ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-net



Peter Reutemann

November 21, 2026

©2012-2026



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato



Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-sa/4.0/

Contents

1 Email **7**
1.1 SMTP 7
1.1.1 Global SMTP settings 7
1.1.2 Command-line 8
1.2 Flow components 9
1.3 Addressbook 10
1.4 Support email 10

2 SSH **11**

3 SSL **13**

4 SMB **15**

5 Remote commands **17**

6 Miscellaneous **19**

Bibliography **21**

List of Figures

1.1 Email preferences 8
1.2 Email address book 10

3.1 SSL preferences 13

Chapter 1

Email

Flows are ideal for being run as background jobs ("-headless" flag). For example, importing or processing data in batches can be done at night time. Of course, you want to be notified if something went wrong or some predictions are off. Adding the *SendEmail* sink to existing flows, allows for automatic sending of emails that were generated by the *CreateEmail* transformer: if everything is OK then send an email to the customer, otherwise send an email to sysadmin. The *CreateEmail* actor adds all incoming file names, e.g., the array output of a *DirectoryLister* source, as attachments before sending the email off to the specified recipients. You can also define a custom subject and body. Variables can be placed in subject and body alike, as they get expanded when creating the email.

With the built-in IMAP[3] support, you can also monitor email accounts and react to messages and/or their attachments.

1.1 SMTP

In order to be able to send emails, ADAMS needs to know what SMTP[2] server to connect to. The following example configures ADAMS to send emails using a Gmail account[4].

There are two ways of configuring SMTP:

- *globally* – using the preferences
- *per flow* – using the *SMTPConnection* standalone actor

1.1.1 Global SMTP settings

For configuring email globally, use the dialog available from the main menu (*Program* → *Preferences* → *Email*), as depicted in Figure 1.1. The placeholders *USER* and *PASSWORD* have to be replaced with the actual user credentials and *YOUR NAME* with the actual user's name, of course.

Alternatively, you can also simply create a properties file in the `$HOME/.adams` directory, called `Email.props`. The content for the Gmail setup[5] would look like this:

```
Enabled=true
SmtpUseSsl=false
```

```
SmtpStartTls=true
SmtpProtocols=TLSv1.2
SmtpPassword=YOUR_APP_PASSWORD  # https://support.google.com/accounts/answer/185833?hl=en
SmtpServer=smtp.gmail.com
SmtpTimeout=30000
SmtpPort=587
SmtpRequiresAuthentication=true
SmtpUser=USER
DefaultAddressFrom=YOUR NAME <USER@gmail.com>
DefaultSignature=
```

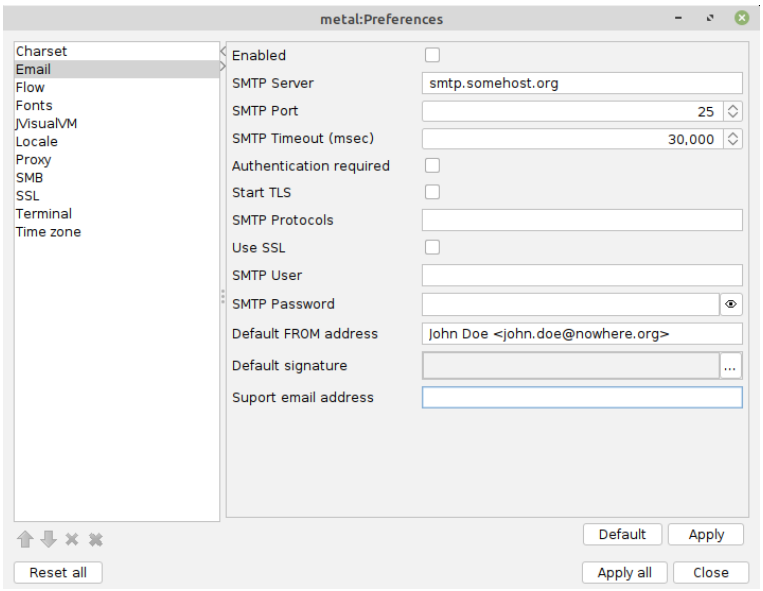


Figure 1.1: Email preferences

1.1.2 Command-line

Using the *adams.core.net.SimpleMailer* command-line tool, you can read and send previously saved emails. The following command-line loads/sends all files in the directory */some/where* that end with *.props* using the *PropertiesEmailFileReader* reader:

```
java -cp lib/* adams.core.net.SimpleMailer
  -env adams.env.Environment
  -reader adams.data.io.input.PropertiesEmailFileReader
  -watch-dir /some/where
  -regexp ".*.props"
  -D 1
```

After a file got sucessfully loaded and sent, it gets (by default) the *.sent* extension appended. If the process failed, the *.failed* extension is used (by default). Use the *-h* or *-help* option on the command-line to list all the available options.

1.2 Flow components

The following standalone actors are available:

- *SMTPConnection* – for configuring a SMTP server connection.
- *IMAPConnection* - defines the IMAP server parameters.

The following source actors are available:

- *IMAPOperation* - for IMAP operations that only generate output

The following transformers are available:

- *CreateEmail* – creates an Email object. Interprets incoming files as attachments.
- *EmailFileReader* – reads email files with the specified email reader.
- *IMAPOperation* - for operations that take input and generate output
- *SaveEmailAttachments* - saves attachments of received emails to the specified directory

The following sinks are available:

- *EmailFileWriter* – writes Email objects to files using the specified email writer.
- *EmailViewer* – displays an Email object; can be used in conjunction with the *DisplayPanelManager* as well.
- *SendEmail* – sends Email objects to a SMTP server.

The following conversions are available:

- *ReceivedEmailToEmail* - changes the type of email object
- *ReceivedEmailToMap* - turns the properties of a received email into a Java map, to make them easier to access in the flow.

SMTPConnection/IMAPConnection
If you do not want to store the password with the flow - after all, the password is only obscured with base64 encoding - you can enable the *promptForPassword* option. This will prompt the user with a dialog for entering a password to be used for the connection.

For Gmail, you will need to enable app passwords[5].

1.3 Addressbook

ADAMS also offers a very simple addressbook for emails. Figure 1.2 shows a screenshot of it.

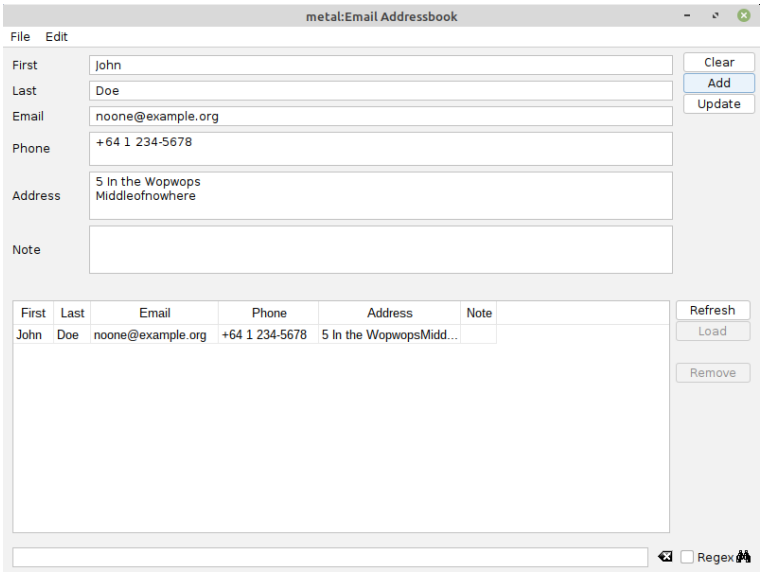


Figure 1.2: Email address book

1.4 Support email

In order to make it easier for users or clients, you can set up a *support email*, which makes it easy to send through error reports. These error reports get the *system info* and current content of the *console window* automatically attached.

To enable this feature, configure the email setup through the preferences and fill in a *Support email address*. After the application has been restarted, you get a *Send error report* menu item in the *Program* menu and the *Send error report* action in the flow's notification area is enabled as well (this one also attaches the flow that generated the error for further analysis).

Chapter 2

SSH

ADAMS also comes with SSH support, thanks to the JSch library [1]. This allows for secure remote access. At the time of writing, RSA [2] and DSA [3] are supported for public key encryption schemes.

The following actors are available:

- *SSHConnection* – standalone actor that defines the server connection (user/password or public key authentication).
- *SSHExec* – for executing remote commands.
- *ScpFrom* – for copying a file *from* a remote host using secure copy.
- *ScpTo* – for copying a file *to* a remote host using secure copy.
- *SFTPDelete* – for delete a remote file via secure FTP.
- *SFTPGet* – for obtaining a remote file via secure FTP.
- *SFTPSend* – for sending a file via secure FTP to a remote host.

The *SSHConnection* standalone allows you to prompt the user as runtime to enter a password, in case you do not want to store the connection's password with the flow setup. The user has to enter the password through a dialog that pops up when the flow gets executed.

There are also several *searchlets* available for the *FileSystemSearch* source actor that allow the listing of remote files.

[1] `http://www.jcraft.com/jsch/`
[2] `http://en.wikipedia.org/wiki/RSA_%28algorithm%29`
[3] `http://en.wikipedia.org/wiki/Digital_Signature_Algorithm`

Chapter 3

SSL

How ADAMS handles SSL, e.g., for HTTPS connections, can be configured in the *SSL* preferences. Figure 3.1 shows a screenshot of the default settings.

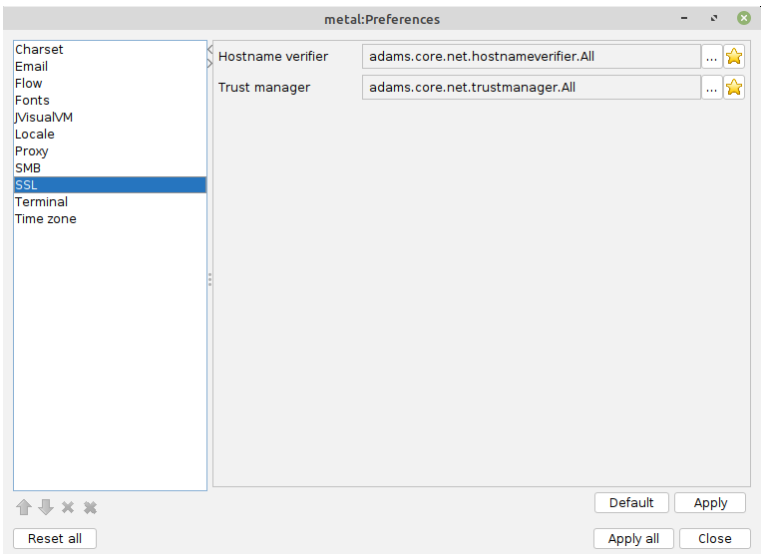


Figure 3.1: SSL preferences

Chapter 4

SMB

SMB support is available through the following actors are available:

- *SMBConnection* – standalone actor that defines the server connection (domain/user/password).
- *SMBGet* – for obtaining a remote file from a Windows host.
- *SMBSend* – for sending a file to a remote Windows host.

There are also several *searchlets* available for the *FileSystemSearch* source actor that allow the listing of remote files.

Chapter 5

Remote commands

The *adams-net* module adds several connection schemes for the remote command framework:

- *ScpConnection* – sends commands as files using secure copy (*scp*).
- *SSHConnection* – offers SSH tunnelling, useful when remote servers are locked down.

Chapter 6

Miscellaneous

Some other basic, but useful actors are the following:

- *Browser* – opens the system's default browser with the specified URL.
- *DownloadFile* – downloads a file via HTTP and saves it to disk.
- *DownloadContent* – downloads (textual) content via HTTP and forwards it.[1]
- *Html4Display* – this sink displays HTML 4 code (and supports clicking on links)
- *HttpPostFile* – the transformer posts a file as multipart/form-data to the specified URL and returns the response.
- *HttpRequest* – the source allows accessing URLs (POST/GET, sending data, cookies), returns the reponse as text; the transformer allows sending the incoming text via HTTP request (and optional headers).
- *MimeType* – determines the mime-type of files.[2]
- *Socket* – source and sink allow you to receive and send strings and byte arrays via sockets.
- *URLSupplier* – outputs one or more URLs.
- *WebSocketClient* – allows executing of clients that implement the web-socket protocol[3].
- *WebSocketServer* – allows running of websocket servers.

The following conversions are available:

- *Base64ToByteArray* – decodes a Base64 string as byte array.
- *Base64ToString* – decodes a Base64 string as string.
- *ByteArrayToBase64* – encodes a byte array as Base64 string.
- *StringArrayToURLParameters* – converts a string array of key-value pairs into a string to used in a URL.
- *StringToBase64* – encodes a string as Base64.
- *StringToURL* – converts a string into a URL object.
- *URLEncode* – encodes a string into a URL-conform string.
- *URLDecode* – decodes a URL string into a regular string.

[1] adams-net-download_content.flow
[2] adams-net-mimetype.flow
[3] `https://en.wikipedia.org/wiki/WebSocket`

- *URLParametersToStringArray* – converts the parameters of a URL (after the '?') back into a string array of key-value pairs.
- *URLToString* – converts a URL object into a string.

Bibliography

[1] *ADAMS* – Advanced Data mining and Machine learning System
`https://adams.cms.waikato.ac.nz/`

[2] *SMTP* – Simple Mail Transfer Protocol
`https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol`

[3] *IMAP* – Internet Message Access Protocol
`https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol`

[4] *Gmail* – Configuring other mail clients
`https://support.google.com/mail/answer/7104828?hl=en`

[5] *Gmail app passwords* – Sign in with app passwords
`https://support.google.com/mail/answer/185833?hl=en`