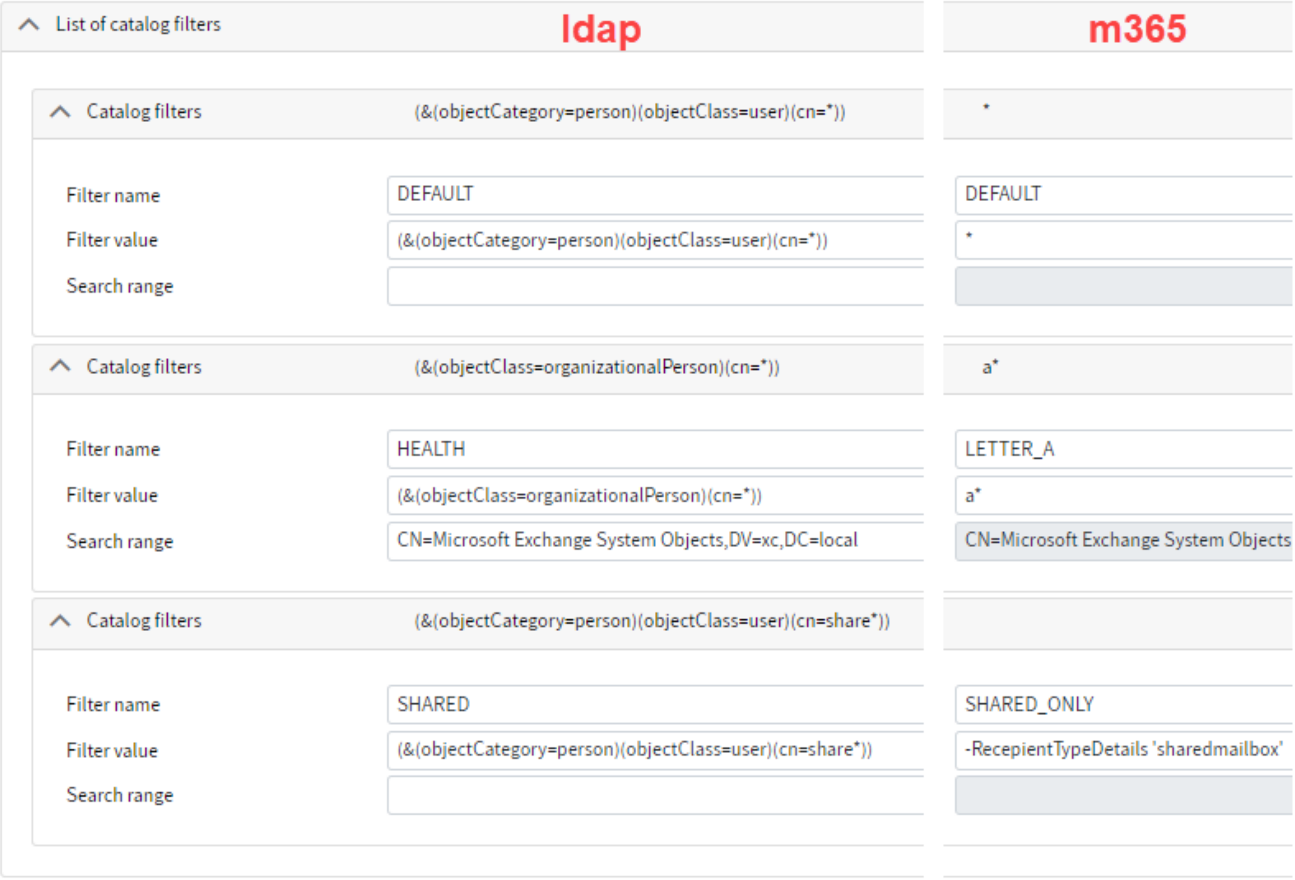
The catalog settings determine which mailboxes can be processed by an instance. The *ldap* and *m365* types automatically retrieve mailboxes with configurable filters through the LDAP and PowerShell interfaces respectively. With the *manual* type, all available mailboxes have to be configured individually.



Service connection	xc.local - xcservice	xcdev.onmicrosoft.com - ı[redacted]
Directory name	xc.local	xcdev.onmicrosoft.com
SMTP address	xcservice@xc.local	xc1@xcdev.onmicrosoft.com
Authentication	xcservice	[redacted]
Key	••	••

The catalog type determines the service connection parameters.

- Directory name: The local domain name or name of the *m365* directory (*tenant*).
- SMTP address: Optional address used as the sender for automatic error messages (see Additional settings) by e-mail.
- Authentication: Credentials for the local domain or ID of the app registration in *m365*.
- Key: Password for the local domain or the fingerprint of the certificate from the app registration in *m365*.

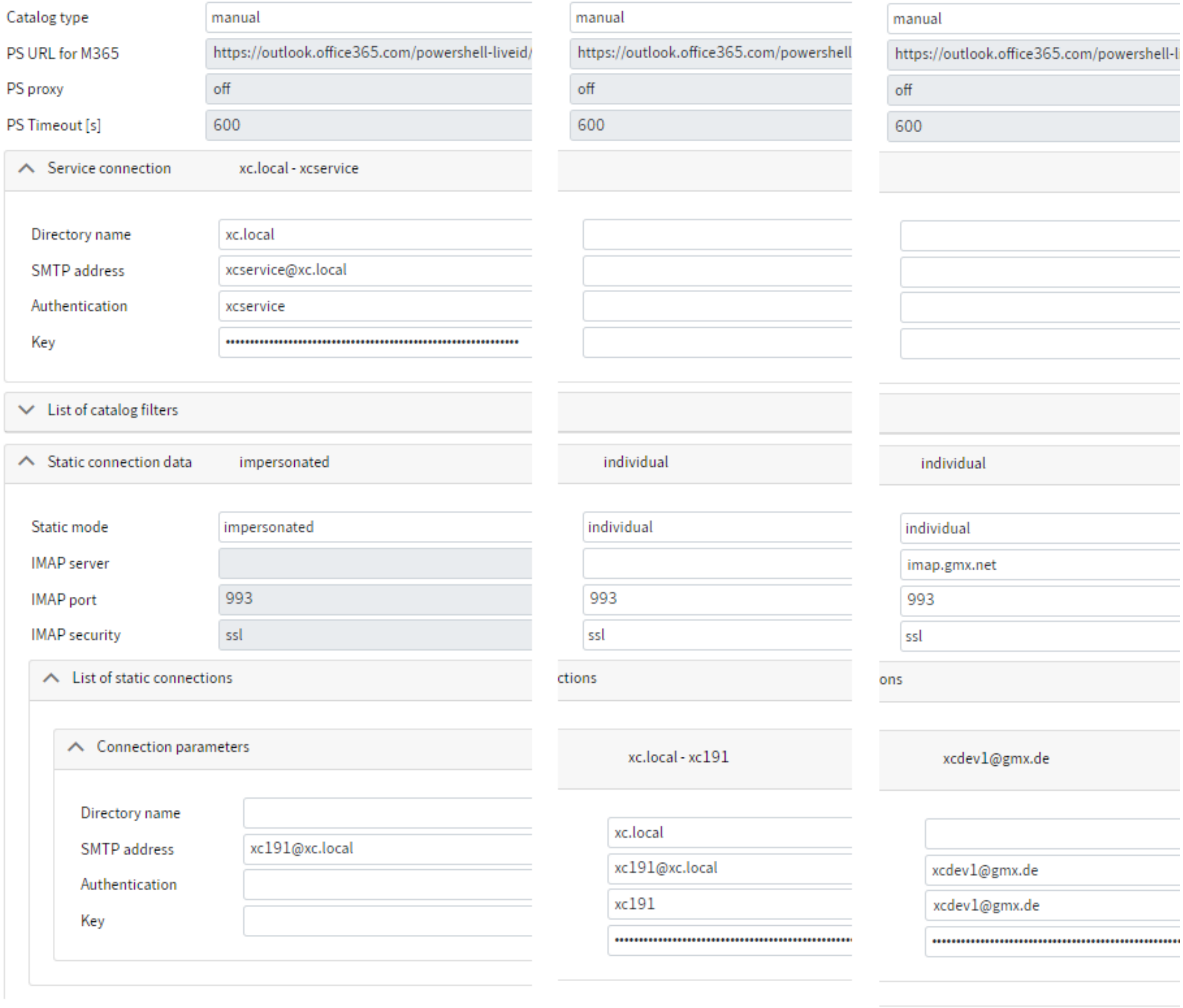


List of catalog filters

	ldap	m365
Catalog filters	(&(objectCategory=person)(objectClass=user)(cn=*))	*
Filter name	DEFAULT	DEFAULT
Filter value	(&(objectCategory=person)(objectClass=user)(cn=*))	*
Search range		
Catalog filters	(&(objectClass=organizationalPerson)(cn=*))	a*
Filter name	HEALTH	LETTER_A
Filter value	(&(objectClass=organizationalPerson)(cn=*))	a*
Search range	CN=Microsoft Exchange System Objects,DV=xc,DC=local	CN=Microsoft Exchange System Objects
Catalog filters	(&(objectCategory=person)(objectClass=user)(cn=share*))	
Filter name	SHARED	SHARED_ONLY
Filter value	(&(objectCategory=person)(objectClass=user)(cn=share*))	-RecepientTypeDetails 'sharedmailbox'
Search range		

The catalog filters determine how the catalogs are queried in LDAP or PowerShell. All the standard filters can be used for LDAP. You can also restrict the search range to an LDAP container (e.g. organizational unit). Only the *Get-Mailbox* cmdlet filters are allowed in the PowerShell catalog.

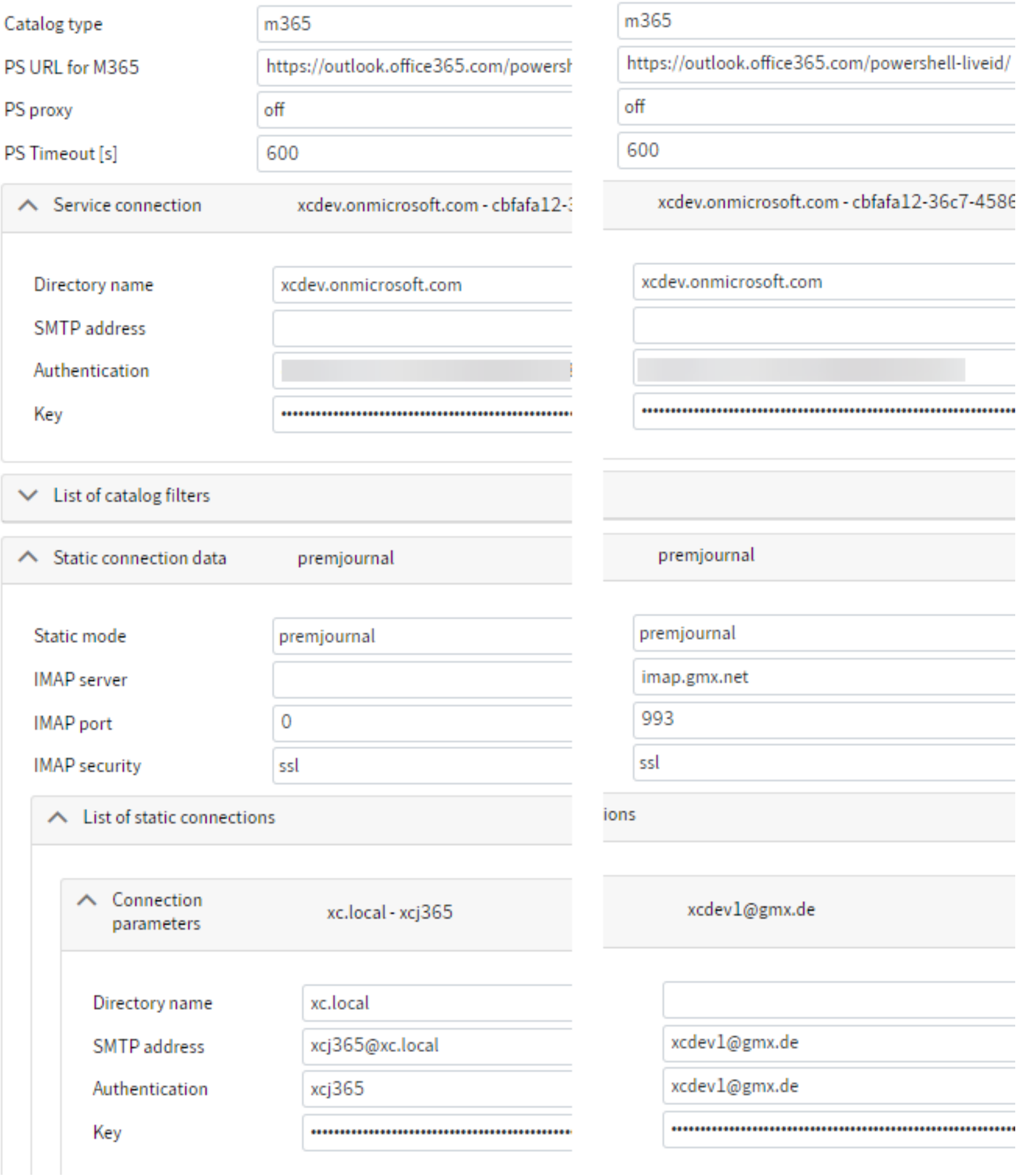
The mailbox query is executed once for each catalog filter. The corresponding filter name is stored internally for each hit. If you want to use the *filter* mailbox type in mailbox list templates or action trees, you have to use one of these filter names.



Catalog type	manual	manual	manual
PS URL for M365	https://outlook.office365.com/powershell-liveid/	https://outlook.office365.com/powershell	https://outlook.office365.com/powershell-l
PS proxy	off	off	off
PS Timeout [s]	600	600	600
Service connection	xc.local - xcservice		
Directory name	xc.local		
SMTP address	xcservice@xc.local		
Authentication	xcservice		
Key	••		
List of catalog filters			
Static connection data	impersonated	individual	individual
Static mode	impersonated	individual	individual
IMAP server			imap.gmx.net
IMAP port	993	993	993
IMAP security	ssl	ssl	ssl
List of static connections		ctions	ons
Connection parameters		xc.local - xc191	xcdev1@gmx.de
Directory name		xc.local	
SMTP address	xc191@xc.local	xc191@xc.local	xcdev1@gmx.de
Authentication		xc191	xcdev1@gmx.de
Key		••	••

If you set the *manual* catalog type, mailboxes are not automatically retrieved according to the catalog connection and filter settings. You need to manually enter them as static connections in the configuration. The static mode determines the configuration scope.

- impersonated: As with automatic catalogs, you need to enter the service connection here. However, no catalogs are retrieved. The mailboxes are added to the static connection list with their SMTP addresses.
- individual: Each mailbox in the static connection list must be entered in full, including directory, authentication data, and key/password. If you specify a parameter for IMAP servers, ELO XC attempts to establish the connection using IMAP protocol. You can also configure the IMAP port and IMAP security settings.



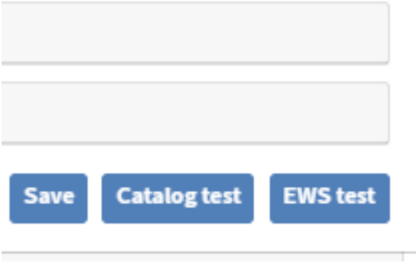
Catalog type	m365	m365
PS URL for M365	https://outlook.office365.com/powersh	https://outlook.office365.com/powershell-liveid/
PS proxy	off	off
PS Timeout [s]	600	600
Service connection	xcdev.onmicrosoft.com - cbfafa12-3	xcdev.onmicrosoft.com - cbfafa12-36c7-4586
Directory name	xcdev.onmicrosoft.com	xcdev.onmicrosoft.com
SMTP address		
Authentication	[redacted]	[redacted]
Key	••	••
List of catalog filters		
Static connection data	premjournal	premjournal
Static mode	premjournal	premjournal
IMAP server		imap.gmx.net
IMAP port	0	993
IMAP security	ssl	ssl
List of static connections		ions
Connection parameters	xc.local - xcj365	xcdev1@gmx.de
Directory name	xc.local	
SMTP address	xcj365@xc.local	xcdev1@gmx.de
Authentication	xcj365	xcdev1@gmx.de
Key	••	••

If you want to configure journaling for a tenant in Microsoft 365, you must be sure that the journal function has access to a suitable catalog for resolving the recipients and that there is no journal mailbox in Microsoft 365. The journal mailbox must be external. The catalog queries are carried out according to the catalog filters to retrieve the expected journal recipients, but only the statically configured mailboxes are processed as journal mailboxes.

The following table lists the allowed connection configurations:

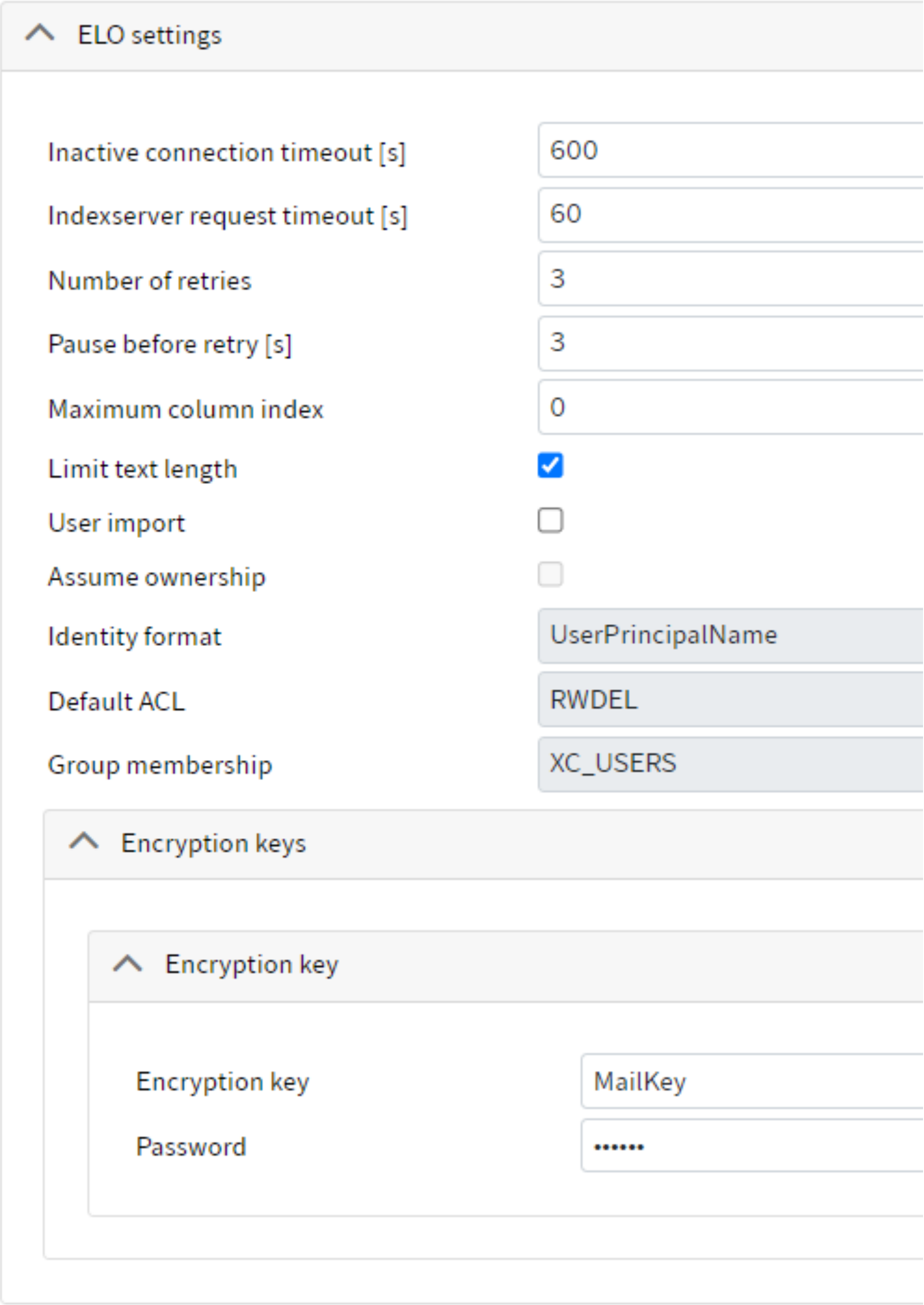
Catalog type	Static mode	Service connection	Static connections	IMAP
ldap	-	Mailbox/catalog authentication	-	No
m365	-	Mailbox/catalog authentication	-	No
manual	premjournal	Catalog authentication	Journal mailbox authentication	Yes
manual	impersonated	Mailbox authentication	SMTP address only	No
manual	individual	No	Mailbox authentication	Yes

The *Connection test* tool allows you to test the connection parameters.



You will find more information about the *catalog and EWS test* connection tests in the chapter Tools > Connection test.

ELO settings



ELO settings

Inactive connection timeout [s]	600
Indexserver request timeout [s]	60
Number of retries	3
Pause before retry [s]	3
Maximum column index	0
Limit text length	☑
User import	☐
Assume ownership	☐
Identity format	UserPrincipalName
Default ACL	RWDEL
Group membership	XC_USERS

Encryption keys

Encryption key

Encryption key	MailKey
Password	••••••

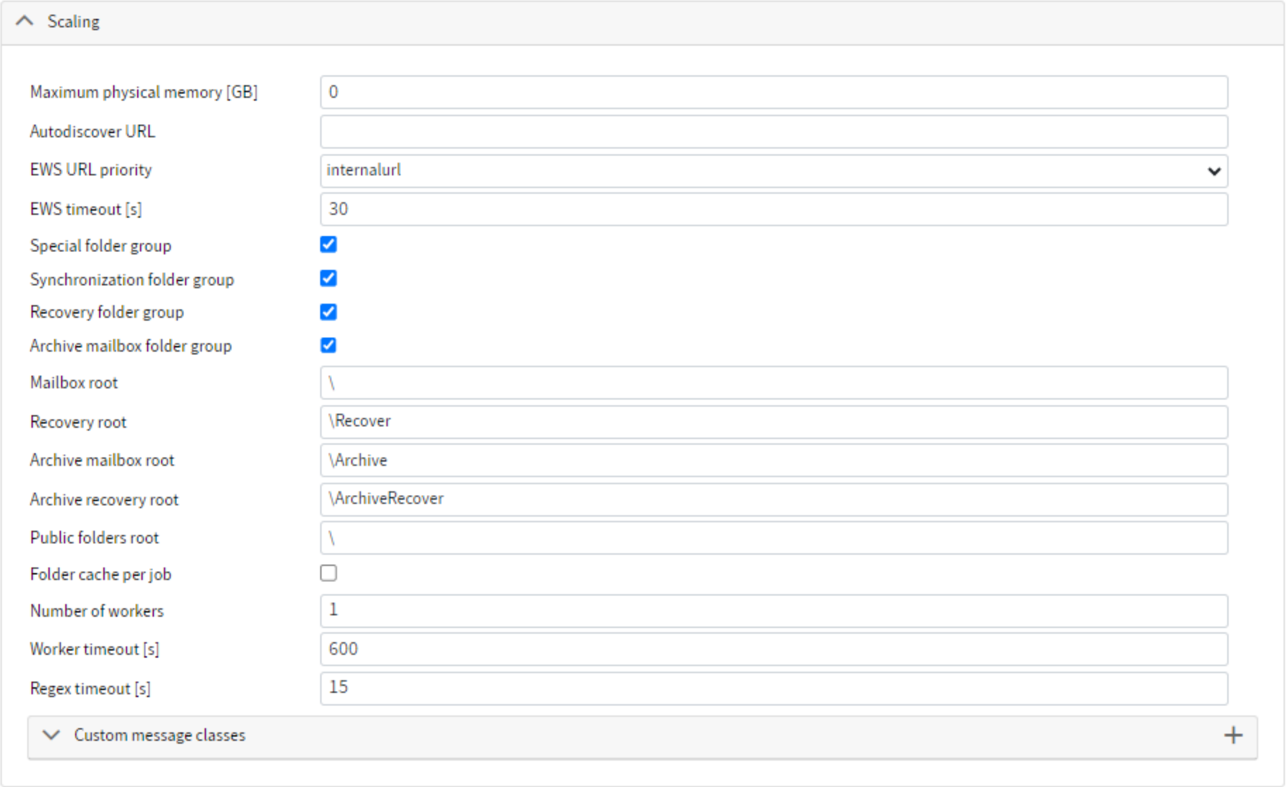
You can use the following parameters to connect to the ELO repository:

- Indexserver request timeout [s]: Time allowed for calls to the ELO Indexserver
- Inactive connection timeout [s]: Idle time allowed for connections to the ELO Indexserver
- Maximum column index: Lists such as e-mail recipients can be stored in metadata fields as column indexes, but this can lead to a significant load on the database if you store very large

lists of recipients in this way. A value greater than 0 limits the number of column index entries created per message.

- Limit text length: Stored metadata is truncated if the field length in the database is exceeded. Without this option, ELO XC would run into processing errors if the field length were exceeded.
- User import: If this option is enabled, external users are imported into ELO.
- Identity format: *UserPrincipalName*, *SAMAccountName*, and *CN* are the properties used to generate the ELO user name on import. If you enter the value *IxLdap*, users are exclusively imported using the ELO Indexserver LDAP interface.
- Assume ownership: With this option, the respective mailbox owner/ELO user is set as the owner of metadata and documents in the ELO repository. Otherwise, the ELO XC service account is used.
- Default ACL: This ACL is set for folders and messages transferred to the ELO repository.
- Group membership: Imported users are assigned to this ELO group.

Scaling



Scaling

Maximum physical memory [GB]: 0
Autodiscover URL:
EWS URL priority: internalurl
EWS timeout [s]: 30
Special folder group: ☑
Synchronization folder group: ☑
Recovery folder group: ☑
Archive mailbox folder group: ☑
Mailbox root: \
Recovery root: \Recover
Archive mailbox root: \Archive
Archive recovery root: \ArchiveRecover
Public folders root: \
Folder cache per job: □
Number of workers: 1
Worker timeout [s]: 600
Regex timeout [s]: 15

Custom message classes +

- Maximum physical memory [GB]: This value allows you to set a memory limit at runtime.

- Autodiscover URL: You can enter an Autodiscover URL to call a specific Autodiscover service. If the field is empty, the Autodiscover endpoint is determined implicitly from the e-mail domain of the SMTP address using DNS.

- EWS URL priority: To speed up EWS routing, you can specify whether to call the internal or external EWS URL first.

-

EWS timeout [s]: This value in seconds specifies when to timeout the EWS interface and Autodiscover.

- Folder groups (Special, Synchronization, Recovery, Archive mailbox): You can use folder variables to ensure language-independent configuration of mailbox folders (e.g., in entry points). The folder types in EWS are assigned as follows:

  Standard: MsgFolderRoot, Calendar, Contacts, DeletedItems, Inbox, Notes, SentItems, Tasks, PublicFoldersRoot

  Special: Drafts, Journal, VoiceMail, JunkEmail, SearchFolders

  Synchronization: RecipientCache, ConversationHistory, QuickContacts, MyContacts, IMContactList, PeopleConnect, Favorites, AllContacts

  Recovery: RecoverableItemsRoot, RecoverableItemsDeletions, RecoverableItemsVersions, RecoverableItemsPurges

  Archive mailbox: ArchiveRoot, ArchiveMsgFolderRoot, ArchiveInbox, ArchiveDeletedItems

  Archive mailbox and *recovery*: ArchiveRecoverableItemsRoot, ArchiveRecoverableItemsDeletions, ArchiveRecoverableItemsVersions, ArchiveRecoverableItemsPurges

- Mailbox root: Determines the plain text of the root variable *{%MSGROOT}*. At least \ must be used.

- Recovery root: Determines the plain text of the root variable *{%RECITEMSROOT}*. At least \ must be used.

- Archive mailbox root: Determines the plain text of the root variable *{%ARCMSGROOT}*. At least \ must be used.

- Archive recovery root: Determines the plain text of the root variable *{%ARCRECITEMSROOT}*. At least \ must be used.

- Public folders root: Determines the plain text of the *public folders* root. At least \ must be used.

- Folder cache per job: When an action trees starts to process a mailbox, the entire folder hierarchy, i.e. language-independent folders and user folders, is always retrieved first. If a mailbox is processed in multiple action trees, this option enables you to stop folders being retrieved multiple times.

- Number of workers: Determines the number of parallel execution paths allocated to a selected message set.

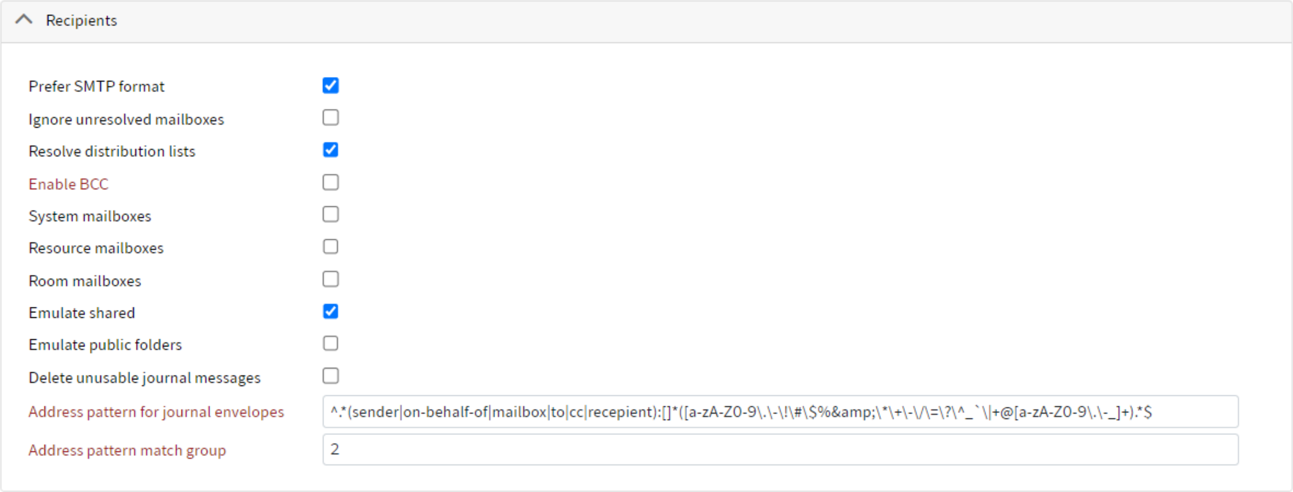
- Worker timeout [s]: The maximum timeout limit for a response from execution paths. If this limit is exceeded, ELO XC will terminate in order to free up resources.

- Regex timeout [s]: The maximum timeout limit for regex operations.

-

Custom message classes: Custom derivatives of the standard message class *IPM.Note* must be entered here so that they can be configured in action trees in addition to the permanent Exchange message classes.

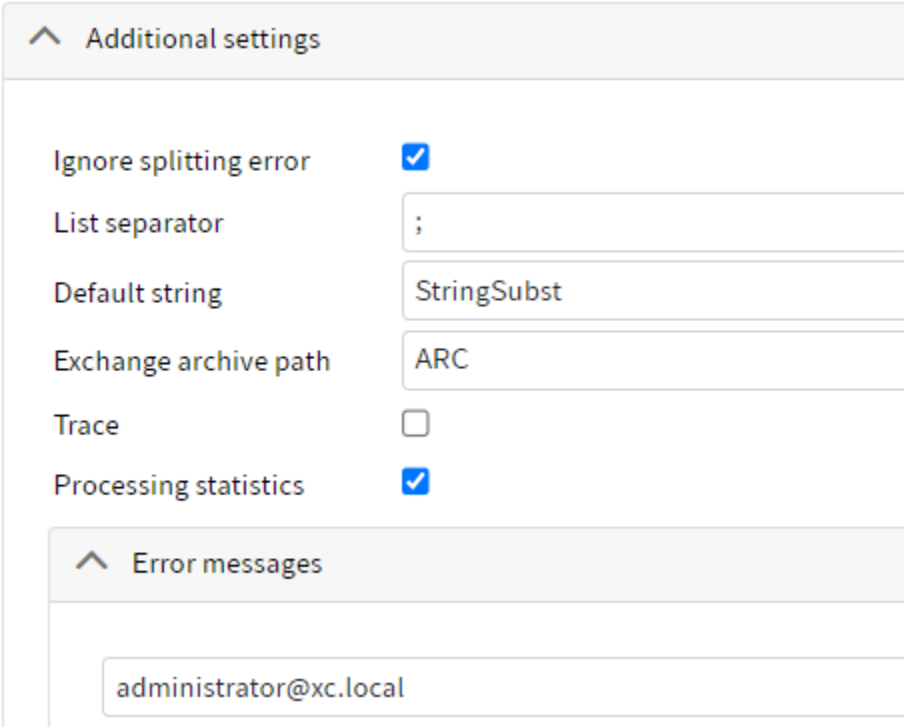
Recipients



Prefer SMTP format	☑
Ignore unresolved mailboxes	☐
Resolve distribution lists	☑
Enable BCC	☐
System mailboxes	☐
Resource mailboxes	☐
Room mailboxes	☐
Emulate shared	☑
Emulate public folders	☐
Delete unusable journal messages	☐
Address pattern for journal envelopes	^.*(sender|on-behalf-of|mailbox|to|cc|recepient):[]*([a-zA-Z0-9\.\-\!\#\$%&amp;*\+\-\/\=\?\^_`\\|+@[a-zA-Z0-9\.\-_]+).*$
Address pattern match group	2

- Prefer SMTP format: When using hybrid properties (e.g. EloSender), this option determines whether to use SMTP format or the internal LDAP format.
- Ignore unresolved mailboxes: If errors occur while retrieving mailboxes processed by an action tree, the program aborts processing. If you enable this option, the error is ignored.
- Resolve distribution lists: When transferring recipients to metadata, this option resolves distribution lists completely or recursively into SMTP addresses.
- Enable BCC: BCC recipients may appear in sender messages and in journals. If you don't want this to happen, disable this option.
- System mailboxes, Resource mailboxes, Room mailboxes: You need to enables these options if you want the different Exchange mailbox types to be included in the mailbox catalogs.
- Emulate shared: The *Emulate shared* option determines whether or not to emulate *shared mailboxes* (*SharedMailbox* type). This option is enabled by default. If it is disabled, *shared mailbox* delegates or the associated user mailboxes are not resolved.
- Emulate public folders: If public folders and their SMTP addresses appear on journal envelopes, they can be emulated as separate mailboxes if you enable this option.
- Delete unusable journal messages: This option moves all unusable journal messages to the *Deleted items* folder. A journal message is unusable when no recipient has been cataloged for the envelope, or there is no e-mail located in the envelope, i.e. no e-mail attached.
- Address pattern for journal envelopes and Address pattern match group: Both parameters can be used to customize the default identifier for recipients on journal envelopes. However, you should only change the default setting in exceptions.

Additional settings



Ignore splitting error ☑

List separator ;

Default string StringSubst

Exchange archive path ARC

Trace □

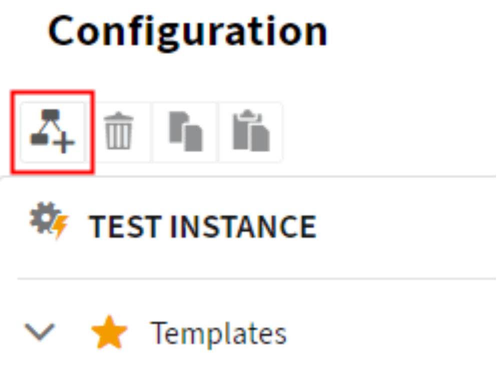
Processing statistics ☑

administrator@xc.local

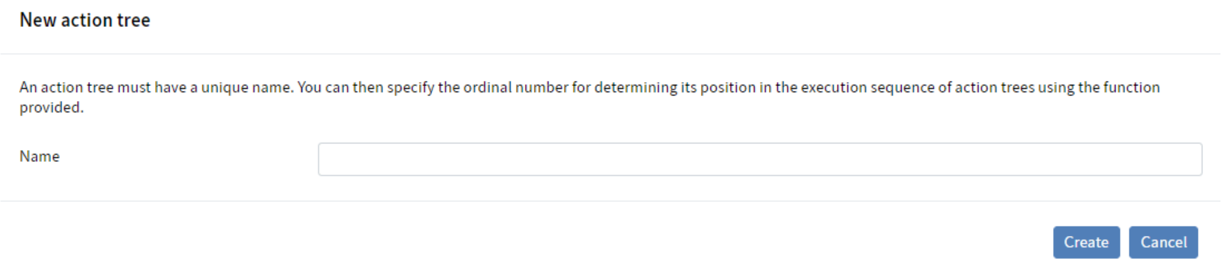
- Ignore splitting error: If errors occur when extracting attachments, this option can be used to control whether processing of the corresponding message should fail or whether processing should continue.
- List separator: This separator is used wherever properties consisting of lists are copied into simple metadata fields.
- Default string: This global string is used as a substitute if required field values are missing.
- Exchange archive path: Mailbox archives or secondary mailboxes have their own folder hierarchy. This means it is not possible to distinguish their folder structure from that of the primary mailbox. If you want to use entry points, this parameter value is used in the configuration of action trees as the virtual root of the secondary hierarchy.
- Trace: This option is used to write advanced EWS logs.
- Processing statistics: This option is used to record the processing statistics of an instance and store them in the ELO repository.
- Error messages: This list contains all the recipients who will be notified by e-mail if processing errors occur.

Action trees

Action trees are the entry points for processing data in ELO XC. They have an ordinal number, which means that ELO XC follows a predefined processing sequence when running in the default execution modes *once*, *interval*, and *fix*. They also have configuration properties that determine which message set is selected for processing.



1. To create an action tree, click the action tree icon in the instance.



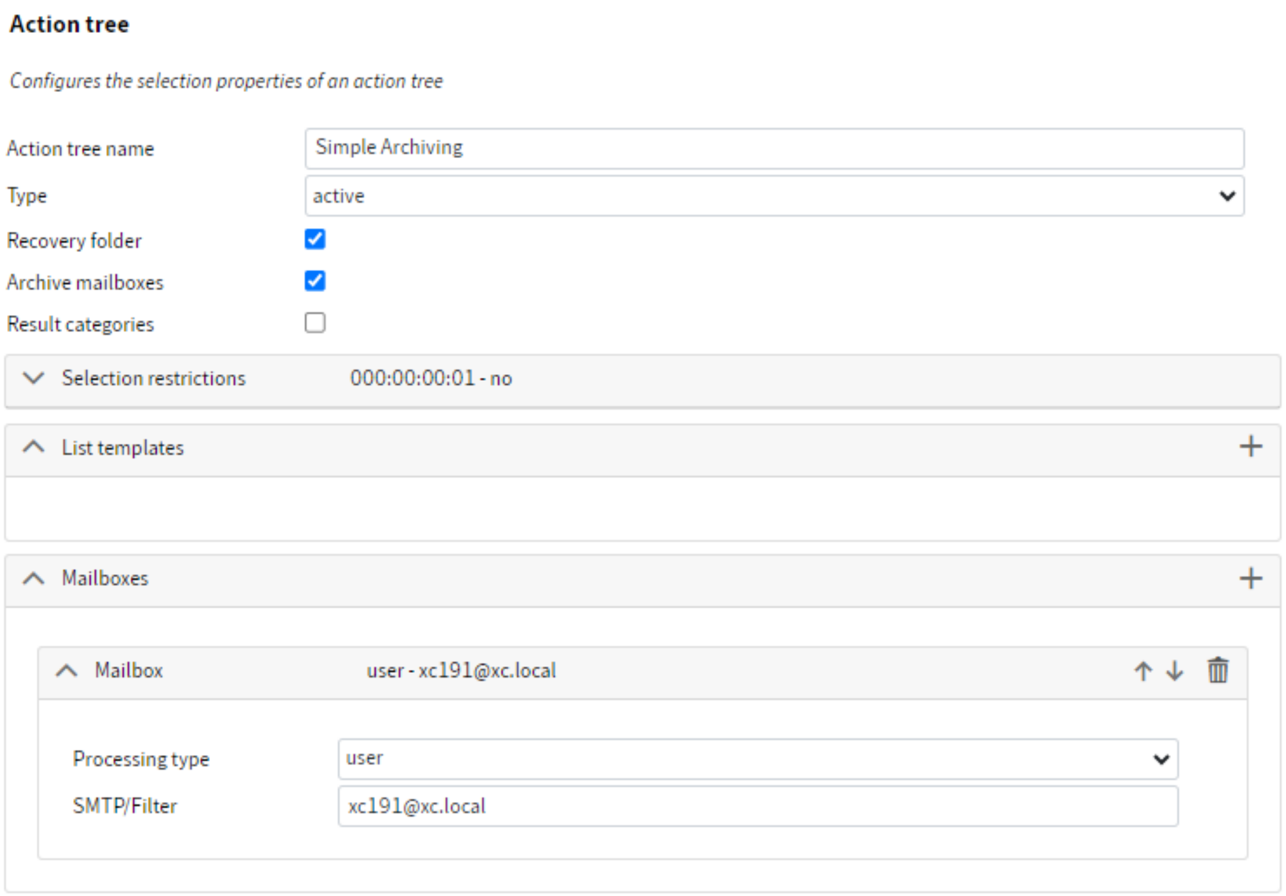
New action tree

An action tree must have a unique name. You can then specify the ordinal number for determining its position in the execution sequence of action trees using the function provided.

Name

The *New action tree* dialog box opens.

2. Select a name for the action tree and click the *Create* button.



Action tree

Configures the selection properties of an action tree

Action tree name	Simple Archiving
Type	active
Recovery folder	☑
Archive mailboxes	☑
Result categories	☐

Selection restrictions 000:00:00:01 - no

List templates

Mailboxes

Mailbox user - xc191@xc.local

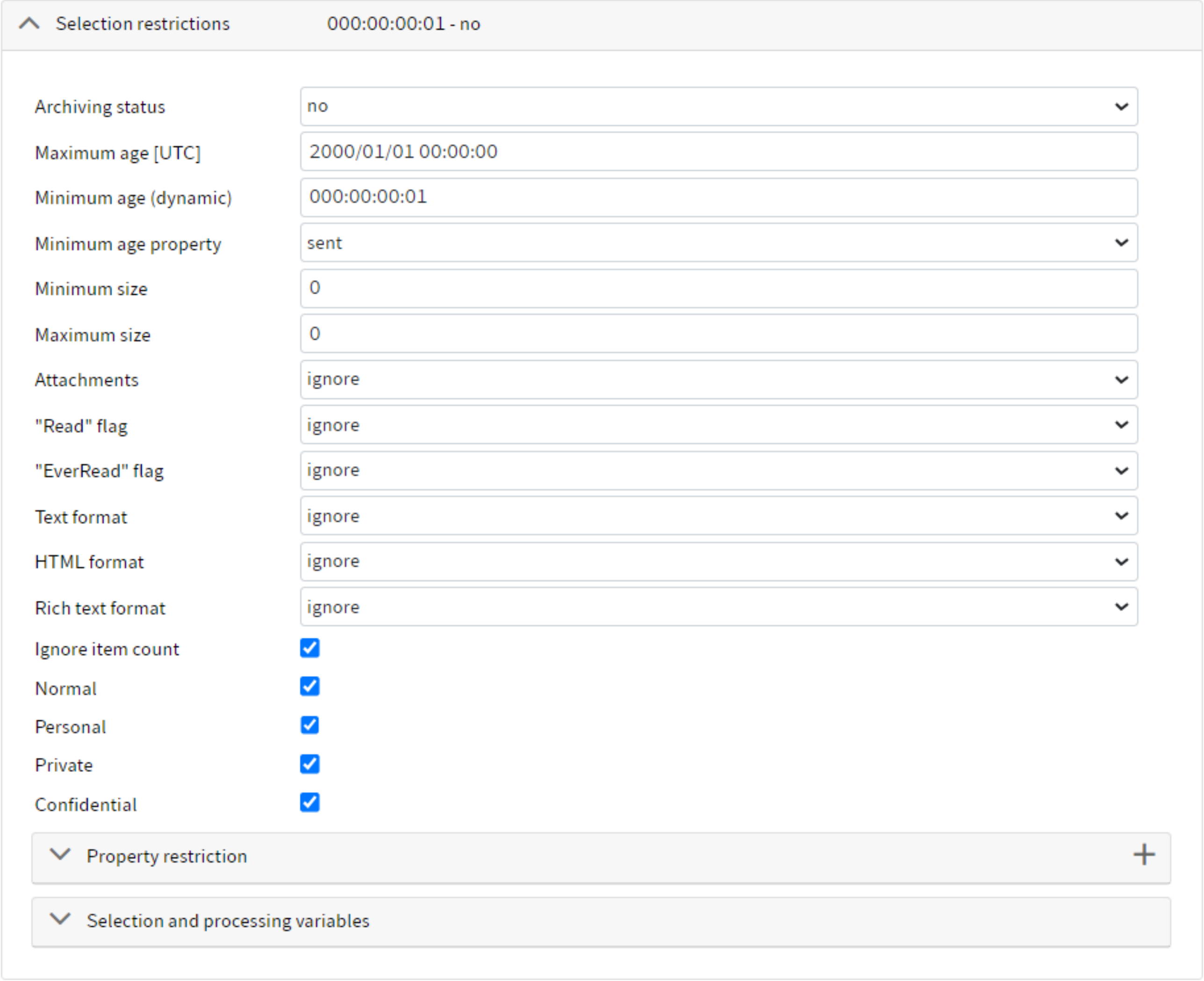
Processing type	user
SMTP/Filter	xc191@xc.local

The *inactive* action tree is inserted at the end of the configuration structure. The parameters of the action tree are displayed in the form view. You need to enable the type *active* so that the tree can be processed.

If you only want to create and enable a subtree using a *Call subtree* action, you can omit the parameter configuration and enter *subtree* under the type. A subtree automatically inherits the parameters of the action tree being called.

Selection restrictions

The selection restrictions configure the message search that an action tree performs for each permitted mailbox folder according to various criteria.



Selection restrictions 000:00:00:01 - no

Archiving status	no
Maximum age [UTC]	2000/01/01 00:00:00
Minimum age (dynamic)	000:00:00:01
Minimum age property	sent
Minimum size	0
Maximum size	0
Attachments	ignore
"Read" flag	ignore
"EverRead" flag	ignore
Text format	ignore
HTML format	ignore
Rich text format	ignore
Ignore item count	☑
Normal	☑
Personal	☑
Private	☑
Confidential	☑

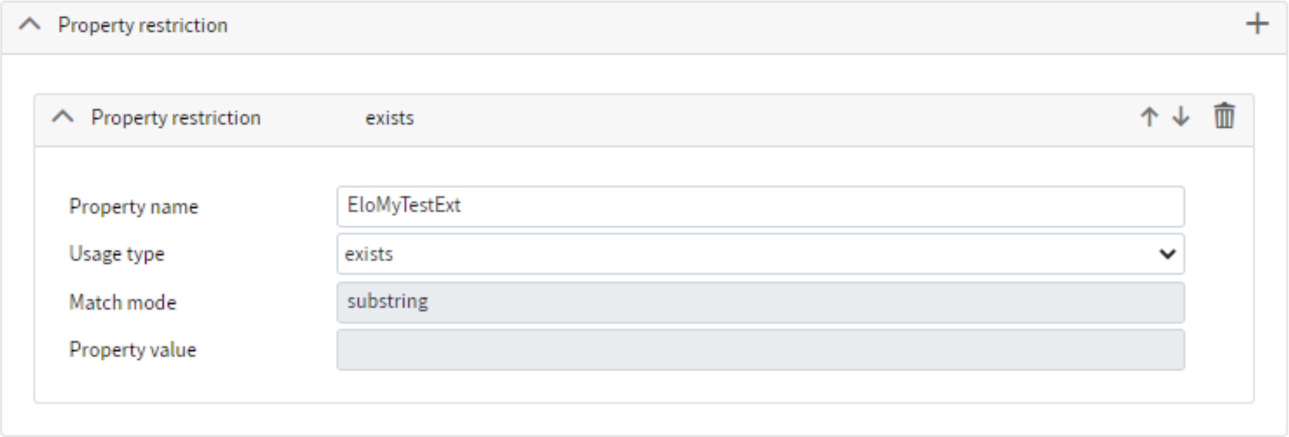
- Archiving status: This setting determines whether a message can already be marked as archived or not. If the archiving status is irrelevant, it can be ignored.
- Maximum age: This parameter is set to 01.01.2000 by default and determines the oldest timestamp of a message. This value is included in the selection of messages.
- Minimum age: Enter the minimum age in the format `ddd:hh:mm:ss`. This specifies the required age of a message in relation to the selected minimum age property. Newer messages are not selected.
- Minimum age property: The property usually used is *PidTagMessageDeliveryTime* (Option *sent*). Use *PidTagEndDate* (Option *enddate*) when processing calendar items. If a minimum age is required depending on the last change to an item, you can also configure *PidTagLastModificationTime* (Option *lastmodified*).
- Minimum size/Maximum size: These values are interpreted as the interval of the permissible item size. If you set the value `0` (zero), only interval values that are not `0` (zero) are checked.
- Attachments: The messages can have attachments (Option *ignore*), must have attachments (Option *yes*), or may not have attachments (Option *no*).
- Flag 'Read'/Flag 'EverRead': The flag *Read* determines whether the current message is marked as *read* or *unread*. *EverRead* indicates whether the message has ever been read,

regardless of the current view. Both flags can be ignored. The option *yes* indicates that the respective flag must be set, whereas the option *no* indicates that it must not be set.

- Body formats: *Text format*, *HTML format*, and *Rich text format* refer to the format of the message body. If you want to process a message independently of this setting, we recommend that you set *ignore* in all three values (default setting). In some cases, such as when stubbing messages, it can be useful to only select messages in HTML or text format. You could also exclude only *Rich text format* with the option *no*. However, if you archive whole messages and do not carry out any special actions that affect the message body, you can ignore these three selection restrictions.
- Ignore item count: With this option, ELO XC also processes folders that do not appear to contain any items. This can be useful when processing *public folders* where the item count is not synchronized.
- Sensitivity: Message selection can be configured according to sensitivity with the options *Normal*, *Personal*, *Private*, and *Confidential*.

Property restriction

The list of property restrictions can also be used to include application-specific processing states in the selection. A processing state is usually set with the *Tag* action. Technically speaking, this is a *named property* that is set as an additional property for a processed item by another action tree. You can use property restrictions to select these properties later on. The *Tag* action always creates a property pair *Elo<tag name>Base* and *Elo<tag name>Ext* for a tag name. If you want to use the tag name as a property restriction, you have to use the whole internal name.

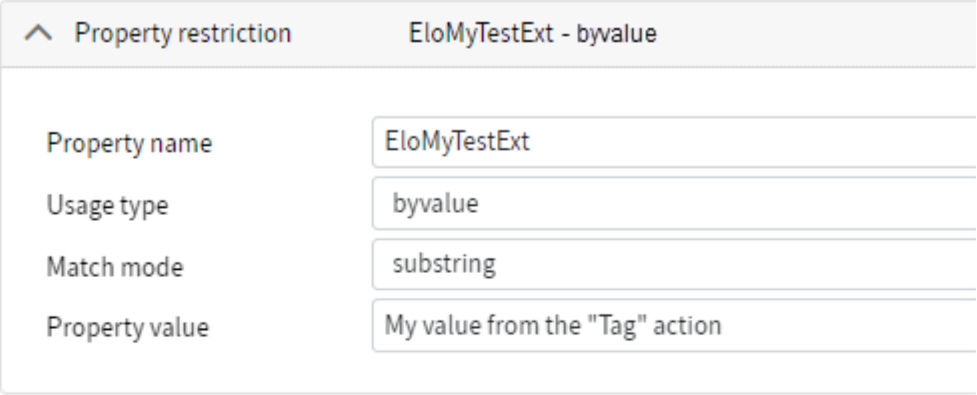


Property restriction

Property restriction exists

Property name	EloMyTestExt
Usage type	exists
Match mode	substring
Property value	

In our example, the name *MyTest* is configured in *Tag*. ELO XC creates the property pair *EloMyTestBase* and *EloMyTestExt*. If you want to configure a property restriction for a new action tree, you need to enter *EloMyTestBase* or *EloMyTestExt* as the property name. *Base* always contains the processing timestamp and *Ext* the configured value of the *Tag* action.



EloMyTestExt - byvalue

Property name	EloMyTestExt
Usage type	byvalue
Match mode	substring
Property value	My value from the "Tag" action

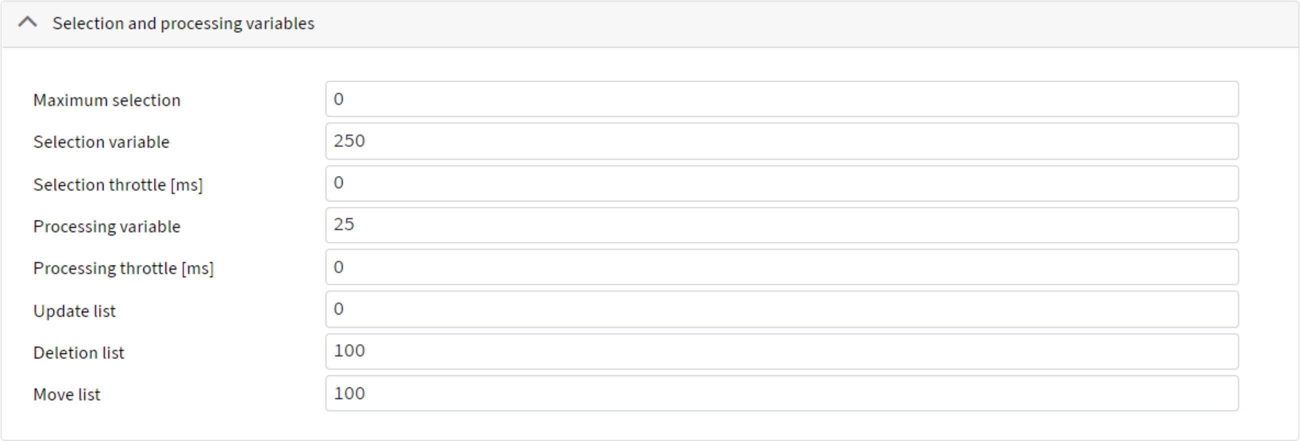
If the usage type *exists* (default value) is configured, Exchange server only checks whether for example *EloMyTestExt* exists (the option *missing* also exists as well as *exists*). If, on the other hand, you also want to include the value of the property in the selection, you must select *byvalue*. The match mode controls other aspects of the string comparison.

Selection restrictions by Outlook category are made with the separate property name *Keywords*. This property is not one of the Exchange properties (*PidTags*) but a list of category names managed by Outlook.

Selection and processing variables

The selection and processing variables help to optimize the processing of an action tree, which takes place in three steps:

1. Selection: The program searches for and selects messages.
2. Processing: The actions of the tree are executed for each message.
3. Completion: The message is passed to the change, deletion, or move list.



Maximum selection	0
Selection variable	250
Selection throttle [ms]	0
Processing variable	25
Processing throttle [ms]	0
Update list	0
Deletion list	100
Move list	100

- Maximum selection: This option determines the maximum number of selectable items per mailbox. `0` removes the restriction.
-

Selection variable and Selection throttle [ms]: The selection variable determines the maximum number of hits for each Exchange search. The selection throttle reduces the frequency of search requests.

- Processing variable and Processing throttle [ms]: The processing variable determines the number of messages that a worker can load from the Exchange server in a single request. The processing throttle reduces the frequency of these requests.
- Update list: The update list determines how many internally cached items there are to update (Exchange server only). The items are updated when this maximum is reached or ELO XC has finished processing a folder. Attachments are deleted immediately.
- Deletion list: The deletion list determines how many internally cached items there are to delete. The items are deleted when this maximum is reached or ELO XC has finished processing a folder.
- Move list: The move list determines how many internally cached items there are to move. The items are moved when this maximum is reached or ELO XC has finished processing a folder.

List templates

Under *Templates*, you can create list templates for use in all action trees. Under *List templates* in an action tree, you can reference the list templates by name. There are templates for mailboxes, entry points, folder filters, and message classes.

Mailboxes

This area contains all mailboxes that the action tree will process.

This list is taken into account in addition to possible references under the list templates. Duplicate mailbox configurations under list templates and mailboxes are automatically resolved. If an address was configured with different mailbox types, the mailboxes are sorted according to the internally assigned processing type.

For more information on configuring different mailbox types, refer to the Basics > Exchange > Mailbox types chapter in the documentation.

Entry points

Entry points are configured as mailbox paths and designate mailbox folders that need to be entered during processing in order for messages to be selected. Mailbox folders that are not entered using an entry point, directly or after recursion, are not processed. As in the case of the list templates, duplicate configurations are automatically removed when the configuration is read. If you click the *Value* field, you get a list of available language-independent variables.

> **Please note**
>
> Variables for language-independent folders also depend on the selected global scaling options of the instance. Entry points for Exchange archives (secondary mailboxes) must use the appropriate global prefix of the instance configuration.

Folder filters

Folder filters are subject to the same purge logic as mailboxes or entry points. They contain folder names that, when evaluating the folder hierarchy of a mailbox, ensure that positive matches lead to exclusion of folders.

Message classes

Message classes represent the item type in Exchange. From an Exchange point of view, all items are messages that fall into a certain class. Messages can in fact be calendar entries or contacts, which is determined by the class property *PidTagMessageClass*, which has to be configured as an additional selection filter in this area. As with mailboxes, entry points and folder filters, the additional inclusion mechanisms of any possible list templates also apply.

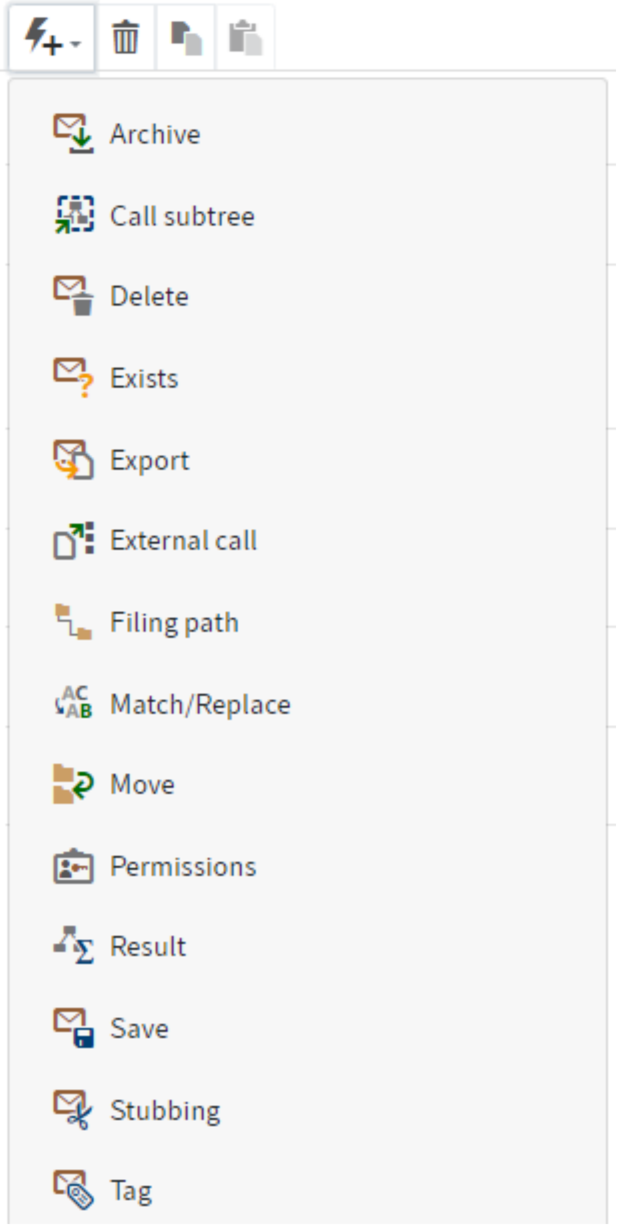
Actions

Actions are created for action trees, have their own action type, comparable to a function type, and can be assigned individual names. Whereas an action tree selects messages, the individual actions are responsible for processing. They are largely configured separately from one another. Exceptions or automatic dependencies are described in the Automated processes section. The overall arrangement of actions describes the purpose of an action tree. You will get examples of an action configuration when you create a new instance (see Instance overview > Register instance).

The result of processing by an action can be successful (*TRUE*) or unsuccessful (*FALSE*). Based on the result, you can configure follow-up actions, which again can be succeeded by follow-up actions depending on the result. This series of actions creates a binary tree structure.

Messages are processed based on the data loaded by Exchange (message properties). It starts with the first action and ends when there is no follow-up action. The last determined result of an action is automatically the processing result of the action tree for the respective message.

Before you create the first action, click an action tree in ELO XC Manager and then click the lightning icon.



A drop-down menu showing all available actions appears.

Meaning of actions

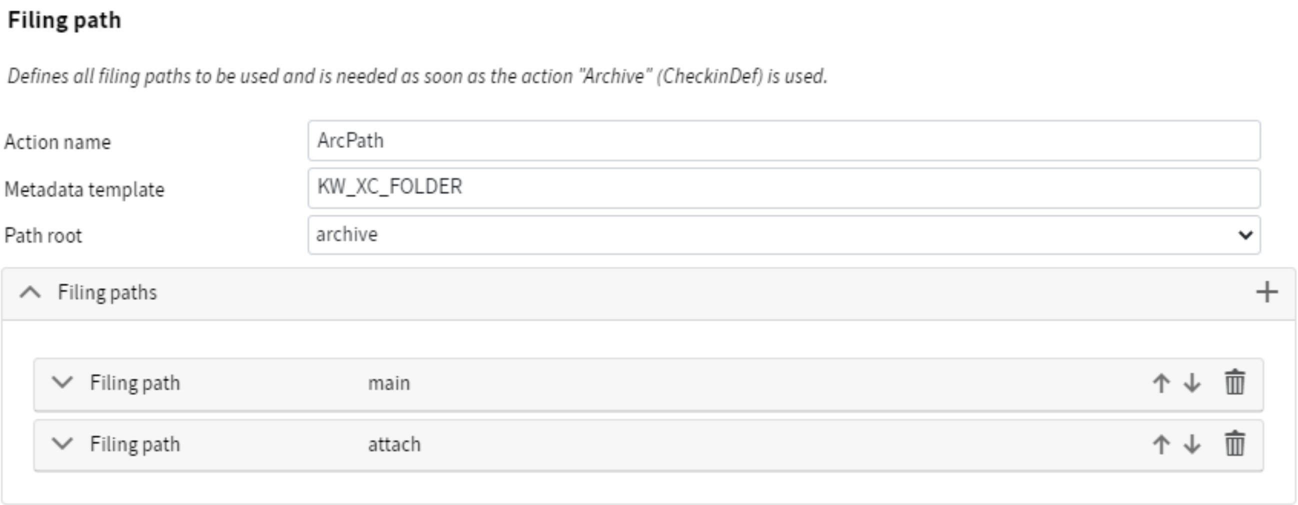
Action	**Schema**	**Description**
Filing path	*ArcPathDef*	Defines all the paths for storing messages and attachments
Archive	*CheckinDef*	Stores processed messages
Permissions	*ItemSecurityDef*	Defines the permissions for the messages being stored
Result	*ResultDef*	Specifies a constant for the result of the executing action tree
Exists	*ExistsDef*	Checks whether messages exist in the repository
Export	*ExportDef*	Determines the parts of a message
External call	*CallExternalDef*	Executes an external call

Action	Schema	Description
Delete	*DeleteDef*	Deletes messages or attachments in the mailbox
Tag	*TagDef*	Tags messages with user-defined properties or Outlook categories
Stubbing	*StubbingDef*	Stubs the body of processed messages
Save	*CommitDef*	Saves all changes to a message in the mailbox
Call subtree	*CallDef*	Calls another action tree that is configured as a subtree
Match/ Replace	*MatchReplaceDef*	Matches properties and can also replace property values
Move	*MoveDef*	Moves messages within the folder hierarchy of a mailbox

Filing path

A filing path to the repository must exist so that e-mails can be stored. A filing path consists of segments that are created in the repository as SORDs.

> **Please note**
>
> If a path or a path segment already exists, it is used repeatedly.



Filing path

Defines all filing paths to be used and is needed as soon as the action "Archive" (CheckinDef) is used.

Action name: ArcPath

Metadata template: KW_XC_FOLDER

Path root: archive

Filing paths

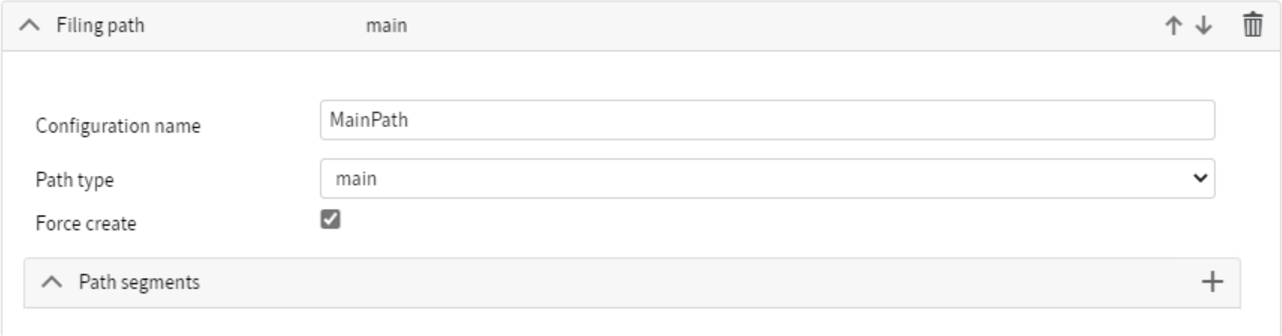
Filing path: main

Filing path: attach

The *Filing path* action allows you to configure three separate path types for the message (main document), for attachments, and for logical references.

- Metadata template: The metadata template is required for the inheritance of segment metadata. The path segments can override this parameter, which is valid for all segments, with their own parameters if needed.
- Path root: The default *archive* option creates the filing path globally and in relation to the repository. The *user* value requires user import to be enabled.

'Path segments' options



Filing path main

Configuration name: MainPath

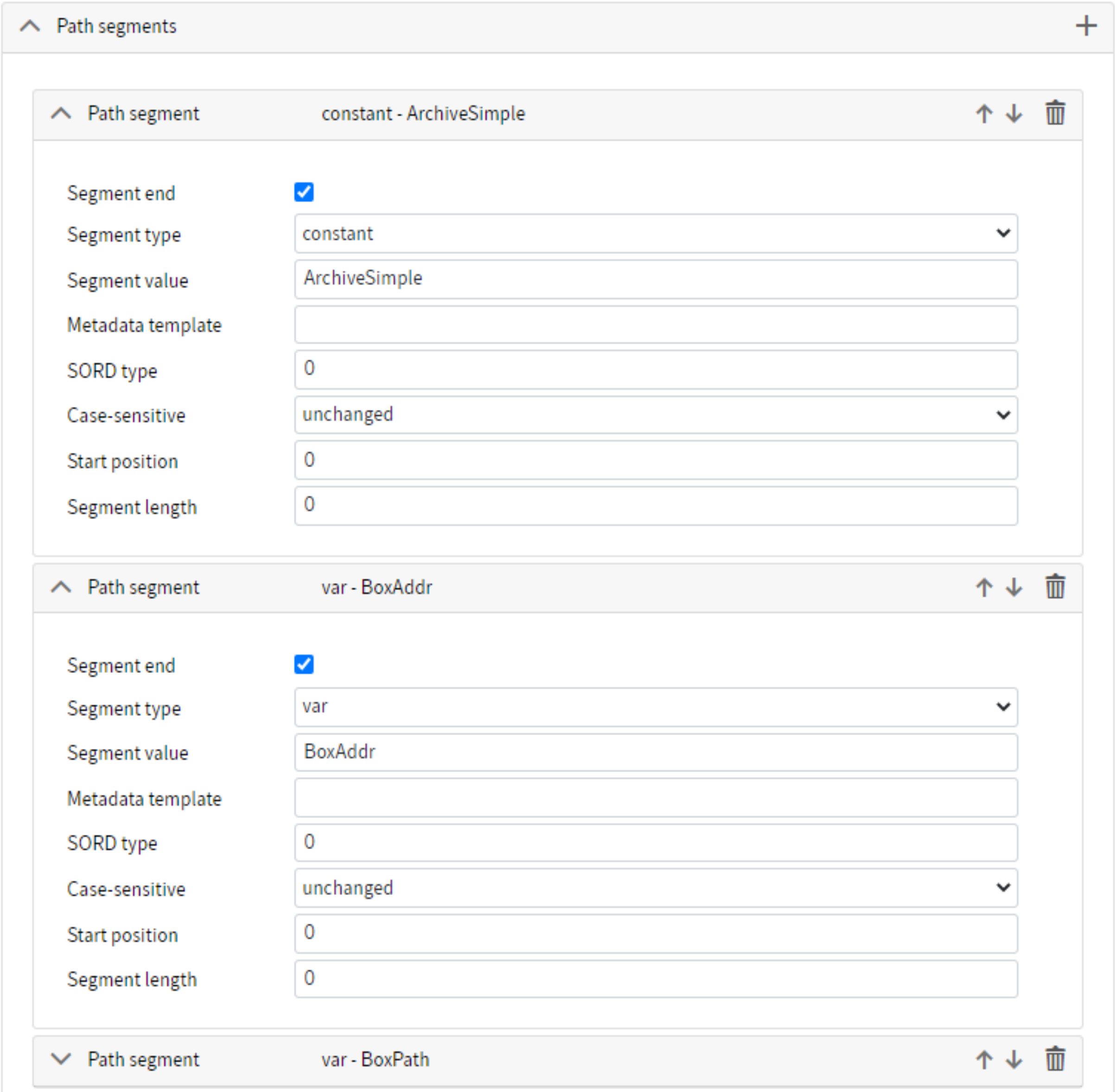
Path type: main

Force create: ☑

Path segments

Each filing path configures its own *path segments*.

- Unique: *Unique* means that the configured path must exist only once and should always be used. If this option is not enabled, the path must be identical except for the last segment. However, a new SORD is always created for the last segment. This is particularly useful if you want to break down messages and store each part in a separate folder.



Path segments

Path segment constant - ArchiveSimple

Segment end	☑
Segment type	constant
Segment value	ArchiveSimple
Metadata template	
SORD type	0
Case-sensitive	unchanged
Start position	0
Segment length	0

Path segment var - BoxAddr

Segment end	☑
Segment type	var
Segment value	BoxAddr
Metadata template	
SORD type	0
Case-sensitive	unchanged
Start position	0
Segment length	0

Path segment var - BoxPath

- Segment end: *Segment end* is enabled by default. This means that each configured path segment is a separate SORD. If you want a SORD to consist of multiple variables or properties, you must disable this option until the last path segment is configured for the SORD.

- Segment type, Segment value: *Segment type* determines how the segment value is to be used. Setting the *constant* type means that the segment value is transferred as text. The *var* type means that the *segment value* contains the name of a path variable. The *propname* and *cachedname* types set the segment value to a property name.

- Metadata template: This parameter can be used to override the action metadata template for the respective segment.

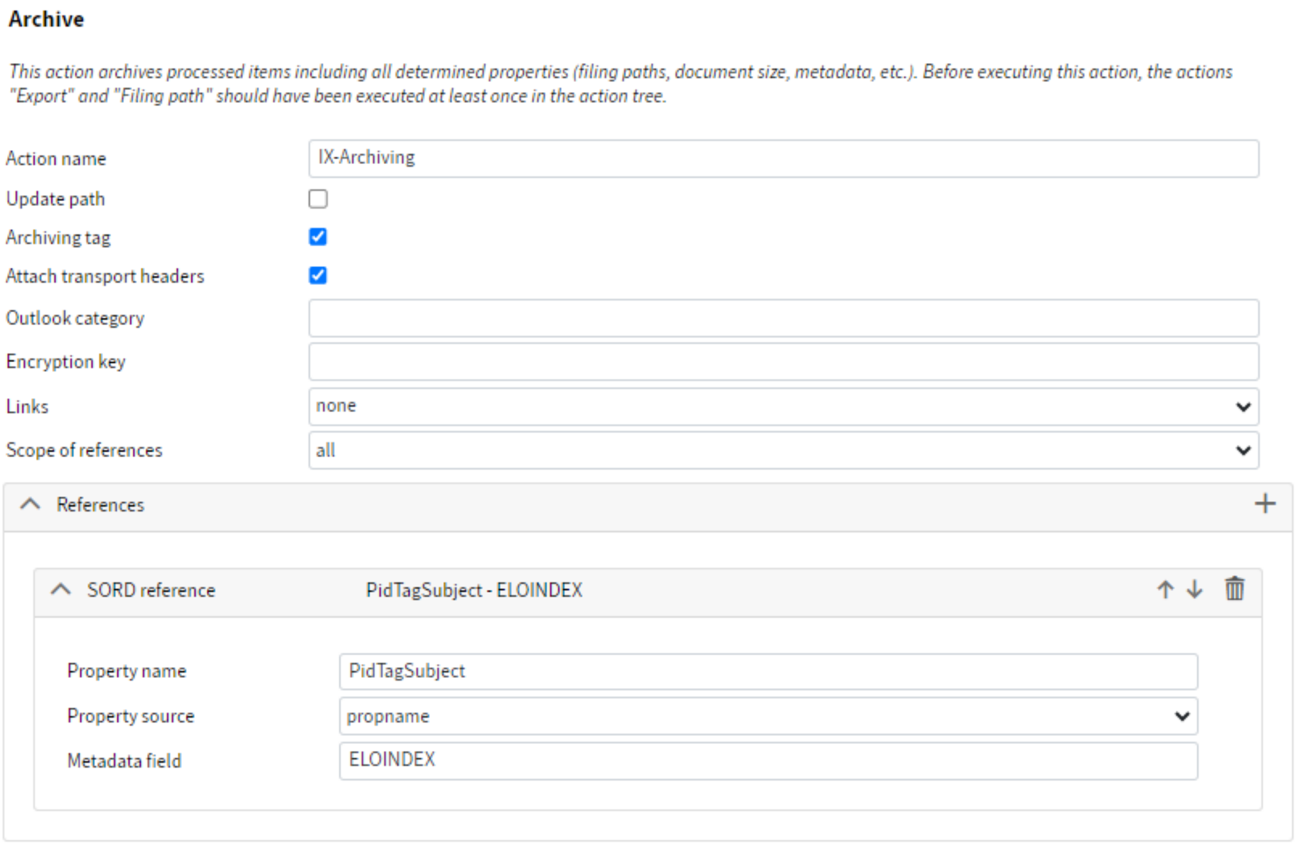
-

SORD type: With the default value `0`, the *SORD type* is set by the ELO Indexserver, which is the recommended setting. If you require an exception, values between 1 and 253 can be configured.

- Case-sensitive, Start position, Segment length: *Case-sensitive, Start position* and *Segment length* determine how the character string is subsequently processed.

Archive

The *Archive* action stores the data collected during the *Filing path and Export* actions in the repository.



Archive

This action archives processed items including all determined properties (filing paths, document size, metadata, etc.). Before executing this action, the actions "Export" and "Filing path" should have been executed at least once in the action tree.

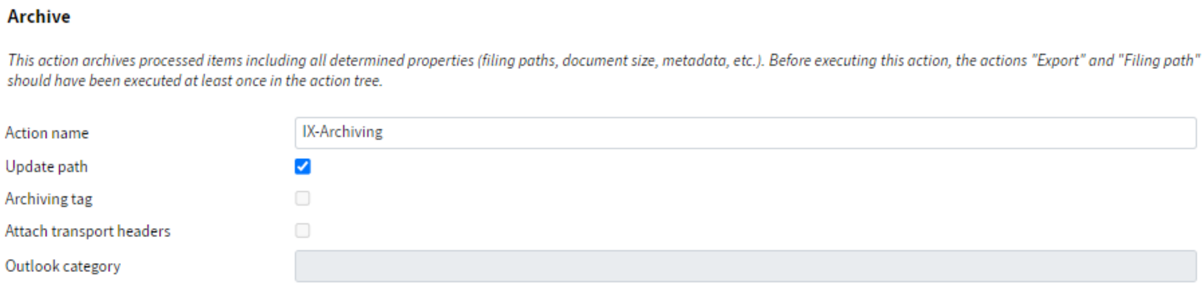
Action name	IX-Archiving
Update path	□
Archiving tag	☑
Attach transport headers	☑
Outlook category	
Encryption key	
Links	none
Scope of references	all

References

SORD reference PidTagSubject - ELOINDEX

Property name	PidTagSubject
Property source	propname
Metadata field	ELOINDEX

- Update path: If the *Update path* option is enabled, the action can also change the filing paths of archived messages. All other parameters are ignored in that case.



Archive

This action archives processed items including all determined properties (filing paths, document size, metadata, etc.). Before executing this action, the actions "Export" and "Filing path" should have been executed at least once in the action tree.

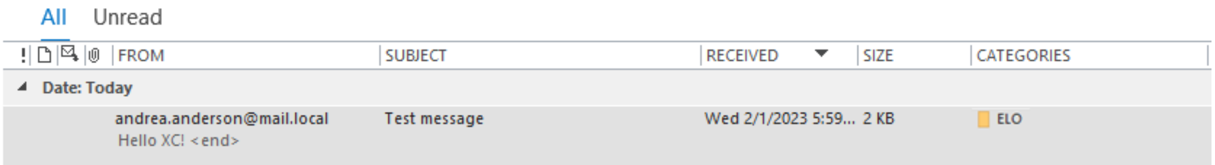
Action name	IX-Archiving
Update path	☑
Archiving tag	□
Attach transport headers	□
Outlook category	

The action searches for the message in the repository and updates the filing path according to the current configuration. This action mode only works if messages have already been stored.

- Archiving tag: The *Archiving tag* option means that the action generates an invisible property after filing, which is useful as selection criterion for action trees.

- Attach transport header: With this option, transport headers of the MIME file are stored as an ELO attachment.

- Outlook category: The *Outlook category* allows you to color-code a stored message in Outlook.



All Unread

!	FROM	SUBJECT	RECEIVED	SIZE	CATEGORIES
	Date: Today				
	andrea.anderson@mail.local Hello XC! <end>	Test message	Wed 2/1/2023 5:59...	2 KB	🟨 ELO

- Encryption key: The *Encryption key* is an optional parameter. It is recommended to set the encryption keys in the metadata form definition and to leave encryption to the ELO Indexserver. However, if the action uses its own encryption keys, you first have to configure these under the ELO settings in the instance configuration. You can only use references to encryption keys in the Archive action.

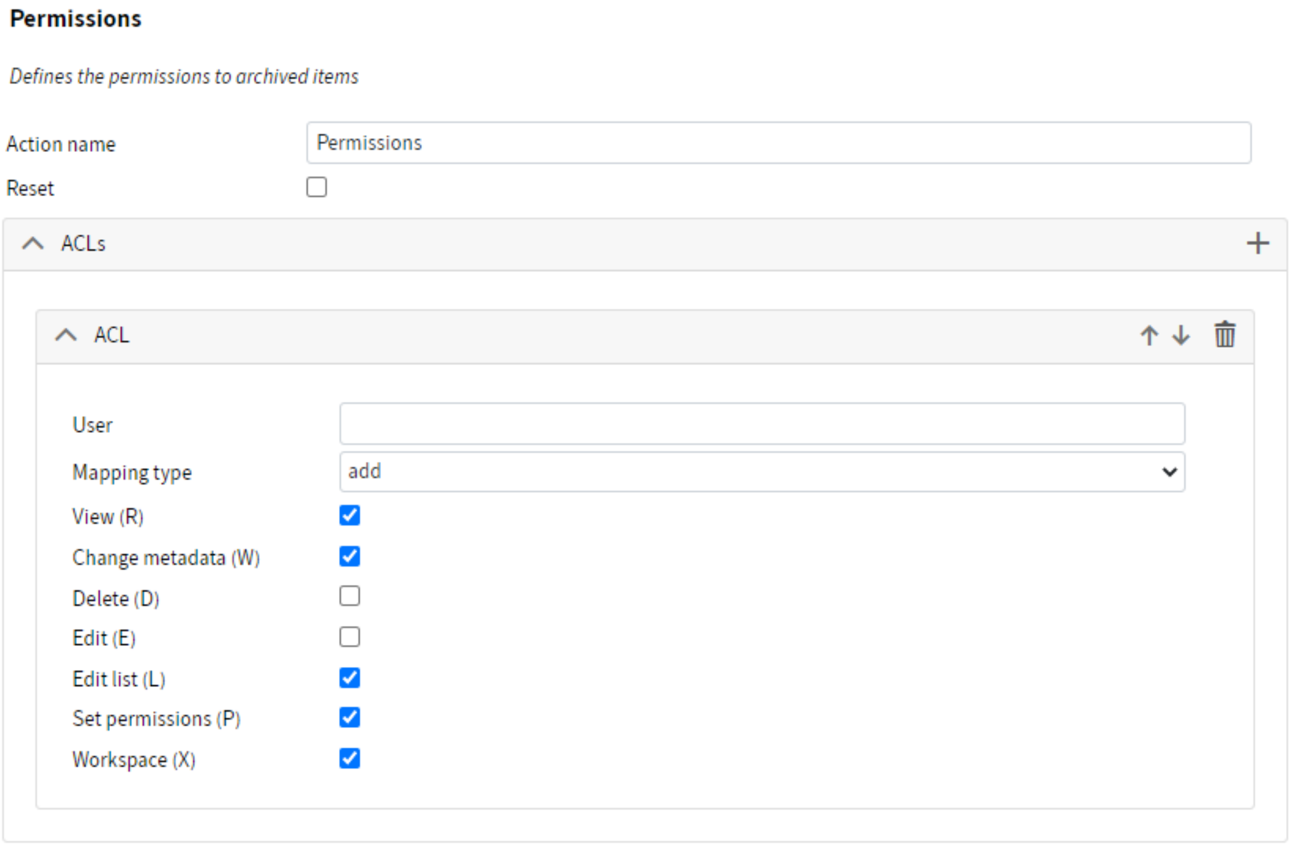
- Links: *Links* are used to link all stored parts of a message. If you don't want to link them, you need to set the default value *none*.

- Scope of references: The *scope of references* determines which message parts to logically reference. This requires a corresponding filing path of type *logref* and the configuration of at least one SORD reference in this action. A metadata search is used to locate the origin of the reference. If the configured value of the property name matches the value of the configured metadata field, a logical reference to the stored message parts is created based on the SORDs.

> **Information**
>
> You'll find more information about the metadata search under Configuration > Automated processes > Metadata search (gen. 1 and gen. 2).

Permissions

The ACLs (*permissions*) of SORDs are usually set using the metadata form definition to achieve standardized form-dependent results in a repository. If user import is enabled, the ACLs are also set for the ELO users of the mailbox owners. If both options are not sufficient, this action can be used to specifically adjust the permissions of stored messages.



Permissions

Defines the permissions to archived items

Action name	Permissions
Reset	☐

ACLs

ACL

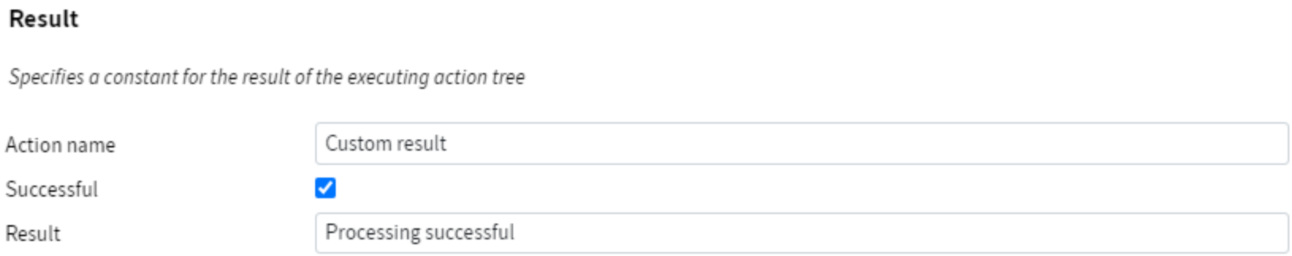
User	
Mapping type	add
View (R)	☑
Change metadata (W)	☑
Delete (D)	☐
Edit (E)	☐
Edit list (L)	☑
Set permissions (P)	☑
Workspace (X)	☑

When an action tree starts processing, there are no ACLs set. Every time the *Permissions* action is called within an action tree, the ACLs are set according to the selected configuration. The reset option enables you to remove ACL settings constructed in this way.

An ACL is always assigned to a user or group. The configured permission settings can be added (*add*) or removed (*remove*).

Result

The *Result* action configures the processing result of the action tree and immediately stops processing the action tree.



Result

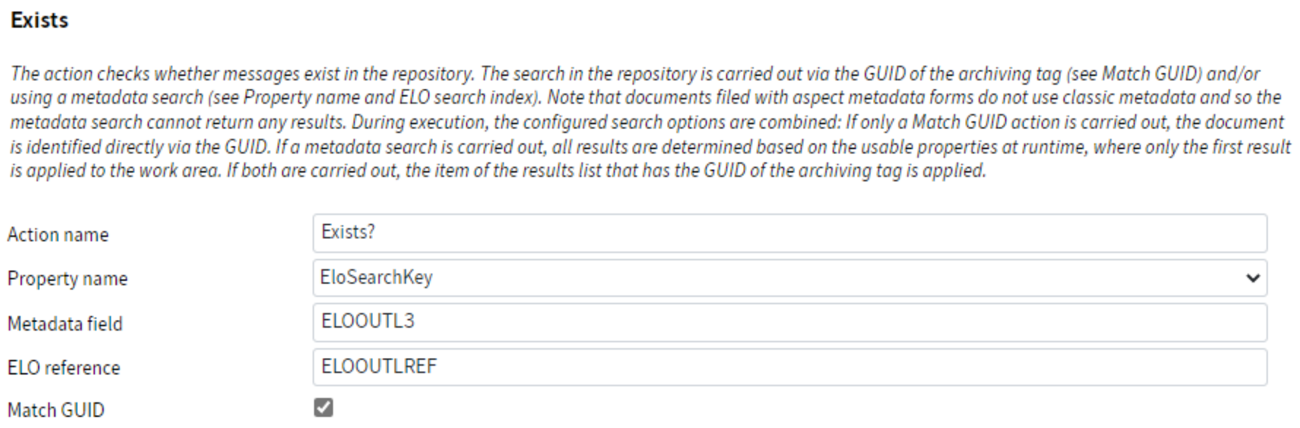
Specifies a constant for the result of the executing action tree

Action name	Custom result
Successful	☑
Result	Processing successful

It makes sense to use this action if you want to reverse the last action result or generate a specific result message for the log. The result value is particularly important when using result categories. In this case, the wording of the result value can be used to control which counter categories create the action trees of the entire instance configuration.

Exists

The *Exists* action performs two tasks: The first checks whether a message has already been filed to the connected repository. The second task reads the *EloGuid* value of the archived message and saves it to the workspace of the message.



Exists

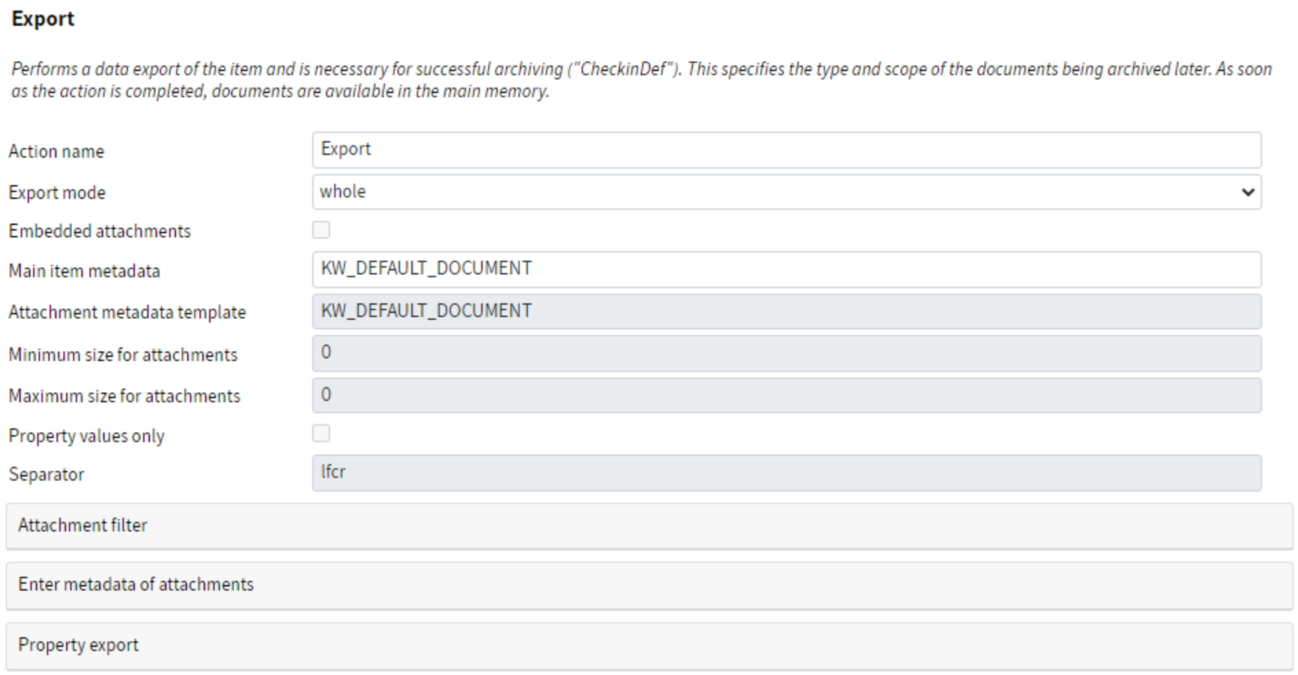
The action checks whether messages exist in the repository. The search in the repository is carried out via the GUID of the archiving tag (see Match GUID) and/or using a metadata search (see Property name and ELO search index). Note that documents filed with aspect metadata forms do not use classic metadata and so the metadata search cannot return any results. During execution, the configured search options are combined: If only a Match GUID action is carried out, the document is identified directly via the GUID. If a metadata search is carried out, all results are determined based on the usable properties at runtime, where only the first result is applied to the work area. If both are carried out, the item of the results list that has the GUID of the archiving tag is applied.

Action name	Exists?
Property name	EloSearchKey
Metadata field	ELOOUTL3
ELO reference	ELOOUTLREF
Match GUID	☑

> **Information**
>
> You'll find more information about the metadata search under Configuration > Automated processes > Metadata search (gen. 1 and gen. 2).

Export

The *Export* action determines which message parts and associated metadata templates to store. The message is always the main document/item. With the default setting, the entire message is exported in MIME format.



Export

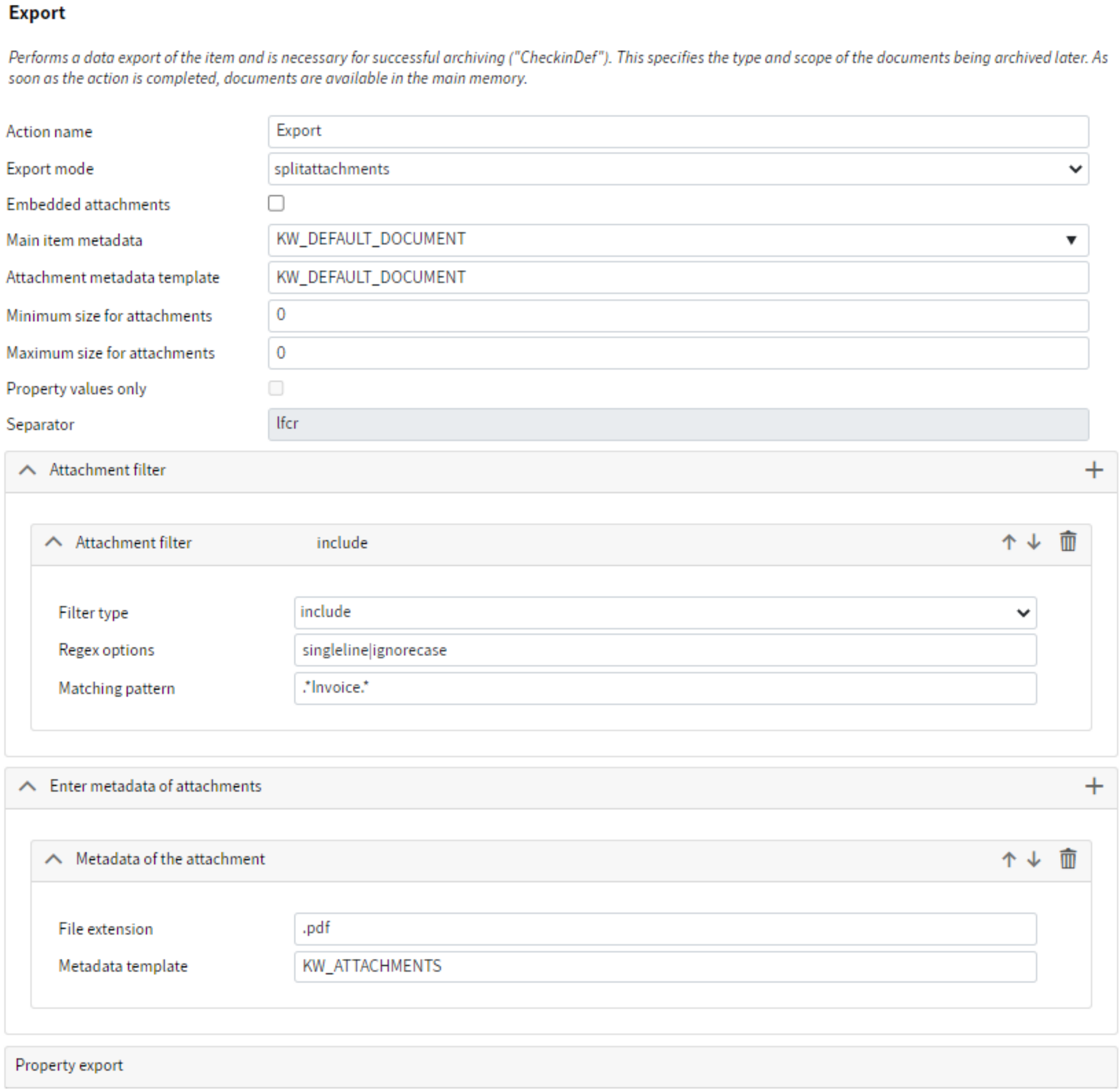
Performs a data export of the item and is necessary for successful archiving ("CheckinDef"). This specifies the type and scope of the documents being archived later. As soon as the action is completed, documents are available in the main memory.

Action name	Export
Export mode	whole
Embedded attachments	☐
Main item metadata	KW_DEFAULT_DOCUMENT
Attachment metadata template	KW_DEFAULT_DOCUMENT
Minimum size for attachments	0
Maximum size for attachments	0
Property values only	☐
Separator	lfcr

- Export mode: The *Export mode* settings determines how to export the message parts.
- whole (default): The whole message is exported.
- split: The message is split into parts.
- maindoconly: The message is stored without attachments.
- attachmentsonly: Only the attachments are stored.
- splitattachments: The whole message and its attachments are stored.
- profile: The configured message properties are written to a text file.
- jsonfile: The configured message properties are written to a text file in JSON format.
- Inline attachments: Inline attachments can optionally be exported when separating file attachments.
- Main item metadata: The name of the metadata template used for the main item.
- Attachment metadata template: The attachment metadata template is used for all extracted attachments unless other templates are configured for specific file extensions (e.g. PDF files).
- Minimum/Maximum size for attachments: The minimum and maximum size of a file attachment can be used as an additional filter to recognize relevant data based on the expected size.
- Property values only, Separator: If *propfile* export mode is selected, item properties are stored as name-value pairs in a text file. With the default settings, the properties are output by line (*lfcr* separator). You can choose a different separator to separate the properties, and you can prevent properties from being output.

'Attachment filter' options

If attachments are separated, additional parameters can be configured.



Export

Performs a data export of the item and is necessary for successful archiving ("CheckinDef"). This specifies the type and scope of the documents being archived later. As soon as the action is completed, documents are available in the main memory.

Action name: Export

Export mode: splitattachments

Embedded attachments: □

Main item metadata: KW_DEFAULT_DOCUMENT

Attachment metadata template: KW_DEFAULT_DOCUMENT

Minimum size for attachments: 0

Maximum size for attachments: 0

Property values only: □

Separator: lfcr

Attachment filter

Attachment filter include

Filter type: include

Regex options: singleline|ignorecase

Matching pattern: .*Invoice.*

Enter metadata of attachments

Metadata of the attachment

File extension: .pdf

Metadata template: KW_ATTACHMENTS

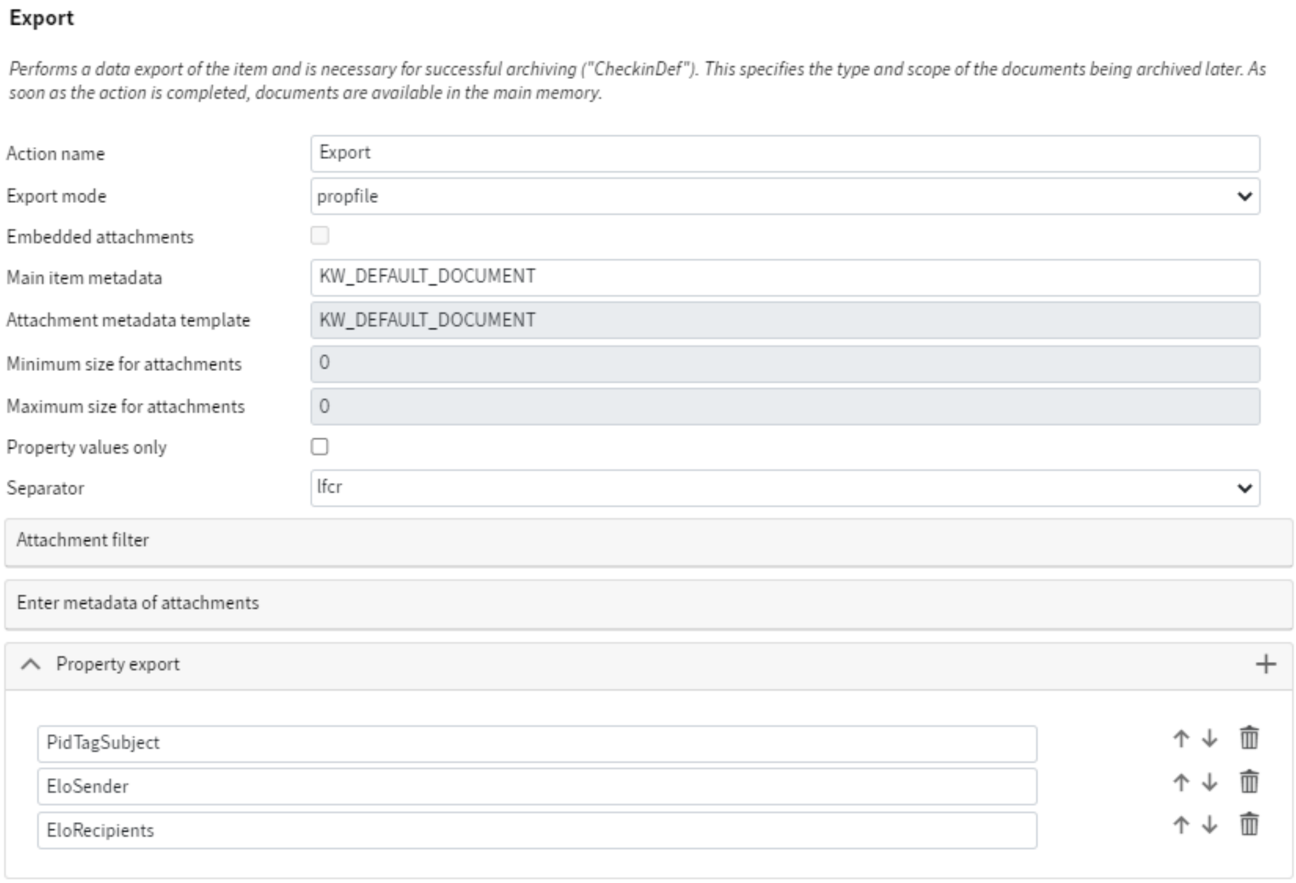
Property export

The list of filters for attachments allows you to perform additional regex checks on the names or file names.

Filter type: The *Filter type* parameter can be used to specify whether to include or exclude an attachment during export.

'Property export' options

Property exports are configured in *profile* or *jsonfile* mode.



Export

Performs a data export of the item and is necessary for successful archiving ("CheckinDef"). This specifies the type and scope of the documents being archived later. As soon as the action is completed, documents are available in the main memory.

Action name	Export
Export mode	propfile
Embedded attachments	□
Main item metadata	KW_DEFAULT_DOCUMENT
Attachment metadata template	KW_DEFAULT_DOCUMENT
Minimum size for attachments	0
Maximum size for attachments	0
Property values only	□
Separator	lfcr

Attachment filter

Enter metadata of attachments

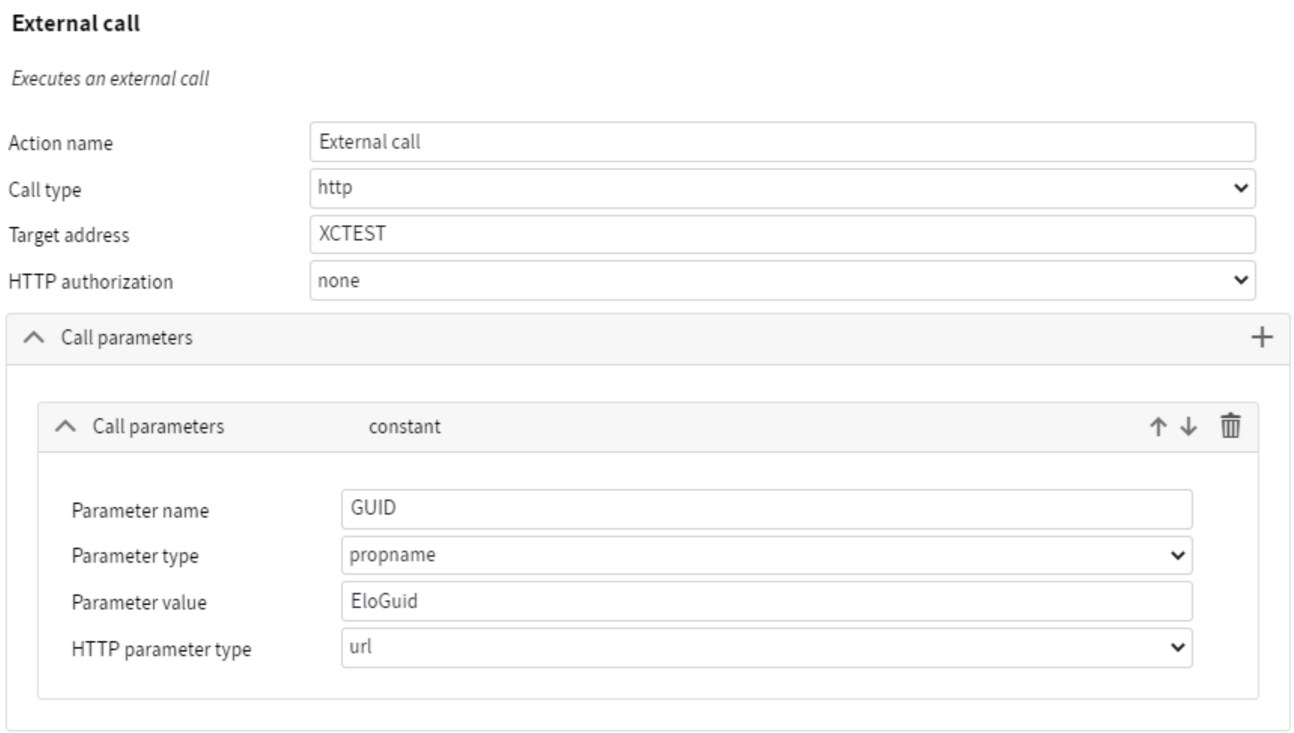
Property export

- PidTagSubject
- EloSender
- EloRecipients

With the *profile* setting, name-value pairs of the *Property export* list are written to a text file. The configured separator is used to separate the name from the value. You also have the option not to write the property names. If you set *jsonfile*, the configured properties including their name are written to the export file in JSON format.

External call

The *External call* action allows you to call additional interfaces.



External call

Executes an external call

Action name	External call
Call type	http
Target address	XCTEST
HTTP authorization	none

Call parameters

Call parameters constant

Parameter name	GUID
Parameter type	propname
Parameter value	EloGuid
HTTP parameter type	url

The call type and target address determine the endpoint of the interface being called. The list of call parameters is passed to the interface.

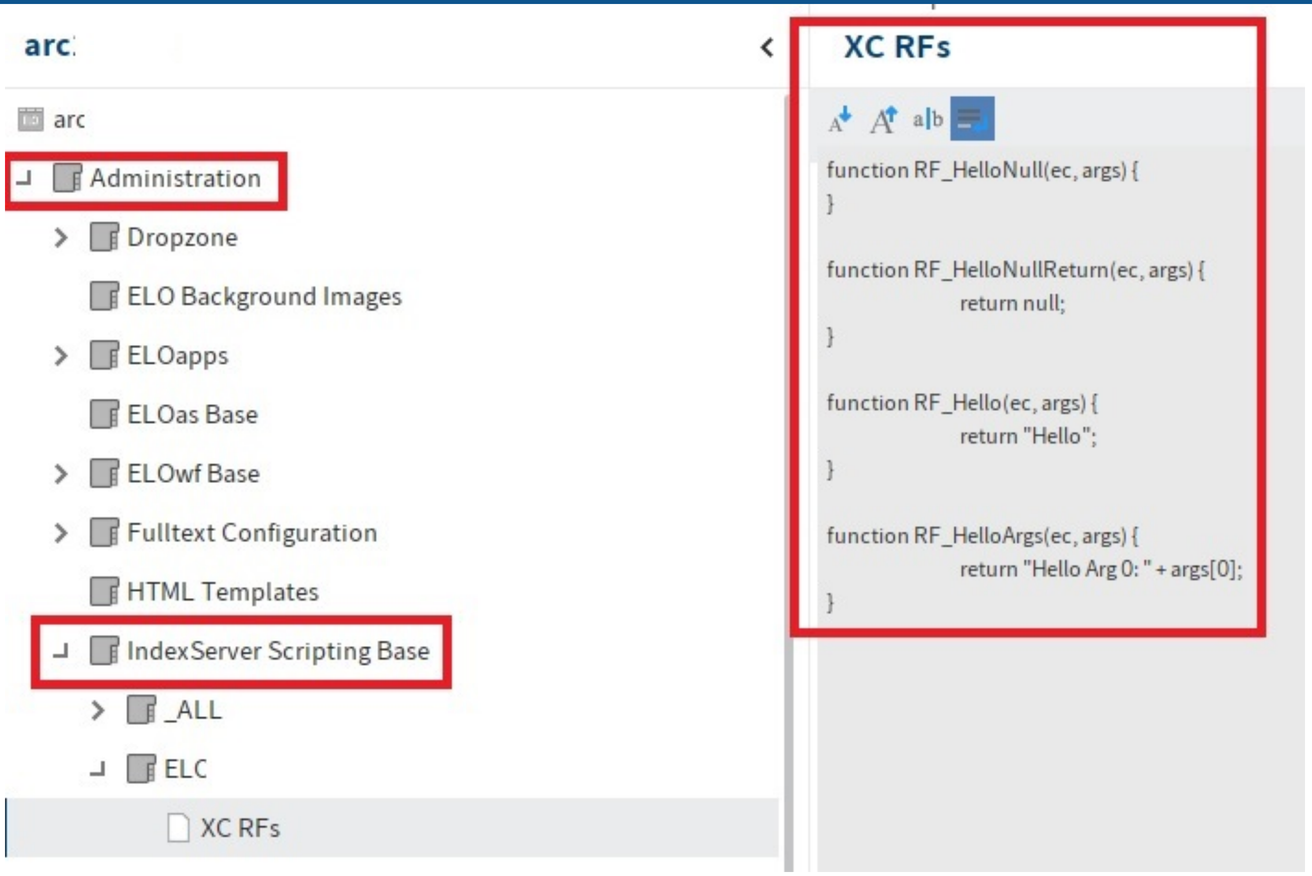
Call type	Target address
http	URL of web interface
ix	Name of the *registered function*
wf	Name of workflow template
feed	Feed action

If you select *http* as the type, you can use a HTTP parameter type to define how a parameter is passed. HTTP authorization allows you to specify whether to authenticate using the ELO Indexserver or the JSession cookie. Both options are needed for browser access using the ELO Indexserver (e.g. for plug-ins).

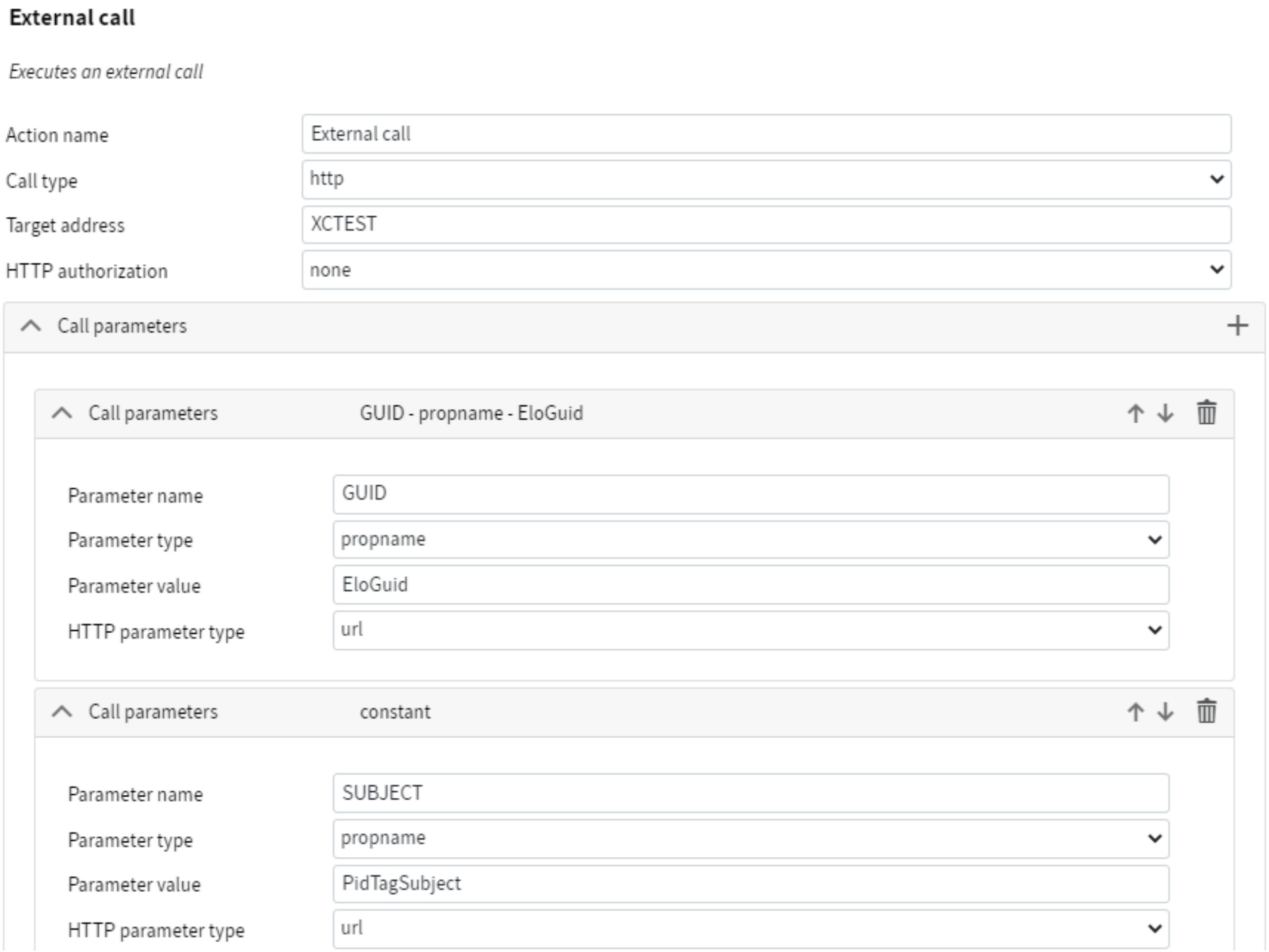
Configuration example for a registered IX function

An *external call* (*CallExternalDef*) from registered IX functions resulted in errors if returns were missing. This change prevents the error. Return values are also logged. The following examples illustrate the principle of registered IX functions.

> **Information**
>
> These examples are only a short demonstration and do not include a parameter check in Javascript.



```
function RF_HelloNull(ec, args) {
}

function RF_HelloNullReturn(ec, args) {
        return null;
}

function RF_Hello(ec, args) {
        return "Hello";
}

function RF_HelloArgs(ec, args) {
        return "Hello Arg 0: " + args[0];
}
```



External call

Executes an external call

Action name	External call
Call type	http
Target address	XCTEST
HTTP authorization	none

Call parameters

Call parameters GUID - propname - EloGuid

Parameter name	GUID
Parameter type	propname
Parameter value	EloGuid
HTTP parameter type	url

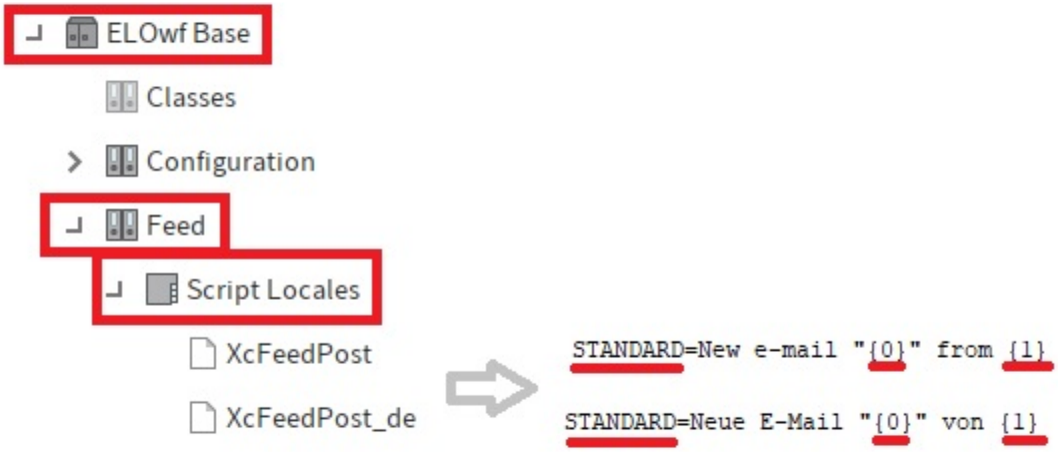
Call parameters constant

Parameter name	SUBJECT
Parameter type	propname
Parameter value	PidTagSubject
HTTP parameter type	url

The name of the registered function has to be used as the target address. The parameters are always passed in the order of the configured call parameters. On transfer/call, a parameter field contains the value determined at runtime based on the configuration. The example configures the call for the first example function. In the worker log, you will therefore find the result `null`. If XC calls `RF_Hello`, `Hello` appears in the log. The example `RF_HelloArgs` should show the SORD GUID of the message filed previously in the log.

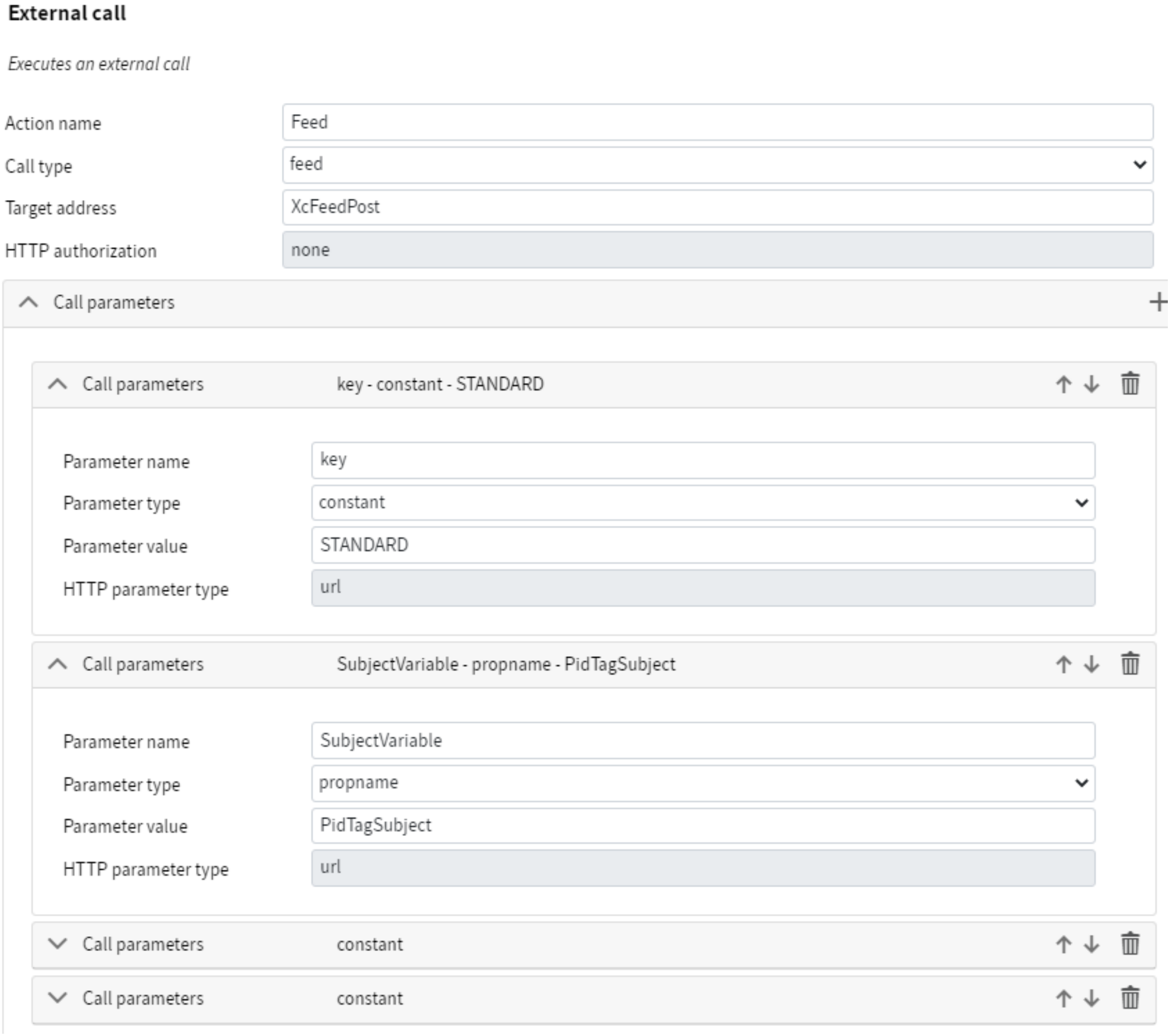
> **Information**
>
> Registered IX functions offer the most flexible and extensive options for completing supplementary tasks for XC processing and should therefore always be considered for follow-up processes in e-mail archiving. In such cases, long-running IX operations (e.g. search operations, volume-dependent/document-dependent evaluations, etc.) should be avoided. Durations in the IX calls of multiple seconds are already beyond the limit of what is considered sensible in bulk processing.

Fig.: Configuration example for a feed action



```
STANDARD=New e-mail "{0}" from {1}
STANDARD=Neue E-Mail "{0}" von {1}
```

The *external call* (*CallExternalDef*) for feeds does not allow you to pass ELO properties. This change extends *CallExternalDef* so that these properties can be used. The goal of this external call type is to configure a message as a feed action, for example to inform users of incoming messages. The properties files define the localized message text to be shown. The freely selectable variables/ parameters of the message are listed numbered in curly brackets.



External call

Executes an external call

Action name	Feed
Call type	feed
Target address	XcFeedPost
HTTP authorization	none

Call parameters

Call parameters key - constant - STANDARD

Parameter name	key
Parameter type	constant
Parameter value	STANDARD
HTTP parameter type	url

Call parameters SubjectVariable - propname - PidTagSubject

Parameter name	SubjectVariable
Parameter type	propname
Parameter value	PidTagSubject
HTTP parameter type	url

Call parameters constant

Call parameters constant

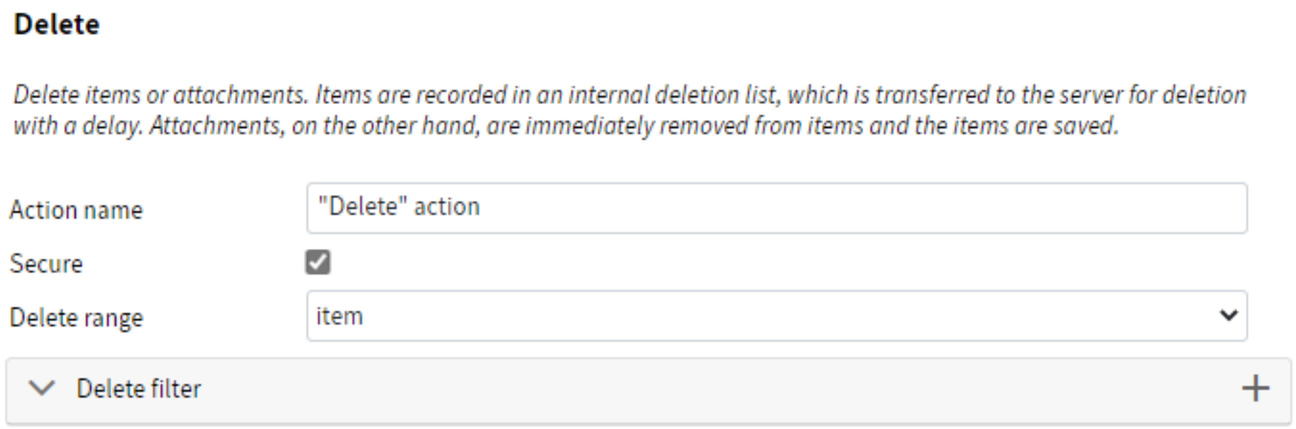
In the figure above, you can see how the feed action can be configured in ELO XC.

The name of the properties file is the call target. The text key named `key` has to be entered as the first call parameter (both are mandatory). The other call parameters are determined at runtime and transferred in the order they are configured to generate the feed action. The order of the call parameters should correspond to the numbering of the text variables.

Delete

The *Delete* action is used to delete messages from mailboxes.

> **Please note**
>
> Because the *Delete* action takes effect immediately, no other follow-up actions may be configured with the exception of the *Result* action.



Delete

Delete items or attachments. Items are recorded in an internal deletion list, which is transferred to the server for deletion with a delay. Attachments, on the other hand, are immediately removed from items and the items are saved.

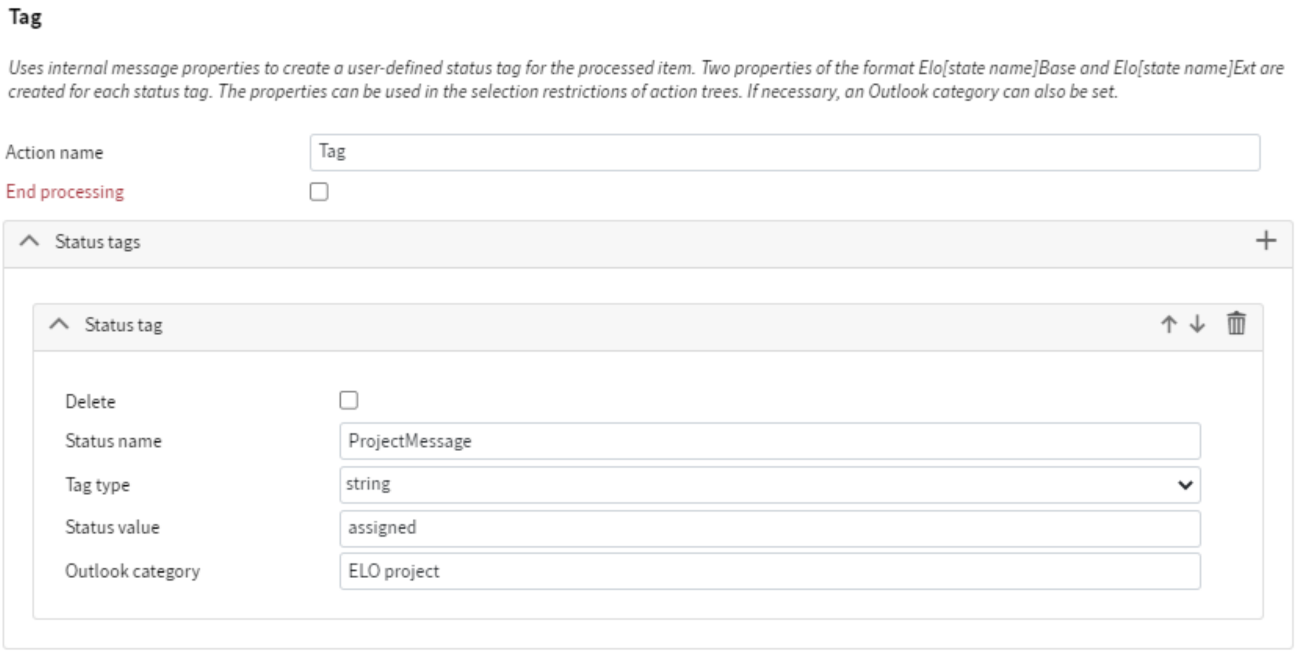
Action name	"Delete" action
Secure	☑
Delete range	item

Secured: The *Secured* parameter specifies whether the message must exist in the repository before deletion. This is checked using the archiving tag and requires the message to have been stored in the repository.

Delete range: The delete range specifies whether to delete the entire message or just the attachments. The list of deletion filters allows you to perform additional optional checks. If *item* is selected, the program performs a regex match on the subject line. If *attachment* is selected, the attachment name is checked.

Tag

The *Tag* action provides different options to tag messages.



Tag

Uses internal message properties to create a user-defined status tag for the processed item. Two properties of the format Elo[state name]Base and Elo[state name]Ext are created for each status tag. The properties can be used in the selection restrictions of action trees. If necessary, an Outlook category can also be set.

Action name	Tag
End processing	□

Delete	□
Status name	ProjectMessage
Tag type	string
Status value	assigned
Outlook category	ELO project

Please note

End processing: If this check box is enabled, ELO XC is instructed to never select or process the message again. This can be a useful option to set if you want to reduce the selection scope of action trees. However, you should consider whether you really want to do this.

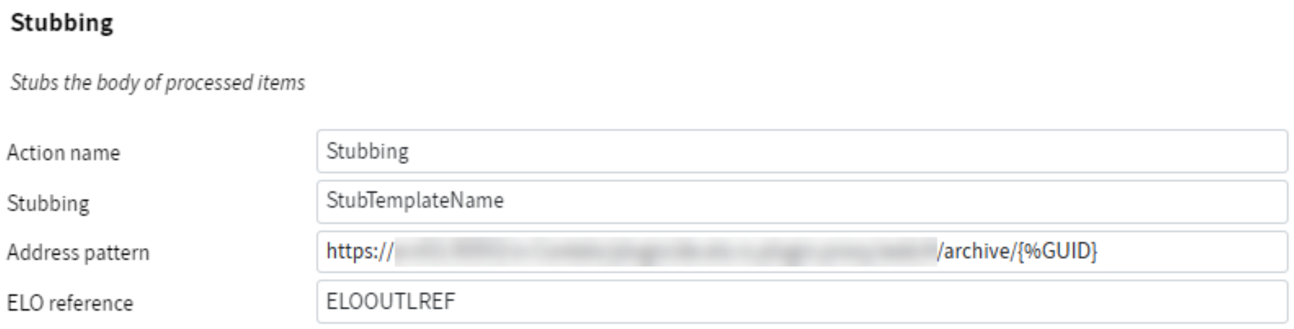
You set the tags in the Status tags area.

- Delete: You also have the option to remove existing tags.
- Status name, Status value: A tag consists of a *status name* and a *status value*. The *status name* is converted into an internal format during processing.

  In the example, the property names *EloProjectMessageBase* and *EloProjectMessageExt* are generated from the status name *ProjectMessage*. The base property is a timestamp, and the *Ext* property is *assigned* the value.
- Tag type: If you select the *string* tag type, the tag is a constant property value. If you select the *prop* tag type, the value must be a property name. In this case, the action inherits the property as the status value. The tag property value is a copy of another property.
- Outlook category: You can also tag the e-mail with a specific *Outlook category*.

Stubbing

The *Stubbing* action replaces message bodies according to a specific template.



Stubbing

Stubs the body of processed items

Action name	Stubbing
Stubbing	StubTemplateName
Address pattern	https://[redacted]/archive/{%GUID}
ELO reference	ELOOUTLREF

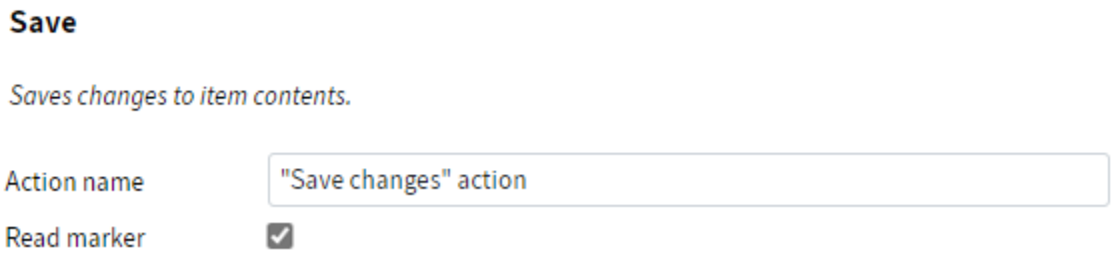
- Stubbing: *Stubbing* is the name of the *stubbing template* (see Templates > Stubbing templates.
- Address pattern: The *Address pattern* field is required for generating links to archived message parts. For this to work, the message parts must have been stored in the repository and their GUIDs must be available (see the Archive and Exists sections).
- Reference field: The *reference field* contains the metadata name that can be used to determine extracted attachments with their GUIDs.

Information

You'll find more information about the metadata search under Configuration > Automated processes > Metadata search (gen. 1 and gen. 2).

Save

Changes to a message do not take effect until the *Save* action is executed. This does not apply to changes to the *Delete* and *Move* actions.



Save

Saves changes to item contents.

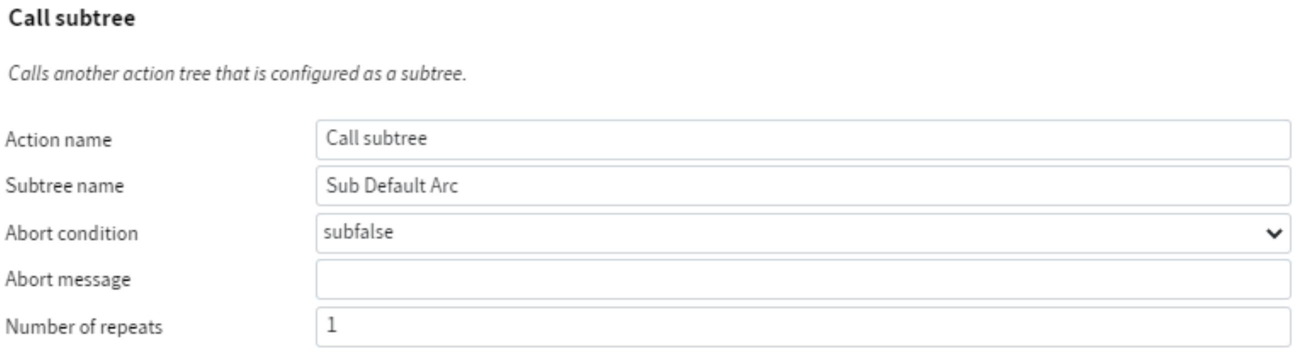
Action name: "Save changes" action

Read marker ☑

- Read marker: The *read marker* sets the bit in *PidTagMessageFlags* that displays the Outlook message as read.

Call subtree

The *Subtree call* action delegates execution to another action tree that is configured with the *subtree* type.



Call subtree

Calls another action tree that is configured as a subtree.

Action name: Call subtree

Subtree name: Sub Default Arc

Abort condition: subfalse

Abort message:

Number of repeats: 1

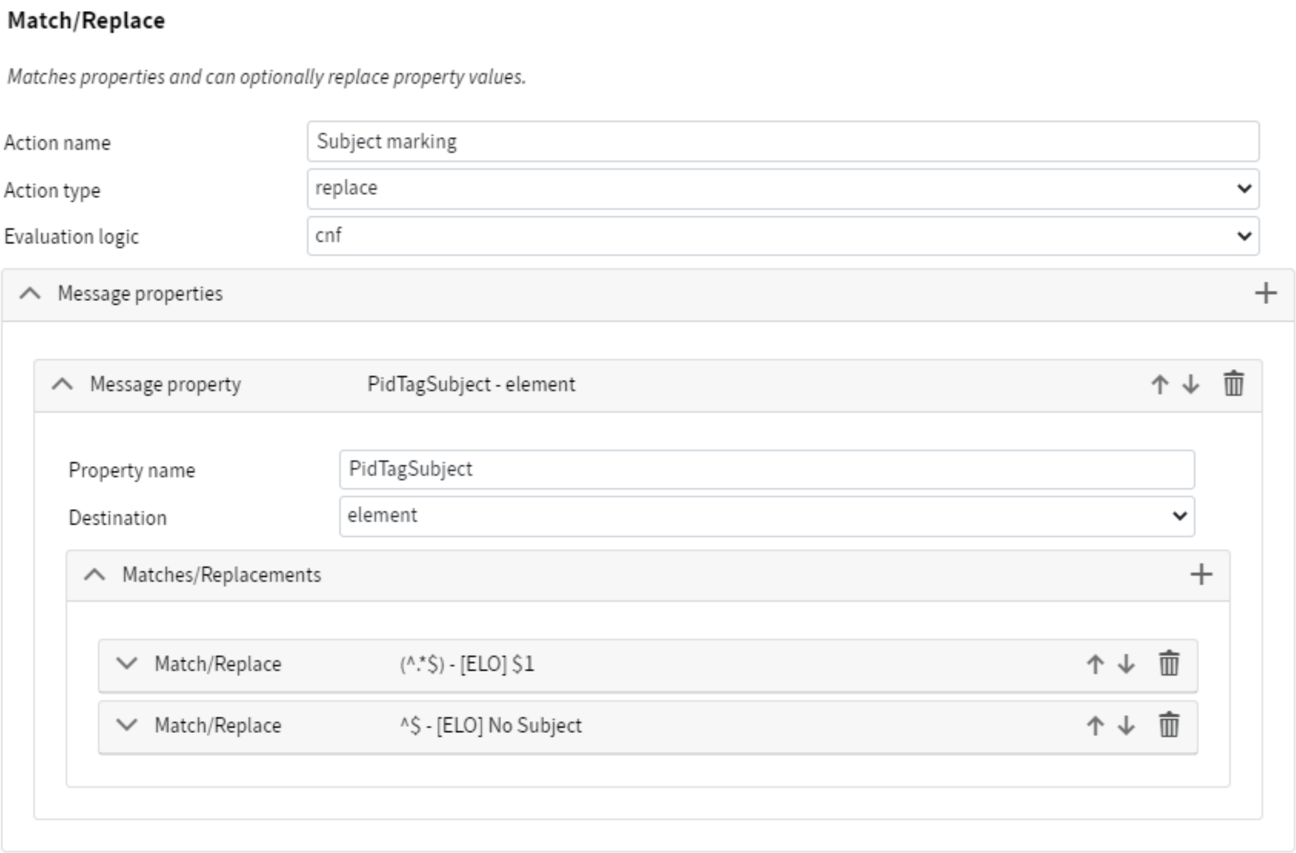
- Subtree name: ELO XC identities the subtree being called by name. If the name you enter does not exist, validation of the instance configuration will fail.

During processing, the action goes straight from the subtree call to the first action in the subtree. Message selection does not occur with subtree calls. In this case, the respective parameters are ignored but not deleted.

- Abort condition: The *Abort condition* determines the subtree result (*subtrue*) or (*subfalse*) which causes the loop to stop processing.
- Abort message: A loop can also be terminated if it matches a configured *abort message*.
- Number of retries: *Number of retries* determines how many times to call the subtree when the action is executed. This usually happens once but it can be useful to make loop calls in some cases. The maximum number of retries is 10.

Match/Replace

The *Match/Replace* action allows you to read out and change message properties. This action can be used to make decisions, configures branches, and change properties in an action tree.



Match/Replace

Matches properties and can optionally replace property values.

Action name: Subject marking

Action type: replace

Evaluation logic: cnf

Message properties

Message property — PidTagSubject - element

Property name: PidTagSubject

Destination: element

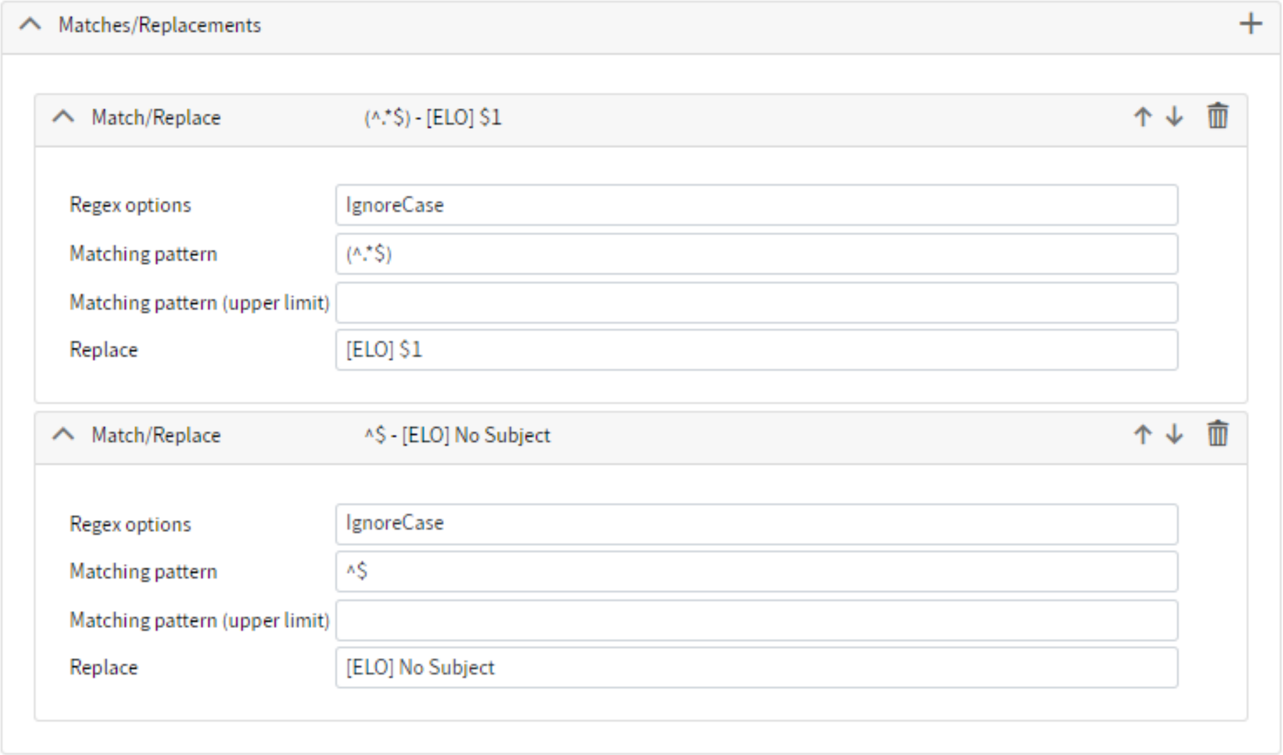
Matches/Replacements

Match/Replace — (^.*$) - [ELO] $1

Match/Replace — ^$ - [ELO] No Subject

- Action type: The action type determines whether to only read (*match*) or to replace (*replace*) configured properties. The action configures a list of message properties, which it then checks and/or replaces using a list of regex configurations.
- Evaluation logic: If the evaluation logic *cnf* (default) is selected, the action is successful if there is at least one successful regex configuration for all properties. If you select *dnf*, the action is successful if all regex configurations are successful for at least one message property.

  The action terminates as soon as the required condition is not met. In addition, all property values used are stored and restored if the entire action fails.
- Property value: The allowed property names are generally *PidTags*. ELO properties are only allowed with the *match* type as they are only available for read access.
- Destination: With the action type *replace* and the destination *element* (default setting), all changes to properties occur in the message itself. If *cache* is selected as the destination, ELO XC writes the changed value to an internal cache, which is available during processing. This allows you to change properties without changing the message. These values can also be accessed with the source type cachedname (see Automated processes > *Properties*).



Matches/Replacements

Match/Replace (^.*$) - [ELO] $1

Regex options: IgnoreCase

Matching pattern: (^.*$)

Matching pattern (upper limit):

Replace: [ELO] $1

Match/Replace ^$ - [ELO] No Subject

Regex options: IgnoreCase

Matching pattern: ^$

Matching pattern (upper limit):

Replace: [ELO] No Subject

- Regex options: The configuration of a match/replacement is bound to the internal options for regular expressions.
- Matching pattern, Matching pattern (upper limit): The matching pattern determines the regex match condition. If you configure an interval check, the property value must be within the lower limit and the upper limit of the *matching pattern*. This option can be used for calendar, time, and number properties.
- Replace The *Replace* parameter for the *replace* type determines the regex expression that is used to replace the property value if the pattern was successfully matched.

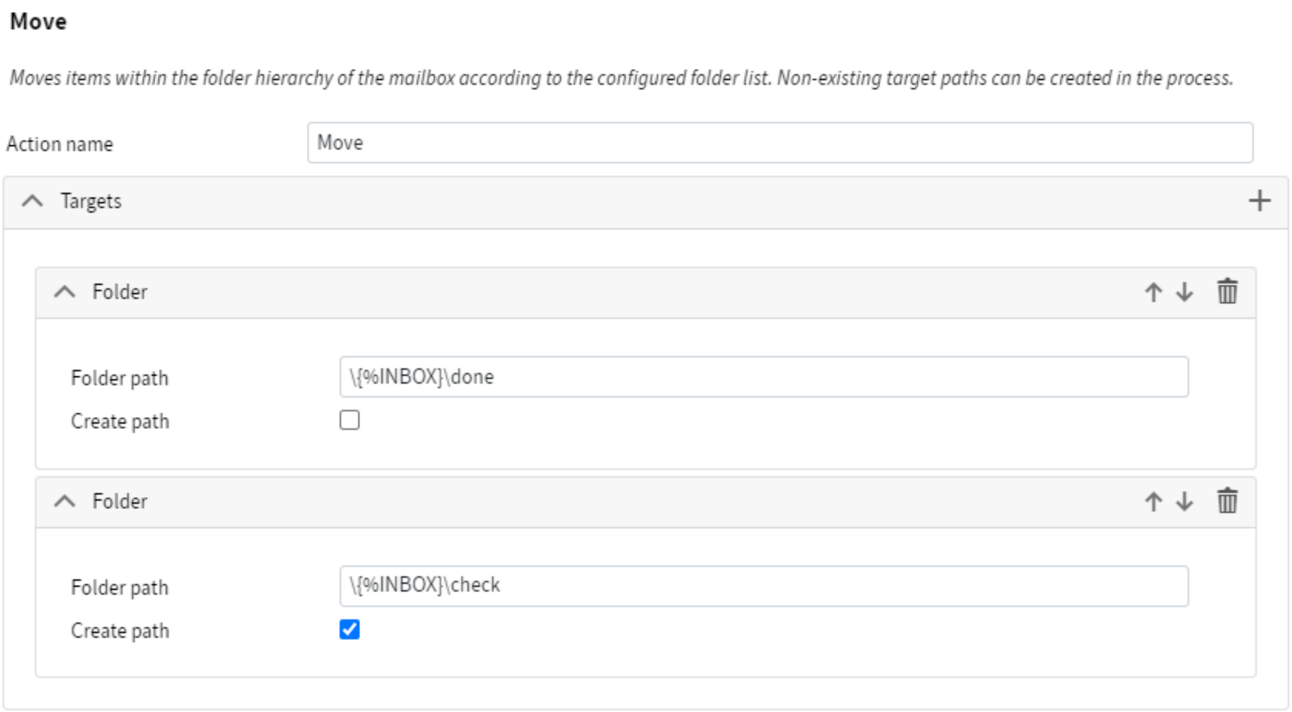
The calendar formats allowed to configure the lower and upper limits of the interval check are (`0`stands for valid numbers of the corresponding date part):

- 00000000000000 (14 digits): This is the ISO date format in the order year, month, day, hour, minute, and second. The year has four digits and the other parts have two digits. The interval limits are interpreted statistically.
- Age000000000 (9 digits): This format triggers and interval check of the age in relation to the job start(difference between job start and matching pattern). The order of the digits is day, hour, minute, and second. The days have three digits and the other parts have two digits. The *Age* prefix is case-sensitive.
- Map0000000000 (10 digits): This format requires the *Map* prefix. This option is case-sensitive. The digits represent month, day, hour, minute, and second. All parts have two digits. Besides numbers, XX are allowed instead of valid date parts. If part of the date in the pattern contains numbers, they are used in the interval threshold. If a part of the pattern contains XX, the numbers in the part of the date are copied from the property value.

Move

The *Move* action can be used to move messages to other mailbox folders.

> **Please note**
>
> Because the *Move* action takes effect immediately, no other follow-up actions may be configured with the exception of the *Result* action.



Move

Moves items within the folder hierarchy of the mailbox according to the configured folder list. Non-existing target paths can be created in the process.

Action name: Move

Targets

Folder

Folder path: \{%INBOX}\done

Create path: □

Folder

Folder path: \{%INBOX}\check

Create path: ☑

The configuration of this action supports multiple target folders, which are selected in sequence during execution. If the action fails, the next mailbox folder is selected as the target until the message is successfully moved. This is often the case if you don't want ELO XC to create the path.

- Create path: If you select the *Create path* option, ELO XC creates the mailbox folder to be used as the target if it doesn't already exist.

> **Please note**
>
> Moving a message will render the data invalid. This is because the mailbox path is part of the message identifier. It is therefore necessary to save all changes before moving a message (see Save section). Once it has been moved, you should not configure any actions in the action tree other than the *Result* action because there is no more data linked to the message.

Templates

Usage

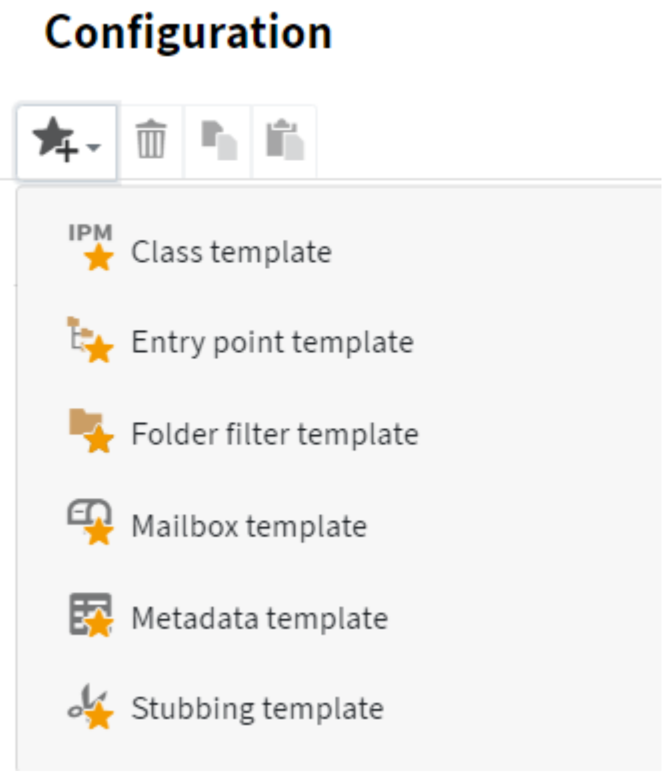
Templates allow you create configuration fragments that you can use again. They have unique names. Most template types are lists that you can refer to in the configuration of an action tree, which not only makes it easier for you to create the overall configuration, but can also be used to create processing instructions for the instance.

For example, if you only wanted process certain message classes, you could configure a template of the allowed classes. You could also use a folder filter template to exclude specific mailbox folders, regardless of whether they appear in the mailbox structure or not. You can create list templates for action trees for catalogs, entry points, folder filters, and classes.

The most important template is the one for importing metadata into the ELO repository. ELO XC therefore provides a default metadata template for stored messages and filing paths (see Metadata templates).

The configuration of a stubbing template is mandatory when using the associated action (*StubbingDef*). This defines the content that is created when the message is stubbed. It is necessary to distinguish between HTML and text format.

If you click the *Templates* node, you will find the templates that have already been created. You can create new templates by clicking the star icon and selecting the desired template type from the submenu.



A drop-down menu with available configuration templates opens.

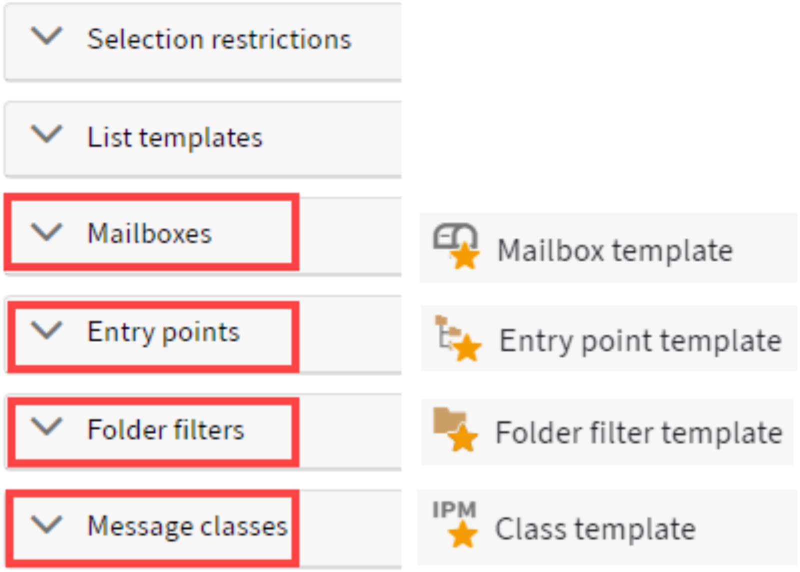
The following configuration templates are available:

- Entry point template
- Class template
- Metadata template
- Folder filter template
- Mailbox template
- Stubbing template

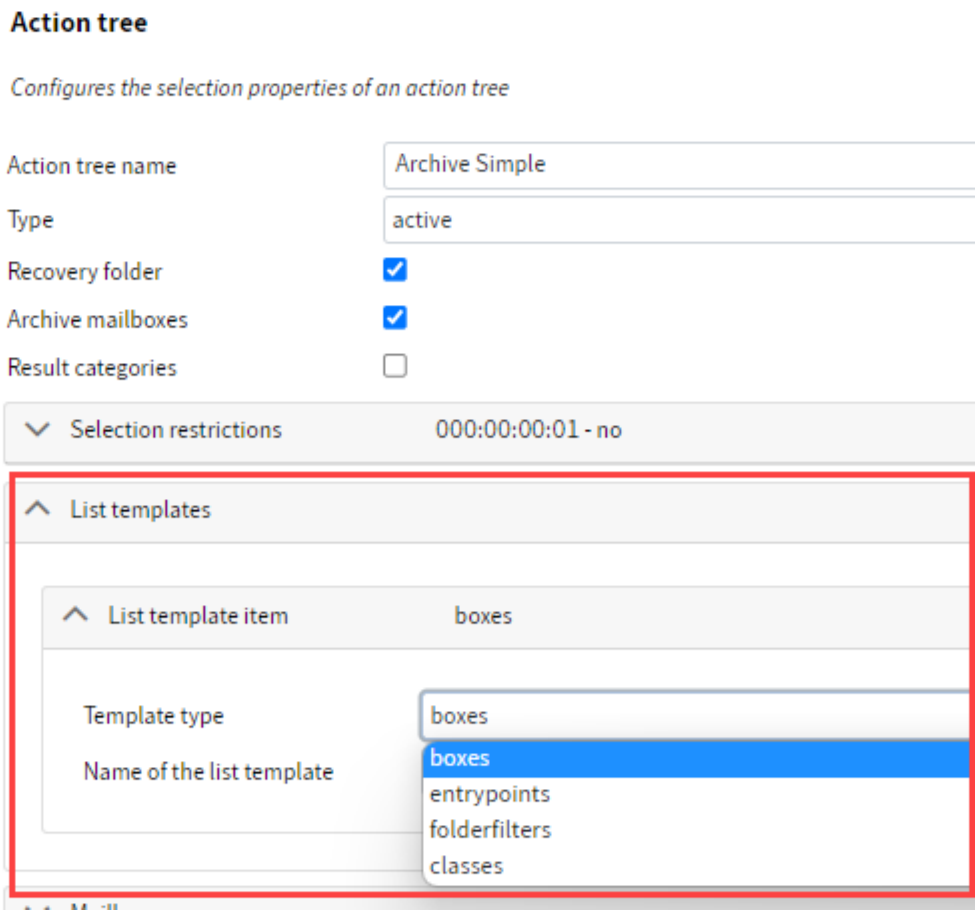
List templates

List templates are lists used by action trees to reduce the number of configuration steps required and to ensure that different trees use the same selection lists.

You have the option to extend the configuration areas of an action tree by importing additional lists.



They are added under *List templates*.



Action tree

Configures the selection properties of an action tree

Action tree name: Archive Simple

Type: active

Recovery folder ☑

Archive mailboxes ☑

Result categories □

Selection restrictions 000:00:00:01 - no

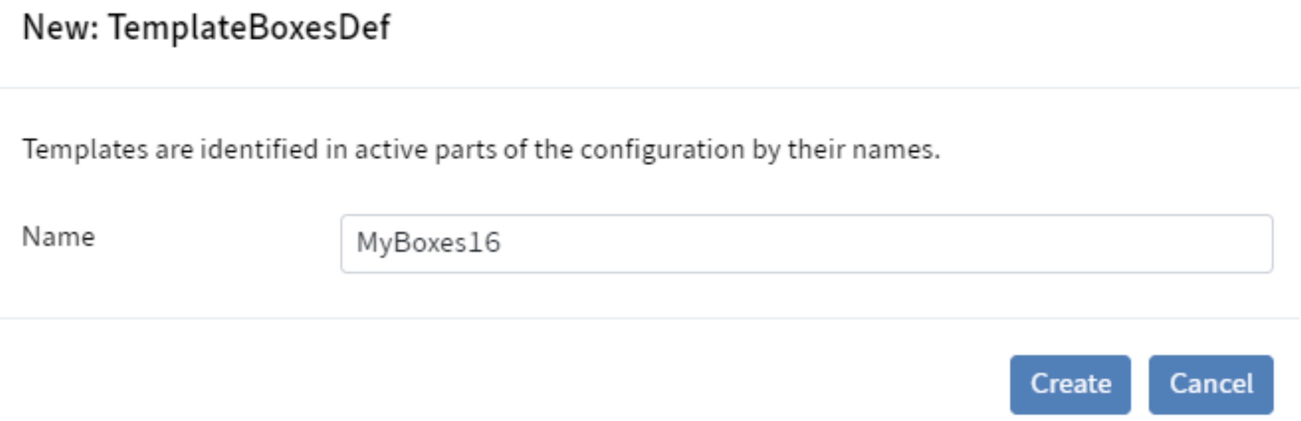
List templates

List template item boxes

Template type: boxes

Name of the list template

Each action tree can use any number of templates. Multiple occurrences of list items are automatically detected and removed. Additional configurations of list items in the action tree are merged with the imported items from the respective list template.

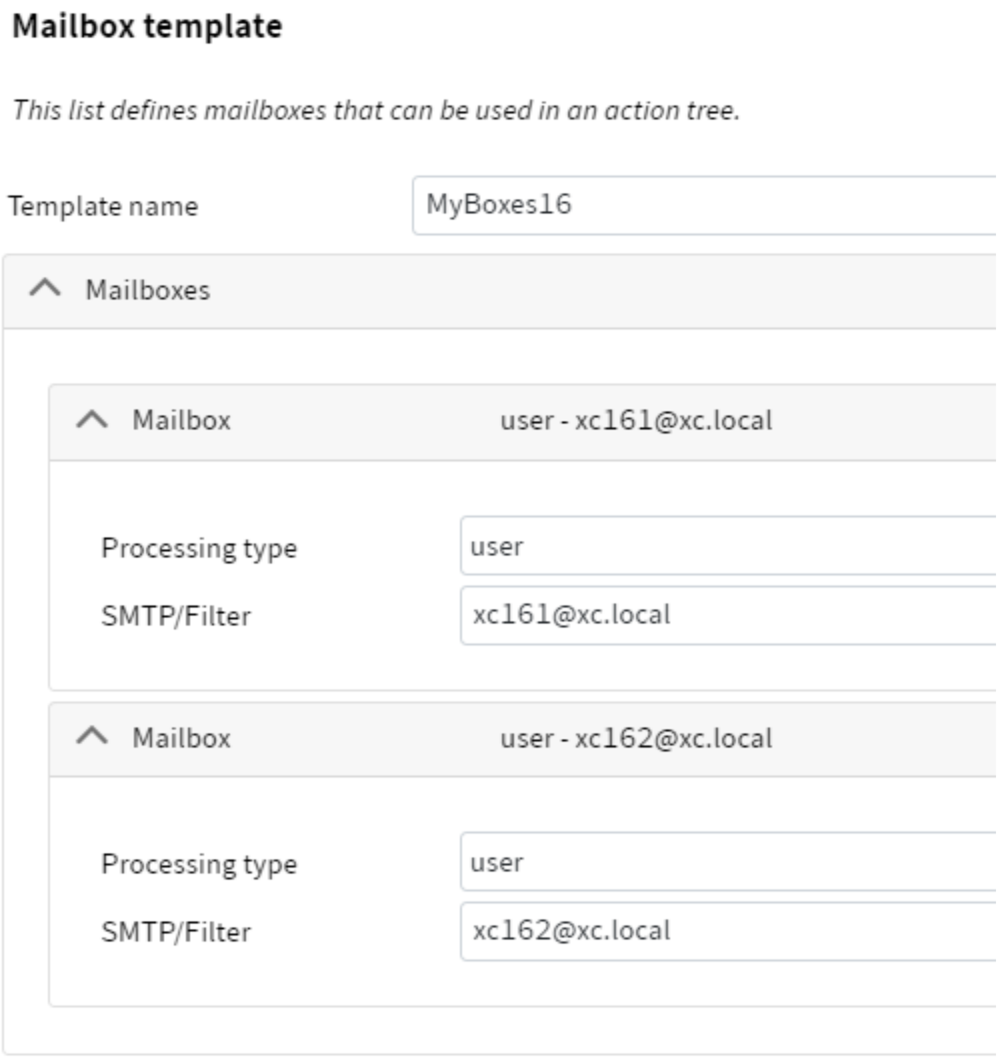


New: TemplateBoxesDef

Templates are identified in active parts of the configuration by their names.

Name: MyBoxes16

For example, you can create a mailbox template.



Mailbox template

This list defines mailboxes that can be used in an action tree.

Template name: MyBoxes16

Mailbox user - xc161@xc.local

Processing type: user

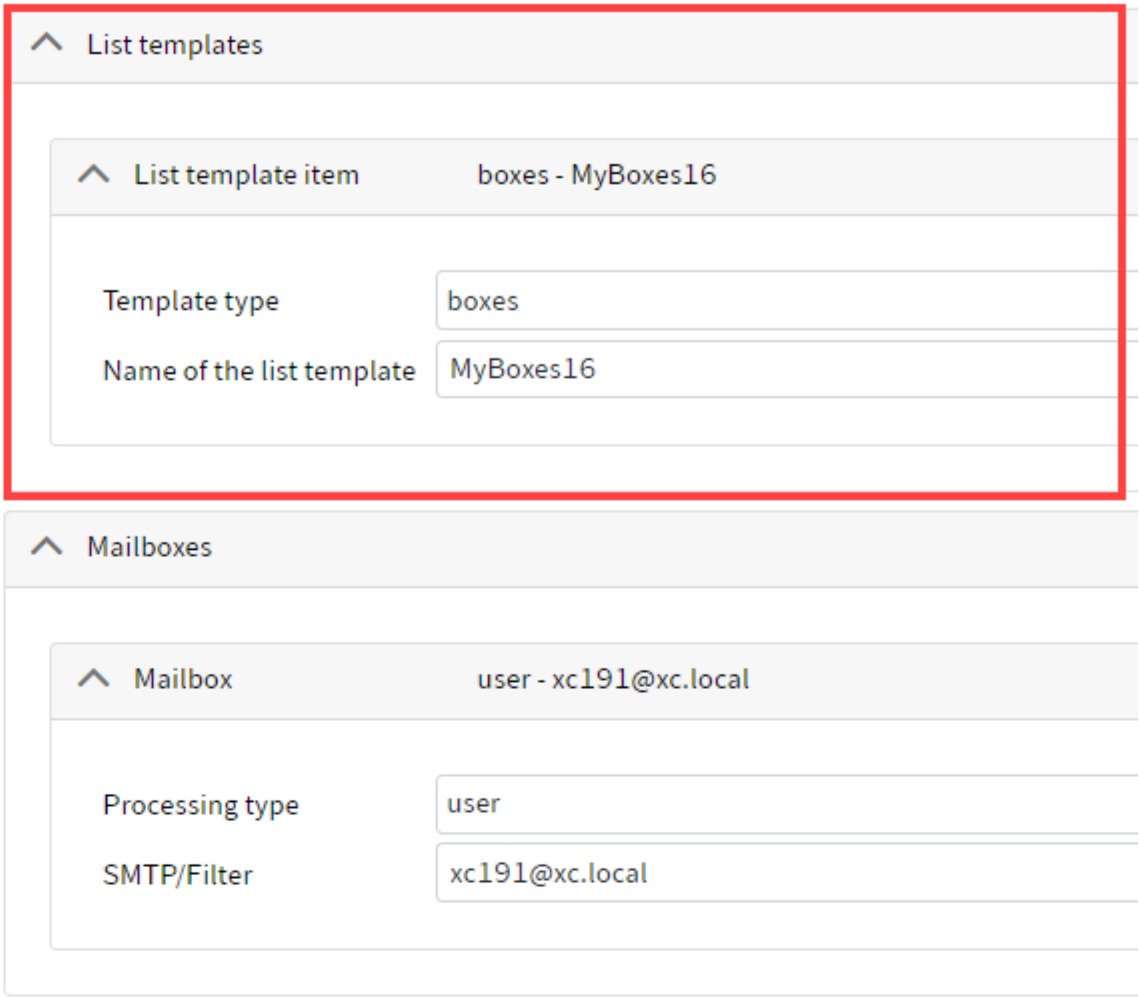
SMTP/Filter: xc161@xc.local

Mailbox user - xc162@xc.local

Processing type: user

SMTP/Filter: xc162@xc.local

Add mailboxes.



List templates

List template item boxes - MyBoxes16

Template type	boxes
Name of the list template	MyBoxes16

Mailboxes

Mailbox user - xc191@xc.local

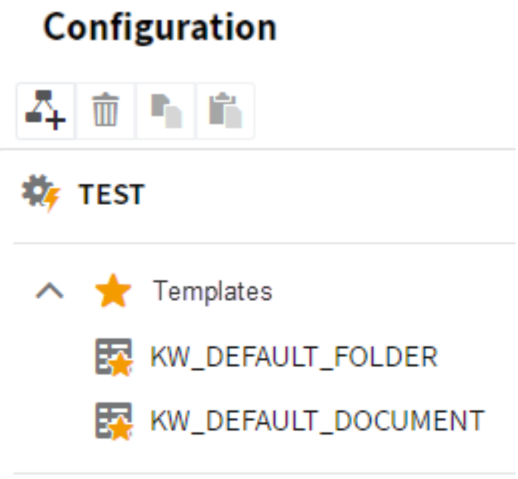
Processing type	user
SMTP/Filter	xc191@xc.local

Refer to this template in the action tree by name.

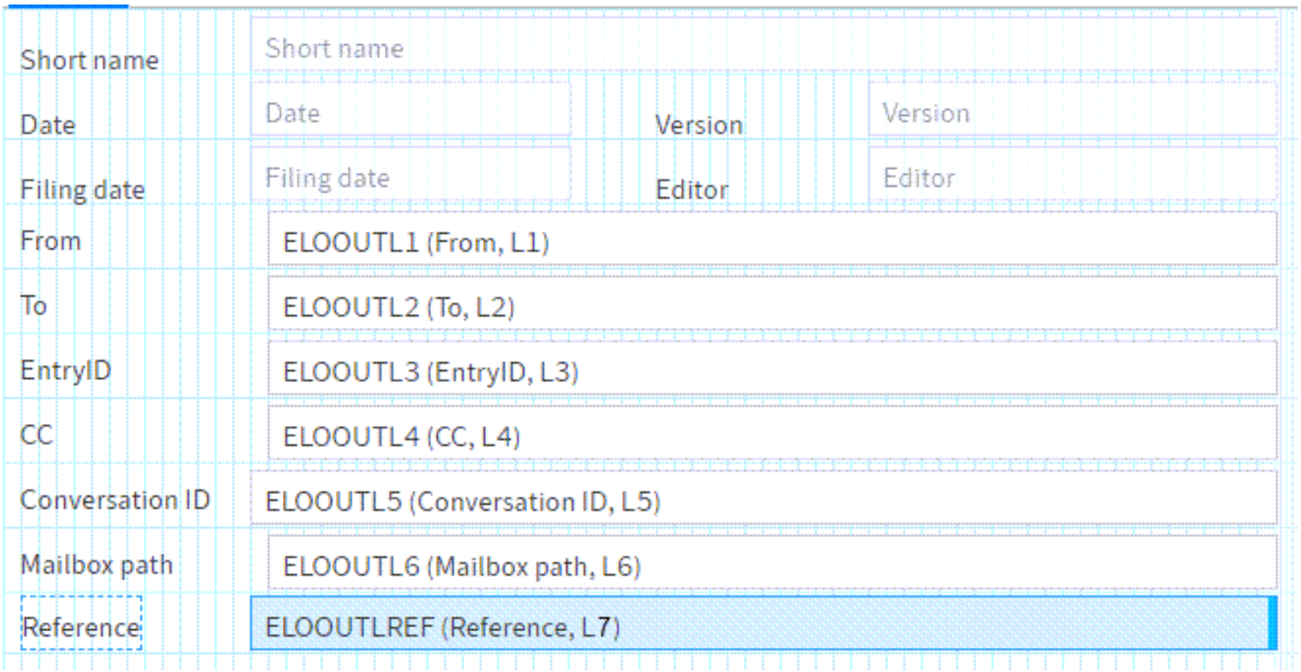
The imported mailboxes *xc161* and *xc162* with the mailbox *xc191* specifically for this action tree will be merged into one list {*xc161, xc162, xc191*}.

The list templates for *folder filters*, *entry points*, and *message classes* are configured in the same way.

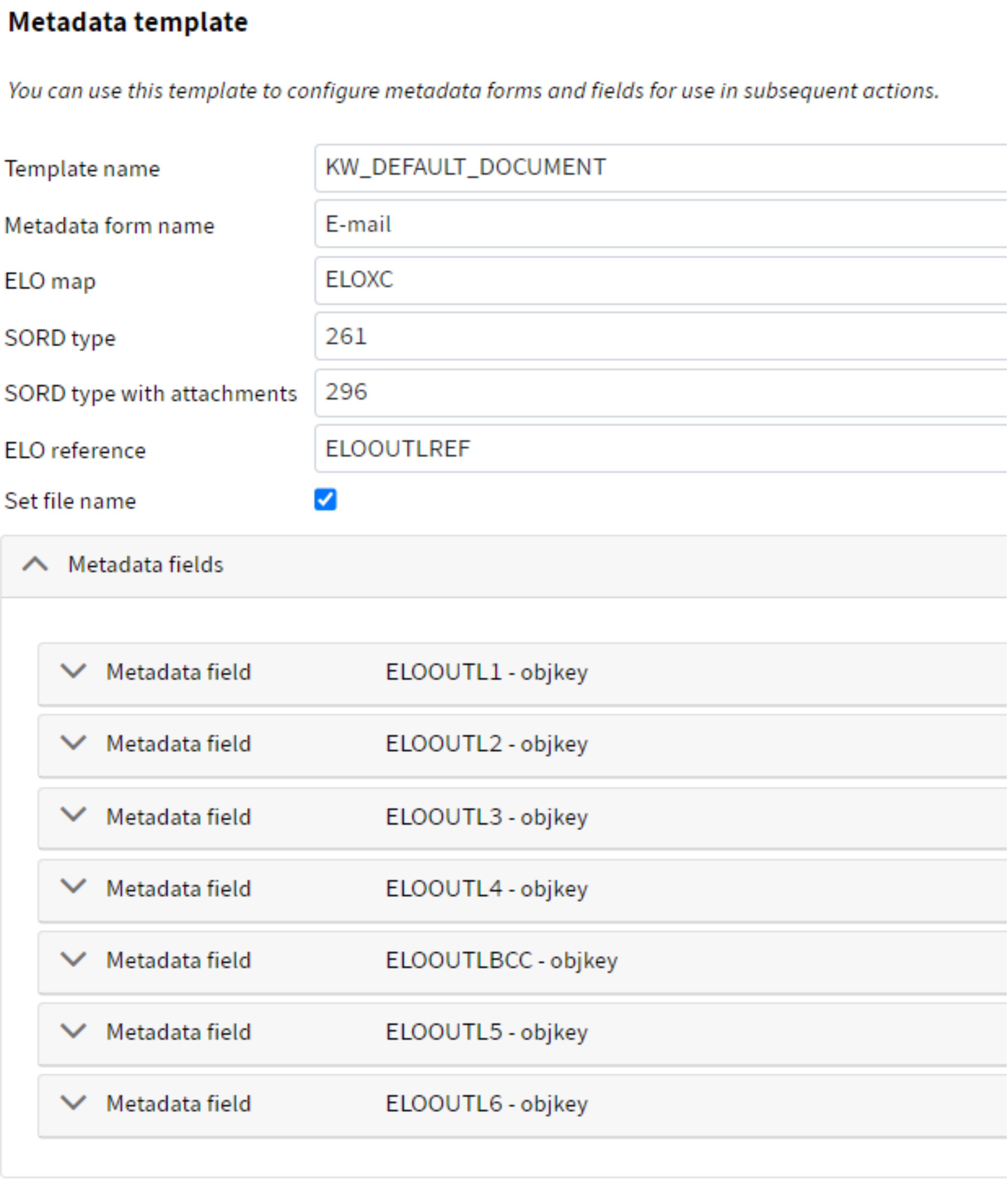
Metadata templates



ELO XC uses metadata templates to transfer properties to the metadata fields of an ELO metadata form so that additional information can be transferred to the repository and used along with the mailbox items. The templates *KW*DEFAULT*DOCUMENT* for e-mails, and *KW*DEFAULT*FOLDER* for folders in the filing path, are statically integrated by default. They are created again if missing.



Short name	Short name		
Date	Date	Version	Version
Filing date	Filing date	Editor	Editor
From	ELOOUTL1 (From, L1)		
To	ELOOUTL2 (To, L2)		
EntryID	ELOOUTL3 (EntryID, L3)		
CC	ELOOUTL4 (CC, L4)		
Conversation ID	ELOOUTL5 (Conversation ID, L5)		
Mailbox path	ELOOUTL6 (Mailbox path, L6)		
Reference	ELOOUTLREF (Reference, L7)		



Metadata template

You can use this template to configure metadata forms and fields for use in subsequent actions.

Template name	KW_DEFAULT_DOCUMENT
Metadata form name	E-mail
ELO map	ELOXC
SORD type	261
SORD type with attachments	296
ELO reference	ELOOUTLREF
Set file name	☑

Metadata fields

Metadata field	ELOOUTL1 - objkey
Metadata field	ELOOUTL2 - objkey
Metadata field	ELOOUTL3 - objkey
Metadata field	ELOOUTL4 - objkey
Metadata field	ELOOUTLBCC - objkey
Metadata field	ELOOUTL5 - objkey
Metadata field	ELOOUTL6 - objkey

In addition to assigning values to the metadata fields, a template is used to set the corresponding form, whose settings (e. g. access rights) the Indexserver takes into account when transferring the messages to the repository.

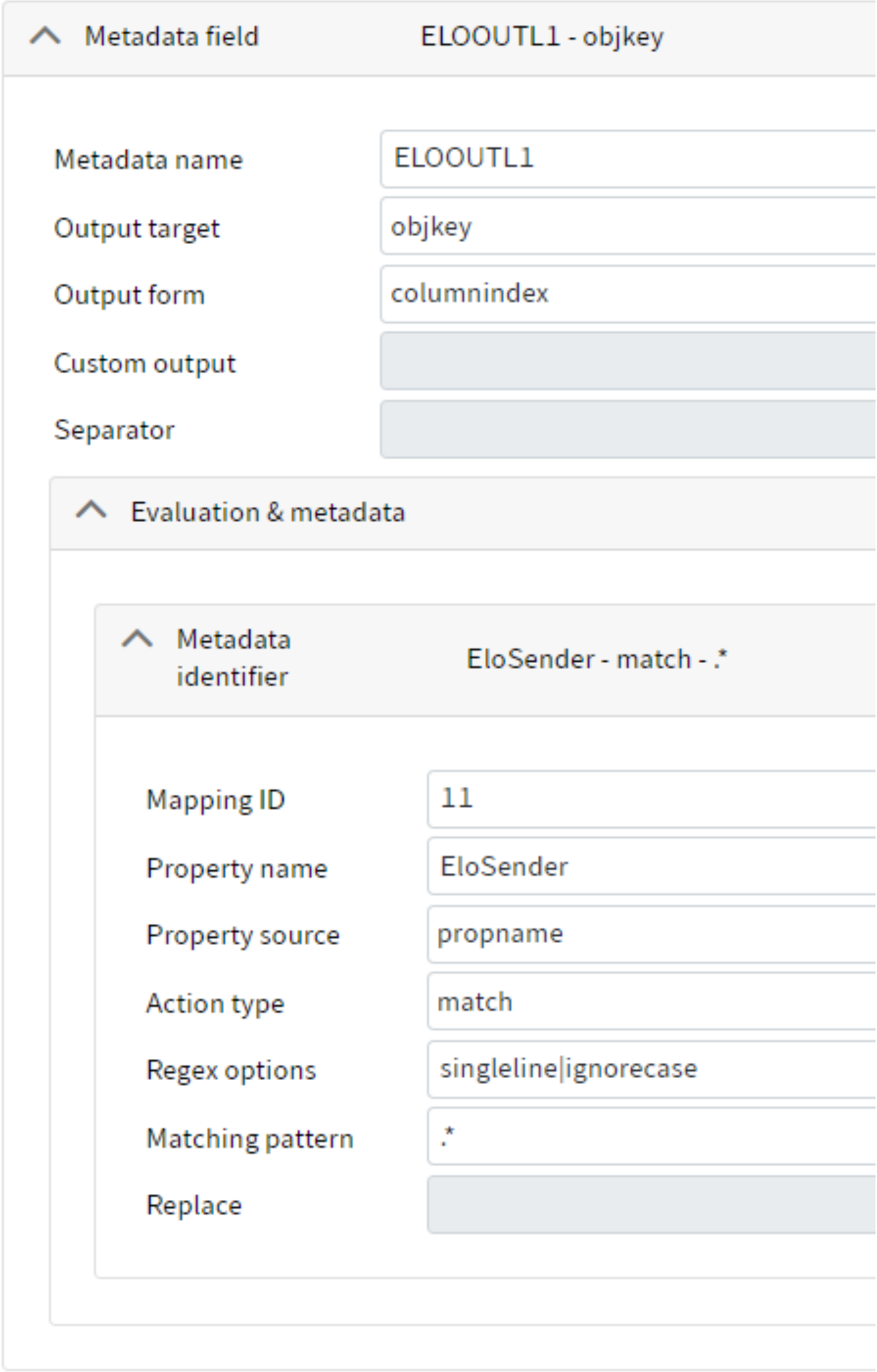
- Template name: The template name is referenced in the *Export* action.
- Metadata form name: The form is specified by its name (see ELO Administration Console).
- SORD types: These parameters assign the appropriate icon types to messages and messages with attachments in the repository.
-

ELO reference: If an attachment is extracted from a message and transferred to ELO, the GUID of the message is assigned to this metadata field. This enables actions such as *Exists* and *Stubbing* to establish a relation between the message and attachments in ELO.

- ELO map: This value is the name of the map domain if you want to store metadata internally in an ELO map. For example, the *Objects* value is required to store metadata in *Additional information*.
- Set file name: This option populates the *ELO_FNAME* global metadata field.

Metadata fields

The metadata fields can be determined in different ways at runtime. It is possible to return a list of value matches for each metadata field. The metadata value can be set if these actions are successful.



Metadata field ELOOUTL1 - objkey

Metadata name	ELOOUTL1
Output target	objkey
Output form	columnindex
Custom output	
Separator	

Evaluation & metadata

Metadata identifier EloSender - match - .*

Mapping ID	11
Property name	EloSender
Property source	propname
Action type	match
Regex options	singleline|ignorecase
Matching pattern	.*
Replace	

The values of the property *EloRecipientsTo* are transferred to the metadata field *ELOOUTL2* if they correspond to the pattern .*. Using the wildcard .* means that the match is always successful and all property values are transferred to the metadata field.

- Metadata name: This is the group name of the metadata field. If using an aspect metadata form, you enter the name of the aspect mapping here.
- Output target: Specify here whether you want the result to be written to metadata fields, an ELO map in ELO XC, or both. The *aspect* setting means that the configuration is based on organization of data with aspects and the metadata template should use aspect mappings.
-

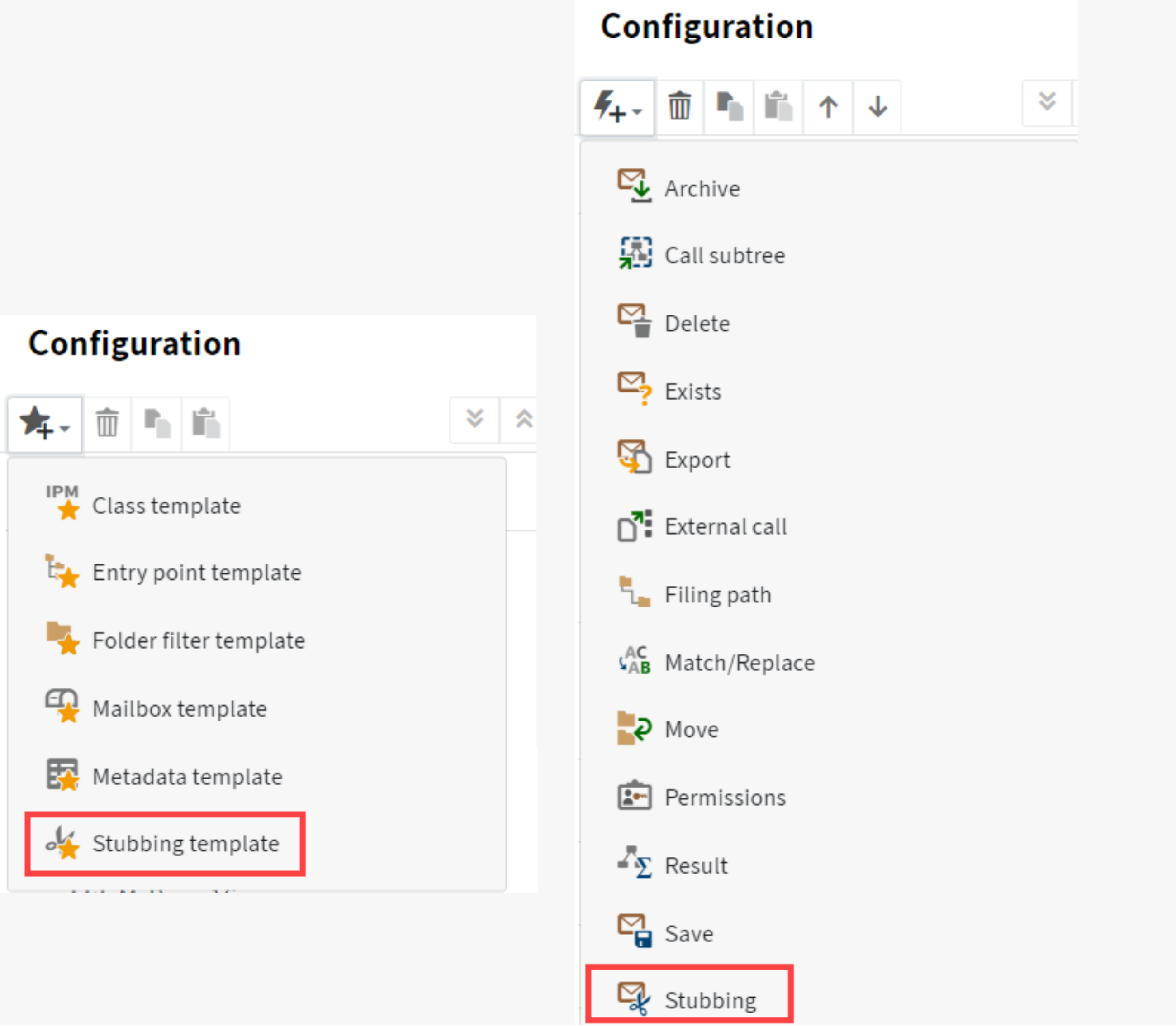
Output form: By default, the results are output to a *column index*. Alternatively, you can string lists without column separators as a single character string or configure them as individual output with *custom* (see Output and Match ID).

- Custom output: If you select the output form *custom*, this parameter contains the output pattern that you can define individually and assign the individual results of the evaluation via the match IDs. A match ID is entered as `{ID}`.
- Separator: You apply the output separator to properties whose values are based on lists if the output form is *concat*.
- Mapping ID: This field is completed for custom output of the property values in metadata fields or mapping to aspect fields. In case of custom output, you can choose any identifier. To map the value on an aspect field, the technical name of the aspect field has to be selected.
- Property name: This field contains the name of an Exchange message property or ELO property. Double-click to display a filtered list of available properties.
- Property source: The default value *propname* determines that the field should be assigned a property value. You can use *cachedname* to specify that the field is assigned a previously modified and cached value of the property. The *constant* and *var* options are required in other fragments and are not valid in metadata templates.
- Action type: With the default value *match*, the property value is inherited on a successful match. With *replace*, the value is modified according to the *Replace* parameter before being transferred to the metadata field.
- Regex options, Matching pattern, and Replace evaluate the property using regular expressions.

Stubbing templates

Please note

ELO XC provides the stubbing function for compatibility with previous versions. However, it should not be used. Microsoft discourages the use of *message stubbing* because it does not improve Exchange performance as originally intended. The number of mailbox items increases the load on Exchange more than their collective size. The best way to improve performance is to reduce the total number of mailbox items. In addition, administrative issues can arise in the long term due to obsolete references, so stubbing only makes sense if the ELO references created in the message bodies and Exchange server performance are not important in the long term.



Stubbing is a function that replaces parts of an e-mail with external links or new contents. In ELO XC, stubbing is based on templates (*stubbing template*) that are used in an action (*Stubbing*).

The template contains two parts. One part is for stubbing messages in text format and the other part is for messages in HTML format. ELO XC recognizes the format of each message and uses the corresponding part of the template. Messages in rich text format cannot be stubbed and are ignored with a corresponding error message in the log.

Variables

A template represents the new body and supports the use of stubbing variables. These variables are always written to the stubbing template as `{%VARIABLE}`. The available variables are:

- IX: This variable is replaced by the URL of the corresponding Indexserver.
- ARCNAME: This writes the name of the repository to the body.
- ARCDATE: This variable returns the date and time of archiving.
- DELDATE: This variable returns the date and time of stubbing.
-

LINK: This generates a Web Client URL that refers to the archived e-mail.

- ATTLINK: As with the variable *Link*, this variable is used for extracted attachments.
- ECD: This variable ensures that ELO links of the archived e-mail and all its attachments are attached to the e-mail.
- BODY: This variable returns the whole message body.
- HTMLBODY: This variable can only be used in HTML format and ensures that the previous message body remains in HTML format. At the same time, all variables except ECD are ignored.
- GUID: This variable is not used in *templates* but in the *Stubbing* action, where the ELO Web Client URL is configured as a link template.

Time formats

The standard assignment of the time variables *ARCDATE* and *DELDATE* is as follows: `{%DELDATE:UTC:ISO:en-US}`.

It is possible to configure additional format requirements by separating them with colons. If `UTC` is omitted, the local time is used. If you do not want to generate an ISO date with the format `YYYYMMDDhhmmss`, you can use one of the following specifications instead:

- D: Short date pattern
- d: Long date pattern
- f: Complete date/time pattern (short term)
- F: Complete date/time pattern (long term)
- g: General date/time pattern (short term)
- G: General date/time pattern (long term)
- t: Short time pattern
- T: Long time pattern

However, these abbreviations sometimes also generate language-specific output, which is why an additional language/country abbreviation (here: *en-US*) is required.

For more information on the *standard format string* for date and time, refer to this Microsoft documentation.

Example

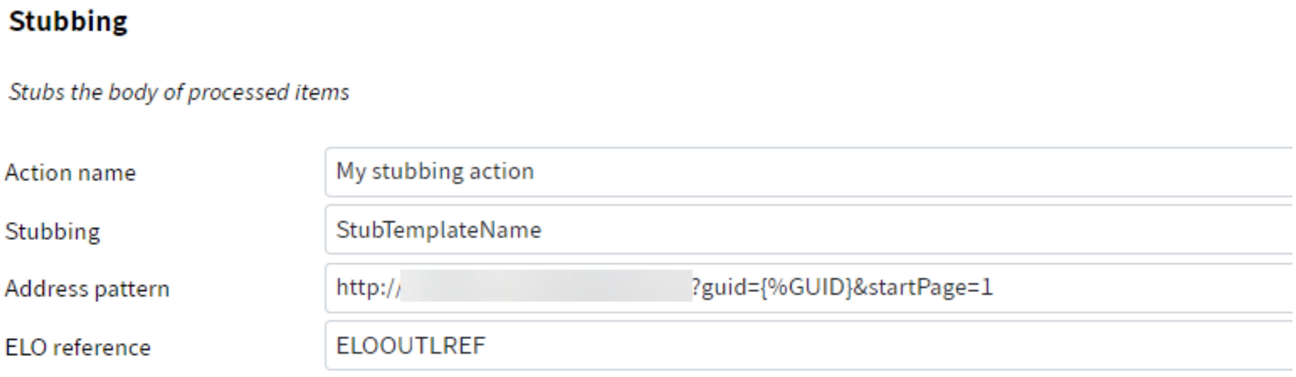
Here is an example of a body template (text).

```
This message was stored in the repository {%ARCNAME} via the ELO Indexserver {%IX} on {%ARC

Click {%LINK} to view the message.
------------------------------------------------------------------------


Extracted attachments:
{%ATTLINK}
------------------------------------------------------------------------
```

```
Whole message:
{%BODY}
------------------------------------------------------------------------
```

Here is an example of a body template (HTML):

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
  </head>
  <body>
    Repository<br />
    IX1={%IX} IX2={%IX}<br />
    ARCNAME={%ARCNAME} ARCNAME={%ARCNAME}<br />
    <hr />
    <br />
    Calendar dates
    <br />ARCDATE={%ARCDATE} DELDATE={%DELDATE} <br />UTC-US
    ARCDATE={%ARCDATE:UTC} DELDATE={%DELDATE:UTC} <br />UTC-DE
    ARCDATE={%ARCDATE:UTC:U:de-DE} DELDATE={%DELDATE:UTC:U:de-DE} <br />LOC-DE
    ARCDATE={%ARCDATE:F:de-DE} DELDATE={%DELDATE:F:de-DE} <br />UTC-DE
    ARCDATE={%ARCDATE:UTC:F:de-DE} DELDATE={%DELDATE:UTC:F:de-DE} <br />
    <hr />
    <br />
    ECD {%ECD} (should not be visible in the body) LINK {%LINK} {%LINK}
    <br />
    <hr />
    <br />
    ATTLINKS {%ATTLINK} {%ATTLINK}
    <br />
    <hr />
    <br />
    Body
    <br />{%BODY} <br />
    <hr />
    <br />
  </body>
</html>
```



Stubbing

Stubs the body of processed items

Action name	My stubbing action
Stubbing	StubTemplateName
Address pattern	http://[redacted]?guid={%GUID}&startPage=1
ELO reference	ELOOUTLREF

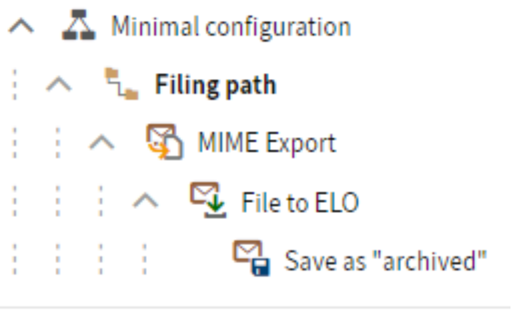
- Stubbing: In the *Stubbing* action, enter the name of the template in the *Stubbing* field.
- Address pattern: The *Address pattern* field is required for the ELO Web Client URL if the variables *LINK* and *ATTNLINK* need to be supported. This URL must also contain the *GUID* variable in this case. ELO XC generates the correct URLs in the stubbing process.
- ELO reference: In addition, you need to set the *ELO reference* parameter so that the GUID of the e-mail is saved to this metadata field when extracting attachments. Otherwise, ELO XC cannot determine the values of the variable *ATTLINK*.

Automated processes

Action principle

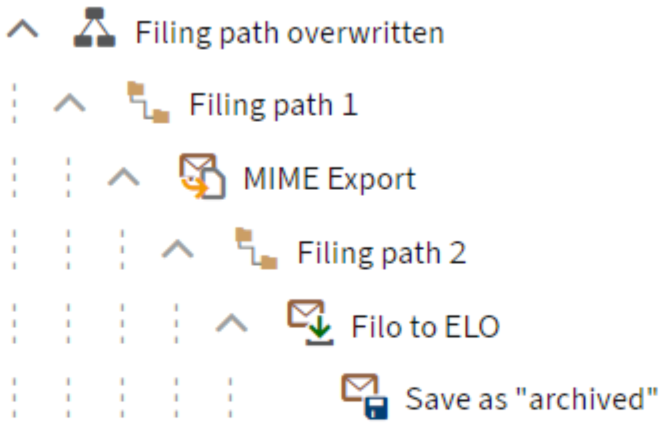
Configuration using the action principle enables you to create complex instance configurations with a modular structure. A job traverses the action trees of the configuration one after the other and uses an Exchange search to select the items within the retrieved mailboxes that are to be processed.

The tree structure is traversed for each Exchange item found. The item data is loaded according to the configuration. The item properties are always unique in the entire action tree, regardless of how often an action type is traversed.



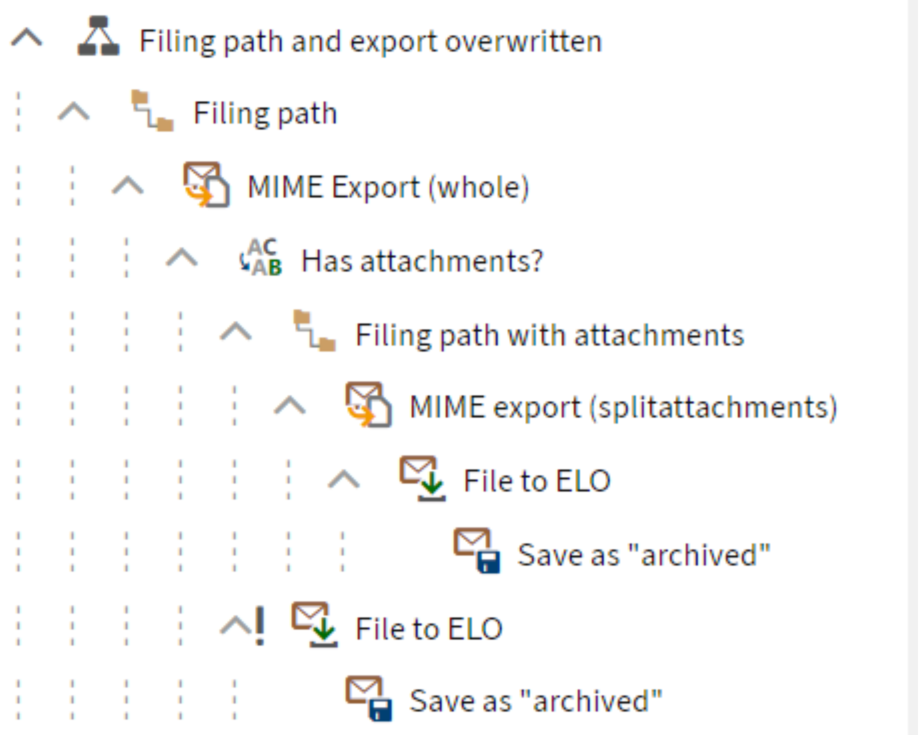
The minimal configuration of an action tree for storing messages consists of a filing path in the repository, a MIME export, the storage action, and subsequently tagging and saving the message.

The filing path in the repository and the MIME document are required for successful storage. The *"archived"* tag is set during storage and subsequently saved.



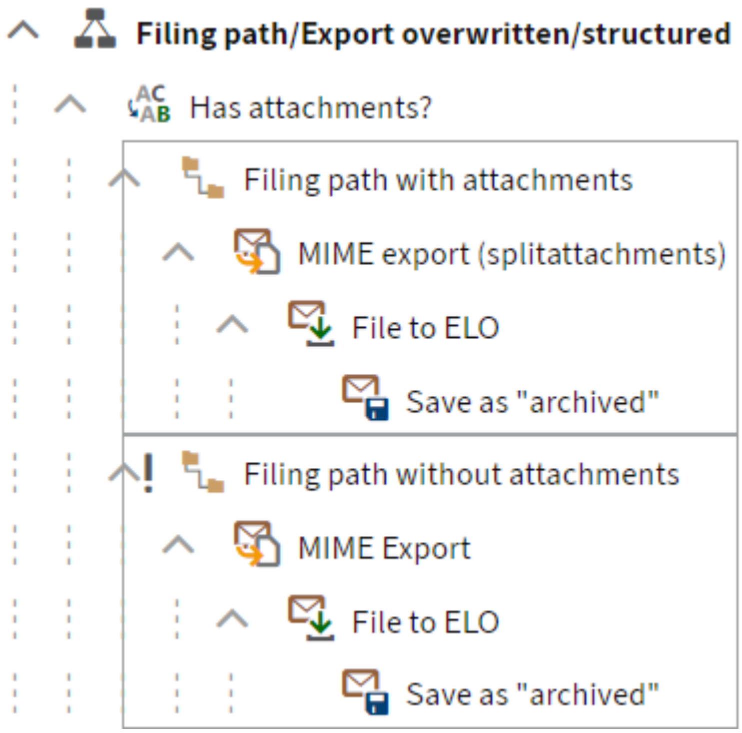
If an action type is traversed multiple times prior to storage, only the most recently used value is allowed.

For example, *Filing path 1* is overwritten with *Filing path 2* before storage. Only *Filing path 2* will be used from now on. This process of automatically writing a value when an action is repeated also applies to message exports.



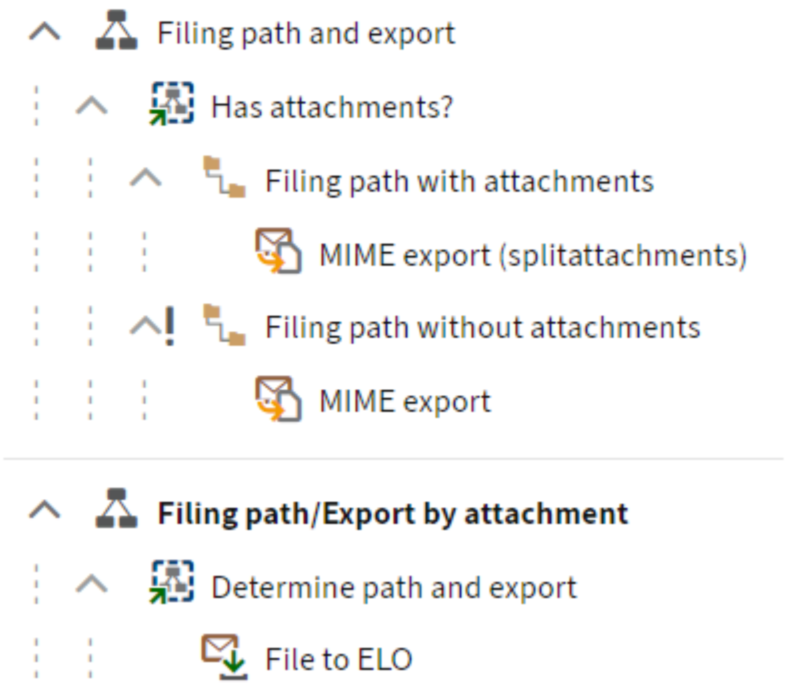
- Filing path and export overwritten
  - Filing path
    - MIME Export (whole)
      - Has attachments?
        - Filing path with attachments
          - MIME export (splitattachments)
            - File to ELO
              - Save as "archived"
        - ! File to ELO
          - Save as "archived"

This tree uses different filing path and export types for messages with and without attachments. It shows how instance configurations become more complex as they grow and actions are not structured in a meaningful way.



- **Filing path/Export overwritten/structured**
  - Has attachments?
    - Filing path with attachments
      - MIME export (splitattachments)
        - File to ELO
          - Save as "archived"
    - ! Filing path without attachments
      - MIME Export
        - File to ELO
          - Save as "archived"

This figure clearly illustrates that the action is split up into two similar branches. This structure is possible because the order of the actions before storage is irrelevant. Filing path and export actions can be executed any number of times in succession.



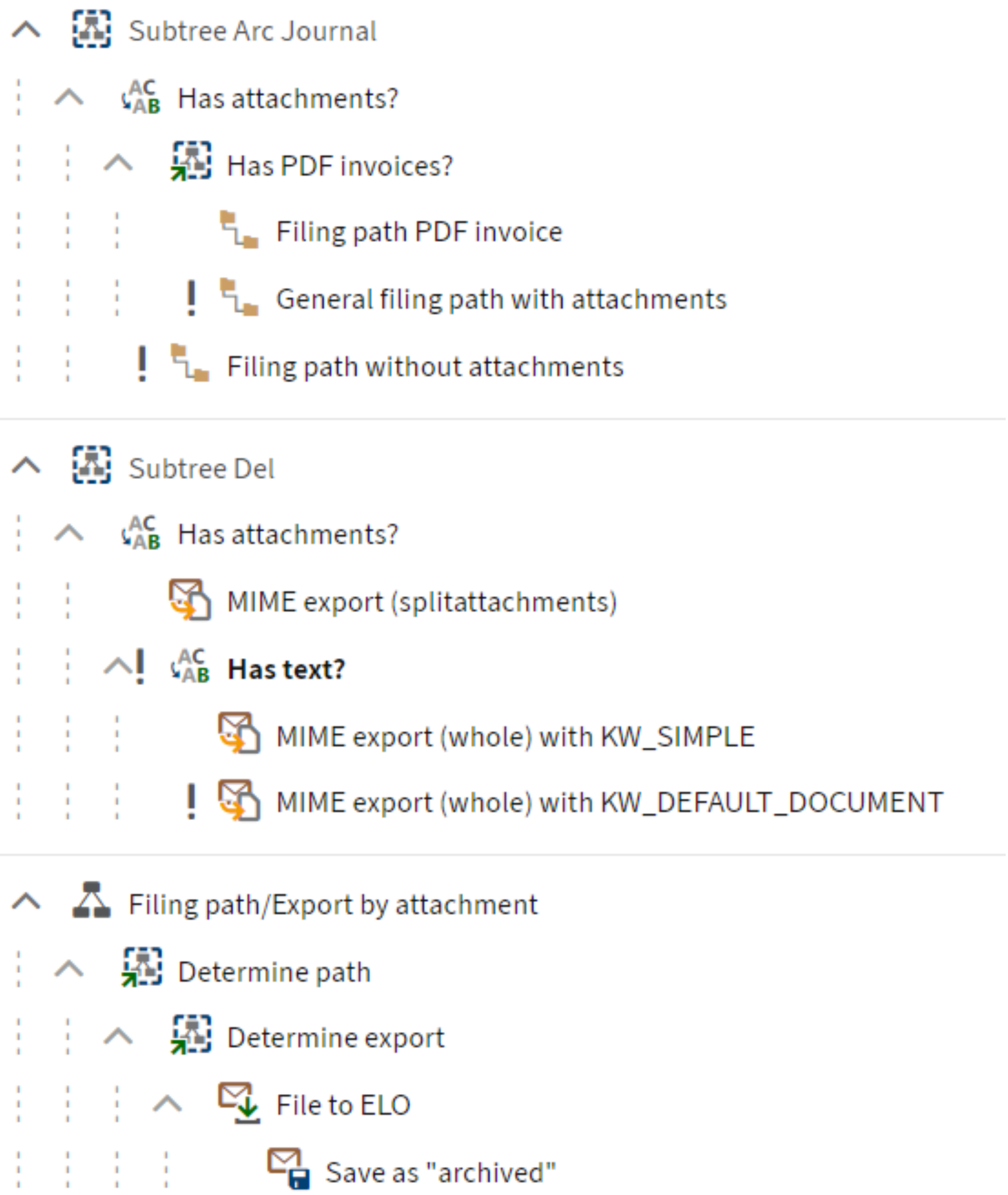
Filing path and export
Has attachments?
Filing path with attachments
MIME export (splitattachments)
Filing path without attachments
MIME export

Filing path/Export by attachment
Determine path and export
File to ELO

However, this tree also shows that despite the structure with two main branches, the actions *File to ELO* and *Save as "archived"* must be configured twice. This can also be prevented by configuring and calling subtrees.

If you call the *Filing path and export* subtree, the message is not filed to ELO and saved. Both actions are only required once.

Subtrees are executed irrespective of the selection settings since the items being processed have already been selected by the active action tree. A subtree therefore only requires minimal configuration.



- Subtree Arc Journal
  - Has attachments?
    - Has PDF invoices?
      - Filing path PDF invoice
      - ! General filing path with attachments
    - ! Filing path without attachments
- Subtree Del
  - Has attachments?
    - MIME export (splitattachments)
    - ! **Has text?**
      - MIME export (whole) with KW_SIMPLE
      - ! MIME export (whole) with KW_DEFAULT_DOCUMENT
- Filing path/Export by attachment
  - Determine path
    - Determine export
      - File to ELO
        - Save as "archived"

In this example, however, the amount of configuration has not yet been reduced since although there was no need to configure *File to ELO* and *Save as "archived"* twice, the amount of configuration stays the same by adding the subtree and the subtree call.

As soon as additional case distinctions are added, the benefits of the subtree outweigh the disadvantages.

The advantages of using subtrees only become really apparent when decisions (e.g. type of filing path or export) are required in different action trees.

Properties

There are two types of properties: *PidTags* and ELO properties.

PidTags are defined by Microsoft Exchange. Their availability depends on various factors such as message class, transmission status, message formats, and occasionally internal Exchange settings. There is no way to determine a complete set of properties für a message before a property is accessed.

The number of properties read at the start of processing depends on the configuration of an action tree. Every time a *PidTag* is used in an action, it is loaded with the message and is available until the action tree is completed.

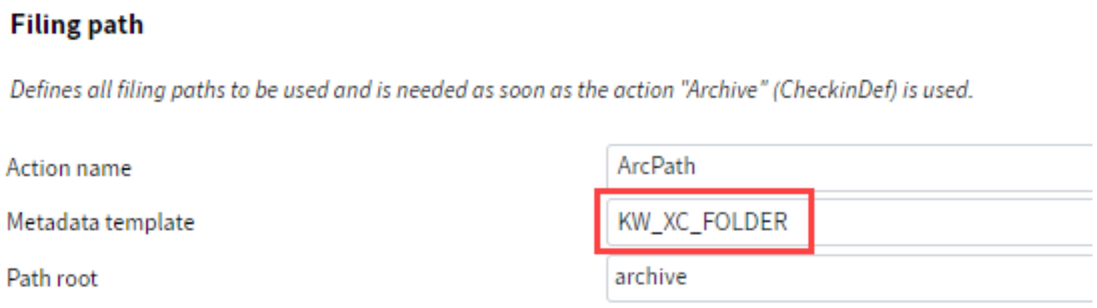
These properties can be changed as the actions are executed. However, changes to a message are not saved in Exchange until the *Save* action is executed. Some actions allow the use of working copies (see *cachedname*). These copies allow you to change message properties during processing and use them in ELO without overwriting them in the original message on saving.

PidTags are designed for use with different data types. ELO XC supports individual values of strings, numbers, time values, and currencies. Binary data, Exchange references, and value lists of any type, including those of the supported values, are ignored and are not available in XC.

ELO properties are recognized by their prefix. They provide calculated values that are relevant for message processing in ELO XC and for which there are no *PidTags*. They are calculated internally according to the current processing state but cannot be changed.

Metadata

ELO metadata is required whenever SORDs have to be created. This applies to filing paths in the repository and the storage of messages and/or attachments as documents. They are configured with metadata templates that are incorporated in the *Filing path* and *Export* actions. The metadata are retrieved during the execution of these actions and written to the repository when the *Archive* action is executed.

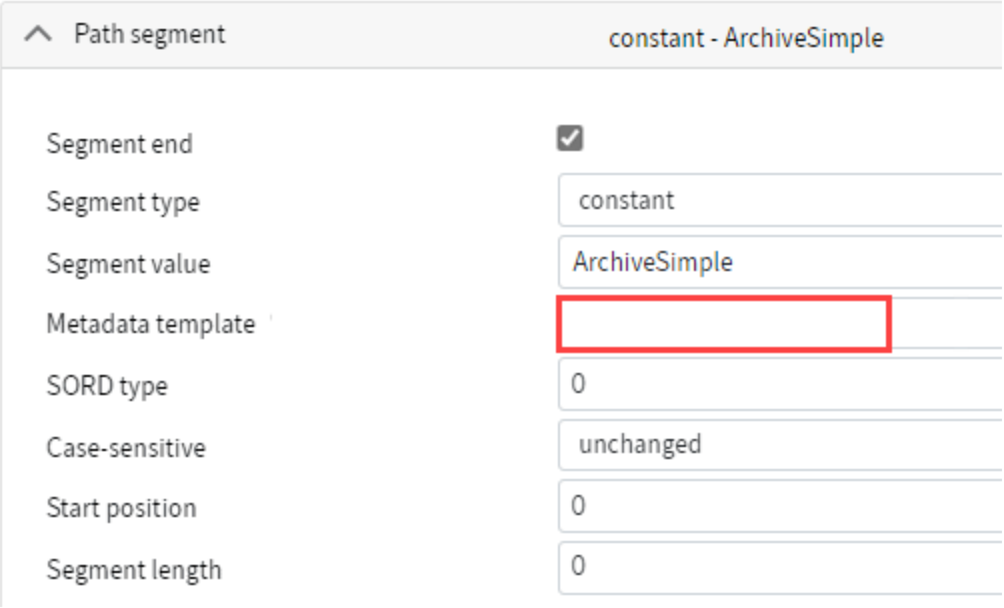


Filing path

Defines all filing paths to be used and is needed as soon as the action "Archive" (CheckinDef) is used.

Action name	ArcPath
Metadata template	KW_XC_FOLDER
Path root	archive

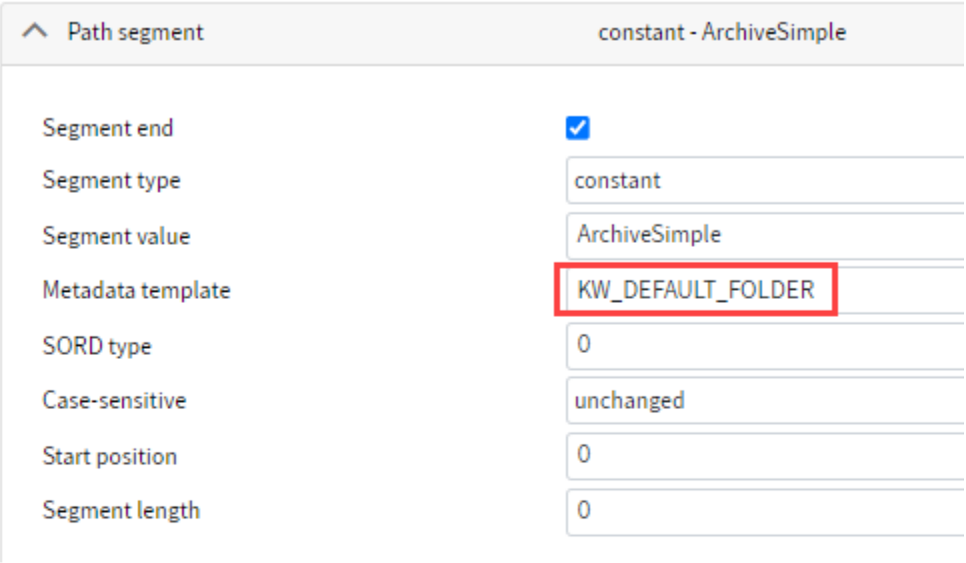
Each generated SORD can have its own metadata in filing paths.



Path segment — constant - ArchiveSimple

Segment end	☑
Segment type	constant
Segment value	ArchiveSimple
Metadata template	
SORD type	0
Case-sensitive	unchanged
Start position	0
Segment length	0

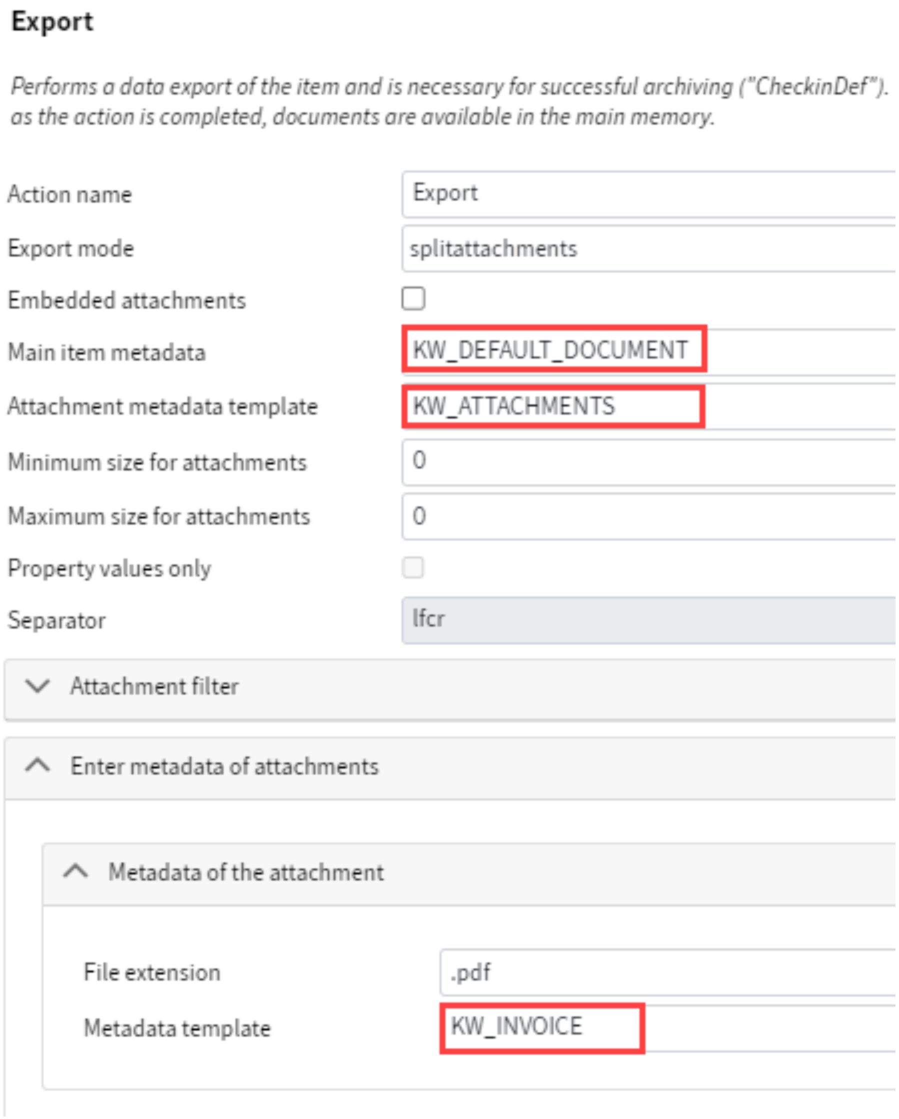
The configured template is valid for all path segments if the *Metadata template* parameter is empty.



Path segment — constant - ArchiveSimple

Segment end	☑
Segment type	constant
Segment value	ArchiveSimple
Metadata template	KW_DEFAULT_FOLDER
SORD type	0
Case-sensitive	unchanged
Start position	0
Segment length	0

You need to configure a custom template to write alternative metadata for a path segment.



Export

Performs a data export of the item and is necessary for successful archiving ("CheckinDef"). as the action is completed, documents are available in the main memory.

Action name	Export
Export mode	splitattachments
Embedded attachments	☐
Main item metadata	KW_DEFAULT_DOCUMENT
Attachment metadata template	KW_ATTACHMENTS
Minimum size for attachments	0
Maximum size for attachments	0
Property values only	☐
Separator	lfcr

File extension	.pdf
Metadata template	KW_INVOICE

The *Export* action distinguishes between the main document, i.e. the message and its attachments.

Metadata search (gen. 1 and gen. 2)

At various points when configuring an instance, you have the option to specify parameters that trigger a metadata search. For example, the *Exists* action uses an *identification field* that identifies stored messages based on a search term (e.g. *PidTagInternetMessageId* or *PidTagSearchKey*). The internal reorganization from gen. 1 to gen. 2 metadata also brings a change in the naming convention. A metadata field (index field in gen. 1) was referenced by its group name alone (*objkeys* table, *okeyname* column). In this case, you only needed to enter the field name in the configuration. With the more flexible data model in gen. 2, it is only possible to determine a field by taking into account what it is being used for, which is done through the concept of aspects, the associated fields, and mappings.

An aspect-based metadata search also requires the configuration of a field. If there are ambiguous possibilities, you need to specify the associated aspect of the data set that a field belongs to. This creates a type of search path with the following general input syntax:

```
[aspect]¶[field]
```

Aspect mappings configure how aspects and their fields are used in metadata forms. The metadata search also takes the mappings into account with the following general input syntax:

```
[mapping]¶[aspect]¶[field]
```

This three-digit notation enables you to search for specific fields by their membership in aspects. The aspects are associated with specific metadata forms. It is also possible to search by specifying the field name only, ignoring metadata form assignments and aspect restrictions. However, this affects the search time due to the complexity of the process. For details, refer to the *findByIndex* section in the Indexserver interface documentation.

ELO XC always uses the metadata search independently of the metadata form. This means that aspect mappings, the first part of the notation, are not supported. The third part can be interpreted as the counterpart to the group name in gen. 1 and therefore does not require any special consideration. The second part of the notation is important in terms of the potential number of hits and the search time.

Aspects are associated with a specific package (*package* or *namespace*). For example, an *ASP*ADDRESS* aspect can exist in the *MESSAGING* package and in the *DELIVERY* package at the same time. Therefore, an aspect is always noted with specification of the package, in this case *MESSAGING.ASP*ADDRESS* or *DELIVERY.ASP_ADDRESS*. If you look at the *aspect* table, you will see that there are two different aspects. Normally, the aspects that are relevant for ELO XC are all in one package. In most cases, notations of the form `¶[aspect]¶[field]` are sufficient, which would be `¶MESSAGING.ASP_ADDRESS¶name` in this case. You can also search for the field without specifying the aspect, e.g. with `¶¶name`, but it will take considerably more time and ELO XC triggers the search queries for each processed item. This is why it is important to avoid having ambiguous packages, although this of course cannot be ruled out entirely.

ELO XC combines the simple group name notation with the more complex notation of the aspect-based metadata search using the `|` separator symbol. The general input syntax looks like this:

```
[gen. 1]|[gen. 2]
```

For example, if 90% of all stored messages were organized with gen. 1 and 10% with gen. 2, and `ELOOUTL3` was used for gen. 1 and `EMAIL.ASPECT_ID.EMAILID` was used for gen. 2, 100% of the stored messages are taken into account in the metadata search for the `ELOOUTL3|¶EMAIL.ASPECT_ID¶EMAILID` notation.

Permissions

ELO XC supports several different options for assigning permissions to stored messages. However, keep in mind that not every mailbox has its own ELO user account. It is therefore recommended that you create a main folder with exclusive access for the mailbox owner before processing a mailbox for the first time so that the stored message contents are not compromised. The default configuration in ELO XC uses the forms *Folder* and *E-mail* whose permission settings are already configured for this case and automatically inherit the permissions of the main folder.

The ELO Indexserver LDAP interface is the standard way to import users from the external directory. For more information on LDAP interface configuration, refer to the documentation at Configuration and administration > User management > LDAP > LDAP interface configuration.

ELO XC can use this interface directly. When importing users from the external directory using ELO XC, you must ensure that no conflicts with existing user data occur. ELO XC identifies existing users with the properties *objectGuid*, *SAMAccountName*, *UserPrincipalName*, and *mail*. If there is no match, the mailbox owners are created as new ELO users.

If the storage paths are configured accordingly, mailbox owners who are identified as ELO users can have personal folders in the repository that only they have access to, which ensures that message contents are not compromised on initial storage.

The *Permissions* action can be used to configure existing users/groups who will have additional access to the stored messages.