# Python Kopano

***Release 8.2.0***

**Kopano BV**

September 22, 2017

Contents

This document describes how to use Python-Kopano, a “Pythonic” abstraction layer above Python-MAPI.

CHAPTER 1

# Introduction

While Python-MAPI provides complete Python bindings to interface with Kopano Core, it is not very easy to use. Most importantly, this is because it really operates at the MAPI level, which is to say, it is very low-level and often strange. When looking at programs written using Python-MAPI, one easily gets the impression that a much nicer abstraction layer should be possible.

Python-Kopano aims to provide a high-level, more "Pythonic", abstraction layer above Python-MAPI. While internally it still uses Python-MAPI, it tries to hide as much of the details as possible from the developer, while still allowing her to dive down into MAPI if needed. Simply put, its main goal is to try and bring the productivity often associated with the Python language itself to developing against Kopano Core.

More specifically, some of the goals of Python-Kopano are as follows:

- To be fully object-oriented, pythonic, layer above MAPI
- To be usable for many common system administration tasks
- To provide full access to the underlying MAPI layer if needed
- To return all text as unicode strings
- To return/accept binary identifiers in readable (hex-encoded) form
- To raise well-described exceptions if something goes wrong

## 1.1 Target audience

This document is mainly targeted at Python developers looking to interact with Kopano Core. MAPI knowledge is not required in most cases, although it can be useful to have at some prior experience with basic MAPI concepts such as properties.

CHAPTER 2

# Getting started

## 2.1 Connecting to the server

To be able to do anything, of course we first need to connect to a Kopano server. For non-interactive use, we recommend to do this as follows:

```
import kopano

server = kopano.Server()
```

If the Kopano server is on another system, we can specify how to connect to it as follows:

```
import kopano

server = kopano.Server(
    server_socket='<https://remoteip:237/kopano>',
    sslkey_file='</etc/kopano/ssl/server.pem>',
    sslkey_pass='<password>'
)
```

Python-kopano will first look at any specified settings, then /etc/kopano/admin.cfg, and if there is nothing, it will fall back to the local server.

## 2.2 Opening a store

Now that we are connected to a Kopano server, let us open a certain user store.

```
store = server.user('<username>').store
```

One can also use the following shortcut, so that an explicit server object is not needed. Such shortcuts are especially meant to be used in interactive-mode (that is, from a Python shell). In non-interactive mode, we recommend to always use an explicit server object.

```
store = kopano.user('<username>').store
```

## 2.3 Opening a folder

A folder can be opened from a store using a folder name (or path):

```
inbox = store.folder('Inbox')
other_folder = store.folder('Inbox/some_subfolder')
```

For special folders, there is usually a simpler way:

```
inbox = store.inbox
junk = store.junk
```

To know the number of items in a folder:

```
count = inbox.count
```

Note that it is usually not needed to use an explicit store object. If there's no ambiguity, we can directly use the user object.

```
inbox = user.inbox
```

## 2.4 Looping over folder items and their properties

Let's loop over all items in a certain folder:

```
for item in user.junk:
    print(item.received, item.subject)
```

A MAPI item is basically just a collection of MAPI properties. Here's how to loop over them:

```
for prop in item:
    print(prop, prop.value)
```

Let's access a specific property, which holds the subject of the respective item:

```
from MAPI.Tags import PR_SUBJECT_W # python-mapi
print(item.prop(PR_SUBJECT_W).value)
```

## 2.5 Sending an email

```
item = user.outbox.create_item(
    to='john@doe.com',
    subject='test email',
    body='Hi john!')
item.send()
```

## 2.6 Where to go from here

After going through these basic examples, we hope our *Reference Guide* can help you further.

CHAPTER 3

# Reference Guide

## 3.1 Address

**class** kopano.**Address**
Address class

**email**
Email address

**name**
Full name

**props**()
Return all properties.

## 3.2 Attachment

**class** kopano.**Attachment**
Attachment class

**create_prop**(*proptag*, *value*, *proptype=None*)
Create property with given proptag

**Parameters**

- **proptag** – MAPI property tag
- **value** – property value (or a default value is used)

**data**
Binary data

**filename**
Filename

**get_prop**(*proptag*)
Return property with given proptag or *None* if not found

**Parameters** **proptag** – MAPI property tag

**get_value**(*proptag*)
Return property value for given proptag or *None* if property does not exist

**Parameters** **proptag** – MAPI property tag

**mimetype**
Mime-type

**prop**(*proptag*, *create=False*, *proptype=None*)
Return `property` with given property tag

**Parameters**

- **proptag** – MAPI property tag
- **create** – create property if it doesn't exist

**props**(*namespace=None*)
Return all `properties`

**set_value**(*proptag*, *value*)
Set `property` value for given proptag, creating the property if it doesn't exist

**size**
Size

**value**(*proptag*)
Return `property` value for given proptag

## 3.3 Autoaccept

**class** kopano.**AutoAccept**
AutoAccept class

**conflicts**
Conflicting appointments are accepted.

**enabled**
Auto-accept is enabled.

**recurring**
Recurring appointments are accepted.

## 3.4 Company

**class** kopano.**Company**
Company class

**admin**
Company `administrator` in multi-tenant mode.

**create_public_store**()
Create company public `store`.

**create_user**(*name*, *password=None*)
Create a new `user` within the company.

**Parameters** **name** – user name

**get_user**(*name*)
Return `user` with given name or *None* if not found.

**Parameters** **name** – user name

**group**(*name*)
Return `group` with given name.

**Parameters** **name** – group name

**groups**()
Return all `groups` within the company.

**hidden**
The company is hidden from the addressbook.

**hook_public_store**(*store*)
Hook company public `store`.

**Parameters** **store** – store to hook as public store

**name**
Company name.

**public_store**
Company public `store`.

**quota**
Company `Quota`.

**store**(*guid*)
Return `store` with given GUID.

**stores**()
Return all company `stores`.

**unhook_public_store**()
Unhook company public `store`.

**user**(*name*, *create=False*)
Return `user` with given name.

**Parameters**

- **name** – user name
- **create** – create user if it doesn't exist (default False)

**users**(*parse=True*, *system=False*)
Return all `users` within company.

## 3.5 Delegation

**class** kopano.**Delegation**
Delegation class

**send_copy**
Delegate receives copies of meeting requests.

## 3.6 Errors

**class** kopano.**Error**

**class** kopano.**ConfigError**

**class** kopano.**DuplicateError**

**class** kopano.**NotFoundError**

**class** kopano.**LogonError**

**class** kopano.**NotSupportedError**

## 3.7 Folder

**class** kopano.**Folder**
Folder class

**archive_folder**
Archive Folder or *None* if not found

**associated**
Associated folder containing hidden items

**container_class**
Property which describes the type of items a folder holds, possible values * IPF.Appointment * IPF.Contact * IPF.Journal * IPF.Note * IPF.StickyNote * IPF.Task

https://msdn.microsoft.com/en-us/library/aa125193(v=exchg.65).aspx

**copy**(*objects*, *folder*, *_delete=False*)
Copy items or subfolders to folder.

**Parameters**

- **objects** – The items or subfolders to copy
- **folder** – The target folder

**count**
Number of items in folder

**create_prop**(*proptag*, *value*, *proptype=None*)
Create property with given proptag

**Parameters**

- **proptag** – MAPI property tag
- **value** – property value (or a default value is used)

**delete**(*objects*, *soft=False*)
Delete items, subfolders, properties or permissions from folder.

**Parameters**

- **objects** – The object(s) to delete
- **soft** – In case of items or folders, are they soft-deleted

**empty**(*recurse=True*, *associated=False*)
Delete folder contents

**Parameters**

- **recurse** – delete subfolders
- **associated** – delete associated contents

**entryid**
Folder entryid

**folder**(*path=None*, *entryid=None*, *recurse=False*, *create=False*)
Return Folder with given path or entryid; raise exception if not found

**Parameters** **key** – name, path or entryid

**folders**(*recurse=True*)
Return all sub-folders in folder

**Parameters** **recurse** – include all sub-folders

**get_folder**(*path=None*, *entryid=None*)
Return folder with given name/entryid or *None* if not found

**get_prop**(*proptag*)
Return `property` with given proptag or *None* if not found

**Parameters** **proptag** – MAPI property tag

**get_value**(*proptag*)
Return `property` value for given proptag or *None* if property does not exist

**Parameters** **proptag** – MAPI property tag

**item**(*entryid=None*, *sourcekey=None*)
Return `Item` with given entryid or sourcekey

**Parameters**

- **entryid** – item entryid
- **sourcekey** – item sourcekey

**items**(*restriction=None*)
Return all `items` in folder, reverse sorted on received date

**move**(*objects*, *folder*)
Move items or subfolders to folder

**Parameters**

- **objects** – The items or subfolders to move
- **folder** – The target folder

**name**
Folder name

**parent**
Return `parent` or None

**permission**(*member*, *create=False*)
Return `permission` for user or group set for this folder.

**Parameters**

- **member** – user or group
- **create** – create new permission for this folder

**permissions**()
Return all `permissions` set for this folder.

**prop**(*proptag*, *create=False*, *proptype=None*)
Return `property` with given property tag

**Parameters**

- **proptag** – MAPI property tag
- **create** – create property if it doesn't exist

**props**(*namespace=None*)
Return all `properties`

**set_value**(*proptag*, *value*)
Set `property` value for given proptag, creating the property if it doesn't exist

**size**
Folder size

**state**
Current folder state

**subfolder_count**
Number of direct subfolders

**sync** (*importer*, *state=None*, *log=None*, *max_changes=None*, *associated=False*, *window=None*, *begin=None*, *end=None*, *stats=None*)
Perform synchronization against folder

**Parameters**

- **importer** – importer instance with callbacks to process changes
- **state** – start from this state; if not given sync from scratch

**Log** logger instance to receive important warnings/errors

**unread**
Number of unread items

**value** (*proptag*)
Return `property` value for given proptag

## 3.8 FreeBusy

**class** kopano.**FreeBusyBlock**
FreeBusyBlock class

**class** kopano.**FreeBusy**
FreeBusy class

**blocks** (*start=None*, *end=None*)
Freebusy blocks

**Parameters**

- **start** – start of period
- **end** – end of period

**publish** (*start=None*, *end=None*)
Publish freebusy data

**Parameters**

- **start** – start of period
- **end** – end of period

## 3.9 Group

**class** kopano.**Group**
Group class.

**add_send_as** (*user*)
Add `user` to send-as list.

**Parameters** **user** – user to add

**add_user** (*user*)
Add `user` to group.

**Parameters** **user** – user to add

**email**
Group email address.

**groupid**
Group identifier.

**groups**()
Return all groups in group.

**hidden**
The group is hidden from the addressbook.

**members**(*groups=True*, *users=True*)
Return all members in group (users or groups).

**name**
Group name.

**remove_send_as**(*user*)
Remove user from send-as list.

**Parameters** **user** – user to remove

**remove_user**(*user*)
Remove user from group.

**Parameters** **user** – user to remove

**send_as**()
Return users in send-as list.

**users**()
Return all users in group.

## 3.10 Item

**class** kopano.**Item**
Item class

**attachments**(*embedded=False*)
Return item attachments

**Parameters** **embedded** – include embedded attachments

**body**
Item body

**body_type**
original body type: 'text', 'html', 'rtf' or 'None' if it cannot be determined

**copy**(*folder*, *_delete=False*)
Copy item to folder; return copied item

**Parameters** **folder** – target folder

**create_attachment**(*name*, *data*)
Create a new attachment

**Parameters**

- **name** – the attachment name
- **data** – string containing the attachment data

**create_item**(*message_flags=None*)
Create embedded item

**create_prop**(*proptag*, *value*, *proptype=None*)
Create property with given proptag

**Parameters**

- **proptag** – MAPI property tag

- **value** – property value (or a default value is used)

**delete**(*objects*)
Delete properties or attachments from item.

**Parameters** **objects** – The object(s) to delete

**eml**(*received_date=False*)
Return .eml version of item

**entryid**
Item entryid

**folder**
Parent `Folder` of an item

**from_**
From `Address`

**get_prop**(*proptag*)
Return `property` with given proptag or *None* if not found

**Parameters** **proptag** – MAPI property tag

**get_value**(*proptag*)
Return `property` value for given proptag or *None* if property does not exist

**Parameters** **proptag** – MAPI property tag

**header**(*name*)
Return transport message header with given name

**headers**()
Return transport message headers

**html**
HTML representation

**importance**
Importance

**items**(*recurse=True*)
Return all embedded `items`

**move**(*folder*)
Move item to folder; return moved item

**Parameters** **folder** – target folder

**name**
Item (display) name

**normalized_subject**
Normalized item subject

**private**
PidLidPrivate - hint to hide the item from other users

**prop**(*proptag*, *create=False*, *proptype=None*)
Return `property` with given property tag

**Parameters**

- **proptag** – MAPI property tag
- **create** – create property if it doesn't exist

**props**(*namespace=None*)
Return all `properties`

**read**
Return boolean which shows if a message has been read

**received**
Datetime instance with item delivery time

**recipients** (*_type=None*)
Return recipient `addresses`

**reminder**
PidLidReminderSet - Specifies whether a reminder is set on the object

**rtf**
RTF representation

**sender**
Sender `Address`

**set_value** (*proptag*, *value*)
Set `property` value for given proptag, creating the property if it doesn't exist

**size**
Item size

**sourcekey**
Item sourcekey

**stubbed**
Is item stubbed by archiver?

**subject**
Item subject

**text**
Plain text representation

**value** (*proptag*)
Return `property` value for given proptag

## 3.11 MeetingRequest

**class** kopano.**MeetingRequest**
MeetingRequest class

**accept** (*tentative=False*, *response=True*, *add_bcc=False*)
Accept meeting request

**Parameters**

- **tentative** – accept tentatively?
- **response** – send response message?

**basedate**
Exception date

**calendar**
Respective `calendar` (possibly in delegator store)

**calendar_item**
Global calendar item `item` (possibly in delegator store)

**decline** (*message=None*, *response=True*)
Decline meeting request

**Parameters** **response** – send response message?

**process_cancellation**(*delete=False*)
Process meeting request cancellation

**Parameters** **delete** – delete appointment from calendar

**process_response**()
Process meeting request response

**processed**
Has the request/response been processed

**track_status**
Track status

**update_counter**
Update counter

## 3.12 OutOfOffice

**class** kopano.**OutOfOffice**
OutOfOffice class

Class which contains a store out of office properties and can set out-of-office status, message and subject.

**Parameters** **store** – store

**enabled**
Out of office enabled status

**end**
Out-of-office is activated until the particular datetime

**message**
Message

**start**
Out-of-office is activated from the particular datetime onwards

**subject**
Subject

**update**(***kwargs*)
Update function for outofoffice

## 3.13 Permission

**class** kopano.**Permission**
Permission class

**member**
User or group given specific rights.

**rights**
Rights given to member.

Possible rights:

read_items, create_items, create_subfolders, edit_own, edit_all, delete_own, delete_all, folder_owner, folder_contact, folder_visible

## 3.14 Property

**class** kopano.**Property**
Property class

**proptag = None**
Property tag, e.g. 0x37001f for PR_SUBJECT

**strval**
String representation of value; useful for overviews, debugging.

**value**
Property value, e.g. 'hello' for PR_SUBJECT

## 3.15 Quota

**class** kopano.**Quota**
Quota class

Quota limits are stored in bytes.

**hard_limit**
Hard limit

**soft_limit**
Soft limit

**update**(***kwargs*)
Update function for Quota limits, currently supports the following kwargs: *warning_limit*, *soft_limit* and *hard_limit*.

TODO: support defaultQuota and IsuserDefaultQuota

**warning_limit**
Warning limit

## 3.16 Recurrence

**class** kopano.**Recurrence**
Recurrence class

**class** kopano.**Occurrence**
Occurrence class

## 3.17 Restriction

**class** kopano.**Restriction**
Restriction class

## 3.18 Rule

**class** kopano.**Rule**
Rule class

## 3.19 Server

**class** kopano.**Server**
Server class

**__init__**(*options=None*, *config=None*, *sslkey_file=None*, *sslkey_pass=None*, *server_socket=None*, *auth_user=None*, *auth_pass=None*, *log=None*, *service=None*, *mapisession=None*, *parse_args=True*)
Create Server instance.

By default, tries to connect to a storage server as configured in /etc/kopano/admin.cfg or at UNIX socket /var/run/kopano/server.sock

Looks at command-line to see if another server address or other related options were given (such as -c, -s, -k, -p)

**Parameters**

- **server_socket** – similar to 'server_socket' option in config file
- **sslkey_file** – similar to 'sslkey_file' option in config file
- **sslkey_pass** – similar to 'sslkey_pass' option in config file
- **config** – path of configuration file containing common server options, for example /etc/kopano/admin.cfg
- **auth_user** – username to user for user authentication
- **auth_pass** – password to use for user authentication
- **log** – logger object to receive useful (debug) information
- **options** – OptionParser instance to get settings from (see parser())
- **parse_args** – set this True if cli arguments should be parsed

**companies**(*remote=False*, *parse=True*)
Return all companies on server.

**Parameters** **remote** – include companies without users on this server node

**company**(*name*, *create=False*)
Return company with given name.

**Parameters**

- **name** – company name
- **create** – create company if it doesn't exist

**create_company**(*name*)
Create a new company on the server.

**Parameters** **name** – the name of the company

**create_group**(*name*, *fullname=''*, *email=''*, *hidden=False*, *groupid=None*)
Create a new group on the server.

**Parameters**

- **name** – the name of the group
- **fullname** – the full name of the group (optional)
- **email** – the email address of the group (optional)

**create_public_store**()
Create public store in single-tenant mode.

**create_user**(*name*, *email=None*, *password=None*, *company=None*, *fullname=None*, *create_store=True*)
Create a new `user` on the server.

**Parameters**

- **name** – the login name of the user
- **email** – the email address of the user
- **password** – the login password of the user
- **company** – the company of the user
- **fullname** – the full name of the user
- **create_store** – should a store be created for the new user

**Returns** `<User>`

**delete**(*objects*)
Delete users, groups, companies or stores from server.

**Parameters** **objects** – The object(s) to delete

**get_company**(*name*)
Return `company` with given name or *None* if not found.

**get_store**(*guid*)
Return `store` with given GUID or *None* if not found.

**get_user**(*name*)
Return `user` with given name or *None* if not found.

**group**(*name*, *create=False*)
Return `group` with given name.

**Parameters**

- **name** – group name
- **create** – create group if it doesn't exist

**groups**()
Return all `groups` on server.

**guid**
Server GUID.

**hook_public_store**(*store*)
Hook public `store` in single-tenant mode.

**Parameters** **store** – store to hook

**id_to_name**(*proptag*)
Give the name representation of an property id. For example 0x80710003 => 'task:33025'.

**Parameters** **proptag** – the property identifier

**multitenant**
The server is multi-tenant.

**public_store**
Public `store` in single-tenant mode.

**state**
Current server state.

**store**(*guid=None*, *entryid=None*)
Return `store` with given GUID.

**stores**(*system=False*, *remote=False*, *parse=True*)
Return all stores on server node.

**Parameters**

- **system** – include system stores
- **remote** – include stores on other nodes

**sync**(*importer*, *state*, *log=None*, *max_changes=None*, *window=None*, *begin=None*, *end=None*, *stats=None*)
Perform synchronization against server node.

**Parameters**

- **importer** – importer instance with callbacks to process changes
- **state** – start from this state (has to be given)

**Log** logger instance to receive important warnings/errors

**unhook_public_store**()
Unhook public store in single-tenant mode.

**user**(*name=None*, *email=None*, *create=False*)
Return user with given name or email address.

**Parameters**

- **name** – user name
- **email** – email address
- **create** – create user if it doesn't exist (name required)

**users**(*remote=False*, *system=False*, *parse=True*)
Return all users on server.

**Parameters**

- **remote** – include users on remote server nodes
- **system** – include system users

## 3.20 Store

**class** kopano.**Store**
Store class

**archive_folder**
Archive Folder or *None* if not found

**archive_store**
Archive Store or *None* if not found

**calendar**
Folder designated as calendar

**common_views**
Folder contains folders for managing views for the message store

**config_item**(*name*)
Retrieve the config item for the given name.

**Parameters** **name** – The config item name

**contacts**
Folder designated as contacts

**create_prop**(*proptag*, *value*, *proptype=None*)
Create `property` with given proptag

**Parameters**

- **proptag** – MAPI property tag
- **value** – property value (or a default value is used)

**delete**(*objects*)
Delete properties, delegations or permissions from store.

**Parameters** **props** – The object(s) to delete

**drafts**
`Folder` designated as drafts

**entryid**
Store entryid

**favorites**()
Returns all favorite folders

**findroot**
`Folder` designated as search-results root

**folder**(*path=None*, *entryid=None*, *recurse=False*, *create=False*)
Return `Folder` with given path or entryid; raise exception if not found

**folders**(*recurse=True*, *parse=True*)
Return all `folders` in store

**Parameters**

- **recurse** – include all sub-folders
- **system** – include system folders

**get_folder**(*path=None*, *entryid=None*)
Return `folder` with given name/entryid or *None* if not found

**get_prop**(*proptag*)
Return `property` with given proptag or *None* if not found

**Parameters** **proptag** – MAPI property tag

**get_value**(*proptag*)
Return `property` value for given proptag or *None* if property does not exist

**Parameters** **proptag** – MAPI property tag

**guid**
Store GUID

**inbox**
`Folder` designated as inbox

**item**(*entryid*)
Return `Item` with given entryid; raise exception of not found

**journal**
`Folder` designated as journal

**junk**
`Folder` designated as junk

**last_logoff**
Return :datetime of the last logoff of a user on this store

**last_logon**
Return :datetime Last logon of a user on this store

**name**
User name ('public' for public store), or GUID

**notes**
`Folder` designated as notes

**outbox**
`Folder` designated as outbox

**outofoffice**
Return `OutOfOffice`

**permission**(*member*, *create=False*)
Return `permission` for user or group set for this store.

**Parameters**

- **member** – user or group
- **create** – create new permission for this store

**permissions**()
Return all `permissions` set for this store.

**prop**(*proptag*, *create=False*, *proptype=None*)
Return `property` with given property tag

**Parameters**

- **proptag** – MAPI property tag
- **create** – create property if it doesn't exist

**props**(*namespace=None*)
Return all `properties`

**reminders**
`Folder` designated as reminders

**root**
`Folder` designated as store root

**rss**
:class'Folder' designated as RSS items

**searches**()
Returns all permanent search folders

**send_only_to_delegates**
When sending meetingrequests to delegates, do not send them to the owner.

**sentmail**
`Folder` designated as sentmail

**set_value**(*proptag*, *value*)
Set `property` value for given proptag, creating the property if it doesn't exist

**size**
Store size

**subtree**
`Folder` designated as IPM.Subtree

**suggested_contacts**
:class'Folder' designated as Suggested contacts

**tasks**
`Folder` designated as tasks

**todo_search**
:class'Folder' designated as To-Do Search folder

**user**
Store owner

**value** (*proptag*)
Return property value for given proptag

**wastebasket**
Folder designated as wastebasket

## 3.21 User

**class** kopano.**User**
User class

**add_feature** (*feature*)
Add a feature for a user

**Parameters** **feature** – the new feature

**company**
Company the user belongs to

**create_prop** (*proptag*, *value*, *proptype=None*)
Create property with given proptag

**Parameters**

- **proptag** – MAPI property tag
- **value** – property value (or a default value is used)

**email**
Email address

**features**
Enabled features (pop3/imap/mobile)

**fullname**
Full name

**get_prop** (*proptag*)
Return property with given proptag or *None* if not found

**Parameters** **proptag** – MAPI property tag

**get_value** (*proptag*)
Return property value for given proptag or *None* if property does not exist

**Parameters** **proptag** – MAPI property tag

**hidden**
The user is hidden from the addressbook.

**name**
Account name

**prop** (*proptag*, *create=False*, *proptype=None*)
Return property with given property tag

**Parameters**

- **proptag** – MAPI property tag
- **create** – create property if it doesn't exist

**props** (*namespace=None*)
Return all properties

**quota**
User Quota

**remove_feature** (*feature*)
Remove a feature for a user

**Parameters** **feature** – the to be removed feature

**set_value** (*proptag*, *value*)
Set property value for given proptag, creating the property if it doesn't exist

**store**
Default Store for user or *None* if no store is attached

**userid**
Userid

**value** (*proptag*)
Return property value for given proptag

CHAPTER 4

# Legal Notice



Adobe, Acrobat, Acrobat Reader and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Apache is a trademark of The Apache Software Foundation.

Apple, Mac, Macintosh, Mac OS, iOS, Safari and TrueType are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Blackberry is the trademark or registered trademark of BlackBerry Limited, the exclusive rights to which are expressly reserved. Kopano is not affiliated with, endorsed, sponsored, or otherwise authorized by BlackBerry Limited.

Collax is a trademark of Collax GmbH.

Debian is a registered trademark of Software in the Public Interest, Inc.

ECMAScript is the registered trademark of Ecma International.

Gentoo is a trademark of Gentoo Foundation, Inc.

Google, Android and Google Chrome are trademarks or registered trademarks of Google Inc.

IBM and PowerPC are trademarks of International Business Machines Corporation in the United States, other countries, or both.

MariaDB is a registered trademark of MariaDB Corporation AB.

Microsoft, Microsoft Internet Explorer, the Microsoft logo, the Microsoft Internet Explorer logo, Windows, Windows Phone, Office Outlook, Office 365, Exchange, Active Directory and the Microsoft Internet Explorer interfaces are trademarks or registered trademarks of Microsoft, Inc.

Mozilla, Firefox, Mozilla Firefox, the Mozilla logo, the Mozilla Firefox logo, and the Mozilla Firefox interfaces are trademarks or registered trademarks of Mozilla Corporation.

MySQL, InnoDB, JavaScript and Oracle are registered trademarks of Oracle Corporation Inc.

NDS and eDirectory are registered trademarks of Novell, Inc.

NGINX is a registered trademark of Nginx Inc. NGINX Plus is a registered trademark of Nginx Inc.

Opera and the Opera "O" are registered trademarks or trademarks of Opera Software AS in Norway, the European Union and other countries.

Postfix is a registered trademark of Wietse Zweitze Venema.

QMAIL is a trademark of Tencent Holdings Limited.

Red Hat, Red Hat Enterprise Linux, Fedora, RHCE and the Fedora Infinity Design logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

SUSE, SLES, SUSE Linux Enterprise Server, openSUSE, YaST and AppArmor are registered trademarks of SUSE LLC.

# Symbols



## A



## B



## C



## D



## E



## F

## G



## H



## I



## J



## L



## M



## N



## O



## P

## Q



## R



## S



## T



## U



## V



## W