Mail

Mail to Database Smart Service

Takes all unread email from a mailbox and adds entries directly to a database table.

- The original email is saved as an EML based on three attachment type options
  - EML with attachments removed
  - EML with attachments included
  - Both (2 EMLs of original email) - EML with attachments removed and EML with attachments included
- Each email attachment is saved as its own document
- The database tables store the Appian document ids, email recipients, subject and body (text and HTML)

This smart service is intended for use with a single mailbox. Where multiple mailboxes exist, work with the mail server administrator to consolidate them together by either:

1. Copying mail from individual mailboxes into a master mailbox
2. Replace the individual mailboxes with email aliases that point to a single mailbox

While it may technically be possible to poll multiple mailbox, this is not recommended or supported by this smart service.
Where either options above cannot be implemented an alternative solution should be sought and this plug-in should not be used.

The designer is responsible for ensuring only one instance of this node is running at one time. The behaviour for multiple instances running at the same time is undefined.
While this node attempts to detect if it is already running, designers should not rely on this behaviour, especially in environments that utilise multiple application servers.

Emoji's are removed from the subject and body before saving to the database. The EML file is unchanged and will contain the emoji's.

The number of email this smart service can read per minutes will depends on the network speed and average size of the email.
Where a good network connection is available and average total size of 2.5MB this node is expected to read around 10-15 mail per minutes (your mileage may vary)

If updating from an older version of the plug-in, the APP_MAIL_POLER table/cdt will need to be updated to add the DOC_ID_WITH_ATTACHMENTS column. Prior versions of the smart service node have been deprecated and the new Configure Mail Server to Database node will need to be used.

Node Inputs

Input	Data Type	Required	Description
Protocol	Text	Yes	Mail server protocol (e.g `imap`, `imaps`, `pop3`, `pop3s`)
Host	Text	Yes	Mail server IP or hostname
Port	Integer	Yes	Mail server port
Mail Folder	Text	No	Mail server folder name. Defaults to "INBOX"
SCS System Key	Text	Yes	The secure credential store key as defined in the Administration Console. Fields for both "username" and "password" are required. Target mailbox (if different to users default) can be set by appending to the username, e.g: domain\user\mailbox
Time Limit	Integer	No	The maximum amount of time in milliseconds to continue processing email while the mailbox is not empty. Default 60,000 (1 minute). If more email remain after the time limit, they will be processed on the next execution of this smart service. If all email is processed before the time limit is reached the node will end early. A minimum of 1 and maximum of 59 minutes can be configured.
On Success	Text	Yes	The action to take on the email after successfully adding to the database
On Failure	Text	Yes	The action to take on the email when failing to add to the database
Attachment Folder	Folder	Yes	The folder to save attachments to
JNDI Name	Text	Yes	The JNDI name of the data source that contains the tables to save the email to

Input	Data Type	Required	Description
Java Mail Keys	Text	Yes	See JavaMail Properties section below
Java Mail Values	Text	Yes	See JavaMail Properties section below
EML Attachment Type	Text	Yes	Options include “Separate Attachments from EML”, “Include Attachments in EML”, or “Both”. Defaults to “Separate Attachments from EML”

JavaMail Properties

This Smart Service uses the JavaMail library to poll the mailbox. There are many different properties (imap, pop3) that can be configured to improve performance and compatibility.

The following properties are always recommended to be set:

Key	Suggested Value	Description
mail.<protocol>.connectiontimeout	10000	Socket connection timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.<protocol>.timeout	10000	Socket read timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.<imap/imaps>.fetchsize	512000	Partial fetch size in bytes. Defaults to 16,000

The following properties are known to help with authentication compatibility for IMAP with Microsoft Exchange. This is needed for access to shared mailboxes (all need to be set to “true”. See https://www.mulesoft.org/jira/browse/MULE-9355 for additional info.

Key	Value	Description
mail.<imap/imaps>.auth.plain.disable	true	If true, prevents use of the AUTHENTICATE PLAIN command. Default is false.
mail.<imap/imaps>.auth.ntlm.disable	true	If true, prevents use of the AUTHENTICATE NTLM command. Default is false.
mail.<imap/imaps>.auth.gssapi.disable	true	If true, prevents use of the AUTHENTICATE GSSAPI command. Default is false.

Node Outputs

Output	Data Type	Description
Error Occurred	Boolean	Returns true if an error has occurred while processing any email
Error Message	Text	The last error message that occurred. Previous errors messages are output to application server log
Mail Count	Integer	Count of email that were successfully read
Error Count	Integer	Count of email that could not be read

Building

Development Build

To create a plug-in jar for testing, run the following Maven command:

- `mvn clean package`
- The jar can be found in `/target`

Release Build

To create a new public release, run the following Maven commands:

- `mvn release:clean`
- `mvn release:prepare -DautoVersionSubmodules=true -DpushChanges=false`
- `git push origin master --tags`
- The release jar can be found in `/target`