MS Graph Mail Poller

This plug-in provides an alternative to sending emails to an Appian process model when inbound email integration is requested. Instead of the email being forwarded to Appian, this plug-in reads the emails directly from the Exchange mailbox using the MS Graph API as described below:

- Reads the mailbox using the MS Graph API
- (Optionally) Converts the email to an EML file stored as an Appian document, with (optionally) attachments removed from it
- Store all email attachments as separate Appian documents
- Store all email metadata (subject, author, recipients, etc...) into a set of tables in the database

Deployment First Time Instructions

To deploy and use the plugin, follow the steps below:

- Download the component from the App Market.
- Deploy the plugin through the admin console of your Appian site or by copying the plugin zip file to the plugins folder
- The zip file "MS Graph Mail Poller applications.zip" contains an example Appian application (version 21.4+) and the admin console settings needed to use the plugin. Use this to understand how to use it.
- Run the SQL scripts that correspond to your database. This will create the necessary tables in the database.
- The application is built for MySQL database. If you use Oracle, update the CDTs in the application with the XDSs found in the app/Oracle folder. ** When upgrading from the EWS Mail Poller, you can reuse the same tables, but be aware that there are addition columns to add.
- Deploy and configure the admin console settings using the properties customization file.

Alternatively, the credentials store required by the plugin can be created manually. It requires the following attributes:

- secret: The client secret belonging to the Azure application for this Appian application
- tenant: The tenant to use. Usually something like `<CLIENT_TENANT_NAME>.onmicrosoft.com`
- appicationID: the application ID (client ID) provided by the Azure team
- proxyUsername: In case you need to connect through an authenticated proxy
- proxyPassword: The password in case you need to connect through an authenticated proxy

In Azure, register an application for Appian to use client credentials. This application needs to have the relevant mailboxes in scope that it will need to connect to. For MSGraph permissions, the following are required: Type: Application Permission: Mail.ReadWrite

Upgrade Instructions

DISCLAIMER: be aware that this version is NOT backwards compatible. The MS Graph API has undergone some larger changes, especially in the Authentication part. Deploying this plugin will thus break any existing mail poller nodes and you have to replace them with the smartservice found in this plugin.

To upgrade to this version of the plugin, follow the steps below.

- Download the component from the App Market.
- Deploy the plugin through the admin console of your Appian site or by copying the plugin zip file to the plugins folder
- The zip file "MS Graph Mail Poller applications.zip" contains an example Appian application and the admin console settings needed to use the plugin. Use this to understand how to use it.
- Run the upgrade SQL script 'MySQLv3.1.0 changes.sql' from the MySQL folder. These will add the columns to the database that are not there yet, but required for this version. If you are not using mySQL / MariaDB, use it to generate a version for your own database platform.
- Update the admin console settings for the secure credential store. One of the changes is in the tenant that should no longer contain the full URL, but only the tenant name (or id). remove the https://login.microsoftonline.com/ part from the tenant, as well as the trailing /.
- Replace the old smartservice with the new one. Remap both input and output parameters.

MS Graph Mail Poller To DB Smart Service

Takes all emails from the given account inbox (including possible subfolders) and adds entries directly to a database table.

- The original email is saved as an EML document (with attachments optionally removed).
- 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), importance and the date fields related to the email. Additionally, an immutable (within the target mailbox) Graph message id is stored, which can be used to interact further with the email message (outside of this plugin).
- Enable/disable generation of the EML
- Toggle eml persistence behaviour between including attachments, without attachments, or potentially both
- Toggle item attachment persistence behaviour between as Appian document and in-line within email body
- Toggle inline image swapping, which if enabled will convert inline HTML images to their corresponding Appian (opaque) document URLs

The database tables store the Appian document ids, email recipients, subject and body (text and HTML), importance and the date fields related to the email. Additionally, an immutable (within the target mailbox) Graph message id is stored, which can be used to interact further with the email message (outside of this plugin).

NB If you choose to store the email as an EML document *and* choose to have the attachments removed from the EML document, you will not see any **mailbox aliases** recorded in the to, cc or bcc fields. Otherwise, if a mailbox alias is used in the to, cc or bcc field it will appear in the EML file and database fields. Mailbox aliases can be useful in identifying the type or purpose of an email received to a common single mailbox, so it is recommended to keep attachments in the stored EML document for this reason.

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. MSGraph will strip any potential unsafe HTML content from the Body if the ContentType is HTML.

The number of emails this smart service can read per minutes will depend on the network speed and average size of the email. It is typically between 10 and 30 per minute. If you expect a fairly constant load of emails, using the Transaction manager to handle the emails themselves is highly recommended

Node Inputs

Input	Data Type	Required	Description
Mailbox	Text	Yes	Name of the account (including domain) for which to read emails. E.g username@domain.com
contentType	Text	Yes	What format to extract from Exchange: Text, HTML or Both. Both is the standard and recommended setting; when selected the marked up body will be saved in the database in the BODY_HTML column and a plain text version will be stored in BODY_TEXT.
SCS External System Key	Text	Yes	The secure credential store key as defined in the Administration Console. Fields for 'secret', 'tenant' and 'applicationId' are required.
Connected By Proxy	Boolean	Yes	Indicates if the connection to MS online services are going through a proxy (only False is yet supported)
Proxy URL	Boolean	No	URL of the proxy, not used
Proxy Port	Boolean	No	Port to connect to the proxy, not used
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 emails 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.
Message Page Size	Integer	Yes	For proper FIFO reading of emails, the page size should be large enough to hold all emails that you will process within the time limit, without loading a page that uses memory unnecessarily. Using a time limit of 60000 (one minute) and a paging size of 50 is the recommended approach.

Input	Data Type	Required	Description
Attachment Folder	Folder	Yes	Folder to save the email (EML) document and all attachments into.
JNDI Name	Text	Yes	JNDI name of the data source that contains the tables to save the email to
Exception Folder Name	Text	Yes	Name of the Exchange folder to move the emails that failed to be processed into. If not found, emails are moved to the folder Junk Emails
Processed Folder Name	Text	Yes	Name of the Exchange folder to move the emails that were processed successfully into. If not found, emails are moved to the folder Deleted Items
Generate Eml	Boolean	No	Indicates whether to generate a copy of the email as an EML file saved as an Appian document. Default True.
Keep File Attachments in Eml	String	No	Indicates whether to keep attachments in the generated EML (INCLUDE), to remove them (SEPARATE) or to keep both copies (BOTH). Default INCLUDE. NB Setting this to SEPARATE will result in mailbox primary email addresses rather than **mailbox aliases** being recorded in the to, cc and bcc fields of the EML and the database row. For example, if the email is sent to complaints@test.com and this is an alias of the comms@test.com mailbox, then comms@test.com will be recorded in the To field. Setting this to INCLUDE or BOTH (and therefore not stripping attachments out of the EML via a copy of the email) will preserve the original alias and complaints@test.com will be stored.
Save Item Attachments as Doc	Boolean	No	Indicates whether to store "item attachments" (e.g. attached email messages and calendar items) from the email as Appian documents. Default False. NB "file attachments" are always stored as Appian documents regardless.
Swap Inline Images	Boolean	No	Indicates whether to swap inline HTML images in the email body to their corresponding Appian (opaque) document URLs. If enabled, this additionally populates the BODY_SWAPPED_IMAGES field with the updated HTML body. Default False.

Debugging

If debugging is required, add the following to your log properties file (<APPIAN_HOME>/deployment/web.war/WEB-INF/resources/appian_log4j.properties).
log4j.logger.com.appiancs.msgraphmail=DEBUG

NB If you have a webserver in front of Appian, make sure to put this on the webserver as well in the documentRoot.

Node Outputs

Output	Data Type	Description
Error Occurred	Boolean	TRUE if an error occurred while processing emails
Error Message	Text	Detailed message of the error
Time Limit Reached	Boolean	TRUE if processing was halted because of passing the time limit set via input parameters
Mail Count	Number	Number of emails successfully processed during the smart service execution
Error Count	Number	Number of errors thrown during the smart service execution