Python Kopano

Release 8.2.0

Kopano BV

May 10, 2019

Contents

1 Introduction **2**
- 1.1 Target audience . . . 2
- 1.2 Conventions . . . 2

2 Getting started **4**
- 2.1 Connecting to a server . . . 4
- 2.2 Accessing stores . . . 4
- 2.3 Accessing folders . . . 4
- 2.4 Looping over items and their properties . . . 5
- 2.5 Accessing items . . . 5
- 2.6 Sending an email . . . 6
- 2.7 Where to go from here . . . 6

3 Reference Guide **7**
- 3.1 Address . . . 7
- 3.2 Appointment . . . 7
- 3.3 Attachment . . . 8
- 3.4 Attendee . . . 9
- 3.5 AutoAccept . . . 9
- 3.6 AutoProcess . . . 10
- 3.7 Company . . . 10
- 3.8 Config . . . 12
- 3.9 Contact . . . 12
- 3.10 Delegation . . . 14
- 3.11 DistList . . . 14
- 3.12 Errors . . . 14
- 3.13 Folder . . . 15
- 3.14 FreeBusy . . . 19
- 3.15 Group . . . 20
- 3.16 Item . . . 21
- 3.17 MeetingRequest . . . 24
- 3.18 Notification . . . 25
- 3.19 OutOfOffice . . . 25
- 3.20 Permission . . . 26
- 3.21 Picture . . . 26
- 3.22 Property . . . 27
- 3.23 Properties . . . 28
- 3.24 Quota . . . 28
- 3.25 Recurrence . . . 29
- 3.26 Restriction . . . 30
- 3.27 Rule . . . 30
- 3.28 Server . . . 30

3.29 Service . . . 34
3.30 Store . . . 35
3.31 Table . . . 39
3.32 User . . . 39

4 Synchronization **42**
4.1 ICS . . . 42
4.2 Notifications . . . 43

5 Legal Notice **44**

Index **46**

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 a 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/accept binary identifiers in readable (hex-encoded) form
- To raise well-described exceptions if something goes wrong

As of Kopano Core 9.0, Python-Kopano is Python3-only.

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 least some experience with basic MAPI concepts such as properties.

1.2 Conventions

When dealing with dates, Python-Kopano accepts and returns standard datetime objects. These are currently always, or always have to be, timezone-unaware and system-local. We intend to add a global flag in the future to enable timezone-aware UTC datetimes.

Usually when an attribute can be read, it can also be set. This is a work in progress however. Let us know when you run into a missing setter which would be useful to have.

```
item.subject = '!!' + item.subject + '!!'
```

The Item class uses mixin classes to group specific functionality, such as appointment or contact-related functionality, which means there is currently a _lot_ of functionality in the Item class. We do however separately document the mixin classes.

When contributing patches to Python-Kopano, please follow the PEP-8 style guide.

CHAPTER 2

Getting started

2.1 Connecting to a 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()
```

Note that for Kopano Core 8.7 and older, one needs to use kopano.Server() instead.

If the Kopano server is not in a default location, 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 the provided arguments to determine how and where to connect. If there are no such arguments, it will try to get useful settings from /etc/kopano/admin.cfg. If this also doesn't exist, it will fall-back to the default UNIX socket.

2.2 Accessing stores

Now that we are connected to a Kopano server, let us access 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.

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

In general, the kopano module can like a server object. However, we recommend to only use this functionality in interactive mode (that is, from a Python shell), to save on typing.

2.3 Accessing folders

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 exist special shortcuts:

```
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 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)
```

Or to avoid the MAPI import:

```
print(item.prop('PR_SUBJECT_W').value)
```

So-called named properties can be accessed as follows:

```
PidLidReminderSignalTime = "PT_SYSTIME:common:0x8560"
print(item.prop(PidLidReminderSignalTime).value)
```

2.5 Accessing items

An item (which can be a mail, appointment, contact..) can be opened by entryid:

```
item = store.item(entryid)
```

Where the entryid comes from a log-file, for example, or is previously determined:

```
entryid = item.entryid
```

Item attributes can be changed as follows:

```
if 'cheese' in item.subject:
    item.subject = item.subject.upper()
```

2.6 Sending an email

Attributes can also be passed when creating an item. Using this we can create and send a simple email:

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

2.7 Where to go from here

After going through these basic examples, we hope our *Reference Guide* can help you further. Please do let us know if something remains unclear and/or not sufficiently documented.

For real-life examples, you may also want to look at Kopano Core (related) components written using Python-Kopano, for example:

- grapi
- kopano-search
- kopano-backup
- kopano-cli
- kopano-spamd
- kopano-migration-pst
- kopano-msr

CHAPTER 3

Reference Guide

3.1 Address

class kopano.**Address**
Address class

Abstraction for addresses, usually of type SMTP or ZARAFA. Most commonly used to resolve full names and/or email addresses.

addrtype
Address type (usually SMTP or ZARAFA)

email
Email address

entryid
User entryid (for addrtype ZARAFA)

name
Full name

props()
Return associated *properties*.

3.2 Appointment

class kopano.**Appointment**
Appointment mixin class

Appointment-specific functionality, mixed into the *Item* class.

all_day
Appointment is all-day.

attendees()
Appointment *attendees*.

busy_status
Appointment busy status (free, tentative, busy, out_of_office, working_elsewhere or unknown)

canceled
Is appointment canceled.

color
Appointment color (old clients).

create_attendee(*type_*, *address*)
Create appointment *attendee*.

Parameters

- **type** – attendee type (required, optional or resource)
- **address** – attendee address (str or *address*)

end
Appointment end.

icaluid
Appointment iCal UID.

location
Appointment location.

occurrences(*start=None*, *end=None*)
Appointment *occurrences* (expanding recurring appointments).

Parameters

- **start** – start from given date (optional)
- **end** – end at given date (optional)

recurrence
Appointment *recurrence*.

recurring
Appointment is recurring.

reminder
Is appointment reminder set.

reminder_minutes
Reminder minutes before appointment.

response_requested
Is appointment response requested.

start
Appointment start.

timezone
Appointment timezone description.

tzinfo
Appointment timezone as datetime compatible tzinfo object.

3.3 Attachment

class kopano.**Attachment**
Attachment class

Attachment abstraction. Attachments may contain binary data or embedded *items*.

Includes all functionality from *Properties*.

content_id
Identifier used to reference (inline) attachment.

content_location
URI pointing to contents of (inline) attachment.

data
Binary data.

delete(*objects*)
Delete properties from attachment

Parameters **objects** – The properties to delete

embedded
Is attachment an embedded *item*.

filename
Filename

hidden
Is attachment hidden from end user.

hierarchyid
Hierarchy (SQL) id.

inline
Is attachment inline.

item
Embedded *item*.

last_modified
Last modification time.

mapiobj
Underlying MAPI object.

mimetype
MIME type

number
Attachment number (respective to parent item).

parent
Parent *item*.

size
Storage size (different from binary data size!).

3.4 Attendee

class kopano.**Attendee**
Attendee class

Abstraction for *appointment* attendees.

address
Attendee *address*.

response
Attendee response status (no_response, accepted, declined, tentatively_accepted, organizer).

response_time
Attendee response time.

type_
Attendee type (required, optional or resource)

3.5 AutoAccept

class kopano.**AutoAccept**
AutoAccept class

Manage settings for automatically accepting meeting requests.

conflicts
Conflicting appointments are accepted.

enabled
Auto-accept is enabled.

recurring
Recurring appointments are accepted.

3.6 AutoProcess

class kopano.**AutoProcess**
AutoProcess class

Manage settings for automatically processing meeting requests.

enabled
Auto-processing is enabled.

3.7 Company

class kopano.**Company**
Company class

For a multi-tenant Kopano setup, users are grouped in companies. Regular users are unaware of anything outside their own company.

add_admin(*user*)
Add user to admin list

Parameters **user** – the user class:*user* <*User*>

Raises DuplicateError

add_view(*company*)
Add *remote viewer*.

admin
Company *administrator* in multi-tenant mode.

admins()
Return all company *admins*.

companyid
Company id.

create_group(*name*)
Create *group*.

Parameters **name** – group name

Raises DuplicateError

create_public_store()
Create company public *store*.

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

Parameters

- **name** – user name
- **password** – password (default None)

Raises DuplicateError

get_group(*name*)
Return *group* with given name or *None* if not found.

Parameters **name** – group name

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

Parameters **name** – user name

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

Parameters

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

Raises NotFoundError

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

mapiobj
Underlying MAPI object.

name
Company name.

public_store
Company public *store*.

quota
Company *Quota*.

remove_admin(*user*)
Remove user from admin list

Parameters **user** – the *user*

Raises NotFoundError

remove_view(*company*)
Remove *remote viewer*.

store(*guid*)
store for the given GUID

Parameters **guid** – store 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)

Raises NotFoundError

users(***kwargs*)
Return all *users* within company.

Parameters

- **parse** – filter users on cli argument –user (default True)
- **system** – include system users (default False)

views()
Return all *remote viewers* (other companies to which the address book is visible).

3.8 Config

class kopano.**Config**
Configuration class

Abstraction for kopano-style configation files.

3.9 Contact

class kopano.**Contact**
Contact mixin class

Contact-specific functionality, mixed into the *Item* class.

address1
Primary *address*.

address2
Secondary *address*.

address3
Tertiary *address*.

addresses()
Return all *addresses*.

assistant
Assistant.

birthday
Birthday.

business_address
Business address.

business_homepage
Business homepage.

business_phones
List of business phone numbers.

children
List of children names.

company_name
Company name.

department
Department.

email
Primary email address.

email2
Secondary email address.

email3
Tertiary email address.

file_as
File as.

first_name
First name.

generation
Generation.

home_address
Home `address`.

home_phones
List of home phone numbers.

im_addresses
List of instant messaging addresses.

initials
Initials.

job_title
Job title.

last_name
Last name.

manager
Manager.

middle_name
Middle name.

mobile_phone
Mobile phone.

nickname
Nickname.

office_location
Office location.

other_address
Other `address`.

photo
Contact *photo*

profession
Profession.

set_photo(*name*, *data*, *mimetype*)
Set contact *photo*

Parameters

- **name** – file name

- **data** – file binary data
- **mimetype** – file MIME type

spouse
Spouse.

title
Title.

yomi_company_name
Yomi (phonetic) company name.

yomi_first_name
Yomi (phonetic) first name.

yomi_last_name
Yomi (phonetic) last name.

3.10 Delegation

class kopano.**Delegation**
Delegation class

Abstraction for delegate users and settings.

flags
Delegation flags (see_private, send_copy).

see_private
Delegate user can see private *items*.

send_copy
Delegate user receives copies of meeting requests.

user
Delegate user.

3.11 DistList

class kopano.**DistList**
DistList class

A distribution list contains users, groups and/or sub distribution lists.

members(*expand=True*)
Return all distribution list members, as *addresses* and/or (sub) *distribution lists*.

Parameters **expand** – expand sub distribution lists (optional)

3.12 Errors

class kopano.**Error**

class kopano.**ConfigError**

class kopano.**DuplicateError**

class kopano.**NotFoundError**

class kopano.**LogonError**

class kopano.**NotSupportedError**

3.13 Folder

class kopano.**Folder**
Folder class

Abstraction for collections of *items*. For example, folders, calendars and contact folders.

Includes all functionality from *Properties*.

archive_folder
Archive *Folder*.

associated
Folder containing hidden items.

container_class
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
Folder item count.

create_folder(*path=None*, ***kwargs*)
Create folder with given (relative) path

Parameters **path** – Folder path

create_item(*eml=None*, *ics=None*, *vcf=None*, *load=None*, *loads=None*, *attachments=True*, *save=True*, ***kwargs*)
Create *item*.

Parameters

- **eml** – pass eml data/file (optional)
- **ics** – pass iCal data/file (optional)
- **vcf** – pass vcf data/file (optional)
- **load** – pass serialized item file (optional)
- **loads** – pass serialized item data (optional)

create_rule(*name=None*, *restriction=None*)
Create rule

Parameters

- **name** – Rule name (optional)
- **restriction** – Rule *restriction* (optional)

created
Folder creation date.

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

Parameters

- **objects** – The object(s) to delete
- **soft** – soft-delete items/folders (default False)

deleted
Folder containing soft-deleted items.

dumps()
Serialize folder contents.

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

Parameters

- **recurse** – delete subfolders (default True)
- **associated** – delete associated contents (default False)

entryid
Folder entryid.

folder(*path=None*, *entryid=None*, *recurse=False*, *create=False*)
Return *Folder* with given path or entryid

Parameters

- **path** – Folder path (optional)
- **entryid** – Folder entryid (optional)
- **create** – Create folder if it doesn't exist (default False)

folders(*recurse=True*, *restriction=None*, *page_start=None*, *page_limit=None*, *order=None*)
Return all *sub-folders* in folder

Parameters

- **recurse** – include all sub-folders
- **restriction** – apply *restriction*
- **page_start** – skip this many items from the start
- **page_limit** – return up to this many items

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

Parameters

- **path** – Folder path (optional)
- **entryid** – Folder entryid (optional)

hierarchyid
Folder hierarchy (SQL) id.

ics(*charset='UTF-8'*)
Export all calendar items in the folder to iCal data

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

Parameters

- **entryid** – item entryid (optional)
- **sourcekey** – item sourcekey (optional)

items(*restriction=None*, *page_start=None*, *page_limit=None*, *order=None*, *query=None*)
Return all *items* in folder, reverse sorted on received date.

Parameters

- **restriction** – apply *restriction*
- **order** – order by (limited set of) attributes, e.g. 'subject', '-subject' (reverse order), or ('subject', '-received').
- **page_start** – skip this many items from the start
- **page_limit** – return up to this many items
- **query** – use search query

last_modified
Folder last modification date.

loads(*data*)
Deserialize folder contents

Parameters **data** – Serialized data

maildir(*location='.'*)
Export all *items* to mailbox (using python mailbox module, maildir variant).

Parameters **location** – mailbox location

mapiobj
Underlying MAPI object.

mbox(*location*)
Export all *items* to mailbox (using python mailbox module, mbox variant).

Parameters **location** – mailbox location

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

Parameters

- **objects** – The item(s) or subfolder(s) to move
- **folder** – The target folder

name
Folder name.

occurrences(*start=None*, *end=None*, *page_start=None*, *page_limit=None*, *order=None*)
For applicable folder types (e.g., calendars), return all *occurrences*.

Parameters

- **start** – start time (optional)
- **end** – end time (optional)

parent
Parent folder

path
Folder path.

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.

permissions_dumps(***kwargs*)
Serialize permissions.

permissions_loads(*data*, ***kwargs*)
Deserialize permissions.

Parameters **data** – Serialized data

primary_folder
Primary *folder* (for archive folders).

primary_store
Primary *store* (for archive folders).

read_ics(*ics*)
Import all *items* from iCal calendar

Parameters **ics** – the iCal data

read_maildir(*location*)
Import all *items* from mailbox (python mailbox module, maildir variant).

Parameters **location** – mailbox location

readmbox(*location*)
Import all *items* from mailbox (python mailbox module, mbox variant).

Parameters **location** – mailbox location

rights(*member*)
Determine rights on this folder for a given *user* or *group*.

Parameters **member** – User or group

rules()
Return all folder *rules*.

rules_dumps(*stats=None*)
Serialize folder rules.

rules_loads(*data*, *stats=None*)
Deserialize folder rules.

Parameters **data** – Serialized data

settings_dumps()
Serialize folder settings (rules, permissions).

settings_loads(*data*)
Deserialize folder settings (rules, permissions).

Parameters **data** – Serialized data

size
Folder storage size.

sourcekey
Folder sourcekey.

state
Folder state (for use with ICS synchronization).

subfolder_count
Direct subfolder count.

subfolder_count_recursive
Subfolder count (recursive).

subscribe(*sink*, ***kwargs*)
Subscribe to folder notifications

Parameters

- **sink** – Sink instance with callbacks to process notifications
- **object_types** – Tracked objects (*item*, *folder*)
- **folder_types** – Tracked folders (*mail*, *contacts*, *calendar*)
- **event_types** – Event types (*created*, *updated*, *deleted*)

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

sync_hierarchy(*importer*, *state=None*, *stats=None*)
Perform hierarchy synchronization against folder. In other words, receive changes to the folder and its subfolders (recursively).

Parameters

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

type_
Folder type (mail, contacts, calendar or just folder).

unread
Unread item count.

unsubscribe(*sink*)
Unsubscribe from folder notifications

Parameters **sink** – Previously subscribed sink instance

3.14 FreeBusy

class kopano.**FreeBusyBlock**
FreeBusyBlock class

end = None
Freebusy block end

start = None
Freebusy block start

status = None
Freebusy block status (free, tentative, busy or outofoffice)

class kopano.**FreeBusy**
FreeBusy class

Freebusy data consists of high level, privacy insensitive, information about user availability. This is mostly useful for scheduling meetings, without requiring full access to other users' calendars.

blocks(*start=None*, *end=None*)
Return all *freebusy blocks* for the given period.

Parameters

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

publish(*start=None*, *end=None*)
Publish freebusy information for the given period.

Parameters

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

3.15 Group

class kopano.**Group**
Group class

A group is a collection of *users* and/or subgroups of *users*.

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 id.

groups()
Return all *groups* in group.

hidden
The group is hidden from the addressbook.

mapiobj
Underlying MAPI object.

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.

3.16 Item

class kopano.**Item**
Item class

Abstraction for basic items, such as emails, calendar appointments and contacts.

Includes all functionality from *Appointment*, *Contact* and *Properties*.

attachments(*embedded=False*, *page_start=None*, *page_limit=None*, *order=None*)
Return all item *attachments*

Parameters **embedded** – include embedded attachments (default False)

bcc
Return all item BCC recipient *addresses*.

body_preview
Item body preview (plaintext, up to 255 characters).

body_type
Original body type: text, html, rtf or *None*.

categories
Item categories (user defined).

cc
Return all item CC recipient *addresses*.

changekey
Item changekey.

codepage
Windows codepage for item.

conversationid
Item conversation ID.

copy(*folder*, *_delete=False*)
Copy item to folder (returning copied item).

Parameters **folder** – Target folder

create_attachment(*name=None*, *data=None*, *filename=None*, ***kwargs*)
Create an item attachment

Parameters

- **name** – The attachment name
- **data** – Attachment data

create_item(*message_flags=None*, *hidden=False*, ***kwargs*)
Create embedded *item*.

created
Creation time.

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

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

delivery_receipt
A delivery receipt was requested.

distlist
Distribution List object.

dump(*f*, *attachments=True*, *archiver=True*)
Serialize item into open file.

Parameters **f** – Open file

dumps(*attachments=True*, *archiver=True*, *skip_broken=False*)
Serialize item.

eml(*received_date=False*, *stored=True*)
Convert the item to eml (RFC 2822) data

Parameters

- **received_date** – add delivery date as received date
- **stored** – if available, use the stored version instead of converting (default True)

encoding
HTML body encoding.

entryid
Item entryid.

filtered_html
Item filtered HTML representation.

folder
Item parent *folder*.

from_
Item from *Address* (not necessarily the sender).

has_attachments
Item has attachments.

header(*name*)
Item transport message header for given name

Parameters **name** – Header name

headers()
Item transport message headers.

hierarchyid
Hierarchy (SQL) id.

html
Item HTML representation.

html_utf8
Item HTML representation (utf-8 encoded).

ics(*charset='UTF-8'*)
Convert the item to iCal data.

is_distlist
Is the item a distribution list.

is_meetingrequest
Is the item a meeting request.

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

last_modified
Last modification time.

load(*f*, *attachments=True*)
Deserialize item from open file

Parameters **f** – Open file

loads(*s*, *attachments=True*)
Deserialize item from data

Parameters **s** – Serialized data

mapiobj
Underlying MAPI object.

match(*restriction*)
Does the given restriction match with the item.

Parameters **restriction** – *Restriction*

meetingrequest
Meeting request object.

message_class
Item message class.

messageid
Item internet message ID.

move(*folder*)
Move item to folder (returning moved item).

Parameters **folder** – Target folder

name
Item (display) name.

normalized_subject
Normalized item subject.

primary_item
Primary item (for archived *items*).

private
Should client hide item from other users.

read
Item is read.

read_receipt
A read receipt was requested.

received
Delivery time.

recipients(*_type=None*)
Return all item recipient *addresses*.

reply(*folder=None*, *all=False*)
Create reply message

Parameters

- **folder** – Folder to save reply (default in drafts)
- **all** – Reply to all (default False)

replyto
Return all *Addresses* where reply should be sent.

restore()
Restore item (for archived *items*).

rtf
Item RTF representation.

searchkey
Item searchkey.

send(*copy_to_sentmail=True*, *_basedate=None*, *cal_item=None*)
Send item (mail or appointment as meeting request).

Parameters **copy_to_sentmail** – Save copy in sentmail (default True)

sender
Item sender *Address*.

sensitivity
Item sensitivity (normal, personal, private or confidential).

sent
Client submit time.

size
Item storage size.

sourcekey
Item sourcekey.

stubbed
Item is stubbed by archiver.

subject
Item subject.

text
Item plaintext representation.

to
Return all item TO recipient *addresses*.

urgency
Item urgency (low, high or normal).

user
Item *owner*.

vcf()
Convert the item to vCard data.

3.17 MeetingRequest

class kopano.**MeetingRequest**
MeetingRequest class

A meeting request is basically a copy of an appointment, being sent to the attendees (or sent back to the organizer).

It may be an actual request, response or cancellation for the given appointment.

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

Parameters

- **tentative** – accept tentatively (default False)
- **response** – send response message (default True)

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 (default True)

is_cancellation
Is it a cancellation.

is_request
Is it a request.

is_response
Is it a response.

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

Parameters **delete** – delete appointment from calendar (default False)

process_response()
Process meeting request response.

processed
Has the request/response been processed.

response_requested
Is a response requested.

update_counter
Update counter.

3.18 Notification

class kopano.**Notification**
Notification class

A notification instance indicates a change to *items* or *folders*.

event_type = None
The type of change (*created*, *updated*, *deleted*)

mapiobj = None
The underlying MAPI notification object

object = None
The changed *item* or *folder*.

object_type = None
The type of the changed object (*item*, *folder*).

3.19 OutOfOffice

class kopano.**OutOfOffice**
OutOfOffice class

Manage out-of-office settings, such as status, subject and message.

active
Out-of-office is currently active (start <= now < end).

enabled
Out-of-office is enabled.

end
Out-of-office is active until the given date.

message
Out-of-office message.

period_desc
English description of out-of-office date range.

start
Out-of-office is active starting from the given date.

subject
Out-of-office subject.

3.20 Permission

class kopano.**Permission**
Permission class

A permission instance combines a store or folder with a user or group and a set of permissions.

Permissions for a given folder are resolved by following the parent chain (and ultimately store), until there is a match on user or group.

member
The associated *User* or *group*.

rights
The rights given to the associated member.

Possible rights:

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

3.21 Picture

class kopano.**Picture**
Picture class

Typical pictures are *contact* and *user* profile photos.

data = None
Picture binary data.

height
Picture pixel height.

mimetype
Picture MIME type.

name = None
Picture name.

scale (*size*)
Return new picture instance, scaled to given size.

Parameters **size** – (width, height) tuple

size
Picture pixel (width, height) tuple.

width
Picture pixel width.

3.22 Property

class kopano.**Property**
Property class

Low-level abstraction for MAPI properties.

guid
For a named property, return the GUID.

id_
Id part of proptag, for example 0x37 for PR_SUBJECT_W.

idname
MAPI proptag name, for example *PR_SUBJECT_W*.

kind
For a named property, return the name kind.

kindname
For a named property, return the readable name kind.

name
For a named property, return the name.

named
Is the property a named property.

namespace
For a named property, return the readable namespace.

proptag = None
MAPI proptag, for example 0x37001f for PR_SUBJECT_W

strid
Readable identifier for named/unnamed property.

strval
String representation of property value.

type_
Proptag type, for example 0x1f for PR_SUBJECT_W.

typename
Proptag type name, for example *PT_UNICODE* for PR_SUBJECT_W.

value
Property value.

For PT_SYSTIME properties, convert to/from timezone-unaware system-local datetimes.

3.23 Properties

class kopano.**Properties**
Property mixin class

MAPI property-specific functionality, mixed into several classes whose instances represent sets of MAPI properties, such as *Item* and *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)

get(*proptag*, *default=None*)
Return *property* value for given proptag or *None* if property does not exist.

Parameters **proptag** – MAPI property tag

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

Parameters **proptag** – MAPI property tag

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*.

3.24 Quota

class kopano.**Quota**
Quota class

Manage *user* or *company* quota settings.

add_recipient(*user*, *company=False*)
Add *recipient* of quota messages.

hard_limit
Hard limit.

recipients()
Return all *recipients* of quota messages.

remove_recipient(*user*, *company=False*)
Remove *recipient* of quota messages.

soft_limit
Soft limit.

update(***kwargs*)
Update function for Quota limits.

Parameters

- **warning_limit** – Warning limit.

- **soft_limit** – Soft limit.
- **hard_limit** – Hard limit.

use_default
Use default quota.

warning_limit
Warning limit.

3.25 Recurrence

class kopano.**Recurrence**
Recurrence class

Abstraction for recurring *items*, such as appointments.

Provides equivalent functionality as iCal, but on top of MAPI and dateutil.rrule.

For example: "an appointment occurs every last monday of the month, except in june 2020".

count
Recurrence count. Depending on range type, indicates an absolute number of occurrences.

end
End of recurrence range (using recurrence timezone!). Used depending on range type.

first_weekday
Recurrence first weekday (*sunday*, *monday*, ..). Depending on the pattern, this indicates the first day of the week.

index
Recurrence index (*first*, *second*, *third*, *fourth*, *last*). Depending on the pattern, this indicates the occurrence within each month where the recurrence occurs. For example, on the second monday or the last wednesday.

interval
Recurrence interval (number). Depending on the pattern, this indicates how many weeks/months/years there should be between each occurrence. For example, every 3 weeks.

month
Recurrence month (1..12). Depending on the pattern, this indicates the month where the recurrence occurs.

monthday
Recurrence month day (1..31). Depending on the pattern, this indicates the day of the month where the recurrence occurs.

occurrences(*start=None*, *end=None*)
Return all recurrence *occurrences*, optionally overlapping with a given time window.

Parameters

- **start** – start of time window (optional)
- **end** – end of time window (optional)

pattern
Recurrence pattern (*daily*, *weekly*, *monthly*, *monthly_rel*, *yearly*, *yearly_rel*).

range_type
Recurrence range type (*end_date*, *count*, *forever*). Indicates how the recurrence ends: by end date, by count, or never.

start
Start of recurrence range (using recurrence timezone!).

weekdays
Recurrence weekdays (*sunday*, *monday*, ..). Depending on the pattern, this indicates days of the week where the recurrence should occur.

class kopano.**Occurrence**
Occurrence class.

Abstraction for specific occurrences of a *recurrence*.

Works similar to class *Item*, except there may be multiple occurrences for a single (recurring) item.

item = None
Recurring *item* which this occurrence belongs to

3.26 Restriction

class kopano.**Restriction**
Restriction class

3.27 Rule

class kopano.**Rule**
Rule class.

A rule consist of a *condition* and a set of actions. When a mail is delivered (usually in the inbox), and it matches the condition, the actions are executed.

Typical actions are deletion, forwarding, copying and moving.

actions()
Rule actions.

active = None
Is the rule active.

name = None
Rule name.

provider = None
Rule provider.

restriction
Rule *condition*.

3.28 Server

class kopano.**Server**
Server class.

Abstraction for Kopano servers. A MAPI session is automatically setup, according to the passed arguments and environment.

__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*, *notifications=False*, *store_cache=True*, *oidc=False*, *_skip_check=False*)
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
- **options** – OptionParser instance to get settings from (see:func:*parser*)
- **parse_args** – set this True if cli arguments should be parsed

admin_store
Admin store.

clear_cache()
Clear caches.

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

Parameters

- **remote** – include companies without users on this server node (default False)
- **parse** – take cli argument –companies into account (default True)

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

Parameters

- **name** – Company name
- **create** – Create company if it doesn't exist (default False)

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

Parameters **name** – Company name

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)
- **hidden** – hide the group (optional)
- **groupid** – the id of the group (optional)

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

create_store(*user*, *_msr=False*)
Create store for *User*.

Parameters **user** – User

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

Parameters

- **name** – User login name
- **email** – User email address (optional)
- **password** – User login password (optional)
- **company** – User company (optional)
- **fullname** – User full name (optional)
- **create_store** – Should a store be created (default True)

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

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

get_company(*name*)
Company with given name or *None* if not found.

Parameters **name** – Company name

get_stat(*key*, *default=None*)
Specific server statistic or *None* if not found.

Parameters **key** – Statistic key

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 (default False)

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

guid
Server GUID.

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

Parameters **store** – store to hook

multitenant
The server is multi-tenant (multiple *companies*).

name
Server name.

nodes()
For a multi-server setup, return all servers (nodes).

public_store
Public *store* (single-tenant mode).

purge_deferred()
Purge deferred updates.

purge_softdeletes(*days*)
Purge soft-deletes older than certain amount of days.

Parameters **days** – Amount of days in the past.

remove_company(*name*)
Remove *Company*.

Parameters **name** – Company name

remove_group(*name*)
Remove *group*.

Parameters **name** – Group name

remove_store(*store*)
Remove *Store*.

Parameters **store** – Store

remove_user(*name*)
Remove *User*.

Parameters **name** – User login name

stat(*key*)
Specific server statistic.

Parameters **key** – Statistic key

state
Server state (for use with ICS synchronization).

stats()
Dictionary containing useful server statistics.

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

Parameters

- **guid** – Store GUID (optional)
- **entryid** – Store entryid (optional)

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

Parameters

- **system** – Include system stores (default False)
- **remote** – Include stores on other nodes (default False)

sync(*importer*, *state*, *log=None*, *max_changes=None*, *window=None*, *begin=None*, *end=None*, *stats=None*)
Perform ICS 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

sync_gab(*importer*, *state=None*)
Perform ICS synchronization against global address book.

Parameters

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

sync_users()
Synchronize users with external source.

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

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

Parameters

- **name** – User name (optional)
- **email** – User email address (optional)
- **userid** – User userid (optional)
- **create** – create user if it doesn't exist (default False, name required)

users(*remote=False*, *system=False*, *parse=True*, *page_start=None*, *page_limit=None*, *order=None*, *hidden=True*, *inactive=True*, *_server=None*, *_company=None*, *query=None*)
Return all *users* on server.

Parameters

- **remote** – Include users on remote server nodes (default False)
- **system** – Include system users (default False)
- **hidden** – Include hidden users (default True)
- **inactive** – Include inactive users (default True)
- **query** – Search query (optional)

3.29 Service

class kopano.**Service**
Encapsulates everything to create a simple Kopano service, such as:

•Locating and parsing a configuration file

•Performing logging, as specifified in the configuration file

•Handling common command-line options (-c, -F)

•Daemonization (if no -F specified)

Parameters

- **name** – name of the service; if for example 'search', the configuration file should be called /etc/kopano/search.cfg or passed with -c
- **config** – *Configuration* to use
- **options** – OptionParser instance to get settings from (see parser())

config = None
Service *configuration* instance.

log = None
Service logger.

server
Service *server* instance.

start()
Start service.

3.30 Store

class kopano.**Store**
Store class.

There are three types of stores: *public*, *private* and *archive*.

A *private* store contains a hierarchical collection of *folders*, as well as various user settings, stored in *properties* of the store. For example, out-of-office and webapp settings.

The *root* folder is the user-invisible root folder of the hierarchy, whereas *subtree* folder is the user-visible root folder. Most special folders such as the *inbox* are found directly under the *subtree* folder.

In the *root* folder there are can be certain invisible special folders, such as a folder containing searches, a folder containing archive state and a folder used for notifications.

A *public* store also contains a hierarchical collection of folders, which are accessible to all *users*.

An *archive* store is used in combination with the archiver, and contains 'older' data, which is migrated there according to configurable criteria.

Includes all functionality from *Properties*.

add_favorite(*folder*)
Add *Folder* to favorites.

archive_folder
Archive *Folder* (in case multiple stores are archived to a single archive store).

archive_store
Archive *Store*.

autoaccept
Return *AutoAccept* settings.

autoprocess
Return *AutoProcess* settings.

calendar
The store (default) *calendar*.

common_views
Folder containing sub-folders acting as special views on the message store.

company
Store *company*.

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

Parameters **name** – The config item name

contacts
The store (default) *contacts folder*.

create_folder(*path=None*, ***kwargs*)
Create *folder* under *subtree* with given path.

Parameters **path** – The path of the folder, relative to *subtree*.

delegation(*user*, *create=False*, *see_private=False*)
Return *delegation* for user.

Parameters

- **user** – user
- **create** – create new delegation for this user
- **see_private** – user can see private items

delegations()
Return all *delegations*.

delegations_dumps(*stats=None*)
Serialize *Delegation* settings.

delegations_loads(*data*, *stats=None*)
Deserialize *Delegation* settings.

Parameters **data** – Serialized data

delete(*objects*, *soft=False*)
Delete properties, delegations, permissions, folders or items from store.

Parameters

- **objects** – The object(s) to delete
- **soft** – Soft-delete items and folders (default False)

drafts
The store *drafts folder*.

dumps()
Serialize entire store, including all settings.

entryid
Store entryid.

favorites()
Returns all favorite *folders*.

findroot
The user-invisible store search folder *Folder*.

folder(*path=None*, *entryid=None*, *recurse=False*, *create=False*, *guid=None*)
Return *Folder* with given path/entryid.

Parameters

- **path** – The path of the folder (optional)
- **entryid** – The entryid of the folder (optional)
- **create** – Create folder if it doesn't exist (default False)

folders(*recurse=True*, *parse=True*, ***kwargs*)
Return all *folders* under *subtree*.

Parameters **recurse** – include all sub-folders (default True)

freebusy
`Freebusy` information.

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

Parameters

- **path** – The path of the folder (optional)
- **entryid** – The entryid of the folder (optional)

guid
Store GUID.

hierarchyid
Hierarchy (SQL) id.

inbox
The store *inbox*.

item(*entryid=None*, *guid=None*)
Return *Item* with given entryid.

journal
The store *journal folder*.

junk
The store *junk folder*.

last_logoff
Return :datetime of the last logoff on this store.

last_logon
Return :datetime of the last logon on this store.

loads(*data*)
Deserialize entire store, including all settings.

Parameters **data** – Serialized data

mapiobj = None
Underlying MAPI object.

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

notes
The store *notes folder*.

orphan
The store is orphaned.

outbox
The store *outbox*.

outofoffice
Return *OutOfOffice* settings.

permission(*member*, *create=False*)
Return *permission* for *User* or *Group* set for the store.

Parameters

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

permissions()
Return all *permissions* set for the store.

permissions_dumps(***kwargs*)
Serialize *permissions* set for the store.

permissions_loads(*data*, ***kwargs*)
Deserialize *permissions*.

Parameters **data** – Serialized data

public
The store is a public store.

reminders
The user-invisible store reminder *Folder*.

root
The user-invisible store root *Folder*.

rss
The store :class‘RSS folder <Folder>‘.

searches()
Return all permanent search folders.

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

sentmail
The store *sentmail folder*.

settings_dumps()
Serialize store settings.

settings_loads(*data*)
Deserialize (overriding) all store settings.

Parameters **data** – Serialized data

size
Store storage size.

subscribe(*sink*, ***kwargs*)
Subscribe to store notifications

Parameters

- **sink** – Sink instance with callbacks to process notifications
- **object_types** – Tracked objects (*item*, *folder*)
- **folder_types** – Tracked folders (*mail*, *contacts*, *calendar*)
- **event_types** – Event types (*created*, *updated*, *deleted*)

subtree
The user-visible store root *Folder*.

suggested_contacts
The store :class‘suggested contacts <Folder>‘.

tasks
The store *tasks folder*.

todo_search
The store :class‘todo search folder <Folder>‘.

type_
Store type (*private*, *public*, *archive*).

user
Store *owner*.

views
Folder containing sub-folders acting as special views on the message store.

wastebasket
The store *wastebasket*.

webapp_settings
Webapp settings (JSON).

3.31 Table

class kopano.**Table**
Table class

Low-level abstraction for MAPI tables.

count
Return table row count.

csv(**args*, ***kwargs*)
Return CSV data for table. All arguments are passed to csv.writer.

data(*header=False*)
Return list per row with textual representation for each element.

dict_rows(*batch_size=100*)
Return all table rows as dictionaries.

header
Return all table column names.

index(*key*)
Return key->row dictionary keyed on given column (proptag).

rows(*batch_size=100*, *page_start=None*, *page_limit=None*)
Return all table rows.

sort(*tags*)
Sort table.

Parameters **tags** – Tag(s) on which to sort.

text(*borders=False*)
Return textual table representation.

3.32 User

class kopano.**User**
User class.

Includes all functionality from *Properties*.

Store attributes can be accessed directly from a *User* instance.

active
Is the user active.

add_feature(*feature*)
Enable a feature for a user.

Parameters **feature** – The feature

add_send_as(*user*)
Add *user* as send-as.

Parameters **user** – User to add.

admin
Is the user (regular or system) administrator.

admin_level
User administration level (1 means regular administrator, 2 means system administrator).

archive_server
Name of archive server.

company
Company the user belongs to.

create_store()
Create and attach store for user.

email
User email address.

features
Enabled features (*pop3*, *imap*, *mobile*, ..).

first_name
User first name.

fullname
User full name.

groups()
Return all *groups* to which user belongs.

hidden
The user is hidden from the addressbook.

home_server
Name of user home server.

hook(*store*)
Hook (attach) store.

Parameters **store** – *Store* to hook.

hook_archive(*store*)
Hook archive store.

Parameters **store** – Archive *store* to hook.

job_title
User job title.

last_name
User last name.

mapiobj
Underlying MAPI object.

mobile_phone
User mobile phone number.

name
User account name.

office_location
User office Location.

password
User password (set-only).

photo
User *photo*.

quota
User *Quota*

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

Parameters **feature** – The feature

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

Parameters **user** – User to remove.

rules()
Return all *rules* on user inbox.

send_as()
Return all *users* who can send-as this user.

store
Return user *Store* or *None* if no store is attached.

unhook()
Unhook attached store.

unhook_archive()
Unhook archive store.

userid
User id.

CHAPTER 4

Synchronization

There are two methods to follow changes to certain server contents. The first, ICS (incremental change synchronization) allows for state-based synchronization. The other, notifications, provides real-time updates whenever something changes.

4.1 ICS

Based on a given state, ICS exports all subsequent changes and when there are no more changes, returns the new state. Changes are communicated to a call-back object, with the following (optional) methods:

```
class Importer:
    def update(self, obj, flags=None):
        print('new or updated:', obj, flags)

    def delete(self, obj, flags=None):
        print('deleted:', obj, flags)

    def read(self, obj, status): # items only
        print('read state change:', obj, status)
```

Using this, we can synchronize the items of a specific folder:

```
folder.sync(Importer(), state)
```

Or even all item changes for an entire Kopano server (system-wide ICS):

```
server.sync(Importer(), state)
```

In the folder case, we can omit the state, in which case the entire folder is synchronized.

We can also synchronize the sub-folder hierarchy for a given folder (usually subtree):

```
store.subtree.sync_hierarchy(Importer(), state)
```

Finally, we can also synchronize the addressbook as follows:

```
server.sync_gab(Importer(), state)
```

Note that on deletion, the actual object is gone when the call-back happens. For items and folders only obj.sourcekey remains, and for users only obj.userid.

Note also that old changes are not kept in the system, so intermediate changes are never exported.

4.2 Notifications

Notifications provide real-time tracking of specific classes of changes. Similar to ICS, a call-back object receives the changes:

```
class Sink:
    def update(self, notification):
        print('changed:',
            notification.object,
            notification.object_type,
            notification.event_type)
```

Now, to subscribe to all changes to a specific folder:

```
folder.subscribe(Sink())
```

To only receive changes to the folder items themselves, specify object type (*folder* or *item*):

```
folder.subscribe(Sink(), object_types=['item'])
```

To only receive certain event types for these items (*created*, *updated*, *deleted*):

```
folder.subscribe(Sink(), object_types=['item'], event_types=['created', 'deleted'])
```

To subscribe to all changes to a given store:

```
store.subscribe(Sink())
```

To only receive changes to folders, specify object type (*folder* or *item*):

```
store.subscribe(Sink(), object_types=['folder'])
```

CHAPTER 5

Legal Notice

Copyright © 2016-2019 Kopano

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.

Sendmail is a trademark of Sendmail, Inc.

UNIX is a registered trademark of The Open Group.

Ubuntu and Canonical are registered trademarks of Canonical Ltd.

Univention is a trademark of Ganten Investitions GmbH.

All trademarks are property of their respective owners. Other product or company names mentioned may be trademarks or trade names of their respective owner.

Disclaimer: Although all documentation is written and compiled with care, Kopano is not responsible for direct actions or consequences derived from using this documentation, including unclear instructions or missing information not contained in these documents.

The text of and illustrations in this document are licensed by Kopano under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at the creativecommons.org website. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version. This document uses parts from the Zarafa Collaboration Platform (ZCP) Python Zarafa, located at the Zarafa Documentation Portal, licensed under CC-BY-SA.

Symbols

__init__() (kopano.Server method), 30

A

accept() (kopano.MeetingRequest method), 24
actions() (kopano.Rule method), 30
active (kopano.OutOfOffice attribute), 26
active (kopano.Rule attribute), 30
active (kopano.User attribute), 39
add_admin() (kopano.Company method), 10
add_favorite() (kopano.Store method), 35
add_feature() (kopano.User method), 39
add_recipient() (kopano.Quota method), 28
add_send_as() (kopano.Group method), 20
add_send_as() (kopano.User method), 39
add_user() (kopano.Group method), 20
add_view() (kopano.Company method), 10
Address (class in kopano), 7
address (kopano.Attendee attribute), 9
address1 (kopano.Contact attribute), 12
address2 (kopano.Contact attribute), 12
address3 (kopano.Contact attribute), 12
addresses() (kopano.Contact method), 12
addrtype (kopano.Address attribute), 7
admin (kopano.Company attribute), 10
admin (kopano.User attribute), 39
admin_level (kopano.User attribute), 39
admin_store (kopano.Server attribute), 31
admins() (kopano.Company method), 10
all_day (kopano.Appointment attribute), 7
Appointment (class in kopano), 7
archive_folder (kopano.Folder attribute), 15
archive_folder (kopano.Store attribute), 35
archive_server (kopano.User attribute), 39
archive_store (kopano.Store attribute), 35
assistant (kopano.Contact attribute), 12
associated (kopano.Folder attribute), 15
Attachment (class in kopano), 8
attachments() (kopano.Item method), 21
Attendee (class in kopano), 9
attendees() (kopano.Appointment method), 7
AutoAccept (class in kopano), 9
autoaccept (kopano.Store attribute), 35
AutoProcess (class in kopano), 10
autoprocess (kopano.Store attribute), 35

B

basedate (kopano.MeetingRequest attribute), 24
bcc (kopano.Item attribute), 21
birthday (kopano.Contact attribute), 12
blocks() (kopano.FreeBusy method), 19
body_preview (kopano.Item attribute), 21
body_type (kopano.Item attribute), 21
business_address (kopano.Contact attribute), 12
business_homepage (kopano.Contact attribute), 12
business_phones (kopano.Contact attribute), 12
busy_status (kopano.Appointment attribute), 7

C

calendar (kopano.MeetingRequest attribute), 25
calendar (kopano.Store attribute), 35
calendar_item (kopano.MeetingRequest attribute), 25
canceled (kopano.Appointment attribute), 7
categories (kopano.Item attribute), 21
cc (kopano.Item attribute), 21
changekey (kopano.Item attribute), 21
children (kopano.Contact attribute), 12
clear_cache() (kopano.Server method), 31
codepage (kopano.Item attribute), 21
color (kopano.Appointment attribute), 7
common_views (kopano.Store attribute), 35
companies() (kopano.Server method), 31
Company (class in kopano), 10
company (kopano.Store attribute), 35
company (kopano.User attribute), 39
company() (kopano.Server method), 31
company_name (kopano.Contact attribute), 12
companyid (kopano.Company attribute), 10
Config (class in kopano), 12
config (kopano.Service attribute), 34
config_item() (kopano.Store method), 35
ConfigError (class in kopano), 14
conflicts (kopano.AutoAccept attribute), 9
Contact (class in kopano), 12
contacts (kopano.Store attribute), 35
container_class (kopano.Folder attribute), 15
content_id (kopano.Attachment attribute), 8
content_location (kopano.Attachment attribute), 8
conversationid (kopano.Item attribute), 21

copy() (kopano.Folder method), 15
copy() (kopano.Item method), 21
count (kopano.Folder attribute), 15
count (kopano.Recurrence attribute), 29
count (kopano.Table attribute), 39
create_attachment() (kopano.Item method), 21
create_attendee() (kopano.Appointment method), 7
create_company() (kopano.Server method), 31
create_folder() (kopano.Folder method), 15
create_folder() (kopano.Store method), 35
create_group() (kopano.Company method), 10
create_group() (kopano.Server method), 31
create_item() (kopano.Folder method), 15
create_item() (kopano.Item method), 21
create_prop() (kopano.Properties method), 28
create_public_store() (kopano.Company method), 10
create_public_store() (kopano.Server method), 31
create_rule() (kopano.Folder method), 15
create_store() (kopano.Server method), 31
create_store() (kopano.User method), 40
create_user() (kopano.Company method), 10
create_user() (kopano.Server method), 31
created (kopano.Folder attribute), 16
created (kopano.Item attribute), 21
csv() (kopano.Table method), 39

D

data (kopano.Attachment attribute), 8
data (kopano.Picture attribute), 26
data() (kopano.Table method), 39
decline() (kopano.MeetingRequest method), 25
Delegation (class in kopano), 14
delegation() (kopano.Store method), 35
delegations() (kopano.Store method), 36
delegations_dumps() (kopano.Store method), 36
delegations_loads() (kopano.Store method), 36
delete() (kopano.Attachment method), 8
delete() (kopano.Folder method), 16
delete() (kopano.Item method), 21
delete() (kopano.Server method), 32
delete() (kopano.Store method), 36
deleted (kopano.Folder attribute), 16
delivery_receipt (kopano.Item attribute), 21
department (kopano.Contact attribute), 12
dict_rows() (kopano.Table method), 39
DistList (class in kopano), 14
distlist (kopano.Item attribute), 21
drafts (kopano.Store attribute), 36
dump() (kopano.Item method), 22
dumps() (kopano.Folder method), 16
dumps() (kopano.Item method), 22
dumps() (kopano.Store method), 36
DuplicateError (class in kopano), 14

E

email (kopano.Address attribute), 7
email (kopano.Contact attribute), 13
email (kopano.Group attribute), 20
email (kopano.User attribute), 40
email2 (kopano.Contact attribute), 13
email3 (kopano.Contact attribute), 13
embedded (kopano.Attachment attribute), 9
eml() (kopano.Item method), 22
empty() (kopano.Folder method), 16
enabled (kopano.AutoAccept attribute), 10
enabled (kopano.AutoProcess attribute), 10
enabled (kopano.OutOfOffice attribute), 26
encoding (kopano.Item attribute), 22
end (kopano.Appointment attribute), 8
end (kopano.FreeBusyBlock attribute), 19
end (kopano.OutOfOffice attribute), 26
end (kopano.Recurrence attribute), 29
entryid (kopano.Address attribute), 7
entryid (kopano.Folder attribute), 16
entryid (kopano.Item attribute), 22
entryid (kopano.Store attribute), 36
Error (class in kopano), 14
event_type (kopano.Notification attribute), 25

F

favorites() (kopano.Store method), 36
features (kopano.User attribute), 40
file_as (kopano.Contact attribute), 13
filename (kopano.Attachment attribute), 9
filtered_html (kopano.Item attribute), 22
findroot (kopano.Store attribute), 36
first_name (kopano.Contact attribute), 13
first_name (kopano.User attribute), 40
first_weekday (kopano.Recurrence attribute), 29
flags (kopano.Delegation attribute), 14
Folder (class in kopano), 15
folder (kopano.Item attribute), 22
folder() (kopano.Folder method), 16
folder() (kopano.Store method), 36
folders() (kopano.Folder method), 16
folders() (kopano.Store method), 36
FreeBusy (class in kopano), 19
freebusy (kopano.Store attribute), 36
FreeBusyBlock (class in kopano), 19
from_ (kopano.Item attribute), 22
fullname (kopano.User attribute), 40

G

generation (kopano.Contact attribute), 13
get() (kopano.Properties method), 28
get_company() (kopano.Server method), 32
get_folder() (kopano.Folder method), 16
get_folder() (kopano.Store method), 36
get_group() (kopano.Company method), 11
get_prop() (kopano.Properties method), 28
get_stat() (kopano.Server method), 32
get_store() (kopano.Server method), 32
get_user() (kopano.Company method), 11
get_user() (kopano.Server method), 32
Group (class in kopano), 20
group() (kopano.Company method), 11

group() (kopano.Server method), 32
groupid (kopano.Group attribute), 20
groups() (kopano.Company method), 11
groups() (kopano.Group method), 20
groups() (kopano.Server method), 32
groups() (kopano.User method), 40
guid (kopano.Property attribute), 27
guid (kopano.Server attribute), 32
guid (kopano.Store attribute), 36

H

hard_limit (kopano.Quota attribute), 28
has_attachments (kopano.Item attribute), 22
header (kopano.Table attribute), 39
header() (kopano.Item method), 22
headers() (kopano.Item method), 22
height (kopano.Picture attribute), 26
hidden (kopano.Attachment attribute), 9
hidden (kopano.Company attribute), 11
hidden (kopano.Group attribute), 20
hidden (kopano.User attribute), 40
hierarchyid (kopano.Attachment attribute), 9
hierarchyid (kopano.Folder attribute), 16
hierarchyid (kopano.Item attribute), 22
hierarchyid (kopano.Store attribute), 36
home_address (kopano.Contact attribute), 13
home_phones (kopano.Contact attribute), 13
home_server (kopano.User attribute), 40
hook() (kopano.User method), 40
hook_archive() (kopano.User method), 40
hook_public_store() (kopano.Company method), 11
hook_public_store() (kopano.Server method), 32
html (kopano.Item attribute), 22
html_utf8 (kopano.Item attribute), 22

I

icaluid (kopano.Appointment attribute), 8
ics() (kopano.Folder method), 16
ics() (kopano.Item method), 22
id_ (kopano.Property attribute), 27
idname (kopano.Property attribute), 27
im_addresses (kopano.Contact attribute), 13
inbox (kopano.Store attribute), 37
index (kopano.Recurrence attribute), 29
index() (kopano.Table method), 39
initials (kopano.Contact attribute), 13
inline (kopano.Attachment attribute), 9
interval (kopano.Recurrence attribute), 29
is_cancellation (kopano.MeetingRequest attribute), 25
is_distlist (kopano.Item attribute), 22
is_meetingrequest (kopano.Item attribute), 22
is_request (kopano.MeetingRequest attribute), 25
is_response (kopano.MeetingRequest attribute), 25
Item (class in kopano), 21
item (kopano.Attachment attribute), 9
item (kopano.Occurrence attribute), 30
item() (kopano.Folder method), 16
item() (kopano.Store method), 37
items() (kopano.Folder method), 17
items() (kopano.Item method), 22

J

job_title (kopano.Contact attribute), 13
job_title (kopano.User attribute), 40
journal (kopano.Store attribute), 37
junk (kopano.Store attribute), 37

K

kind (kopano.Property attribute), 27
kindname (kopano.Property attribute), 27

L

last_logoff (kopano.Store attribute), 37
last_logon (kopano.Store attribute), 37
last_modified (kopano.Attachment attribute), 9
last_modified (kopano.Folder attribute), 17
last_modified (kopano.Item attribute), 22
last_name (kopano.Contact attribute), 13
last_name (kopano.User attribute), 40
load() (kopano.Item method), 22
loads() (kopano.Folder method), 17
loads() (kopano.Item method), 23
loads() (kopano.Store method), 37
location (kopano.Appointment attribute), 8
log (kopano.Service attribute), 34
LogonError (class in kopano), 14

M

maildir() (kopano.Folder method), 17
manager (kopano.Contact attribute), 13
mapiobj (kopano.Attachment attribute), 9
mapiobj (kopano.Company attribute), 11
mapiobj (kopano.Folder attribute), 17
mapiobj (kopano.Group attribute), 20
mapiobj (kopano.Item attribute), 23
mapiobj (kopano.Notification attribute), 25
mapiobj (kopano.Store attribute), 37
mapiobj (kopano.User attribute), 40
match() (kopano.Item method), 23
mbox() (kopano.Folder method), 17
MeetingRequest (class in kopano), 24
meetingrequest (kopano.Item attribute), 23
member (kopano.Permission attribute), 26
members() (kopano.DistList method), 14
members() (kopano.Group method), 20
message (kopano.OutOfOffice attribute), 26
message_class (kopano.Item attribute), 23
messageid (kopano.Item attribute), 23
middle_name (kopano.Contact attribute), 13
mimetype (kopano.Attachment attribute), 9
mimetype (kopano.Picture attribute), 26
mobile_phone (kopano.Contact attribute), 13
mobile_phone (kopano.User attribute), 40
month (kopano.Recurrence attribute), 29
monthday (kopano.Recurrence attribute), 29
move() (kopano.Folder method), 17

move() (kopano.Item method), 23
multitenant (kopano.Server attribute), 32

N

name (kopano.Address attribute), 7
name (kopano.Company attribute), 11
name (kopano.Folder attribute), 17
name (kopano.Group attribute), 20
name (kopano.Item attribute), 23
name (kopano.Picture attribute), 26
name (kopano.Property attribute), 27
name (kopano.Rule attribute), 30
name (kopano.Server attribute), 32
name (kopano.Store attribute), 37
name (kopano.User attribute), 40
named (kopano.Property attribute), 27
namespace (kopano.Property attribute), 27
nickname (kopano.Contact attribute), 13
nodes() (kopano.Server method), 32
normalized_subject (kopano.Item attribute), 23
notes (kopano.Store attribute), 37
NotFoundError (class in kopano), 14
Notification (class in kopano), 25
NotSupportedError (class in kopano), 14
number (kopano.Attachment attribute), 9

O

object (kopano.Notification attribute), 25
object_type (kopano.Notification attribute), 25
Occurrence (class in kopano), 30
occurrences() (kopano.Appointment method), 8
occurrences() (kopano.Folder method), 17
occurrences() (kopano.Recurrence method), 29
office_location (kopano.Contact attribute), 13
office_location (kopano.User attribute), 40
orphan (kopano.Store attribute), 37
other_address (kopano.Contact attribute), 13
outbox (kopano.Store attribute), 37
OutOfOffice (class in kopano), 25
outofoffice (kopano.Store attribute), 37

P

parent (kopano.Attachment attribute), 9
parent (kopano.Folder attribute), 17
password (kopano.User attribute), 40
path (kopano.Folder attribute), 17
pattern (kopano.Recurrence attribute), 29
period_desc (kopano.OutOfOffice attribute), 26
Permission (class in kopano), 26
permission() (kopano.Folder method), 17
permission() (kopano.Store method), 37
permissions() (kopano.Folder method), 18
permissions() (kopano.Store method), 37
permissions_dumps() (kopano.Folder method), 18
permissions_dumps() (kopano.Store method), 37
permissions_loads() (kopano.Folder method), 18
permissions_loads() (kopano.Store method), 37
photo (kopano.Contact attribute), 13
photo (kopano.User attribute), 40
Picture (class in kopano), 26
primary_folder (kopano.Folder attribute), 18
primary_item (kopano.Item attribute), 23
primary_store (kopano.Folder attribute), 18
private (kopano.Item attribute), 23
process_cancellation() (kopano.MeetingRequest method), 25
process_response() (kopano.MeetingRequest method), 25
processed (kopano.MeetingRequest attribute), 25
profession (kopano.Contact attribute), 13
prop() (kopano.Properties method), 28
Properties (class in kopano), 28
Property (class in kopano), 27
props() (kopano.Address method), 7
props() (kopano.Properties method), 28
proptag (kopano.Property attribute), 27
provider (kopano.Rule attribute), 30
public (kopano.Store attribute), 37
public_store (kopano.Company attribute), 11
public_store (kopano.Server attribute), 32
publish() (kopano.FreeBusy method), 20
purge_deferred() (kopano.Server method), 32
purge_softdeletes() (kopano.Server method), 33

Q

Quota (class in kopano), 28
quota (kopano.Company attribute), 11
quota (kopano.User attribute), 40

R

range_type (kopano.Recurrence attribute), 29
read (kopano.Item attribute), 23
read_ics() (kopano.Folder method), 18
read_maildir() (kopano.Folder method), 18
read_receipt (kopano.Item attribute), 23
readmbox() (kopano.Folder method), 18
received (kopano.Item attribute), 23
recipients() (kopano.Item method), 23
recipients() (kopano.Quota method), 28
Recurrence (class in kopano), 29
recurrence (kopano.Appointment attribute), 8
recurring (kopano.Appointment attribute), 8
recurring (kopano.AutoAccept attribute), 10
reminder (kopano.Appointment attribute), 8
reminder_minutes (kopano.Appointment attribute), 8
reminders (kopano.Store attribute), 37
remove_admin() (kopano.Company method), 11
remove_company() (kopano.Server method), 33
remove_feature() (kopano.User method), 40
remove_group() (kopano.Server method), 33
remove_recipient() (kopano.Quota method), 28
remove_send_as() (kopano.Group method), 20
remove_send_as() (kopano.User method), 40
remove_store() (kopano.Server method), 33
remove_user() (kopano.Group method), 20
remove_user() (kopano.Server method), 33

remove_view() (kopano.Company method), 11
reply() (kopano.Item method), 23
replyto (kopano.Item attribute), 23
response (kopano.Attendee attribute), 9
response_requested (kopano.Appointment attribute), 8
response_requested (kopano.MeetingRequest attribute), 25
response_time (kopano.Attendee attribute), 9
restore() (kopano.Item method), 23
Restriction (class in kopano), 30
restriction (kopano.Rule attribute), 30
rights (kopano.Permission attribute), 26
rights() (kopano.Folder method), 18
root (kopano.Store attribute), 37
rows() (kopano.Table method), 39
rss (kopano.Store attribute), 38
rtf (kopano.Item attribute), 23
Rule (class in kopano), 30
rules() (kopano.Folder method), 18
rules() (kopano.User method), 41
rules_dumps() (kopano.Folder method), 18
rules_loads() (kopano.Folder method), 18

S

scale() (kopano.Picture method), 26
searches() (kopano.Store method), 38
searchkey (kopano.Item attribute), 24
see_private (kopano.Delegation attribute), 14
send() (kopano.Item method), 24
send_as() (kopano.Group method), 20
send_as() (kopano.User method), 41
send_copy (kopano.Delegation attribute), 14
send_only_to_delegates (kopano.Store attribute), 38
sender (kopano.Item attribute), 24
sensitivity (kopano.Item attribute), 24
sent (kopano.Item attribute), 24
sentmail (kopano.Store attribute), 38
Server (class in kopano), 30
server (kopano.Service attribute), 34
Service (class in kopano), 34
set_photo() (kopano.Contact method), 13
settings_dumps() (kopano.Folder method), 18
settings_dumps() (kopano.Store method), 38
settings_loads() (kopano.Folder method), 18
settings_loads() (kopano.Store method), 38
size (kopano.Attachment attribute), 9
size (kopano.Folder attribute), 18
size (kopano.Item attribute), 24
size (kopano.Picture attribute), 27
size (kopano.Store attribute), 38
soft_limit (kopano.Quota attribute), 28
sort() (kopano.Table method), 39
sourcekey (kopano.Folder attribute), 18
sourcekey (kopano.Item attribute), 24
spouse (kopano.Contact attribute), 14
start (kopano.Appointment attribute), 8
start (kopano.FreeBusyBlock attribute), 19
start (kopano.OutOfOffice attribute), 26
start (kopano.Recurrence attribute), 29
start() (kopano.Service method), 34
stat() (kopano.Server method), 33
state (kopano.Folder attribute), 18
state (kopano.Server attribute), 33
stats() (kopano.Server method), 33
status (kopano.FreeBusyBlock attribute), 19
Store (class in kopano), 35
store (kopano.User attribute), 41
store() (kopano.Company method), 11
store() (kopano.Server method), 33
stores() (kopano.Company method), 11
stores() (kopano.Server method), 33
strid (kopano.Property attribute), 27
strval (kopano.Property attribute), 27
stubbed (kopano.Item attribute), 24
subfolder_count (kopano.Folder attribute), 18
subfolder_count_recursive (kopano.Folder attribute), 18
subject (kopano.Item attribute), 24
subject (kopano.OutOfOffice attribute), 26
subscribe() (kopano.Folder method), 19
subscribe() (kopano.Store method), 38
subtree (kopano.Store attribute), 38
suggested_contacts (kopano.Store attribute), 38
sync() (kopano.Folder method), 19
sync() (kopano.Server method), 33
sync_gab() (kopano.Server method), 33
sync_hierarchy() (kopano.Folder method), 19
sync_users() (kopano.Server method), 34

T

Table (class in kopano), 39
tasks (kopano.Store attribute), 38
text (kopano.Item attribute), 24
text() (kopano.Table method), 39
timezone (kopano.Appointment attribute), 8
title (kopano.Contact attribute), 14
to (kopano.Item attribute), 24
todo_search (kopano.Store attribute), 38
type_ (kopano.Attendee attribute), 9
type_ (kopano.Folder attribute), 19
type_ (kopano.Property attribute), 27
type_ (kopano.Store attribute), 38
typename (kopano.Property attribute), 27
tzinfo (kopano.Appointment attribute), 8

U

unhook() (kopano.User method), 41
unhook_archive() (kopano.User method), 41
unhook_public_store() (kopano.Company method), 11
unhook_public_store() (kopano.Server method), 34
unread (kopano.Folder attribute), 19
unsubscribe() (kopano.Folder method), 19
update() (kopano.Quota method), 28
update_counter (kopano.MeetingRequest attribute), 25
urgency (kopano.Item attribute), 24
use_default (kopano.Quota attribute), 29

User (class in kopano), 39
user (kopano.Delegation attribute), 14
user (kopano.Item attribute), 24
user (kopano.Store attribute), 38
user() (kopano.Company method), 11
user() (kopano.Server method), 34
userid (kopano.User attribute), 41
users() (kopano.Company method), 12
users() (kopano.Group method), 20
users() (kopano.Server method), 34

V

value (kopano.Property attribute), 27
vcf() (kopano.Item method), 24
views (kopano.Store attribute), 38
views() (kopano.Company method), 12

W

warning_limit (kopano.Quota attribute), 29
wastebasket (kopano.Store attribute), 38
webapp_settings (kopano.Store attribute), 38
weekdays (kopano.Recurrence attribute), 29
width (kopano.Picture attribute), 27

Y

yomi_company_name (kopano.Contact attribute), 14
yomi_first_name (kopano.Contact attribute), 14
yomi_last_name (kopano.Contact attribute), 14