Email

IEC 61131 Library for ACSELERATOR RTAC® Projects

SEL Automation Controllers

Table of Contents

RTAC LIBRARY

Email

Introduction

The Email library allows emails to be sent easily from a Real-Time Automation Controller (RTAC) to a Simple Mail Transfer Protocol (SMTP) email server. These emails can contain periodic information about process status, alert on-call staff to process anomalies, or send collected event reports or other attachments directly to email accounts or mobile devices.

Though this library does some parsing of the inputs provided, it is not meant to fully support all features of SMTP. Test all emails to ensure that the formatting of the arguments does not create an email the SMTP server receiving the request will ignore.

This library uses the HELO message and local IP address to open each email. This means that any features requiring the EHLO extensions, including user authentication and encryption, are not supported.

Special Considerations

➤ Copying classes from this library causes unwanted behavior. This means the following:

1. The assignment operator ":=" must not be used on any class from this library; consider assigning pointers to the objects instead.

```
// This is bad and in most cases will provide a compiler error
    such as:
// "C0328: Assignment not allowed for type
    class_EmailClientObject"
myEmailClientObject := otherEmailClientObject;
```

```
// This is fine
someVariable := myEmailClientObject.value;
// As is this
pt_myEmailClientObject := ADR(myEmailClientObject);
```

2. Classes from this library must never be VAR_INPUT or VAR_OUTPUT members in function blocks, functions, or methods. Place them in the VAR_IN_OUT section or use pointers instead.

➤ Classes in this library have memory allocated inside them. As such, they should only be created in environments of permanent scope (e.g., Programs, Global Variable Lists, or VAR_STAT sections).

Supported Firmware Versions

You can use this library on any device configured using ACSELERATOR RTAC® SEL-5033 Software with firmware version R143 or higher.

Versions 3.5.0.6 and older can be used on RTAC firmware version R132 and higher.

Enumerations

Enumerations make code more readable by allowing a specific number to have a readable textual equivalent.

enum_RecipientType

Enumeration	Description
MAIL_TO	This recipient will be a direct target of the email.
MAIL_CC	This recipient will receive a copy of the email.
MAIL_BCC	This recipient will receive a blind copy of the email.

Function Blocks

Function Blocks (FB), if declared in a program or a Global Variable List (GVL), maintain their state from one processing scan to the next. They should be called during each task cycle using their input pins to manage their behavior.

fb_SimpleEmailClient (Function Block)

This function block provides a simple triggered interface for sending email to one or two recipients. It reads all inputs immediately upon receiving a request that data be sent and ignores any further changes until the next trigger is received.

Initialization Inputs

Name	IEC 61131 Type	Description
localIPAddr	STRING(15)	The IP address this RTAC should use to communicate with the email server. Set this to **0.0.0.0** to leave From IP information off of the message request and use an arbitrary interface for communication.
mailServerIP	STRING(15)	The IP address of the email server.

Inputs

Name	IEC 61131 Type	Description
Send	BOOL	Initiate a new email communication.

Inputs/Outputs

Name	IEC 61131 Type	Description
ToEmail	STRING(254)	The email address of the message recipient.
ToName	STRING(255)	The name of the email recipient.
CcEmail	STRING(254)	The email address to receive a copy of the message.
CcName	STRING(255)	The name of the copied recipient.
FromEmail	STRING(254)	The email address of the message sender.
FromName	STRING(255)	The name of the message sender.
Subject	STRING(255)	The subject of the message.
BodyStr	STRING(255)	The body of the message.

Outputs

Name	IEC 61131 Type	Description
Busy	BOOL	Indication that the function block is sending an email.
SentMsg	STRING(255)	The first 255 characters of the last message the function block sent to the SMTP server.
Error	BOOL	TRUE if the last attempt to send an email failed.
ErrorStr	STRING(80)	A description of why the last email transmission failed.

Processing

The fb_SimpleEmailClient function block body does the following:

- Checks to see if a message is processing.
- Validates the email addresses provided. Allowed characters include all alphanumeric characters, ., !, #, $, %, &, +, -, /, =, ?, ^, _, {, }, |, ~, and @.
- Replaces names containing double quotes with empty strings.
- Sends a new message to the defined recipients if the function block is not busy and it detects a rising edge on Send.
- Works to complete the previously begun email transfer if the function block is busy.

➤ Sets Busy to TRUE if it is waiting on the server to complete an email request.

fb_SimpleEmailClient2 (Function Block)

This function block provides functionality and behavior that is identical to *fb_SimpleEmail-Client (Function Block) on page 4*, with the exception that the user may define the outgoing and destination ports of the email client at declaration. These inputs are described below.

Initialization Inputs

Name	IEC 61131 Type	Description
localIPAddr	STRING(15)	The IP address this RTAC should use to communicate with the email server. Set this to **0.0.0.0** to leave From IP information off of the message request and use an arbitrary interface for communication.
localPort	UINT	The outgoing client port from which emails shall be sent. Setting this to zero will allow the OS to select an ephemeral port.
mailServerIP	STRING(15)	The IP address of the email server.
mailServerPort	UINT	The port number of the email server. For SMTP, this is should normally be set to 25, but the user may set it otherwise if their server is configured accordingly.

Classes

Classes are a particular implementation of a function block. They provide methods and properties, which normal function blocks do not provide.

class_EmailClient (Class)

This class provides the ability to craft an email message over time and allows for multiple recipients, dynamic message body lengths, and vector attachments. This class is identical to class_EmailClient2 with the exception that class_EmailClient2 allows the client to define the outgoing and destination ports for the client and server, respectively. By contrast, this class sets the local port to 0, allowing the client OS to select an ephemeral port number. The destination/server port is set to 25 (the standard SMTP interface port). As such, this class can be used when there are no special requirements imposed on local or destination port numbers.

Initialization Inputs

Name	IEC 61131 Type	Description
localIPAddr	STRING(15)	The IP address this RTAC should use to communicate with the email server. Set this to **0.0.0.0** to leave From IP information off of the message request and use an arbitrary interface for communication.
mailServerIP	STRING(15)	The IP address of the email server.

Properties

Name	IEC 61131 Type	Access	Description
Busy	BOOL	R	Indication that the class is sending an email.
SentMsg	STRING(255)	R	The first 255 characters of the last message the class sent to the SMTP server.
ErrorStr	STRING(80)	R	A description of any state that would prevent the attempt to send an email.

Properties are internal values made visible through Get and Set accessors. Access is defined as R (read), W (write), or R/W (read/write).

AddRecipient (Method)

This method adds a recipient for subsequent outgoing emails when they are sent. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

Inputs

Name	IEC 61131 Type	Description
recipientType	enum_RecipientType	To, Cc, or Bcc.

Inputs/Outputs

Name	IEC 61131 Type	Description
emailAddress	STRING(254)	The email address of the recipient.
name	STRING(255)	The name of the recipient.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if emailAddress is valid and the recipient was added.

Processing

The `AddRecipient()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Checks that the emailAddress contains only allowed characters. Allowed characters include all alphanumeric characters, ., !, #, $, %, &, +, -, /, =, ?, ^, _, {, }, |, ~, and @.
- Verifies that names do not contain double quotes.
- Stores the email address and name to add to the recipient field defined by recipientType of subsequent emails.

➤ Returns FALSE if the email address was not added.

AttachVector (Method)

This method allows the user to add a raw byte vector as an attachment. The vector does not need to be encoded in base64-MIME because this will be performed internally, nor will data be modified as a result of calling this function. This method may not be called and will return false while Busy is TRUE and the client is busy sending an email.

Inputs/Outputs

Name	IEC 61131 Type	Description
data	class_ByteVector	A byte vector of data to be attached when email is sent.
name	STRING	The name for the attachment, as it shall appear in each recipient's email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if *data* was successfully stored with *name*.

Processing

The `AttachVector()` method does the following:

➤ Verifies the client is not busy sending an email by checking that Busy is FALSE.

➤ Verifies both *data* and *name* are non-empty.

➤ Copies and encodes *data* in base64-MIME format, storing the encoded data as an attachment to be appended to an email.

➤ Returns FALSE if attachment was not added.

ClearAttachments (Method)

This method removes all stored attachments. Freeing large amounts of memory is expensive, so this should be called infrequently. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

Processing

The `ClearAttachments()` method does the following:

➤ Verifies the class is not busy sending email by verifying that Busy is FALSE.

➤ Deletes all email attachments stored by the email client object.

ClearRecipients (Method)

This method removes all stored recipients. This method may not be called and will return false while Busy is TRUE and the client is busy sending an email.

Processing

The `ClearRecipients()` method does the following:

- Verifies the class is not busy sending email by verifying Busy is FALSE.
- Deletes all stored email recipients.

Run (Method)

Main state machine, which must be called every task cycle. It handles communication with the socket to send emails over multiple scans. This method only performs work after `Send()` is called and Busy is TRUE; otherwise, the call has no effect.

Processing

The `Run()` method does the following:

- Sits idle until `Send()` is called.
- Sets and clears the Busy state.
- Opens communication with the email server.
- Queues data for the email server.
- Closes the connection with the email server when the send is complete.

Send (Method)

This method sends the presently configured email. It does not clear any internal state of the class, so the same email could be sent a second time by calling `Send()` again.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the class will send the message to the email server.

Processing

The `Send()` method does the following:

- Verifies that the class is not already busy sending email.
- Verifies at least one recipient has been defined.
- Verifies the sender has been set.
- Sends a request to the email server to send the configured email.
- Returns FALSE if any check fails.

SetBodyBytes (Method)

Sets the content of the body of the email to the provided bytes. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

Inputs

Name	IEC 61131 Type	Description
pt_data	POINTER TO BYTE	The bytes to use as the body of the email.
numBytes	DINT	The number of bytes to use as the body.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the body content of the email was successfully populated.

Processing

The `SetBodyBytes()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Validates access to pt_data.
- Copies numBytes worth of data into internal storage.
- Replaces any periods beginning lines with two periods to prevent incorrect body termination.
- Returns false if the body construction failed.

SetBodySELString (Method)

Sets the body of the email to the content of the SELString provided. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

NOTE: See the SELString Library documentation for more details on class_SELString objects.

Inputs/Outputs

Name	IEC 61131 Type	Description
data	class_SELString	The string to use as the body of the email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the body content of the email was successfully populated.

Processing

The `SetBodySELString()` method does the following:

- Verifies the class is not busy sending email by verifying Busy is FALSE.
- Copies the contents of *data* into internal storage.
- Replaces any periods beginning lines with two periods to prevent incorrect body termination.
- Returns false if the body construction failed.

SetBodyString (Method)

Sets the body of the email to the provided string. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

Inputs/Outputs

Name	IEC 61131 Type	Description
data	STRING(255)	The string to use as the body of the email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the body content of the email was successfully populated.

Processing

The `SetBodyString()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Copies the contents of *data* into internal storage.
- Replaces any periods beginning lines with two periods to prevent incorrect body termination.
- Returns false if the body construction failed.

SetBodyVector (Method)

Sets the body of the email to the content of the vector provided. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

NOTE: See the DynamicVectors Library documentation for more details on the I_Vector interface.

Inputs

Name	IEC 61131 Type	Description
data	I_Vector	The content to use as the body of the email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the body content of the email was successfully populated.

Processing

The `SetBodyVector()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Validates the data provided can be used.
- Copies the contents of *data* into internal storage.
- Replaces any periods beginning lines with two periods to prevent incorrect body termination.
- Returns FALSE if the body construction failed.

SetSender (Method)

This method sets the sending email address for outgoing email. This method may not be called and will return false while Busy is TRUE and the client is busy sending an email.

Inputs/Outputs

Name	IEC 61131 Type	Description
emailAddress	STRING(254)	The email address of the sender.
name	STRING(255)	The name of the sender.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if emailAddress is valid and the sender was set.

Processing

The `SetSender()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Checks that emailAddress contains only allowed characters. Allowed characters include all alphanumeric characters, ., !, #, $, %, &, +, -, /, =, ?, ^, _, {, }, |, ~, and @.
- Sets emailAddress as the originating email address for outgoing emails.
- Returns FALSE if the sender was not set.

SetSubjectBytes (Method)

Sets the subject of the email to the provided bytes. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

Inputs

Name	IEC 61131 Type	Description
pt_data	POINTER TO BYTE	The bytes to use as the subject of the email.
numBytes	DINT	The number of bytes to use as the subject.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the subject content of the email was successfully populated.

Processing

The `SetSubjectBytes()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Validates access to pt_data.
- Copies numBytes worth of data into internal storage.
- Returns FALSE if the subject construction failed.

SetSubjectSELString (Method)

Sets the subject of the email to the content of the SELString provided. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

NOTE: See the SELString Library documentation for more details on class_SELString objects.

Inputs/Outputs

Name	IEC 61131 Type	Description
data	class_SELString	The string to use as the subject of the email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the subject content of the email was successfully populated.

Processing

The `SetSubjectSELString()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.

- Copies the contents of *data* into internal storage.
- Returns FALSE if the subject construction failed.

SetSubjectString (Method)

Sets the subject of the email to the provided string. This method may not be called and will return false while Busy is TRUE and the client is busy sending an email.

Inputs/Outputs

Name	IEC 61131 Type	Description
data	STRING(255)	The string to use as the subject of the email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the subject content of the email was successfully populated.

Processing

The `SetSubjectString()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Copies the contents of *data* into internal storage.
- Returns false if the subject construction failed.

SetSubjectVector (Method)

Sets the subject of the email to the content of the vector provided. This method may not be called and will return FALSE while Busy is TRUE and the client is busy sending an email.

NOTE: See the DynamicVectors Library documentation for more details on the I_Vector interface.

Inputs

Name	IEC 61131 Type	Description
data	I_Vector	The content to use as the subject of the email.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the subject content of the email was successfully populated.

Processing

The `SetSubjectVector()` method does the following:

- Verifies the class is not busy sending email by verifying that Busy is FALSE.
- Validates the data provided can be used.
- Copies the contents of *data* into internal storage.
- Returns false if the subject construction failed.

class_EmailClient2 (Class)

The functionality, properties, and behavior of this class are identical to that in *class_EmailClient (Class) on page 6*, with the exception of requiring two additional variables in the class declaration: a local port number and a destination email server port number. These inputs allow the user to define the outgoing local port as well as the destination email server port. If the localPort parameter is zero, then the operating system will select an ephemeral port number. The SMTP email server port should normally be set to 25, but may be set otherwise if the SMTP email server is configured accordingly.

Extended Classes

Extending a class provides full inheritance of all that classes features (methods, variables, properties). A class may only extend one other class directly, but class extension can be tiered indefinitely.

- class_EmailClient

Initialization Inputs

These inputs are necessary during instantiation of the class.

Initialization Inputs

Name	IEC 61131 Type	Description
localPort	UINT	The outgoing client port from which emails shall be sent. Setting localPort to zero will allow the OS to select an ephemeral port.
localIPAddr	STRING(15)	The IP address this RTAC should use to communicate with the email server. Set this to **0.0.0.0** to leave From IP information off of the message request and use an arbitrary interface for communication.
mailServerPort	UINT	The port number of the email server. This is nearly always port 25 for SMTP, but the user may set it otherwise if the server listening port is configured accordingly.
mailServerIP	STRING(15)	The IP address of the email server.

Benchmarks

Benchmark Platforms

The benchmarking tests recorded for this library are performed on the following platforms.

- SEL-3505
  - R134-V1 firmware
- SEL-3530
  - R134-V1 firmware
- SEL-3555
  - Dual-core Intel i7-3555LE processor
  - 4 GB ECC RAM
  - R134-V1 firmware

Benchmark Test Descriptions

fb_SimpleEmailClient Average Busy Time

The posted time is the average execution time of 100 calls of the function block body while the Busy output is high. This benchmark is run with all inputs using maximum length strings.

fb_SimpleEmailClient Average Longest Scan Busy Time

It takes multiple calls to the function block to send an email and each call while the function block is busy will take a varying length of time. The posted time is the average time of the longest call of the function block body while the function block is busy. The average is taken over 100 emails with all inputs using maximum length strings.

fb_SimpleEmailClient2 Average Busy Time

The posted time is the average execution time of 100 calls of the function block body while the Busy output is high. This benchmark is run with all inputs using maximum length strings.

fb_SimpleEmailClient2 Average Longest Scan Busy Time

It takes multiple calls to the function block to send an email and each call while the function block is busy will take a varying length of time. The posted time is the average time of the longest call of the function block body while the function block is busy. The average is taken over 100 emails with all inputs using maximum length strings.

class_EmailClient.AddRecipient()

The posted time is the average execution time of 100 method calls adding a recipient when the class is not busy. The email address and name strings are the maximum length.

class_EmailClient.AttachVector()

The posted time is the average execution time of 100 method calls attaching a vector when the class is not busy. The attachment name string is the maximum length and the content is 255 characters.

class_EmailClient.ClearAttachments()

The posted time is the average execution time of 100 method calls when clearing 10 file attachments. To ensure the most work possible, the attachments must already be encoded as Base64 before they are cleared. The class is not busy when the method is called.

class_EmailClient.ClearRecipients()

The posted time is the average execution time of 100 method calls when clearing 10 To recipients, 10 Cc recipients, and 10 Bcc recipients. The class is not busy when the method is called.

class_EmailClient.Run() Average Busy Time

The posted time is the average execution time of 100 method calls while the Busy output is high. The email message contains an 80-character subject with a message body containing 255 characters.

class_EmailClient.Run() Average Busy Time With Event Report

The posted time is the average execution time of 100 method calls while the Busy output is high. The email message contains an 80-character subject with a message body containing a large event report.

class_EmailClient.Run() Average Longest Scan Time

It requires multiple calls to the `Run()` method to send an email and each call while the class is busy will take a varying length of time. The posted time is the average time of the longest call of the method while the method is busy. The average is taken over 100 emails with an 80-character subject and a message body containing 255 characters.

class_EmailClient.Run() Average Longest Scan Time With Event Report

It requires multiple calls to the `Run()` method to send an email and each call while the class is busy will take a varying length of time. The posted time is the average time of the longest call of the method while the method is busy. The average is taken over 100 emails with an 80-character subject and a message body containing a large event report.

class_EmailClient.Send()

The posted time is the average execution time of 100 method calls. The class is not busy when the method is called.

class_EmailClient.SetBodyBytes()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient.SetBodyBytes() With Event Report

The posted time is the average execution time of 100 method calls when setting the email body to the contents of a large event report. The class is not busy when the method is called.

class_EmailClient.SetBodySELString()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient.SetBodySELString With Event Report

The posted time is the average execution time of 100 method calls when setting the email body to the contents of a large event report. The class is not busy when the method is called.

class_EmailClient.SetBodyString()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient.SetBodyVector()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient.SetBodyVector() With Event Report

The posted time is the average execution time of 100 method calls when setting the email body to the contents of a large event report. The class is not busy when the method is called.

class_EmailClient.SetSender()

The posted time is the average execution time of 100 method calls when setting a maximum length email address and name. The class is not busy when the method is called.

class_EmailClient.SetSubjectBytes()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient.SetSubjectSELString()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient.SetSubjectString()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient.SetSubjectVector()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient2.AddRecipient()

The posted time is the average execution time of 100 method calls adding a recipient when the class is not busy. The email address and name strings are the maximum length.

class_EmailClient2.AttachVector()

The posted time is the average execution time of 100 method calls attaching a vector when the class is not busy. The attachment name string is the maximum length and the content is 255 characters.

class_EmailClient2.ClearAttachments()

The posted time is the average execution time of 100 method calls when clearing 10 file attachments. To ensure the most work possible, the attachments must already be encoded as Base64 before they are cleared. The class is not busy when the method is called.

class_EmailClient2.ClearRecipients()

The posted time is the average execution time of 100 method calls when clearing 10 To recipients, 10 Cc recipients, and 10 Bcc recipients. The class is not busy when the method is called.

class_EmailClient2.Run() Average Busy Time

The posted time is the average execution time of 100 method calls while the Busy output is high. The email message contains an 80-character subject with a message body containing 255 characters.

class_EmailClient2.Run() Average Busy Time With Event Report

The posted time is the average execution time of 100 method calls while the Busy output is high. The email message contains an 80-character subject with a message body containing a large event report.

class_EmailClient2.Run() Average Longest Scan Time

It requires multiple calls to the `Run()` method to send an email and each call while the class is busy will take a varying length of time. The posted time is the average time of the longest call of the method while the method is busy. The average is taken over 100 emails with an 80-character subject and a message body containing 255 characters.

class_EmailClient2.Run() Average Longest Scan Time With Event Report

It requires multiple calls to the `Run()` method to send an email and each call while the class is busy will take a varying length of time. The posted time is the average time of the longest call of the method while the method is busy. The average is taken over 100 emails with an 80-character subject and a message body containing a large event report.

class_EmailClient2.Send()

The posted time is the average execution time of 100 method calls. The class is not busy when the method is called.

class_EmailClient2.SetBodyBytes()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetBodyBytes() With Event Report

The posted time is the average execution time of 100 method calls when setting the email body to the contents of a large event report. The class is not busy when the method is called.

class_EmailClient2.SetBodySELString()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetBodySELString With Event Report

The posted time is the average execution time of 100 method calls when setting the email body to the contents of a large event report. The class is not busy when the method is called.

class_EmailClient2.SetBodyString()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetBodyVector()

The posted time is the average execution time of 100 method calls when setting the email body to contain 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetBodyVector() With Event Report

The posted time is the average execution time of 100 method calls when setting the email body to the contents of a large event report. The class is not busy when the method is called.

class_EmailClient2.SetSender()

The posted time is the average execution time of 100 method calls when setting a maximum length email address and name. The class is not busy when the method is called.

class_EmailClient2.SetSubjectBytes()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetSubjectSELString()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetSubjectString()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

class_EmailClient2.SetSubjectVector()

The posted time is the average execution time of 100 method calls when setting the email subject to 255 characters. The class is not busy when the method is called.

Benchmark Results

Operation Tested	Platform (time in μs)		
	SEL-3505	SEL-3530	SEL-3555
fb_SimpleEmailClient Busy Time	714	449	40
fb_SimpleEmailClient Longest Scan Busy Time	81	49	5
fb_SimpleEmailClient2 Busy Time	703	446	40
fb_SimpleEmailClient2 Longest Scan Busy Time	11	52	8
class_EmailClient.AddRecipient()	120	72	13
class_EmailClient.AttachVector()	1543	871	78
class_EmailClient.ClearAttachments()	836	335	24
class_EmailClient.ClearRecipients()	5	2	1
class_EmailClient.Run() Busy Time	178	98	17
class_EmailClient.Run() Busy Time w/ER	245	139	17
class_EmailClient.Run() Longest Scan Time	871	518	56
class_EmailClient.Run() Longest Scan Time w/ER	3839	1640	83
class_EmailClient.Send()	883	537	68
class_EmailClient.SetBodyBytes()	828	498	60
class_EmailClient.SetBodyBytes() w/ER	93158	51279	5579
class_EmailClient.SetBodySELString()	988	539	50
class_EmailClient.SetBodySELString w/ER	115792	61122	5855
class_EmailClient.SetBodyString()	861	581	60
class_EmailClient.SetBodyVector()	807	519	53
class_EmailClient.SetBodyVector() w/ER	93246	51173	5553
class_EmailClient.SetSender()	98	58	14
class_EmailClient.SetSubjectBytes()	49	31	9
class_EmailClient.SetSubjectSELString()	1091	498	48
class_EmailClient.SetSubjectString()	66	42	11
class_EmailClient.SetSubjectVector()	17	9	4
class_EmailClient2.AddRecipient()	118	72	12
class_EmailClient2.AttachVector()	1569	845	80
class_EmailClient2.ClearAttachments()	962	322	24
class_EmailClient2.ClearRecipients()	5	2	1

Operation Tested	Platform (time in μs)		
	SEL-3505	SEL-3530	SEL-3555
class_EmailClient2.Run() Busy Time	183	97	16
class_EmailClient2.Run() Busy Time w/ER	245	142	16
class_EmailClient2.Run() Longest Scan Time	925	526	49
class_EmailClient2.Run() Longest Scan Time w/ER	3874	1628	86
class_EmailClient2.Send()	936	540	73
class_EmailClient2.SetBodyBytes()	842	518	55
class_EmailClient2.SetBodyBytes() w/ER	95720	52103	5609
class_EmailClient2.SetBodySELString()	988	537	49
class_EmailClient2.SetBodySELString w/ER	118489	62911	5864
class_EmailClient2.SetBodyString()	890	506	58
class_EmailClient2.SetBodyVector()	854	497	52
class_EmailClient2.SetBodyVector() w/ER	95879	52390	5596
class_EmailClient2.SetSender()	95	59	14
class_EmailClient2.SetSubjectBytes()	48	31	8
class_EmailClient2.SetSubjectSELString()	1306	531	48
class_EmailClient2.SetSubjectString()	68	43	11
class_EmailClient2.SetSubjectVector()	17	10	2

Examples

These examples demonstrate the capabilities of this library. Do not mistake them as suggestions or recommendations from SEL.

Implement the best practices of your organization when using these libraries. As the user of this library, you are responsible for ensuring correct implementation and verifying that the project using these libraries performs as expected.

Sending an Alert Email After a Failure

Objective

A user wants to notify correct team members of events occurring in the field.

Assumptions

The user has in some way defined triggers that will set Alarm1 or Alarm2 to TRUE when appropriate events occur and allow them to return FALSE afterwards.

Solution

The user can create a fb_SimpleEmailClient to provide brief notifications as shown in *Code Snippet 1*.

This example sends an email to different people determined by the alert received and also sends a text message to an on-call telephone number using Short Message Service (SMS).

Code Snippet 1 prg_EmailAlerts

```
PROGRAM prg_EmailAlerts
VAR
    Emailer : fb_SimpleEmailClient( localIPAddr := '0.0.0.0',
                                    mailServerIP := '192.168.176.99');
    FromEmail : STRING(254) := 'rtac6@myCompany.com';
    FromName : STRING(255) := 'The automation RTAC';
    //The number of the on call phone for an SMS message
    OncallPhone : STRING(255) := '5095552321@txt.att.net';
    DayEmail : STRING(254) := 'day_managers@myCompany.com';
    NightEmail : STRING(254) := 'night_managers@myCompany.com';
    ManagerEmail : STRING(254);
    SubjectLine : STRING(255) := 'Plant on Backup Power';
    BodyString : STRING(255);

    Alarm1 : BOOL;
    Alarm2 : BOOL;
END_VAR
```

```
IF Alarm1 THEN
    ManagerEmail := DayEmail;
    BodyString := 'The daytime plant has lost main power';
END_IF
IF Alarm2 THEN
    ManagerEmail := NightEmail;
    BodyString := 'The nighttime plant has lost main power';
END_IF
Emailer(   Send := Alarm1 OR Alarm2,
           ToEmail := OncallPhone, ToName := '',
           CcEmail := ManagerEmail, CcName := '',
           FromEmail := FromEmail, FromName := FromName,
           Subject := SubjectLine, BodyStr := BodyString);
```

Forwarding a Text Report From the RTAC

Objective

After the RTAC receives confirmation of the completion of an automated task, a user wants to receive a notice from the RTAC, including a log file of actions taken.

Assumptions

The user has a task that runs and builds a readable log file “details.txt” accessible through the File Manager features of the RTAC.

The user has in some way defined a trigger that will set TaskDone to TRUE when appropriate events occur and allow it to return false afterwards.

The user has included FileIo and DynamicVectors in their project.

NOTE: See the FileIO Library documentation for details on how to access this area programmatically.

Solution

The user can create a class_EmailClient to provide brief notifications as seen in *Code Snippet 2*.

This example sends an email containing the text of the log file to the user and a manager.

Code Snippet 2 prg_DailyReport

```
PROGRAM prg_DailyReport
VAR
    Emailer : class_EmailClient(  localIPAddr := '0.0.0.0',
                                  mailServerIP := '192.168.176.99');
    FromEmail : STRING(254) := 'rtac6@myCompany.com';
    FromName : STRING(255) := 'The automation RTAC';
    //Email addresses to receive the information.
    SecretEmail : STRING(254) := 'TheBoss@myCompany.com';
    DeveloperEmail : STRING(254) := 'developerLead@myCompany.com';
    SubjectLine : STRING(255) := 'The Numbers for the Day';

    Initialized : BOOL := FALSE;
    TaskDone : BOOL;
    FileRead : BOOL;

    FileBuffer : class_ByteVector;
    LogFileReader : class_FileReader;
END_VAR
```

```
IF NOT Initialized THEN
    Emailer.SetSender(FromEmail, FromName);
    Emailer.AddRecipient(MAIL_BCC, SecretEmail, 'Head Honcho');
    Emailer.AddRecipient(MAIL_TO, DeveloperEmail, '');
    Emailer.SetSubjectString(SubjectLine);
    Initialized := TRUE;
END_IF
IF TaskDone THEN
    LogFileReader.ReadFile('/details.txt');
    TaskDone := FALSE;
    FileRead := FALSE;
END_IF
IF NOT FileRead AND LogFileReader.BytesInBuffer > 0 THEN
    FileBuffer.Recycle();
    LogFileReader.AppendToVector(startByte := 0, vector := FileBuffer);
    Emailer.SetBodyVector(FileBuffer);
    Emailer.Send();
    FileRead := TRUE;
END_IF
Emailer.Run();
LogFileReader.Run();
```

Attaching Raw Data to Send

Objective

A user wishes to directly forward raw data on the RTAC device to various technicians.

Assumptions

The user has configured Alarm to trigger on events of some manner, and has also written data to ProcessControlData. The user has included DynamicVectors in their project.

Solution

The user can create a class_EmailClient to provide notifications with raw data attachments as shown in *Code Snippet 3*.

This example sends an email with raw data attachments to various technicians or other field personnel.

Code Snippet 3 prg_AttachData

```
PROGRAM prg_AttachData
VAR
    Alarm : BOOL := FALSE;
    Emailer : class_EmailClient( localIPAddr := '0.0.0.0',
                              mailServerIP := '192.168.176.7');
    FromEmail : STRING(254) := 'rtac6@myCompany.com';
    FromName : STRING(255) := 'The automation RTAC';
    Email : STRING(254) := 'tech_leads@myCompany.com';
    AttachmentName : STRING(255) := 'procData.raw';
    ProcessControlData : STRING(255) := '01110110101010';
    ControlData : class_ByteVector;
    SubjectLine : STRING(255) := 'Current Process Control Data';
    BodyString : STRING(255) := 'Raw process control data attached.';
    Initialized : BOOL := FALSE;
    trigger : R_TRIG;
END_VAR
```

Code Snippet 3 prg_AttachData (Continued)

```
trigger(clk := Alarm OR Emailer.Busy);
//Send email when an alarm occurs and we have vector data to send.
IF trigger.Q AND (LEN(ProcessControlData) > 0) THEN
    //Initializes the email parameters; assumes these do not change.
    IF NOT Initialized THEN
        Emailer.AddRecipient(MAIL_TO, Email, 'Lead Process Technicians');
        Emailer.SetSender(FromEmail,FromName);
        Emailer.SetBodyBytes(ADR(BodyString),LEN(BodyString));
        Emailer.SetSubjectBytes(ADR(SubjectLine),LEN(SubjectLine));
        Initialized := TRUE;
    END_IF

    // Clear any previous information from the vector
    ControlData.Recycle();
    // Append the new process information
    ControlData.Append(ADR(ProcessControlData),
         INT_TO_UDINT(LEN(ProcessControlData)));
    // Clear any previous attachments and add the new data.
    Emailer.ClearAttachments();
    Emailer.AttachVector(ControlData,AttachmentName);

    //Initiates the email send operation.
    IF NOT Emailer.Busy THEN
        Emailer.Send();
    END_IF
END_IF

// Call Run() every scan to complete the email.
Emailer.Run();
```

Troubleshooting

As a communication module, this library depends several things:

1. Correct IP address entered for both the local host (either 0.0.0.0 or an IP that can route to the mail server) and the mail server.
   - ➤ If using 0.0.0.0 as the local IP address, ensure the network that can reach the mail server is set as the primary gateway. This is set via the RTAC web HMI.
   - ➤ If the local IP address is set to the real IP address rather than 0.0.0.0, the primary gateway does not need to be set.
2. Access to the mail server via port 25.
3. The mail server must be configured to allow email from the RTAC. This may include but is not limited to: allowing the RTACs IP address to send mail, permitting the “from name” that the RTAC uses, as well as the “from email address.”

For full information as to what is happening to a sent message, check the logs on the mail server handling the request.

To quickly check the behavior of a given setup, one can manually test the communications from a computer on the same subnet as the RTAC by opening a raw TCP channel to the mail server on port 25 and issuing the same sequence of commands issued by this library. All sections in teal should be replaced with values from your environment. All new lines are represented by the ASCII for carriage return followed by new line (many applications will insert this just by pressing enter).

HELO RTAC_IP
MAIL FROM:<RTAC_EMAIL_ADDR>
RCPT TO:<TO_ADDR>
RCPT TO:<CC_ADDR>
DATA
From: "FROM_NAME"<RTAC_EMAIL_ADDR>
To: "TO_NAME"<TO_ADDR>
Cc: "CC_NAME"<CC_ADDR>
Subject: Email from RTAC

BODY OF MESSAGE
.
QUIT

Release Notes

Version	Summary of Revisions	Date Code
3.5.1.0	➤ Allows new versions of ACSELERATOR RTAC to compile projects for previous firmware versions without SEL IEC types "Cannot convert" messages. ➤ Must be used with R143 firmware or later.	20180921
3.5.0.6	➤ Full SMTP client/server handshaking implemented. ➤ Added fb_SimpleEmailClient2 and class_EmailClient2. These items extend the basic objects, providing destination and source port initialization parameters. ➤ Attachment functionality added for multiple files and raw data. ➤ Very large emails, over 10 KB in size, send correctly.	20150722
3.5.0.3	➤ Added correct HELO syntax when using local ip "0.0.0.0". ➤ Allow up to 10 seconds to connect with a mail server instead of a single scan.	20141212
3.5.0.2	➤ Improved sending of long emails.	20141110
3.5.0.1	➤ Initial release.	20141010