JavaMail Adapter

On this Page:

Related Pages:

- Documentation up to version 1.7.0 (obsolete)
- POP3 Adapter
- SMTP Adapter

POP3 Adapter and SMTP Adapter provide basic email receiving and sending. However, these adapters do not offer advanced features such as imap/imaps/smtps support, ssl/starttls encryption or selective access by folder, date and time, subject etc. In scenarios where such advanced features are required, we recommend the use of the **JavaMail** library.

JavaMail library aims at providing the means to interact with all modern email servers through a common API with the convenience of a pre-packaged, ready to use Bridge xUML library. The API is available through one library class that defines all operations, whose parameters refer to the underlying Java implementation. After importing the JavaMail library, there is no need to import any further components, java classes etc. - you can start using it in your project right away.

Compatibility

JavaMail Library 3 The current version 3 introduces a breaking change with version 2 regarding the structure of the connection, so all services which imported version 2.x are affected. In addition, the deleteMessagesFiltered operation now returns an Integer indicating the number of deleted messages.

JavaMail Library 2 The previous version 2 introduced breaking changes with version 1, and dropped support for Bridge Version 6 and earlier.

JavaMail Library 1If you need compatibility with Release 6, please refer to version 1.7.0.
Documentation of version 1 is available on Documentation up to version 1.7.0 (obsolete).
Version 2 of the JavaMail library contains also all deprecated classes from version 1.7.0 to ease migration. However, these are now longer supported and will eventually be dropped.

Example File (Builder project JavaMail):

<your example path>\Libraries\ JavaMail\uml\MailExample.xml

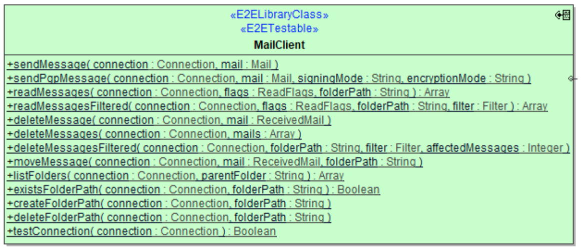
Features

Feature	Support	Comment
Protocols	IMAP, IMAPS, POP3, POP3S, STMP, SMTPS	With username/password authentication
	EWS (MS Exchange)	With username/password (BasicAuth) or OAuth login (for Outlook 365 only) The EWS Java SDK has meanwhile been deprecated, and been succeeded by Microsoft's Java SDK for the Graph API. This library will likely be migrated to use the Graph API in the future, depending on demand. For the time being, the EWS API is used.
Security	SSL and STARTTLS for IMAP/POP3/SMTP, HTTPS for EWS	For host verification the corresponding certificate must either be known by the underlying JVM or provided within the keystore on the connection (see Keys and Certificates)
Signed Mail	S/MIME (RFC 3852) and PGP (RFC 4880, RFC 3156)	• Only for receiving email. • Sending signed emails is not yet supported. Receiving PGP "compressed" signed messages is also not yet supported (PGP signature verification requires the public keys of the signer(s), see Keys and Certificates).

Encrypted Mail	OpenPGP only	- OpenPGP encrypted messages are supported (encryption and decryption). - S/MIME encrypted messages will be read, but not decrypted. You will receive the envelope without content and the status will indicate success=false.
Filtered Reading	receive date, subject, sender, attachment name, and unread status	
Attachments	append file attachments on sending, and receiving attachments	Image attachments can be sent as inline images using CIDs in the HTML message.

Supported Operations

All operations can be accessed through the **MailClient** class.



```
«E2ELibraryClass»
«E2ETestable»
MailClient
+sendMessage( connection : Connection, mail : Mail )
+sendPgpMessage( connection : Connection, mail : Mail, signingMode : String, encryptionMode : String )
+readMessages( connection : Connection, flags : ReadFlags, folderPath : String ) : Array
+readMessagesFiltered( connection : Connection, flags : ReadFlags, folderPath : String, filter : Filter ) : Array
+deleteMessage( connection : Connection, mail : ReceivedMail )
+deleteMessages( connection : Connection, mails : Array )
+deleteMessagesFiltered( connection : Connection, folderPath : String, filter : Filter, affectedMessages : Integer )
+moveMessage( connection : Connection, mail : ReceivedMail, folderPath : String )
+listFolders( connection : Connection, parentFolder : String ) : Array
+existsFolderPath( connection : Connection, folderPath : String ) : Boolean
+createFolderPath( connection : Connection, folderPath : String )
+deleteFolderPath( connection : Connection, folderPath : String )
+testConnection( connection : Connection ) : Boolean
```

All operations require a connection object. Details on it and its attributes are explained on Mail Server Connection.

The library supports

- Receiving and Processing Emails from IMAP, POP3, and MS Exchange Servers. Processing messages typically means either deleting messages or moving them to specific folders
- Sending Messages via SMTP and MS Exchange
- Creation and deletion of folders.
- For verification of PGP signatures and SSL certificates provided by mail services, the library needs access to the relevant certs and public keys.
  More about this on Keys and Certificates.

Operations may throw exceptions. In cases where the underlying Java code throws exceptions that are anticipated, these are translated into xUML exception signatures. For unexpected (not anticipated) exceptions in the Java code, a generic xUML exception is thrown.

Folder Handling

Operations expecting a **folderPath** argument will treat this the following way:

Folder Path	Description	Example
NULL	This is interpreted as meaning the `<Inbox>` folder.	
path beginning with `/`	This is interpreted as a path from the mail connections root folder.	`/Archive/Orders`
path not beginning with `/`	This is interpreted as a relative path to the `<Inbox>` folder, so translates to `/<Inbox>/Archive/Orders`.	`Archive/Orders`

The intention of this behavior is to abstract away the fact that `<Inbox>` has many different names depending on the connection, i.e. IMAP typically uses `INBOX`, Exchange calls it `Inbox`, and a German Exchange calls it `Posteingang`.

Also note that **POP3 does not support folders** other than `<Inbox>`. Hence when you specify a POP3 connection, **folderPath** parameters other than NULL will raise an exception.

Used Open Source Libraries

This library makes use of a number of OpenSource libraries. The following table lists the main dependent Java Open Source libraries that are used.

Library	Information	Version	License
EWS Managed API	https://github.com/OfficeDev/ews-java-api	2.0	MIT License
Oracle Java Mail	https://javaee.github.io/javamail/	1.6.2	CDDL/GPLv2+CE
Bouncy Castle	https://www.bouncycastle.org/index.html	1.64	Bouncy Castle Licence (MIT License)
Apache HTTP Components	https://hc.apache.org/	4.5.3 (client) / 4.4.6 (core)	Apache License 2.0
Apache Commons Email	https://commons.apache.org/proper/commons-email/	1.5	Apache License 2.0
PGPainless	https://github.com/pgpainless/pgpainless	1.0.1	Apache License 2.0