Email Gate User's Guide
javAPRSSrvr 4.3.3b08

Table of Contents

Section 1 - Introduction

EmailGate is written to provide the amateur radio APRS community a simple means to send and receive one line emails.

EmailGate extends net.ae5pl.aprssrvr.MessagingClientRcv. It requires javAPRSSrvr to provide the network interface and do the packet parsing. As a client, there can be multiple EmailGates in a single instance of javAPRSSrvr.

Section 2 - Program Requirements and Description

EmailGate is designed to run on any OS and Java VM supported by javAPRSSrvr.

EmailGate is comprised of a number of classes which Java looks at as objects. The main class is net.ae5pl.emailgate.EmailGate. This class is called at startup, sets parameters, and begins execution. EmailGate assumes jakarta.mail as its basis for email handling.

EmailGate monitors the feed from javAPRSSrvr to determine if a station has sent a message to it. If it receives a message, it responds depending on the message content.

When a message is received from APRS-IS which is properly formatted for processing as email (see section 6), EmailGate will connect to the defined SMTP server and transfer a properly formatted SMTP message to the SMTP server. EmailGate supports both authenticated and non-authenticated connections to the SMTP server. EmailGate will send an ack to the sending station and then send a confirmation/rejection message to the sending station upon completion of the connection to the SMTP server. Logging of all emails sent is generated for tracking purposes.

If POP3 is enabled, EmailGate will continually poll the POP3 server at PollInterval minute intervals. If a message is received, it is parsed for proper format and authorization (see section 6). If IMAP is enabled, EmailGate can create a permanent connection to the IMAP server eliminating the need for polling. However, IMAP does support polling as well.

If an email passes authentication (in POP3OkFile for that station, properly formatted), an APRS message is sent to the requested station 4 times at one minute intervals and a “service” (DSN) message is generated with an action of “relayed” and a status of 2.1.5. If the message does not pass parsing, the message is saved in the directory as an error file for further diagnosis and a rejection DSN email is sent to the email originator (action “failed” and status reflects why the email was rejected). Logging is done for all APRS messages sent and all emails received.

To simplify emailing, a station can create shortcuts for different recipients. The shortcut can be any non-space combination of characters. The station can set this by sending “shortcut email@domain.net” to the server. These are kept in the POP3OkFile so a server restart will not affect them.

Section 3 - Configuration Properties

The configuration properties reside in properties files for each client adjunct, server adjunct, and port. The main properties file is called javaprssrvr.properties by default. You can use any text file for the main properties file if you pass the name into javAPRSSrvr as a command line parameter.

The property names are not case sensitive but the values can be. Defaults are shown below.

NOTE: UNLESS YOU REQUIRE A SETTING OTHER THAN THE DEFAULT, DO NOT INCLUDE ANY PARAMETERS WITH DEFAULT SETTINGS.

List parameters (L) may be defined on the property line or may be defined in a text file with the suffix .lst. If defined on the line, each entry is separated by a semicolon. If defined in a file, each entry is put on a separate line in the .lst file and the file name is the property value. Do not put blank lines in the file. For instance, this could be a definition for ListProperty (example only):

ListProperty=first.aprs.net:1313;second.aprs.net:1313

Or you could have the following 2 lines in a file named hubs.lst:

first.aprs.net:1313
second.aprs.net:1313

with ListProperty=hubs.lst

Properties preceded by a (M) are unchangeable and should not be included in your properties files. They are included in the descriptions below to indicate what common properties are available vs. those that have been forcibly overridden.

javAPRSSrvr Properties

Clients=

(L)This must include the EmailGate properties file.

EmailGate General Properties

Class=

(Deprecated) This must be set to net.ae5pl.emailgate.EmailGate.

ClassPath=

(Deprecated)(L)This must include EmailGate.jar.

ClientType=

Set to email

StationCall=

This is the callsign-SSID for EmailGate.
It must conform to APRS-IS standards, be unique, and it must be different from serverCall (the server's callsign-SSID).
Do not use EMAIL-2 or EMAIL which are reserved.

(M)Upstream=false

(M)FullFeed=false

(M)ReadOnly=false

(M)LocalOnly=false

(M)ServerCommand=

(M)FixedCommand=false

(M)LoginCommands=

(M)MessageHoldTime=-1

(M)LastHeardTime=-1

DSNClass=net.ae5pl.emailgate.DSNProvider

This is the Delivery Status Notification provider class to be used when sending DSN messages to email addresses.
DSN's generated by EmailGate:
2.0.0 – Delivered to station
2.1.9 – Retrieved by callsign
4.2.2 – Rejected by station (message "box" full)
4.4.1 – Delivery delayed (no response from station, queued for later retrieval, see 2.1.9)
4.4.7 – Delayed message timed out and deleted
5.1.3 – Invalid Subject line
5.7.1 – No/invalid userid field (valid Subject line)

OptionsDirectory=.

This is where all options files such as POP3OkList are located. This allows multiple, independent EmailGates to coexist within one instance of javAPRSSrvr.

DomainPrepends=

(L)This prepends the indicated string followed by a space to the subject line of emails sent to the listed domains.
The list is formatted as domain,string with semicolons separating entries. This is primarily designed for Winlink interoperability. To force Winlink whitelisting, set DomainPrepends=winlink.org,//WL2K

POP3OkList=pop3ok.txt

This is where allowed ids for POP3 senders are stored located in OptionsDirectory.
This is a simple text file where each ID is stored on its own line. This list is also the shortcut list for APRS users. The format for each line is CALL,shortcut=email@domain.net

javax.mail Properties

Name	Type	Description
mail.debug	boolean	The initial debug mode. Default is false.
mail.from	String	The return email address of the current user, used by the InternetAddress method getLocalAddress.
mail.mime.address.strict	boolean	The MimeMessage class uses the InternetAddress method parseHeader to parse headers in messages. This property controls the strict flag passed to the parseHeader method. The default is true.
mail.host	String	The default host name of the mail server for both Stores and Transports. Used if the mail.*protocol*.host property isn't set.
mail.store.protocol	String	Specifies the default message access protocol. The Session method getStore() returns a Store object that implements this protocol. By default the first Store provider in the configuration files is returned.
mail.transport.protocol	String	Specifies the default message transport protocol. The Session method getTransport() returns a Transport object that implements this protocol. By default the first Transport provider in the configuration files is returned.
mail.user	String	The default user name to use when connecting to the mail server. Used if the mail.*protocol*.user property isn't set.
mail.*protocol*.class	String	Specifies the fully qualified class name of the provider for the specified protocol. Used in cases where more than one provider for a given protocol exists; this property can be used to specify which provider to use by default. The provider must still be listed in a configuration file.
mail.*protocol*.host	String	The host name of the mail server for the specified protocol. Overrides the mail.host property.
mail.*protocol*.port	int	The port number of the mail server for the specified protocol. If not specified the protocol's default port number is used.
mail.*protocol*.user	String	The user name to use when connecting to mail servers using the specified protocol. Overrides the mail.user property.

The following properties are supported by the EE4J implementation of Jakarta Mail, but are not currently a required part of the specification. The names, types, defaults, and semantics of these properties may change in future releases.

Name	Type	Description
mail.debug.auth	boolean	Include protocol authentication commands (including usernames and passwords) in the debug output. Default is false.

mail.debug.auth.username	boolean	Include the user name in non-protocol debug output. Default is true.
mail.debug.auth.password	boolean	Include the password in non-protocol debug output. Default is false.
mail.transport.protocol.*address-type*	String	Specifies the default message transport protocol for the specified address type. The Session method getTransport(Address) returns a Transport object that implements this protocol when the address is of the specified type (e.g., "rfc822" for standard internet addresses). By default the first Transport configured for that address type is used. This property can be used to override the behavior of the send method of the Transport class so that (for example) the "smtps" protocol is used instead of the "smtp" protocol by setting the property mail.transport.protocol.rfc822 to "smtps".
mail.event.scope	String	Controls the scope of events. (See the javax.mail.event package.) By default, a separate event queue and thread is used for events for each Store, Transport, or Folder. If this property is set to "session", all such events are put in a single event queue processed by a single thread for the current session. If this property is set to "application", all such events are put in a single event queue processed by a single thread for the current application. (Applications are distinguished by their context class loader.)

The Jakarta Mail API also supports several System properties; see the javax.mail.internet package documentation for details.

In addition to printing debugging output as controlled by the Session configuration, the current implementation of classes in this package log the same information using Logger as described in the following table:

Jakarta Mail Loggers

Logger Name	**Logging Level**	**Purpose**
javax.mail	CONFIG	Configuration of the Session
javax.mail	FINE	General debugging output

Transport General Properties

TransportClass=

(Deprecated) Normally set this to net.ae5pl.emailgate.smtpThread.

TransportPassword=

This is the password for the Transport server.
Caution: This password is stored in plain text.

TransportEncPassword=

This is the password for the Transport server. The is the result of encoding the password using encpwd application.

com.sun.mail SMTP Properties

Note that if you're using the "smtps" protocol to access SMTP over SSL, all the properties would be named "mail.smtps.*".

SMTP properties

Name	Type	Description
mail.smtp.user	String	Default user name for SMTP.
mail.smtp.host	String	The SMTP server to connect to.
mail.smtp.port	int	The SMTP server port to connect to, if the connect() method doesn't explicitly specify one. Defaults to 25.
mail.smtp.connectiontimeout	int	Socket connection timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.smtp.timeout	int	Socket read timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.smtp.writetimeout	int	Socket write timeout value in milliseconds. This timeout is implemented by using a java.util.concurrent.ScheduledExecutorService per connection that schedules a thread to close the socket if the timeout expires. Thus, the overhead of using this timeout is one thread per connection. Default is infinite timeout.
mail.smtp.from	String	Email address to use for SMTP MAIL command. This sets the envelope return address. Defaults to msg.getFrom() or InternetAddress.getLocalAddress(). NOTE: mail.smtp.user was previously used for this.
mail.smtp.localhost	String	Local host name used in the SMTP HELO or EHLO command. Defaults to InetAddress.getLocalHost().getHostName(). Should not normally need to be set if your JDK and your name service are configured properly.
mail.smtp.localaddress	String	Local address (host name) to bind to when creating the SMTP socket. Defaults to the address picked by the Socket class. Should not normally need to be set, but useful with multi-homed hosts where it's important to pick a particular local address to bind to.
mail.smtp.localport	int	Local port number to bind to when creating the SMTP socket. Defaults to the port number picked by the Socket class.
mail.smtp.ehlo	boolean	If false, do not attempt to sign on with the EHLO command. Defaults to true. Normally failure of the EHLO command will fallback to the HELO command; this property exists only for

		servers that don't fail EHLO properly or don't implement EHLO properly.
mail.smtp.auth	boolean	If true, attempt to authenticate the user using the AUTH command. Defaults to false.
mail.smtp.auth.mechanisms	String	If set, lists the authentication mechanisms to consider, and the order in which to consider them. Only mechanisms supported by the server and supported by the current implementation will be used. The default is "LOGIN PLAIN DIGEST-MD5 NTLM", which includes all the authentication mechanisms supported by the current implementation except XOAUTH2.
mail.smtp.auth.login.disable	boolean	If true, prevents use of the AUTH LOGIN command. Default is false.
mail.smtp.auth.plain.disable	boolean	If true, prevents use of the AUTH PLAIN command. Default is false.
mail.smtp.auth.digest-md5.disable	boolean	If true, prevents use of the AUTH DIGEST-MD5 command. Default is false.
mail.smtp.auth.ntlm.disable	boolean	If true, prevents use of the AUTH NTLM command. Default is false.
mail.smtp.auth.ntlm.domain	String	The NTLM authentication domain.
mail.smtp.auth.ntlm.flags	int	NTLM protocol-specific flags. See http://curl.haxx.se/rfc/ntlm.html#theNtlmFlags for details.
mail.smtp.auth.xoauth2.disable	boolean	If true, prevents use of the AUTHENTICATE XOAUTH2 command. Because the OAuth 2.0 protocol requires a special access token instead of a password, this mechanism is disabled by default. Enable it by explicitly setting this property to "false" or by setting the "mail.smtp.auth.mechanisms" property to "XOAUTH2".
mail.smtp.submitter	String	The submitter to use in the AUTH tag in the MAIL FROM command. Typically used by a mail relay to pass along information about the original submitter of the message. See also the setSubmitter method of SMTPMessage. Mail clients typically do not use this.
mail.smtp.dsn.notify	String	The NOTIFY option to the RCPT command. Either NEVER, or some combination of SUCCESS, FAILURE, and DELAY (separated by commas).
mail.smtp.dsn.ret	String	The RET option to the MAIL command. Either FULL or HDRS.
mail.smtp.allow8bitmime	boolean	If set to true, and the server supports the 8BITMIME extension, text parts of messages that use the "quoted-printable" or "base64"

		encodings are converted to use "8bit" encoding if they follow the RFC2045 rules for 8bit text.
mail.smtp.sendpartial	boolean	If set to true, and a message has some valid and some invalid addresses, send the message anyway, reporting the partial failure with a SendFailedException. If set to false (the default), the message is not sent to any of the recipients if there is an invalid recipient address.
mail.smtp.sasl.enable	boolean	If set to true, attempt to use the javax.security.sasl package to choose an authentication mechanism for login. Defaults to false.
mail.smtp.sasl.mechanisms	String	A space or comma separated list of SASL mechanism names to try to use.
mail.smtp.sasl.authorizationid	String	The authorization ID to use in the SASL authentication. If not set, the authentication ID (user name) is used.
mail.smtp.sasl.realm	String	The realm to use with DIGEST-MD5 authentication.
mail.smtp.sasl.usecanonicalhostname	boolean	If set to true, the canonical host name returned by InetAddress.getCanonicalHostName is passed to the SASL mechanism, instead of the host name used to connect. Defaults to false.
mail.smtp.quitwait	boolean	If set to false, the QUIT command is sent and the connection is immediately closed. If set to true (the default), causes the transport to wait for the response to the QUIT command.
mail.smtp.quitonsessionreject	boolean	If set to false (the default), on session initiation rejection the QUIT command is not sent and the connection is immediately closed. If set to true, causes the transport to send the QUIT command prior to closing the connection.
mail.smtp.reportsuccess	boolean	If set to true, causes the transport to include an SMTPAddressSucceededException for each address that is successful. Note also that this will cause a SendFailedException to be thrown from the sendMessage method of SMTPTransport even if all addresses were correct and the message was sent successfully.
mail.smtp.socketFactory	SocketFactory	If set to a class that implements the javax.net.SocketFactory interface, this class will be used to create SMTP sockets. Note that this is an instance of a class, not a name, and must be set using the put method, not the setProperty method.
mail.smtp.socketFactory.class	String	If set, specifies the name of a class that implements the javax.net.SocketFactory interface. This class will be used to create SMTP sockets.

mail.smtp.socketFactory.fallback	boolean	If set to true, failure to create a socket using the specified socket factory class will cause the socket to be created using the java.net.Socket class. Defaults to true.
mail.smtp.socketFactory.port	int	Specifies the port to connect to when using the specified socket factory. If not set, the default port will be used.
mail.smtp.ssl.enable	boolean	If set to true, use SSL to connect and use the SSL port by default. Defaults to false for the "smtp" protocol and true for the "smtps" protocol.
mail.smtp.ssl.checkserveridentity	boolean	If set to true, check the server identity as specified by RFC 2595. These additional checks based on the content of the server's certificate are intended to prevent man-in-the-middle attacks. Defaults to false.
mail.smtp.ssl.trust	String	If set, and a socket factory hasn't been specified, enables use of a MailSSLSocketFactory. If set to "*", all hosts are trusted. If set to a whitespace separated list of hosts, those hosts are trusted. Otherwise, trust depends on the certificate the server presents.
mail.smtp.ssl.protocols	string	Specifies the SSL protocols that will be enabled for SSL connections. The property value is a whitespace separated list of tokens acceptable to the javax.net.ssl.SSLSocket.setEnabledProtocols method.
mail.smtp.ssl.ciphersuites	string	Specifies the SSL cipher suites that will be enabled for SSL connections. The property value is a whitespace separated list of tokens acceptable to the javax.net.ssl.SSLSocket.setEnabledCipherSuites method.
mail.smtp.starttls.enable	boolean	If true, enables the use of the STARTTLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands. If the server does not support STARTTLS, the connection continues without the use of TLS; see the mail.smtp.starttls.required property to fail if STARTTLS isn't supported. Note that an appropriate trust store must configured so that the client will trust the server's certificate. Defaults to false.
mail.smtp.starttls.required	boolean	If true, requires the use of the STARTTLS command. If the server doesn't support the STARTTLS command, or the command fails, the connect method will fail. Defaults to false.
mail.smtp.proxy.host	string	Specifies the host name of an HTTP web proxy server that will be used for connections to the mail server.
mail.smtp.proxy.port	string	Specifies the port number for the HTTP web proxy server. Defaults to port 80.

mail.smtp.proxy.user	string	Specifies the user name to use to authenticate with the HTTP web proxy server. By default, no authentication is done.
mail.smtp.proxy.password	string	Specifies the password to use to authenticate with the HTTP web proxy server. By default, no authentication is done.
mail.smtp.socks.host	string	Specifies the host name of a SOCKS5 proxy server that will be used for connections to the mail server.
mail.smtp.socks.port	string	Specifies the port number for the SOCKS5 proxy server. This should only need to be used if the proxy server is not using the standard port number of 1080.
mail.smtp.mailextension	String	Extension string to append to the MAIL command. The extension string can be used to specify standard SMTP service extensions as well as vendor-specific extensions. Typically the application should use the SMTPTransport method supportsExtension to verify that the server supports the desired service extension. See RFC 1869 and other RFCs that define specific extensions.
mail.smtp.userset	boolean	If set to true, use the RSET command instead of the NOOP command in the isConnected method. In some cases sendmail will respond slowly after many NOOP commands; use of RSET avoids this sendmail issue. Defaults to false.
mail.smtp.noop.strict	boolean	If set to true (the default), insist on a 250 response code from the NOOP command to indicate success. The NOOP command is used by the isConnected method to determine if the connection is still alive. Some older servers return the wrong response code on success, some servers don't implement the NOOP command at all and so always return a failure code. Set this property to false to handle servers that are broken in this way. Normally, when a server times out a connection, it will send a 421 response code, which the client will see as the response to the next command it issues. Some servers send the wrong failure response code when timing out a connection. Do not set this property to false when dealing with servers that are broken in this way.

In addition to printing debugging output as controlled by the Session configuration, the com.sun.mail.smtp provider logs the same information using Logger as described in the following table:

Logger Name	**Logging Level**	**Purpose**
com.sun.mail.smtp	CONFIG	Configuration of the SMTPTransport
com.sun.mail.smtp	FINE	General debugging output
com.sun.mail.smtp.protocol	FINEST	Complete protocol trace

Transport Message Properties

SMTPMsgBodyFile=smtpbody.txt

This is the body text to be used in emails originated from APRS. This file will be sent as-is with MessageFormat substitution being done on any {0} (sender's callsign-SSID), {1} (EmailGate's StationCall), and {2} (APRS message text). The conversion of the file from bytes to String is using the UTF-8 character set. The "standard" text file is included with in the .zip file. You may make this anything you want, however, **I warn against placing reply instructions in the text for security and spam-prevention reasons.**

SMTPPositOnlyMsg=Position via APRS

This is the email subject when there is no text in the APRS message.

SMTPNoShortCuts=No shortcuts on file.

This is body text for the shortcut list email if no shortcuts exist for the callsign.

SMTPShortCutList=Shortcut List

This is the subject line for the shortcut list email.

Store General Properties

If you want to connect to Gmail via IMAP, include the appropriate gimap.jar from the jakarta.mail repository in your classpath and use mail.gimaps as your IMAP properties prefix instead of mail.imap.

StoreClass=

(Deprecated) Normally set this to net.ae5pl.emailgate.pop3Task or net.ae5pl.emailgate.SunImapTask.

StorePassword=

This is the password for the Store server.
Caution: This password is stored in plain text.

StoreEncPassword=

This is the password for the Store server. This is the encoded text from the encpwd application.

MsgMaxSize=10240

Maximum email size to download.
Larger emails will be deleted without download.

PollInterval=-1

This is the time between checks on the Store server in minutes. If less than or equal to zero, it is assumed the Store provider is not polled (IMAP, for instance).

com.sun.mail POP3 properties

Note that if you're using the "pop3s" protocol to access POP3 over SSL, all the properties would be named "mail.pop3s.*".

Name	Type	Description
mail.pop3.user	String	Default user name for POP3.
mail.pop3.host	String	The POP3 server to connect to.
mail.pop3.port	int	The POP3 server port to connect to, if the connect() method doesn't explicitly specify one. Defaults to 110.
mail.pop3.connectiontimeout	int	Socket connection timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.pop3.timeout	int	Socket read timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.pop3.writetimeout	int	Socket write timeout value in milliseconds. This timeout is implemented by using a java.util.concurrent.ScheduledExecutorService per connection that schedules a thread to close the socket if the timeout expires. Thus, the overhead of using this timeout is one thread per connection. Default is infinite timeout.
mail.pop3.rsetbeforequit	boolean	Send a POP3 RSET command when closing the folder, before sending the QUIT command. Useful with POP3 servers that implicitly mark all messages that are read as "deleted"; this will prevent such messages from being deleted and expunged unless the client requests so. Default is false.
mail.pop3.message.class	String	Class name of a subclass of com.sun.mail.pop3.POP3Message. The subclass can be used to handle (for example) non-standard Content-Type headers. The subclass must have a public constructor of the form MyPOP3Message(Folder f, int msgno) throws MessagingException.
mail.pop3.localaddress	String	Local address (host name) to bind to when creating the POP3 socket. Defaults to the address picked by the Socket class. Should not normally need to be set, but useful with multi-homed hosts where it's important to pick a particular local address to bind to.
mail.pop3.localport	int	Local port number to bind to when creating the POP3 socket. Defaults to the port number picked by the Socket class.
mail.pop3.apop.enable	boolean	If set to true, use APOP instead of USER/PASS to login to the POP3 server, if the POP3 server supports APOP. APOP sends a digest of the password rather than the clear text password. Defaults to false.
mail.pop3.auth.mechanisms	String	If set, lists the authentication mechanisms to consider, and the order in which to consider them. Only mechanisms supported by the server and supported by the current implementation will be used. The default is "LOGIN PLAIN DIGEST-MD5 NTLM", which includes all the authentication mechanisms supported by the current implementation except XOAUTH2.
mail.pop3.auth.login.disable	boolean	If true, prevents use of the USER and PASS commands. Default is false.

mail.pop3.auth.plain.disable	boolean	If true, prevents use of the AUTH PLAIN command. Default is false.
mail.pop3.auth.digest-md5.disable	boolean	If true, prevents use of the AUTH DIGEST-MD5 command. Default is false.
mail.pop3.auth.ntlm.disable	boolean	If true, prevents use of the AUTH NTLM command. Default is false.
mail.pop3.auth.ntlm.domain	String	The NTLM authentication domain.
mail.pop3.auth.ntlm.flags	int	NTLM protocol-specific flags. See http://curl.haxx.se/rfc/ntlm.html#theNtlmFlags for details.
mail.pop3.auth.xoauth2.disable	boolean	If true, prevents use of the AUTHENTICATE XOAUTH2 command. Because the OAuth 2.0 protocol requires a special access token instead of a password, this mechanism is disabled by default. Enable it by explicitly setting this property to "false" or by setting the "mail.pop3.auth.mechanisms" property to "XOAUTH2".
mail.pop3.auth.xoauth2.two.line.authentication.format	boolean	If true, splits authentication command on two lines. Default is false.
mail.pop3.ssl.enable	boolean	If set to true, use SSL to connect and use the SSL port by default. Defaults to false for the "pop3" protocol and true for the "pop3s" protocol.
mail.pop3.ssl.checkserveridentity	boolean	If set to true, check the server identity as specified by RFC 2595. These additional checks based on the content of the server's certificate are intended to prevent man-in-the-middle attacks. Defaults to false.
mail.pop3.ssl.trust	String	If set, and a socket factory hasn't been specified, enables use of a MailSSLSocketFactory. If set to "*", all hosts are trusted. If set to a whitespace separated list of hosts, those hosts are trusted. Otherwise, trust depends on the certificate the server presents.
mail.pop3.ssl.protocols	string	Specifies the SSL protocols that will be enabled for SSL connections. The property value is a whitespace separated list of tokens acceptable to the javax.net.ssl.SSLSocket.setEnabledProtocols method.
mail.pop3.ssl.ciphersuites	string	Specifies the SSL cipher suites that will be enabled for SSL connections. The property value is a whitespace separated list of tokens acceptable to the javax.net.ssl.SSLSocket.setEnabledCipherSuites method.
mail.pop3.starttls.enable	boolean	If true, enables the use of the STLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands. If the server does not support STARTTLS, the connection continues without the use of TLS; see the mail.pop3.starttls.required property to fail if STARTTLS isn't supported. Note that an appropriate trust store must configured so that the client will trust the server's certificate. Defaults to false.
mail.pop3.starttls.required	boolean	If true, requires the use of the STLS command. If the server doesn't support the STLS command, or the command fails, the connect method will fail. Defaults to false.
mail.pop3.proxy.host	string	Specifies the host name of an HTTP web proxy server that will be used for connections to the mail server.

mail.pop3.proxy.port	string	Specifies the port number for the HTTP web proxy server. Defaults to port 80.
mail.pop3.proxy.user	string	Specifies the user name to use to authenticate with the HTTP web proxy server. By default, no authentication is done.
mail.pop3.proxy.password	string	Specifies the password to use to authenticate with the HTTP web proxy server. By default, no authentication is done.
mail.pop3.socks.host	string	Specifies the host name of a SOCKS5 proxy server that will be used for connections to the mail server.
mail.pop3.socks.port	string	Specifies the port number for the SOCKS5 proxy server. This should only need to be used if the proxy server is not using the standard port number of 1080.
mail.pop3.disabletop	boolean	If set to true, the POP3 TOP command will not be used to fetch message headers. This is useful for POP3 servers that don't properly implement the TOP command, or that provide incorrect information in the TOP command results. Defaults to false.
mail.pop3.disablecapa	boolean	If set to true, the POP3 CAPA command will not be used to fetch server capabilities. This is useful for POP3 servers that don't properly implement the CAPA command, or that provide incorrect information in the CAPA command results. Defaults to false.
mail.pop3.forgettopheaders	boolean	If set to true, the headers that might have been retrieved using the POP3 TOP command will be forgotten and replaced by headers retrieved as part of the POP3 RETR command. Some servers, such as some versions of Microsft Exchange and IBM Lotus Notes, will return slightly different headers each time the TOP or RETR command is used. To allow the POP3 provider to properly parse the message content returned from the RETR command, the headers also returned by the RETR command must be used. Setting this property to true will cause these headers to be used, even if they differ from the headers returned previously as a result of using the TOP command. Defaults to false.
mail.pop3.filecache.enable	boolean	If set to true, the POP3 provider will cache message data in a temporary file rather than in memory. Messages are only added to the cache when accessing the message content. Message headers are always cached in memory (on demand). The file cache is removed when the folder is closed or the JVM terminates. Defaults to false.
mail.pop3.filecache.dir	String	If the file cache is enabled, this property can be used to override the default directory used by the JDK for temporary files.
mail.pop3.cachewriteto	boolean	Controls the behavior of the writeTo method on a POP3 message object. If set to true, and the message content hasn't yet been cached, and ignoreList is null, the message is cached before being written. Otherwise, the message is streamed directly to the output stream without being cached. Defaults to false.
mail.pop3.keepmessagecontent	boolean	The content of a message is cached when it is first fetched. Normally this cache uses a SoftReference to refer to the cached content. This allows the cached content to be purged if memory is low, in which case the content will be fetched again if it's needed. If this property is set to true, a

		hard reference to the cached content will be kept, preventing the memory from being reused until the folder is closed or the cached content is explicitly invalidated (using the invalidate method). (This was the behavior in previous versions of Jakarta Mail.) Defaults to false.
mail.pop3.finalizecleanclose	boolean	When the finalizer for POP3Store or POP3Folder is called, should the connection to the server be closed cleanly, as if the application called the close method? Or should the connection to the server be closed without sending any commands to the server? Defaults to false, the connection is not closed cleanly.

In general, applications should not need to use the classes in this package directly. Instead, they should use the APIs defined by javax.mail package (and subpackages). Applications should never construct instances of POP3Store or POP3Folder directly. Instead, they should use the Session method getStore to acquire an appropriate Store object, and from that acquire Folder objects.
In addition to printing debugging output as controlled by the Session configuration, the com.sun.mail.pop3 provider logs the same information using Logger as described in the following table:

Logger Name	**Logging Level**	**Purpose**
com.sun.mail.pop3	CONFIG	Configuration of the POP3Store
com.sun.mail.pop3	FINE	General debugging output
com.sun.mail.pop3.protocol	FINEST	Complete protocol trace

com.sun.mail IMAP Properties

Note that if you're using the "imaps" protocol to access IMAP over SSL, all the properties would be named "mail.imaps.*". If you want to connect to Gmail via IMAP, include the appropriate gimap.jar from the jakarta.mail repository in your classpath and use mail.gimap as your IMAP properties prefix instead of mail.imap.

Name	Type	Description
mail.imap.user	String	Default user name for IMAP.
mail.imap.host	String	The IMAP server to connect to.
mail.imap.port	int	The IMAP server port to connect to, if the connect() method doesn't explicitly specify one. Defaults to 143.
mail.imap.partialfetch	boolean	Controls whether the IMAP partial-fetch capability should be used. Defaults to true.
mail.imap.fetchsize	int	Partial fetch size in bytes. Defaults to 16K.
mail.imap.peek	boolean	If set to true, use the IMAP PEEK option when fetching body parts, to avoid setting the SEEN flag on messages. Defaults to false. Can be overridden on a per-message basis by the setPeek method on IMAPMessage.
mail.imap.ignorebodystructuresize	boolean	The IMAP BODYSTRUCTURE response includes the exact size of each body part. Normally, this size is used to determine how much data to fetch for each body part. Some servers report this size incorrectly in some cases; this property can be set to work around such server bugs. If this property is set to true, this size is ignored and data is fetched until the server reports the end of data. This will result in an extra fetch if the data size is a multiple of the block size. Defaults to false.
mail.imap.connectiontimeout	int	Socket connection timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.imap.timeout	int	Socket read timeout value in milliseconds. This timeout is implemented by java.net.Socket. Default is infinite timeout.
mail.imap.writetimeout	int	Socket write timeout value in milliseconds. This timeout is implemented by using a java.util.concurrent.ScheduledExecutorService per connection that schedules a thread to close the socket if the timeout expires. Thus, the overhead of using this timeout is one thread per connection. Default is infinite timeout.
mail.imap.statuscachetimeout	int	Timeout value in milliseconds for cache of STATUS command response. Default is 1000 (1 second). Zero disables cache.
mail.imap.appendbuffersize	int	Maximum size of a message to buffer in memory when appending to an IMAP folder. If not set, or set to -1, there is no maximum and all messages are buffered. If set to 0, no messages are buffered. If set to (e.g.) 8192, messages of 8K bytes or less are buffered, larger messages are not buffered. Buffering saves cpu time at the expense of short term memory usage. If you commonly append very large messages to IMAP mailboxes you might want to set this to a moderate value (1M or less).
mail.imap.connectionpoolsize	int	Maximum number of available connections in the connection pool. Default is 1.
mail.imap.connectionpooltimeout	int	Timeout value in milliseconds for connection pool connections. Default is 45000 (45 seconds).
mail.imap.separatestoreconnection	boolean	Flag to indicate whether to use a dedicated store connection for store commands. Default is false.
mail.imap.allowreadonlyselect	boolean	If false, attempts to open a folder read/write will fail if the SELECT command succeeds but indicates that the folder is READ-ONLY. This sometimes indicates that the folder contents can'tbe changed, but the flags are per-user and can be changed, such as might be the case for public shared folders. If true, such open attempts will succeed, allowing the flags to be changed. The getMode method on the Folder object will

		return Folder.READ_ONLY in this case even though the open method specified Folder.READ_WRITE. Default is false.
mail.imap.auth.mechanisms	String	If set, lists the authentication mechanisms to consider, and the order in which to consider them. Only mechanisms supported by the server and supported by the current implementation will be used. The default is "PLAIN LOGIN NTLM", which includes all the authentication mechanisms supported by the current implementation except XOAUTH2.
mail.imap.auth.login.disable	boolean	If true, prevents use of the non-standard AUTHENTICATE LOGIN command, instead using the plain LOGIN command. Default is false.
mail.imap.auth.plain.disable	boolean	If true, prevents use of the AUTHENTICATE PLAIN command. Default is false.
mail.imap.auth.ntlm.disable	boolean	If true, prevents use of the AUTHENTICATE NTLM command. Default is false.
mail.imap.auth.ntlm.domain	String	The NTLM authentication domain.
mail.imap.auth.ntlm.flags	int	NTLM protocol-specific flags. See http://curl.haxx.se/rfc/ntlm.html#theNtlmFlags for details.
mail.imap.auth.xoauth2.disable	boolean	If true, prevents use of the AUTHENTICATE XOAUTH2 command. Because the OAuth 2.0 protocol requires a special access token instead of a password, this mechanism is disabled by default. Enable it by explicitly setting this property to "false" or by setting the "mail.imap.auth.mechanisms" property to "XOAUTH2".
mail.imap.proxyauth.user	String	If the server supports the PROXYAUTH extension, this property specifies the name of the user to act as. Authenticate to the server using the administrator's credentials. After authentication, the IMAP provider will issue the PROXYAUTH command with the user name specified in this property.
mail.imap.localaddress	String	Local address (host name) to bind to when creating the IMAP socket. Defaults to the address picked by the Socket class. Should not normally need to be set, but useful with multi-homed hosts where it's important to pick a particular local address to bind to.
mail.imap.localport	int	Local port number to bind to when creating the IMAP socket. Defaults to the port number picked by the Socket class.
mail.imap.sasl.enable	boolean	If set to true, attempt to use the javax.security.sasl package to choose an authentication mechanism for login. Defaults to false.
mail.imap.sasl.mechanisms	String	A space or comma separated list of SASL mechanism names to try to use.
mail.imap.sasl.authorizationid	String	The authorization ID to use in the SASL authentication. If not set, the authentication ID (user name) is used.
mail.imap.sasl.realm	String	The realm to use with SASL authentication mechanisms that require a realm, such as DIGEST-MD5.
mail.imap.sasl.usecanonicalhostname	boolean	If set to true, the canonical host name returned by InetAddress.getCanonicalHostName is passed to the SASL mechanism, instead of the host name used to connect. Defaults to false.
mail.imap.sasl. xgwtrustedapphack.enable	boolean	If set to true, enables a workaround for a bug in the Novell Groupwise XGWTRUSTEDAPP SASL mechanism, when that mechanism is being used. Defaults to true.
mail.imap.usesocketchannels	boolean	If set to true, use SocketChannels instead of Sockets for connecting to the server. Required if using the IdleManager. Ignored if a socket factory is set. Defaults to false.
mail.imap.ssl.enable	boolean	If set to true, use SSL to connect and use the SSL port by default. Defaults to false for the "imap" protocol and true for the "imaps" protocol.

mail.imap.ssl.checkserveridentity	boolean	If set to true, check the server identity as specified by RFC 2595. These additional checks based on the content of the server's certificate are intended to prevent man-in-the-middle attacks. Defaults to false.
mail.imap.ssl.trust	String	If set, and a socket factory hasn't been specified, enables use of a MailSSLSocketFactory. If set to "*", all hosts are trusted. If set to a whitespace separated list of hosts, those hosts are trusted. Otherwise, trust depends on the certificate the server presents.
mail.imap.ssl.protocols	string	Specifies the SSL protocols that will be enabled for SSL connections. The property value is a whitespace separated list of tokens acceptable to the javax.net.ssl.SSLSocket.setEnabledProtocols method.
mail.imap.ssl.ciphersuites	string	Specifies the SSL cipher suites that will be enabled for SSL connections. The property value is a whitespace separated list of tokens acceptable to the javax.net.ssl.SSLSocket.setEnabledCipherSuites method.
mail.imap.starttls.enable	boolean	If true, enables the use of the STARTTLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands. If the server does not support STARTTLS, the connection continues without the use of TLS; see the mail.imap.starttls.required property to fail if STARTTLS isn't supported. Note that an appropriate trust store must configured so that the client will trust the server's certificate. Default is false.
mail.imap.starttls.required	boolean	If true, requires the use of the STARTTLS command. If the server doesn't support the STARTTLS command, or the command fails, the connect method will fail. Defaults to false.
mail.imap.proxy.host	string	Specifies the host name of an HTTP web proxy server that will be used for connections to the mail server.
mail.imap.proxy.port	string	Specifies the port number for the HTTP web proxy server. Defaults to port 80.
mail.imap.proxy.user	string	Specifies the user name to use to authenticate with the HTTP web proxy server. By default, no authentication is done.
mail.imap.proxy.password	string	Specifies the password to use to authenticate with the HTTP web proxy server. By default, no authentication is done.
mail.imap.socks.host	string	Specifies the host name of a SOCKS5 proxy server that will be used for connections to the mail server.
mail.imap.socks.port	string	Specifies the port number for the SOCKS5 proxy server. This should only need to be used if the proxy server is not using the standard port number of 1080.
mail.imap.minidletime	int	Applications typically call the idle method in a loop. If another thread termiantes the IDLE command, it needs a chance to do its work before another IDLE command is issued. The idle method enforces a delay to prevent thrashing between the IDLE command and regular commands. This property sets the delay in milliseconds. If not set, the default is 10 milliseconds.
mail.imap.enableresponseevents	boolean	Enable special IMAP-specific events to be delivered to the Store's ConnectionListener. If true, IMAP OK, NO, BAD, or BYE responses will be sent as ConnectionEvents with a type of IMAPStore.RESPONSE. The event's message will be the raw IMAP response string. By default, these events are not sent. NOTE: This capability is highly experimental and likely will change in future releases.
mail.imap.enableimapevents	boolean	Enable special IMAP-specific events to be delivered to the Store's ConnectionListener. If true, unsolicited responses received during the Store's idle method will be sent as ConnectionEvents with a type of IMAPStore.RESPONSE. The event's message will be the raw IMAP response string. By default, these events are not sent. NOTE:

		This capability is highly experimental and likely will change in future releases.
mail.imap.throwsearchexception	boolean	If set to true and a SearchTerm passed to the Folder.search method is too complex for the IMAP protocol, throw a SearchException. For example, the IMAP protocol only supports less-than and greater-than comparisons for a SizeTerm. If false, the search will be done locally by fetching the required message data and comparing it locally. Defaults to false.
mail.imap.folder.class	String	Class name of a subclass of com.sun.mail.imap.IMAPFolder. The subclass can be used to provide support for additional IMAP commands. The subclass must have public constructors of the form public MyIMAPFolder(String fullName, char separator, IMAPStore store, Boolean isNamespace) and public MyIMAPFolder(ListInfo li, IMAPStore store)
mail.imap.closefoldersonstorefailure	boolean	In some cases, a failure of the Store connection indicates a failure of the server, and all Folders associated with that Store should also be closed. In other cases, a Store connection failure may be a transient failure, and Folders may continue to operate normally. If this property is true (the default), failures in the Store connection cause all associated Folders to be closed. Set this property to false to better handle transient failures in the Store connection.
mail.imap.finalizecleanclose	boolean	When the finalizer for IMAPStore is called, should the connection to the server be closed cleanly, as if the application called the close method? Or should the connection to the server be closed without sending any commands to the server? Defaults to false, the connection is not closed cleanly.
mail.imap.referralexception	boolean	If set to true and an IMAP login referral is returned when the authentication succeeds, fail the connect request and throw a ReferralException. Defaults to false.
mail.imap.compress.enable	boolean	If set to true and the IMAP server supports the COMPRESS=DEFLATE extension, compression will be enabled. Defaults to false.
mail.imap.compress.level	int	The compression level to be used, in the range -1 to 9. See the Deflater class for details.
mail.imap.compress.strategy	int	The compression strategy to be used, in the range 0 to 2. See the Deflater class for details.
mail.imap.reusetagprefix	boolean	If true, always use "A" for the IMAP command tag prefix. If false, the IMAP command tag prefix is different for each connection, from "A" through "ZZZ" and then wrapping around to "A". Applications should never need to set this. Defaults to false.

Loggers

In addition to printing debugging output as controlled by the Session configuration, the com.sun.mail.imap provider logs the same information using Logger as described in the following table:

Logger Name	Logging Level	Purpose
com.sun.mail.imap	CONFIG	Configuration of the IMAPStore
com.sun.mail.imap	FINE	General debugging output
com.sun.mail.imap.connectionpool	CONFIG	Configuration of the IMAP connection pool
com.sun.mail.imap.connectionpool	FINE	Debugging output related to the IMAP connection pool
com.sun.mail.imap.messagecache	CONFIG	Configuration of the IMAP message cache
com.sun.mail.imap.messagecache	FINE	Debugging output related to the IMAP message cache
com.sun.mail.imap.protocol	FINEST	Complete protocol trace

APRS Message Text Properties

Messages, where indicated, use the java.util.Formatter method for inserting a parameter. In all of these cases, there is, at most, only 1 parameter ({0}) which will be formatted as a string.

MsgNoAddr=No email address!

This is sent to the APRS station when no email address is specified in the APRS message.

MsgInvalidAddr=Invalid email address!

This is sent to the APRS station when the email address is specified in the APRS message is invalid.

MsgShortAdded={0} added.

This is sent to the APRS station when a shortcut is added to the server ({0} is the placeholder for the shortcut name).

MsgNoAddrFound=No email address found!

This is sent to the APRS station when no email address is found in the shortcut list for the specified shortcut.

MsgShortRemoved={0} removed.

This is sent to the APRS station when a shortcut is removed ({0} is the placeholder for the shortcut name).

MsgEmailSent=Email sent to {0}

This is sent to the APRS station when an email is successfully submitted to the SMTP server ({0} is the placeholder for the recipient's email address).

MsgSendFailed=Send to {0} failed.

This is sent to the APRS station when an email cannot be submitted to the Transport server ({0} is the placeholder for the recipient's email address).
This is returned when an email to {0} ({0} is a placeholder) fails to be sent to the SMTP server.

Email to APRS Unacknowledged Message Properties

UnackedBackupFile=unacked.bak

This defines where unacked messages will be held in case of restart. The file location is in the OptionsDirectory.

UnackedMessageHoldTime=0

Set to number of hours to hold unacknowledged APRS messages generated from received emails. The may be retrieved by the “get” command in an APRS message. Zero or less than zero turns off caching.

NoMsgsUnacked=No unacked email messages.

This is sent to the APRS station when there are no unacked messages for that callsign.

Section 4 - Recommended Configurations

Use only the settings necessary to properly communicate with your mail server(s). A transport server (SMTP) is necessary at a minimum.

Section 5 - Installation Instructions

You must include EmailGate.jar in the ClassPath property in your EmailGate properties file if started with the deprecated -jar switch.

Place all of the .txt files into the EmailGate directory (as pointed to by OptionsDirectory). Modify them for your own installation.
Set Clients= to point to your EmailGate properties file.

jakarta.mail.jar and dsn jar 1.x files must be downloaded to your javAPRSSrvr folder. These may be downloaded from a Maven repository under com/sun/mail/jakarta.mail and com/sun/mail/dsn, respectively.

JAVA 9-10 Note (Do NOT do this if you are on Java 8):

javax.mail requires javax.activation to be installed. javax.activation is part of the Java.EE module which will be removed from the Java SE distributions as of Java 11. Add the following entries to your Java startup (before javAPRSSrvr.jar call):

--add-modules java.activation

The above line adds the javax.activation that ships with Java 9 & 10.

If you are using Java 11 or later, please use EmailGate 4.4 which is designed for the modular environment and later versions of Jakarta EE.

Section 6 – Operator Guide

This section describes how users interact with the server.

Send an Email from APRS

Send a numbered APRS message to APRSCall with the email address as the first word in the message text. If no other text is given, an email with a link to your position display will be sent.

AE5PL-10>APRS::AE5PL-EM :pete@ae5pl.net This is a test{00

Create an Email Shortcut from APRS

Send a numbered APRS message to APRSCall with the shortcut (single alphanumeric word) and the desired email address separated by a space.

AE5PL-10>APRS::AE5PL-EM :me pete@ae5pl.net{00

Delete an Email Shortcut from APRS

Send a numbered APRS message to APRSCall with the shortcut (single alphanumeric word) and the single letter r (case-insensitive) separated by a space.

AE5PL-10>APRS::AE5PL-EM :me r{00

List Email Shortcuts via Email from APRS

Send a numbered APRS message to APRSCall with the desired shortcut or email address and the single letter L (case-insensitive) separated by a space. This will send an email to the specified address with all of the sending station's shortcuts and email addresses.

AE5PL-10>APRS::AE5PL-EM :me l{00

Send an Email using a Shortcut from APRS

Send a numbered APRS message to APRSCall with the shortcut as the first word in the message text. If no other text is given, an email with a link to your position display will be sent.

AE5PL-10>APRS::AE5PL-EM :me This is a test{00

Send an APRS message using an Email

Send an email to the POP3 email address of this server. Place the callsign-ssid of the destination station followed by a colon followed by the message text on the subject line. Place userid:shortcut: in the message body where shortcut is the shortcut name associated with the sender's email address.

From: pete@ae5pl.net
To: aprsemail@ae5pl.net
Subject: AE5PL-10:This is a test
userid:me:

Note: Shortcuts are related to callsigns only (no SSID's) so "me" in the above examples is a valid shortcut for any AE5PL stations. Every licensee controls their own shortcuts (AE5PL shortcuts are different from W5EJL shortcuts).

Get all unacknowledged email APRS messages

Send a numbered APRS message to APRSCall with the word "get". EmailGate also responds to a ?APRSM query with the same information.

AE5PL-10>APRS::AE5PL-EM :get{00

Section 7 – XML Status Page

General XML

<clientrcv>
<time>
<connect utc="1341331889731"/>
<lastlinein utc="1341331889731"/>
</time>
<upstream>
false
</upstream>
<readonly>
false
</readonly>
<login>
<callssid verified="true">
EMAIL-2
</callssid>
<software version="4.0.0">
EmailGate
</software>
</login>
<rcvdfrom bytes="0" lines="0" packets="0">
<udp bytes="0" lines="0" packets="0"/>
</rcvdfrom>
<clientxmt>
<upstream>
false
</upstream>
<sentto bytes="0" lines="0" packets="0">
<lastlinems>
1341331889762
</lastlinems>
<udp bytes="0" lines="0" packets="0"/>
</sentto>
<xmtqueue depth="0" depthms="0"/>
<emailxmt>
<aprsmsgsrcvd total="0"/>
<commands invalid="0" valid="0"/>
</emailxmt>
</clientxmt>
<emailgate>
<shortcuts callsigns="167" total="471"/>
<emails failedsend="0" sent="0">
<received invalid="0" valid="0"/>
</emails>
</emailgate>
</clientrcv>

Detail XML


```
<clientrcv>
<time>
<connect utc="1341331889731"/>
<lastlinein utc="1341331889731"/>
</time>
<class name="EmailGate">
<package name="net.ae5pl.emailgate" revision="b01" title="APRS Email Server" version="4.0.0"/>
</class>
<messages callssids="0" unacked="0"/>
<upstream>
false
</upstream>
<readonly>
false
</readonly>
<login>
<callssid verified="true">
EMAIL-2
</callssid>
<software version="4.0.0">
EmailGate
</software>
</login>
<rcvdfrom bytes="0" lines="0" packets="0">
<udp bytes="0" lines="0" packets="0"/>
</rcvdfrom>
<clientxmt>
<class name="EmailXmt">
<package name="net.ae5pl.emailgate" revision="b01" title="APRS Email Server" version="4.0.0"/>
</class>
<upstream>
false
</upstream>
<sentto bytes="0" lines="0" packets="0">
<lastlinems>
1341331889762
</lastlinems>
<udp bytes="0" lines="0" packets="0"/>
</sentto>
<xmtqueue depth="0" depthms="0"/>
<emailxmt>
<aprsmsgsrcvd total="0"/>
<commands invalid="0" valid="0"/>
</emailxmt>
</clientxmt>
```