ELO XC

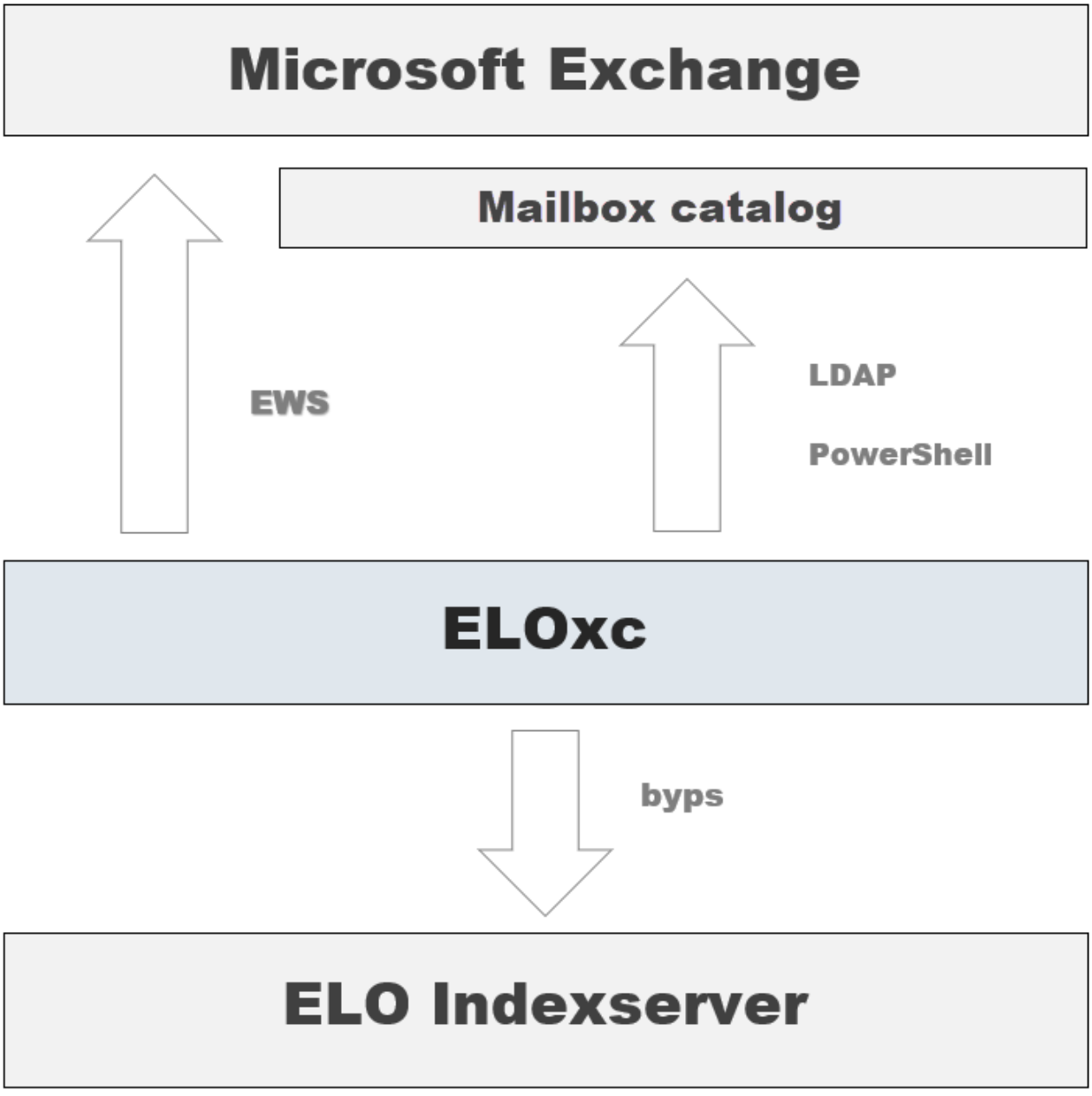
Basics

Table of contents

Basics

Configuration

Product topology



Structure

As a service, ELO XC provides a structure in which the configuration of different instances is processed. An instance constitutes a separate structure for action trees. Action trees are executed

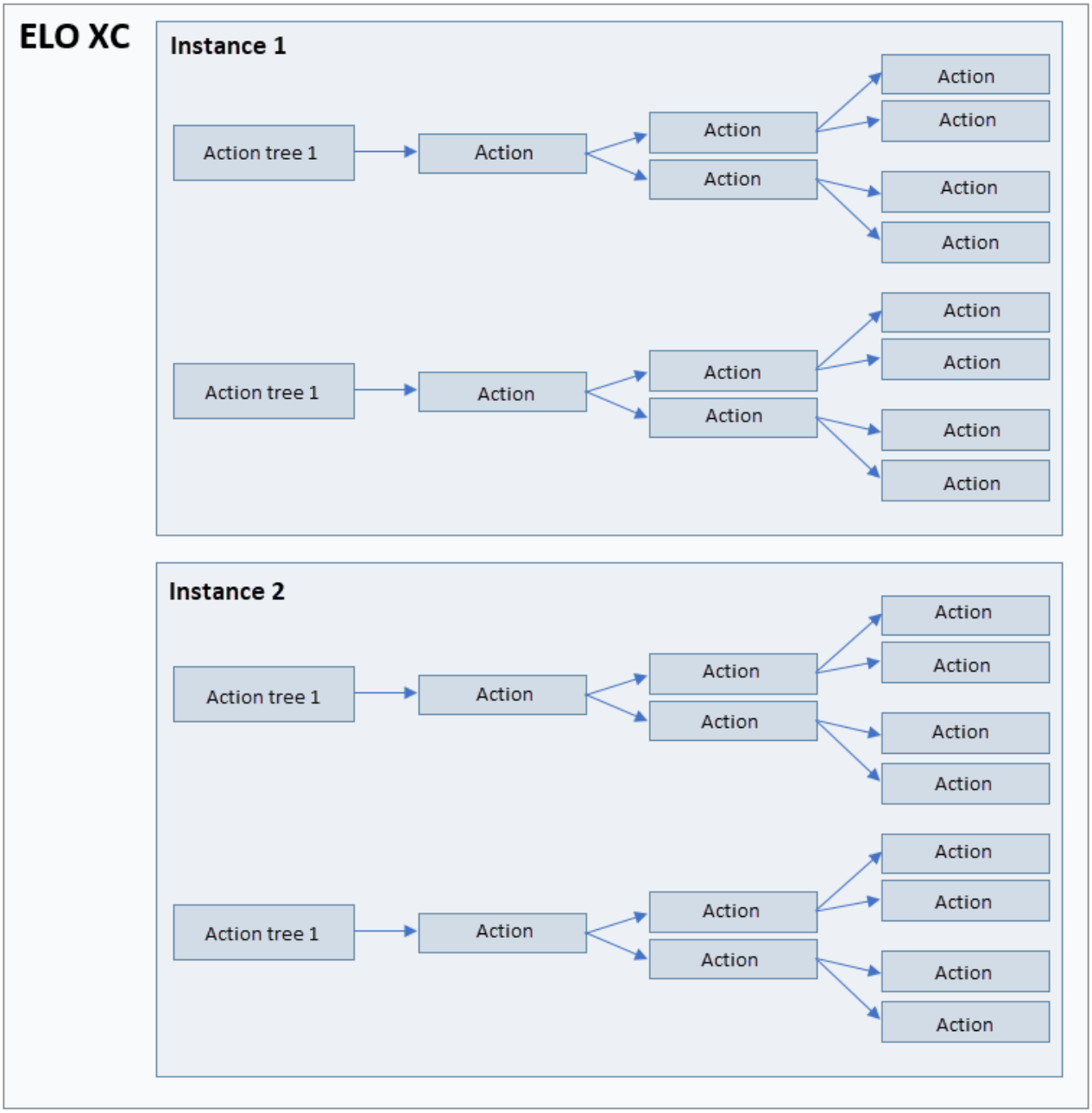
according to a specified order within an instance. They consist of actions that are arranged hierarchically and logically.

ELO XC has flexible configuration options. It offers a variety of parameters for setting the frequency and scope with which the service is processed, data granularity, and storage of e-mails in the ELO repository. The configuration of a service is broken down into separate instances whose configurations are processed independently.

Each instance contains the definition of the underlying processing settings, a catalog of available mailboxes, and any number of action trees and configuration templates. An instance is always linked to one repository.

Each action tree consists of a message selection based on the corresponding mailbox catalog and its actions.

Each action performs a general function and has its own parameters.



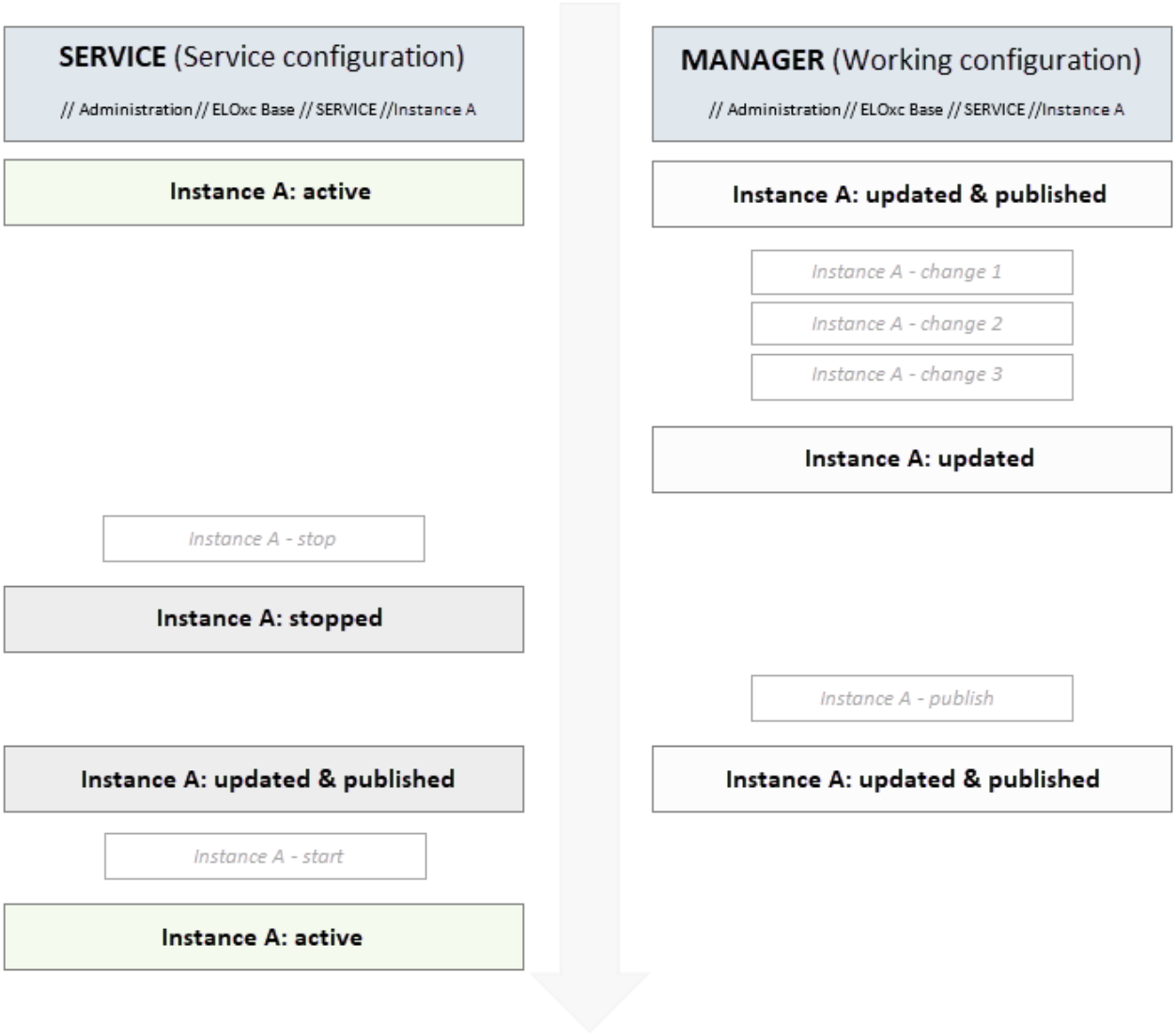
Format and source

When the program is started, the *ELOxc.xml* file (registration file) is read to determine the available instances.

It contains the address and authentication parameters for each instance. The configuration and the filing location of an instance are therefore always linked to a single repository. The instance configuration is stored in the repository under //Administration//XC Base//SERVICE and //Administration//XC Base//MANAGER.

The *SERVICE* area contains the currently active, executable configurations. The *MANAGER* area contains the working versions. This allows you to make more extensive changes to the configuration without having to interrupt processing.

Once you have completed the changes in the working configuration, you must stop the instance and publish the working version as a new service configuration. The configuration of the working version is copied from the *MANAGER* folder to the *SERVICE* folder in the repository. After restarting the instance, e-mails are processed according to the updated configuration.



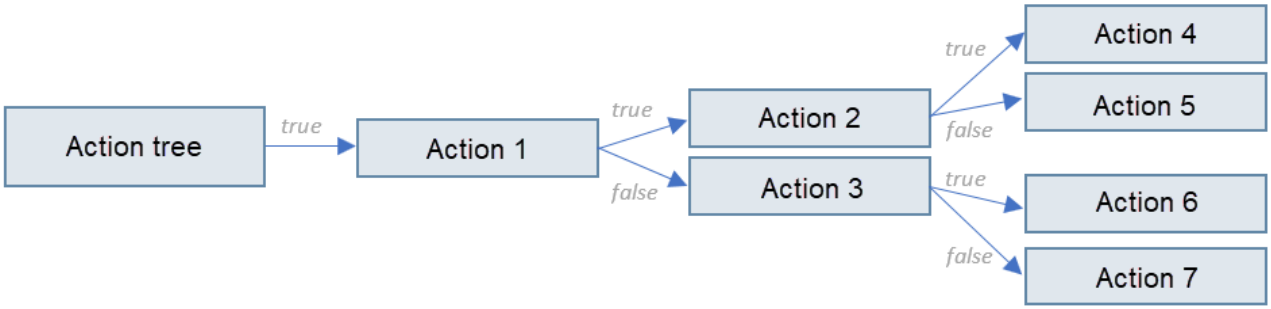
Action principle

An action always returns a binary result: `true` if successful or `false` if it fails. By linking the results to follow-up actions, you create a binary tree structure, which allows you to configure an application-specific logic.

The actions deliver functions that affect the scope of the message data, the location in the repository, and the time of transfer, using and evaluating message properties in the process. You

can create an action tree from any number of actions and call other action trees at any number of places. Processed messages are buffered while an action tree is executing.

An action tree returns a binary result, `true` or `false`, depending on the last executed action for each item processed, which determines how a single message is processed.



Configuration and execution levels

The configuration of an instance is broken down into three levels:

1 Instance level:

The basic settings of an instance, such as log behavior or default values, are processed at this level. Most importantly, this is also where the connection settings for the mailbox catalog and mailbox access during processing are defined.

2 Action tree level:

The action tree configurations are evaluated at the level below the instance level. These determine the scope of selection, which is directly related to the settings in the mailbox catalog at the instance level. This is also where you configure the basic filter settings, e.g. *entry points* or *message classes*.

3 Action level:

The action level consists of any number of individual hierarchically linked actions that change message or repository data according to their type. The order of actions in a tree determines the course of processing.

Validation always takes place while the configuration is loading. This applies both to calling the validation function, which is explicitly limited to checking the syntax of fragments and individual plausibility checks, and at the start of message processing. If the validation of the configuration fails, the messages cannot be processed.

Exchange

Autodiscover

Autodiscover is an Exchange service that determines the Exchange server that corresponds to an SMTP address. An EWS connection to the mailbox can only be established once the Exchange server has been determined. It makes no sense to directly configure Exchange servers since these can change at any time, especially in the Microsoft 365 environment.

The address of the Autodiscover service must be available in the runtime environment and automatically provided by the infrastructure. This is usually done with corresponding entries in the local Active Directory and/or DNS. With the appropriate configuration, ELO XC can therefore find and process mailboxes across domains.

For cases where automatic resolution of the e-mail domain is not possible, the ELO XC connection settings provide the option to manually specify an Autodiscover endpoint.

Process diagram

Status	Component	Action
SMTP address	API	ELO XC attempts to heuristically determine an Autodiscover endpoint based on the mail domain. It generates plausible URLs, localizes them with DNS, and then attempts to determine the Autodiscover endpoint (svc).
Autodiscover endpoint	Autodiscover	The mailbox settings (Exchange version and Exchange URL) are determined based on the SMTP address.
EWS URL	EWS	Mailbox authentication and access is possible.

EWS authentication and access rights

Authentication is required for mailbox access. The mailbox owner usually authenticates with a name and a password. For data privacy reasons, ELO XC should not store these authentication credentials. Instead, it uses a service account (ldap) or an app registration (m365) to access mailboxes. This proxy access (called *impersonation*) requires elevated privileges and must be set up and configured by an administrator accordingly.

If you are only processing a small number of mailboxes, it may also be possible to do without the impersonation function. In this case, you are able to manually configure mailbox catalogs with separate credentials.

Scenario	Configuration
Impersonation	This access method is mainly used because it is possible to process many mailboxes with one account. ELO XC uses a privileged service connection with its own account.

Scenario	Configuration
Direct access	If only a few mailboxes are processed, it is possible to configure access to each mailbox without impersonation (*individual* static mode).
Public folders	This special case requires a configured service connection with an account that has access rights to *public folders*.

Mailbox types

Catalog	user	shared	journal	premjournal	pubfolder
ldap	X	X	X		X
m365	X	X		X	X
manual	X				X

The cataloged mailboxes can be used in ELO XC in different ways. The mailbox types are based on the *RecipientTypeDetails* directory property. Some mailboxes can only be used for their intended purpose (e.g. *user mailboxes*), while others can be processed both natively and with emulation (e.g. *journal recipients*).

Mailbox type 'user'

This type represents a user mailbox and is therefore the standard case. In principle, every available mailbox can be processed as a user mailbox with the exception of *public folders* (type *pubfolder*), in which case there may be no need for advanced processing options. User mailboxes are the norm, so that emulation of *shared* and *journal* mailboxes also traces back to user mailboxes. A license is required as soon as a message in a user's mailbox is stored.

> **Information**
>
> In an on-premises installation, create user mailboxes in the Exchange admin center under *Recipients > Mailboxes > + > User mailbox*. In Microsoft 365, create them in the Microsoft 365 admin center under *Active users > Add a user*.

Mailbox type 'shared'

Shared mailboxes are group mailboxes that have their own SMTP address and are configured for access by multiple users. These users, so-called *delegates*, have automatic access to the mailbox share and can act as the owner of the shared mailbox. If this type is configured in an action tree, the messages it contains are processed for all delegates involved. This is what is known as mailbox emulation. ELO XC stores a message for each of the delegates in the repository as if the message had been found in the respective user mailbox of the delegate. Messages stored in this way require one license for each delegate.

When a shared mailbox is processed in native mode (*user* type), the delegates are not emulated. Only one message is stored for the shared mailbox but all delegates are licensed for the mailbox.

Information

In an on-premises installation, create *shared mailboxes* in the Exchange admin center under *Recipients > Shared > +*. In Microsoft 365, create them in the Exchange admin center under *Mailboxes > Add a shared mailbox*.

Mailbox type 'journal'

The *journal* mailbox type is a journal recipient that must be configured specifically for an Exchange mailbox store. These mailboxes will then automatically receive all messages related to the corresponding mailbox store. The original message is attached to an *envelope* message. This *envelope* contains a list of recipients, which are then interpreted as user mailboxes during processing. This means that this mailbox type is also emulated. Shared mailboxes are also resolved into user mailboxes or their delegates. Each emulated mailbox uses up one license.

If a journal mailbox is processed in native mode (*user* type), the recipients are not resolved, i.e. the mailboxes are not emulated. Only the actual message, the envelope with the original message attached, is stored for the journal mailbox. In this case, all potential journal recipients are automatically licensed.

In addition, all mailbox shares that can also be recipients of the envelope are also resolved.

Please note

BCC recipients are recognized according to the Exchange default method and processed symmetrically if necessary. If BCC recipients appear in recipient lists, they are also retained on storage. If they show up on journal envelopes, they are also processed as part of mailbox emulation. If you don't want this to happen, you need to set *InstanceDef.Addresses.UseBcc* to `false`.

Information

In an on-premises installation, create journal mailboxes in the Exchange admin center under *Servers > Databases > Edit > Maintenance > Journal recipient*.

Mailbox type 'premjournal'

Premium journaling is no longer available for Microsoft Office 365. Journaling in Microsoft 365 is now configured as a transport rule that ensures that journal messages are automatically sent to an external mailbox. This mailbox can be on a locally hosted Exchange server or can be an IMAP mailbox. The connection settings in ELO XC must be configured to use the catalog type *m365* so that the emulated mailboxes are known. The journal mailboxes are configured under *Static authentication*. Select *premjournal* as static mode. Enter the mailboxes as static authentications. As with the *journal* type, each emulated mailbox uses up one license.

> **Information**
>
> In Microsoft 365, create premium journaling mailboxes in the Exchange admin center under *Microsoft Purview > Data lifecycle management > Exchange (legacy) > Journal rules > New rule*.

Mailbox type 'pubfolder'

To process *public folders*, you must select the *pubfolder* type. The configured SMTP address is usually that of the processing account, since it makes sense to grant the ELO XC service user the access rights for all *public folders*. Access to the *public folders* is assigned automatically when establishing the connection to the service user's mailbox. When storing messages from public folders, one license is used up.

> **Information**
>
> In an on-premises installation, create *public folders* in the Exchange admin center under *Public folders*. In Microsoft 365, create them in the Exchange admin center under *Public folders > Add a public folder*.

Mailbox type 'filter'

This type is an exception. It is not used in combination with an SMTP address, but with the name of a configured catalog filter.

Special mailboxes

The Exchange server also has other mailbox types that you can include using the instance configuration. These include *system mailboxes*, *room mailboxes*, and *resource mailboxes*. These mailboxes are not shown by default. Like user mailboxes, these mailboxes use up one license each when storing messages.

Archive mailboxes are secondary user mailboxes used by Exchange to archive old messages. These can be included by enabling an option in the instance configuration. Since repository mailboxes are mapped to normal user mailboxes, no additional licenses are used up.

Prioritization

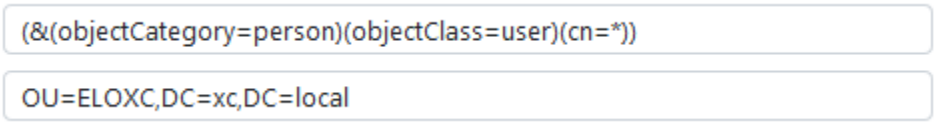
The mailboxes are fully resolved when an action tree starts processing. An internal processing type is defined according to the configuration and duplicate mailbox addresses or SMTP addresses are ignored. If duplicates with different mailbox types are found, the address with the processing type that has a higher priority takes precedence. The following list shows the priority level of the internal types in descending order:

Priority	Processing type	Configuration type
1	Public folders	pubfolder

Priority	Processing type	Configuration type
2	Journal	journal
3	Premium journal	premjournal
4	User mailbox	user
5	Shared mailbox	shared
6	Room mailbox	user
7	Resource mailbox	user
8	System mailbox	user

Mailbox catalogs

Each catalog is determined by filters, and each catalog type has an internal default filter that is automatically used if configuration settings are missing. A filter data set consists of filter names, the filter (value), and a search range (*container*), which is only used for the *ldap* type.



Filter name: Stored internally on every match for use in action trees or list templates.

Value: Value for finding matches, depends on the catalog type.

Search range: Is only used for LDAP.

You can configure as many filters as you like. The mailboxes are identified by their SMTP addresses and added to the mailbox catalog along with the corresponding filter.

The following table contains the default settings for each catalog type.

Catalog	Catalog filter (value)
ldap	`(&(objectCategory=person)(objectClass=user)(cn=*)).`
m365	*

Type 'ldap'

Special permissions are usually not required for LDAP queries. If this is the case, however, the local administrator must configure read access to the local AD.

The special search range option is only available for LDAP catalogs. These are mainly required for selecting organizational units (OU).

Type 'm365'

Mailbox catalogs for Microsoft 365 use a remote PowerShell with the module for Exchange Online (EXO module). This connection type requires app registration and authentication using a certificate.

The permission to use *Get-Mailbox* (and *Get-MailboxPermission* for *SharedMailboxes*) to request available mailboxes is configured in the API permissions of the app registration.

If this catalog type is used with *premjournal* static mode, the retrieved mailboxes can resolve static journal mailbox envelopes. This is because Microsoft 365 does not offer journal mailboxes, but redirects to external mailboxes with *premium journaling*.

Type 'manual'

Static catalogs contain a list of manually configured mailboxes and the authentication data that can be processed. They therefore don't need catalog requests or any filters.

If the *impersonated* static mode is configured, only the SMTP address from the list is used to get mailboxes. The service connection data is used for authentication.

IMAP servers can only be processed with this catalog type. In this case, static mode must be configured as *individual* and there must be a valid server and port. The static mode *premjournal* is only allowed with the *m365* catalog type.

Mailbox folders

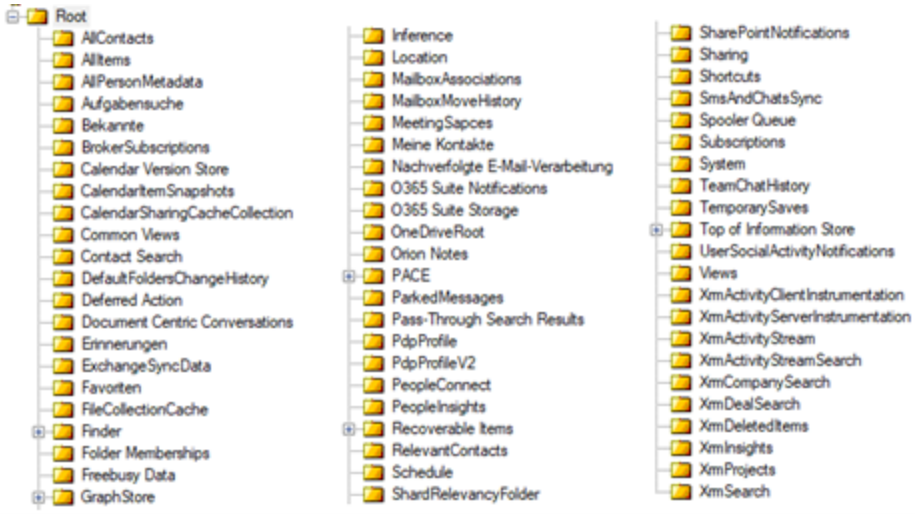
Microsoft Exchange uses various predefined folders, e.g. *Inbox*, *Sent Items*, or *Deleted Items*. These folders are managed and identified with technical names at the internal level, whereas the localized view in Microsoft Outlook depends on the first user authentication with Outlook. To enable processing independently of the language, ELO XC uses the following variables for determination of folder names:

Technical name	ELO XC variable	Group
MsgFolderRoot	*{%MSGROOT}*	Standard selection
Calendar	*{%APPOINTMENT}*	Standard selection
Contacts	*{%CONTACT}*	Standard selection
DeletedItems	*{%DELITEMS}*	Standard selection
Inbox	*{%INBOX}*	Standard selection
Notes	*{%NOTES}*	Standard selection
SentItems	*{%SENTMAIL}*	Standard selection
Tasks	*{%TASK}*	Standard selection
Drafts	*{%DRAFTS}*	Extended selection
Journal	*{%JOURNAL}*	Extended selection
VoiceMail	*{%VOICE}*	Extended selection
JunkEmail	*{%JUNK}*	Extended selection
SearchFolders	*{%SEARCH}*	Extended selection
RecipientCache	*{%RCPTCACHE}*	Synchronized folders
QuickContacts	*{%QUICKCONTACTS}*	Synchronized folders
ConversationHistory	*{%CONVHISTORY}*	Synchronized folders

Technical name	ELO XC variable	Group
MyContacts	*{%MYCONTACTS}*	Synchronized folders
IMContactList	*{%IMCONTACTS}*	Synchronized folders
PeopleConnect	*{%PEOPLECONNECT}*	Synchronized folders
Favorites	*{%FAVORITES}*	Synchronized folders
AllContacts	*{%ALLCONTACTS}*	Synchronized folders
RecoverableItemsRoot	*{%RECITEMSROOT}*	Recovery
RecoverableItemsDeletions	*{%RECITEMSDEL}*	Recovery
RecoverableItemsVersions	*{%RECITEMSVER}*	Recovery
RecoverableItemsPurges	*{%RECITEMSPURG}*	Recovery
ArchiveRoot	*{%ARCROOT}*	Archive mailbox
ArchiveMsgFolderRoot	*{%ARCMSGROOT}*	Archive mailbox
ArchiveInbox	*{%ARCINBOX}*	Archive mailbox
ArchiveDeletedItems	*{%ARCDELITEMS}*	Archive mailbox
ArchiveRecoverableItemsRoot	*{%ARCRECITEMSROOT}*	Archive mailbox recovery
ArchiveRecoverableItemsDeletions	*{%ARCRECITEMSDEL}*	Archive mailbox recovery
ArchiveRecoverableItemsVersions	*{%ARCRECITEMSVER}*	Archive mailbox recovery
ArchiveRecoverableItemsPurges	*{%ARCRECITEMSPURG}*	Archive mailbox recovery

These variables are available when the corresponding group is selected under Scaling in the instance configuration. With the default settings, all folder groups are selected, but this affects performance because these known folders must be determined according to the configuration when ELO XC starts processing a mailbox. In many cases, however, there is no need to fully determine the known folders, and the process can usually be limited to the standard structure of a mailbox.

To understand the different folder groups, look at the internal folder structure of a mailbox in Microsoft Exchange:



You will recognize a number of folders that a Microsoft Outlook user does not see. Messages are usually only stored in the *Top of Information Store* folder. A secondary mailbox or archive mailbox has a similarly complex structure.

The root folders of the structures containing messages are called:

- MsgFolderRoot {%MSGROOT}
- RecoverableItemsRoot {%RECITEMSROOT}
- ArchiveMsgFolderRoot {%ARCMSGROOT}
- ArchiveRecoverableItemsRoot {%ARCRECITEMSROOT} *MsgFolderRoot* is the entry point of the visible folder structure of Outlook, where most of the known and language-dependent folders are stored. If a mailbox also has an *archive mailbox*, the folders visible in it can be found below *ArchiveMsgFolderRoot* and are also language-dependent. The roots of the *recovery folders* are hidden and are not translated but retain their English names.

In addition to these four main entry points, ELO XC uses the *Public folders* root, which cannot be included or excluded in the instance configuration, but depends on the mailbox processing settings in the action tree and the permission settings of its mailbox owner.

The *Synchronization* folder group, on the other hand, has no root of its own, as its only purpose is to split the visible Outlook folders even further. Whereas the default selection is always used and

cannot be excluded through the configuration when ELO XC starts processing the mailbox, the *Synchronization* group allows you to extent the default selection. The *Archive mailbox recovery* group is only available when both the *Recovery* group and the *Archive mailbox* group are selected.

The roots are each a main entry point into an independent folder structure and have their own ELO XC variable. The table of variables shows additional variables that allow targeted entry points into child folder structures. For example, if you only want to process *Inbox* messages and all folders below it, it is sufficient to configure *{%INBOX}* as a recursive entry point. If you want to process the entire mailbox with all its folders, use *{%MSGROOT}* as the recursive entry point. If you use non-recursive entry points, only the configured folders are processed. The folders below them are ignored.

Under Scaling in the instance configuration, you can specify the display names of the five possible roots. The default values are only a recommendation based on standard use cases. A configurable name is required because the roots are not in different languages and are always part of the folder path when the *EloBoxPath* property is used to create the repository path. For example, whereas the Inbox folder is perceived as *\Inbox* due to its visibility in Outlook, this need not apply to the inbox of the *archive mailbox*. It can be assumed that this folder should not be used as *\Inbox* as well, but as Archive\Inbox to better distinguish between stored messages by path. However, there is no reason why you can't configure ** and display both roots identically in the filing path.

The process of folder determination and the provision of entry point variables at the start of processing is as follows:

The known folders are queried by Exchange according to the Scaling settings. The default selection cannot be bypassed. If an option is not selected, its root is not available.

ELO XC uses the available roots to build the internal folder cache and select which folders to include based on the configured name filters, entry points, and the number of items in each folder.

Messages

Properties

Messages in Exchange consist of a variety of properties. *PidTagSubject*, for example, is the property name of the message subject. The *PidTagMessageDeliveryTime* property contains the time of message delivery. There are more than 1,000 defined Exchange properties in total, of which only a fraction are relevant for processing Exchange items in ELO XC.

Which properties exist and can be used depends on the specific use case. The availability of properties depends on the message class. There are usually properties that overlap but some properties are only available for a specific class. For example, while e-mails and calendar entries both use the *PidTagSubject* property, it makes sense that only calendar entries contain the *PidTagStartDate* or *PidTagEndDate* propery.

Processing in ELO XC is based to a large extent on message properties. They are used during selection, validation, or differentiation between cases, or to determine metadata when storing documents in the repository. They can also be changed as a result of processing. In addition to the properties determined by Exchange (*PidTags*), ELO also uses own properties that are calculated at runtime and contain useful additional information that are not or cannot exist in the regular Exchange properties.

ELO XC Manager includes utility functions to display the names and meanings of the properties.

Message classes

The supported message classes are based on the classes officially documented by Microsoft. These classes each have a reliable minimum set of properties that can be processed by ELO XC. User-defined classes may prevent processing in certain circumstances if items of these classes do not contain the minimum set, which means that ELO XC cannot process the messages. It is therefore recommended to use mainly derivatives of IPM.Note as user-defined classes. In any case, you should compare user-defined classes with the officially documented classes and check for missing properties before processing with ELO XC.

The currently integrated classes are listed in the following table.

Message class	File extension
IPM.Appointment	.ics
IPM.Conflict.Message	.vcf
IPM.Contact	.vcf
IPM.Document*	.eml
IPM.DistList	.txt
IPM.InfoPathForm*	.eml
IPM.Note	.eml
IPM.Note.IMC.Notification	.eml

Message class	File extension
IPM.Note.Mobile.MMS	.txt
IPM.Note.Mobile.SMS	.txt
IPM.Note.Microsoft.Voicemail	.txt
IPM.Note.Microsoft.Voicemail.UM	.txt
IPM.Note.Microsoft.Voicemail.UM.CA	.txt
IPM.Note.Microsoft.Fax	.txt
IPM.Note.Microsoft.Missed	.txt
IPM.Note.Microsoft.Conversation	.txt
IPM.Note.NotSupportedICal	.txt
IPM.Note.Rules.OofTemplate.Microsoft	.eml
IPM.Note.Rules.ExternalOofTemplate.Microsoft	.eml
IPM.Note.Rules.ReplyTemplate.Microsoft	.eml
IPM.Note.SMIME	.eml
IPM.Note.SMIME.MultipartSigned	.eml
IPM.Note.StorageQuotaWarning	.eml
IPM.Note.StorageQuotaWarning.Warning	.eml
IPM.Note.StorageQuotaWarning.Send	.eml
IPM.Note.StorageQuotaWarning.SendReceive	.eml
IPM.Post	.txt
IPM.Post.RSS	.txt
IPM.Outlook.Recall	.txt
IPM.Recall.Report	.eml
IPM.Recall.Report.Failure	.eml
IPM.Recall.Report.Success	.eml
IPM.Sharing	.txt
IPM.Schedule.Meeting.Canceled	.ics
IPM.Schedule.Meeting.Request	.ics
IPM.Schedule.Meeting.Resp.Neg	.ics
IPM.Schedule.Meeting.Resp.Pos	.ics
IPM.Schedule.Meeting.Resp.Tent	.ics
IPM.Schedule.Meeting.Notification	.ics
IPM.Schedule.Meeting.Notification.Forward	.ics
IPM.StickyNote	.txt
IPM.Task	.txt
IPM.TaskRequest	.txt
IPM.TaskRequest.Accept	.txt
IPM.TaskRequest.Decline	.txt

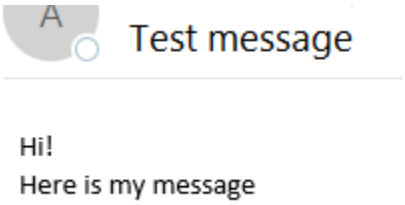
Message class	File extension
IPM.TaskRequest.Update	.txt
REPORT.IPM.Note.Delayed.DR	.eml
REPORT.IPM.Note.Relayed.DR	.eml
REPORT.IPM.Note.NDR	.eml
REPORT.IPM.Note.DR	.eml
REPORT.IPM.Note.DR.NDR	.eml
REPORT.IPM.Note.NDR.NDR	.eml
REPORT.IPM.Note.Expanded.DR	.eml
REPORT.IPM.Note.Delayed	.eml
REPORT.IPM.Note.IPNNDR	.eml
REPORT.IPM.Note.IPNNRN	.eml
REPORT.IPM.Note.IPNRN	.eml
REPORT.IPM.Note.SMIME.NDR	.eml
REPORT.IPM.Note.SMIME.DR	.eml
REPORT.IPM.Note.SMIME.MultipartSigned.NDR	.eml
REPORT.IPM.Note.SMIME.MultipartSigned.DR	.eml
REPORT.IPM.Note.SMIME.MultipartSigned.IPNRN	.eml
REPORT.IPM.Recall.Report.Failure.NDR	.eml
REPORT.IPM.SharingIPNRN	.txt
REPORT.IPM.Schedule.Meeting.Canceled.DR	.eml
REPORT.IPM.Schedule.Meeting.Canceled.NDR	.eml
REPORT.IPM.Schedule.Meeting.Request.DR	.eml
REPORT.IPM.Schedule.Meeting.Request.NDR	.eml
REPORT.IPM.Schedule.Meeting.Request.IPNNRN	.eml
REPORT.IPM.Schedule.Meeting.Request.IPNRN	.eml
REPORT.IPM.Schedule.Meeting.Resp.Neg.DR	.eml
REPORT.IPM.Schedule.Meeting.Resp.Neg.NDR	.eml
REPORT.IPM.Schedule.Meeting.Resp.Neg.IPNRN	.eml
REPORT.IPM.Schedule.Meeting.Resp.Pos.DR	.eml
REPORT.IPM.Schedule.Meeting.Resp.Pos.NDR	.eml
REPORT.IPM.Schedule.Meeting.Resp.Pos.IPNRN	.eml
REPORT.IPM.Schedule.Meeting.Resp.Tent.DR	.eml
REPORT.IPM.Schedule.Meeting.Resp.Tent.NDR	.eml
REPORT.IPM.Schedule.Meeting.Resp.Tent.IPNRN	.eml
REPORT.IPM.TaskRequest.Accept.DR	.txt
REPORT.IPM.TaskRequest.Accept.NDR	.txt
REPORT.IPM.TaskRequest.Decline.DR	.txt

Message class	File extension
REPORT.IPM.TaskRequest.Decline.NDR	.txt
REPORT.IPM.TaskRequest.NDR	.txt
REPORT.IPM.TaskRequest.Update.DR	.txt
REPORT.IPM.TaskRequest.Update.NDR	.txt

Message body

Message bodies can be in text format and HTML format. Microsoft Exchange also supports RTF, which ELO XC does not support due to its age and support issues with EWS. Processing the message body in this format is therefore not recommended. The *Stubbing* action does not work with this format.

If you want ELO XC to change or process message bodies, you have to make sure to use the correct format in some actions. For example, if you want to stub part of the message, the body format is important.



Microsoft Outlook (and any other e-mail client) will display a formatted message, not the source text of the format. If you want ELO XC to replace `Hi! Here is` with `This is` to get the result `This is my message`, you have to pay attention to the body format because it significantly influences the search pattern.

If your message is in text format, you can search for `Hi! Here is` using a regular expression. However, if the message is in HTML format, you must consider that ELO XC needs to process the source text instead of the displayed text:

```
<p class="MsoNormal">Hi!<o:p></o:p></p>
<p class="MsoNormal">This is my message<o:p></o:p></p>
```

This is why it is important that you create the configuration in line with the format of the message body if you want to change it. You need to read out the property *PidTagNativeBody*. If the property value is 1, the message is in text format. If the value is 3, the message is in HTML format, and if the value is 2, it uses RTF. If this property is not read out and, for example, the *Match/Replace* action is used, this could destroy the message format, making it impossible to display the affected messages either partially or completely.

However, if you use the *Stubbing* action, you have to configure a template for both formats, which ELO XC automatically uses.

Here is a summary of how the message format influences the various actions:

- Export: Inline attachments in HTML format are not extracted. Only file attachments are extracted.
- Delete: Inline attachments in HTML format are not deleted. Only file attachments are deleted.
- Stubbing: Stubbing templates require the configuration of both formats, i.e. text and HTML. If variables are used that generate references (ELO links or other links), inline attachments are ignored.
- Match/Replace: The matching pattern must take the message format into account, otherwise there is the risk that the source text will be destroyed.

Duplicate e-mails

E-mails sent to multiple recipients can result in identical MIME files in at least two recipient mailboxes, called *duplicate e-mails*.

Duplicate e-mail: *Duplicate e-mails* are MIME files delivered by the e-mail server that have identical pairs of byte sequences.

Multiple recipient e-mail: When an e-mail is sent to more than one recipient, we refer to this as a *multiple recipient e-mail*.

Duplicate e-mail and multiple recipient e-mail: Sending an e-mail to more than one recipient can result in duplicate e-mails. But this doesn't necessarily happen automatically. It is not possible to deduce the number of duplicate e-mails from the number of multiple recipients.

Consequences

Duplicate e-mails can only be recognized using the full content of two MIME files. This means that there is no way to recognize duplicates unless you compare the content of two MD5 values. Recognizing duplicates would seem to make sense in terms of saving storage space. However, this is based on the false assumption that you can only avoid redundant storage if duplicate copies are only stored once. The correct method is to configure an MD5 path as the physical storage path, which automatically ensures that duplicate files physically exist only once. ELO calls this method the *passive duplicate check*. Attempting to recognize e-mail duplicates before they are stored is an additional, unnecessary step that also reduces performance.

The message properties *PidTagInternetMessageId* and *PidTagSearchKey* provide a direct option to detect multiple recipient e-mails. The pseudo-property *EloSearchKey* initially tries to use *PidTagInternetMessageId*, and in the absence of this property, tries to find *PidTagSearchKey*.

Before you use these properties, you need to take note of the following:

- Determining multiple recipient e-mails is not an adequate substitute for recognizing duplicate e-mails. Pairs of multiple recipient e-mails are not always duplicate e-mails.
- The two properties not available for all Exchange items and cannot therefore be used in all cases to recognize multiple recipient e-mails.
-

Identifying all mailboxes associated with a multiple recipient e-mail can also reveal BCC recipients. It is therefore necessary to consider in each scenario whether privacy regulations are being breached.

Licensing

ELO XC is activated using the license key in ELOam. The maximum number of available ELO XC licenses corresponds to the *usercount4* value. ELO XC calculates one license for each processed or emulated mailbox when a message is stored and frees up the license after about 30 days if no new messages are stored during this period. The available number of licenses can be altered by importing a new license key. In this case, the date of issue applies.

If the number of available licenses is exceeded, ELO XC will continue to operate up to a tolerable limit above the configured threshold. If usage exceeds this tolerance limit, ELO XC is deactivated and cannot be used until sufficient licenses are available. One way to do this is to reuse licenses that are automatically freed up after a certain period or to register a new license key.

> **Please note**
>
> Group mailboxes such as *Journal recipients* or *Shared mailboxes* are always fully resolved with automatic licensing in order to determine the license usage. For example, if there are 20 mailboxes in a mailbox database with one journal recipient, 20 licenses will be used. The same applies for *Shared mailboxes*. If *Shared mailboxes* are also associated with a *Journal recipient*, the delegates of a *Shared mailbox* also use one license each in addition to the user mailboxes of the journal.

Premium journaling is an exception since it is based on transport rules and not a mailbox database. In this case, there is no limit. All recipients found in the catalog are licensed.

Tolerance limit

Since in practice, especially with a small number of licenses, mailboxes accidentally configured incorrectly immediately causes the number of available licenses to be exceeded, the default 20% tolerance limit is only applied starting with 20 mailbox licenses. The following table shows the license tolerance of the purchased licenses.

Purchased	Effective	Tolerance [%]
1	5	400
2	6	200
3	7	133
4	8	100
5	9	80
6	10	66
7	11	57
8	12	50
9	13	44
10	14	40

Purchased	Effective	Tolerance [%]
11	15	36
12	16	33
13	17	30
14	18	28
15	19	26
16	20	25
17	21	23
18	22	22
19	23	21
20	24	20