

Various Ways to Send an Email

Presented by: **Keisuke Miyako**



INTRODUCTION

Sending an email from 4D could be as easy as running a single line of code. Or, it could be more complicated than that. In this presentation I will explore the major updates made in **4D Internet Commands**, to give an up-to-date understanding of the various ways to send an email from 4D.

QUICK HISTORY

There are two standard commands to send an email in 4D Internet Commands, [SMTP_Send](#) and [SMTP_QuickSend](#). The latter is basically a "convenience" shorthand for the former command, so the idea is to **use it whenever possible** and keep your code simple. It doesn't make much sense to call a sequence of SMTP commands yourself, if plugin is prepared do it for you in just a single call.

The key, therefore, is to know the limits and constraints of the *QuickSend* command. For a long time, *QuickSend* accepted just a minimal set of parameters required to send an email. Here is its code signature from v2004/v11:

```
(Integer) - SMTP_QuickSend (host;from;to;subject;message)
```

As you can see, the command didn't even support **authentication**, let alone **encryption**. It dates back to a time when people weren't so sensitive about internet security.

Around the time when v2004/v11 was released, many Internet Services Providers started to adopt an authentication system known as [POP before SMTP](#). The idea was that while sending an email (SMTP) itself doesn't require a user name or password, the ISP would block access to its mail server if a successful POP3 authentication had not been made from the same IP address within the past few minutes.

In that context, one could use the *QuickSend*, which didn't support authentication, as long as a valid [POP3 Login](#), [POP3 Logout](#) call was made immediately before sending the message.







v11:

| | QuickSend | Send |
|-----------------|-----------|------|
| POP before SMTP | | |
| SMTP AUTH | | |

Eventually, [SMTP Authentication](#) became more common, which meant that now a user name and a password must be provided for sending an email. In addition, the introduction of [mail submission agents](#) and/or *Outbound Port 25 Blocking* required messages to be submitted to a port different to the default SMTP port (25), typically port 587.

Meanwhile, some email providers started to require encrypted connection, otherwise known as [SMTP over SSL](#). To keep up with the changes, 4D Internet Commands added SSL support in **v12.1**.

v12.1:

| | QuickSend | Send |
|---------------|---|---|
| SMTP AUTH |  |  |
| SMTP over SSL |  |  |
| STARTTLS |  |  |

Here is a code example to send an email using [SMTP over SSL](#):

```
$hostName:="smtp.gmail.com"
$from:="keisuke.miyako@gmail.com"
$to:="keisuke.miyako@4d.com"
$subject:=...
$body:=...
$user:="keisuke.miyako"
$password:=...
$useSSL:=1
$port:=465

$error:=SMTP_New ($smtpId)
If (0=$error)
  Case of
    : (0#IT_SetPort (12;$port))
    : (0#SMTP_Host ($smtpId;$hostName))
    : (0#SMTP_From ($smtpId;$from))
    : (0#SMTP_To ($smtpId;$to))
    : (0#SMTP_Subject ($smtpId;$subject))
    : (0#SMTP_Body ($smtpId;$body))
    : (0#SMTP_Auth ($smtpId;$user;$password))
  Else
    $error:=SMTP_Send ($smtpId;$useSSL)
  End case
  $error:=SMTP_Clear ($smtpId)
End if
```

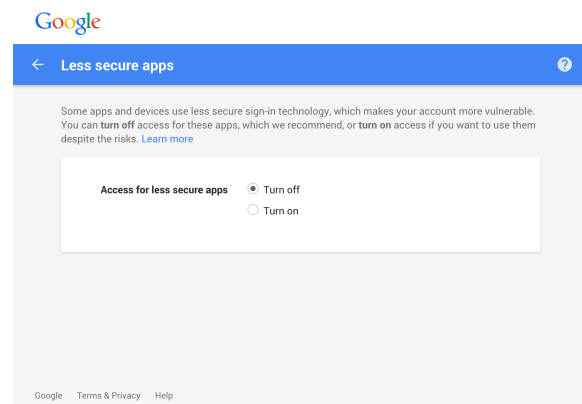
A couple of points:

1. The port number for SMTP over SSL must be set using *IT_SetPort* and the selector 12
2. The new argument to use SSL must be set to 1

The new "useSSL" parameter was added to both the *Send* and *QuickSend* commands. However, in reality, only the *Send* command would work, as most servers expect not just encryption but also user authentication, yet the *QuickSend* command supports only encryption and not authentication.

Support of SMTP over SSL is necessary for sending emails using a **Gmail account**. Note that the user still needs to specifically allow access for "less secure apps"; otherwise, 4D Internet Commands would throw an "Error with AUTHENTICATION" exception.

The basis for calling SMTP over SSL as less secure is that while the connection itself is encrypted by SSL (TLS) and secure, the application only needs to know the user name and password to access an account. OAuth, on the other hand, requires that the specifically consents to delegation, and retain the ability to revoke it at any time.



Support for [STARTTLS](#), the other kind of SSL used by **Microsoft Exchange** and other servers, was added with **v13.2**. Unlike SMTP over SSL, where the connection is **started in SSL using port 465**, the client application must first connect to the server **without using SSL on port 587**, and then switch over to SSL following a negotiation protocol.

Here is a code example to send an email using STARTTLS:

```
$hostName:="exchange.4d.com"
$from:="keisuke.miyako@4d.com"
$to:="keisuke.miyako@gmail.com"
$subject:...
$body:...
$user:="keisuke.miyako"
$password:...
$useSSL:=0
$port:=587

$error:=SMTP_New ($smtpId)
If (0=$error)
  Case of
    : (0#IT_SetPort (2;$port))
    : (0#SMTP_Host ($smtpId;$hostName))
    : (0#SMTP_From ($smtpId;$from))
    : (0#SMTP_To ($smtpId;$to))
    : (0#SMTP_Subject ($smtpId;$subject))
    : (0#SMTP_Body ($smtpId;$body))
    : (0#SMTP_Auth ($smtpId;$user;$password))
  Else
    $error:=SMTP_Send ($smtpId;$useSSL)
  End case
  $error:=SMTP_Clear ($smtpId)
End if
```

In summary:







1. The port number for STARTTLS must be set using *IT_SetPort* and the selector 2 (not 12)
2. The new argument to use SSL must be set to 0 (not 1)

QuickSend was extended to accept 3 more arguments, port, user name and password in **v13.2**.

Here is the new code signature:











```
(Integer) - SMTP_QuickSend (host;from;to;subject;message;param;port;user;password)
```

v13.2:

| | QuickSend | Send |
|---------------|---|---|
| SMTP AUTH |  |  |
| SMTP over SSL |  |  |
| STARTTLS |  |  |

QuickSend was again changed in **v14.0**. The parameter previous used to indicate whether connection is SSL (1) or not (0) has been redefined to mean whether the SSL is a **requirement** (1) or **optional** (0). The value 0 no longer means "no SSL", rather, it implies STARTTLS, that SSL is optional if the server supports it. The option value for "no SSL" has been changed from 0 to 2. The parameter is also used to specify the content type; plain, HTML or MIME.

v14.0:

| | | | | v11 | v12.1 | v13.2 | 14.0 |
|------|-----------|---|--------------------|-----|---|---|---|
| Send | QuickSend | 0 | no SSL | |  | | |
| | | 0 | STARTTLS | | |  |  |
| | | 1 | SMTP over SSL | |  |  |  |
| | | 2 | no SSL | | |  |  |
| | | 4 | HTML/STARTTLS | | | |  |
| | | 5 | HTML/SMTP over SSL | | | |  |
| | | 8 | MIME/STARTTLS | | | | |
| | | 9 | MIME/SMTP over SSL | | | | |

The extended options to specify the content type as well as the encryption mode apply only to *QuickSend*. When using the *Send* command, the content type can be set with *SMTP_Body* (v14.0) or *SMTP_AddHeader* (any version).

ENCODING

We all know that 4D itself has moved to **Unicode** in v11. But what about 4D Internet Commands? As it turns out, the plugin has carried on in non-Unicode world for quite a long time.

the v11 plugin ran in compatibility mode, similar to a 4D database or component running in non-Unicode mode. Unicode passed to the plugin was converted to ANSI and chopped at 32K bytes. As a result, some non-ASCII characters could get lost. Basically, the plugin was just as good (or, not good) as its v2004 predecessor.

UTF-8 was added to the list of character sets in v12. However, the plugin was still running in non-Unicode mode.

Consider the following code:

```
$error:=$SMTP_SetPrefs (1;15;0)
$error:=$SMTP_Charset (1;1)
```

According to the documentation, the option 15 specifies [UTF-8](#) as the character set and [Base64](#) as the encoding. Yet, because the plugin is not running in Unicode mode, what happens is that the 4D text passed to the plugin is first converted to ANSI and then that ANSI is converted back to UTF-8 before it is sent out as an email. As a result, extended characters could be lost, converted to a question mark '?' or displayed wrongly at the recipient's end.

In reality, the only viable options were to use Windows-1252 (1), US-ASCII (2,3,4) or ISO-8859-1 (5, 6, 7, 8).

Note: ANSI, in this context, is used to denote the internal non-Unicode character set used in older version of 4D. It would be a cross platform version of the MacRoman set on a computer with a typical western language setting.

True Unicode support came with **v14.0**. The following line was added to the documentation:

Starting with 4D v14, the "subject" and "body" fields of SMTP commands use the UTF-8 character set by default. This character set will be interpreted correctly by almost all of the e-mail clients. This functioning greatly simplifies the use of SMTP commands and now limits the usefulness of the SMTP_Charset and SMTP_SetPrefs commands.

Notice how it falls a few steps short of calling the use of any character sets other than UTF-8 as obsolete. In fact, the original text which appeared in the [v14 Upgrade documentation](#) used the expression "unnecessary". Indeed, v14 ignored any call to *Charset* or *SetPrefs* and used the UTF-8 character set and base64 encoding anyway, for both the subject and body. Suddenly, it became impossible to send emails using any other options but the default.

That, was arguably a step too far. Some mobile devices, as well as a few email clients, do not have satisfactory Unicode support. Sometimes, it is necessary to send an email in non-Unicode. The bug, ACI0093983 ACI0092449 ACI0090674, was fixed in the following releases: v14.4 HF1, v15.1.HF1, 15R3.

ATTACHMENT

One thing *QuickSend* doesn't support is attaching files to an email. That would be one reason to use *Send* instead.

| | encoding | |
|---|--------------------|---|
| 0 | none | SMTP was originally designed to send plain text messages only using a 7-bit encoding system (ASCII). Any files must be encoding and attached following the MIME standard. Although 4D Internet Commands support a variety of encodings, keep in mind that most of them are deprecated in favour of Base64 . For example, BinHex , a system once popular on Mac OS, isn't even supported in the latest Apple Mail software on iOS or OS X. |
| 1 | BinHex | |
| 2 | Base64 | |
| 3 | AppleSingle | The <i>SMTP_Attachment</i> command expects either a full path or a short name, in which case the file is assumed to be adjacent to the structure file. You can't attach a picture or BLOB directly, the data must come from a file on the disk. |
| 4 | AppleDouble | |
| 5 | AppleSingle+Base64 | The plugin is designed to display a "select file" dialog if an empty string was provided instead of a path. <u>This feature has been removed in the Mac 64 bit version of 4D Internet Commands</u> ; an "unimplemented dialog error" -2201 is returned. If you need to let the user specify a file to attach, you should use native 4D commands prior to passing the path to the plugin. |
| 6 | AppleDouble+Base64 | |
| 7 | UUEncode | |

A new feature was added in **14R4**, which allows one to associate a content ID with an email attachment. The feature was mainly intended especially for the use with HTML emails, where a content ID could be used to "embed" an image, as opposed to a hyperlink reference of an image file that would require an extra trip to a web server.

```
$error:=SMTP_Attachment ($smtpId;$filePath;2;0;$contentID;$contentType)
```

For example, if an image file was attached with the content ID "img1", then its tag in the HTML would look like this:

```

```

There was another addition in **14R4**, which allows one to specify the content type of an attachment. If none is provided, the plugin will add a content type based on the file's extension name. However, not all types are matched automatically. You might want to set the MIME type yourself. (.txt, .docx, for example)

Being more specific about the [MIME type](#) than the default generic "application/octet-stream" helps the recipient email client to display a preview of the document or launch an appropriate application when asked by the user to open it.

| extension | MIME |
|-----------|-------------------------------|
| jpg, jpeg | image/jpeg |
| png | image/png |
| gif | image/gif |
| pdf | application/pdf |
| doc | application/msword |
| xls | application/vnd.ms-excel |
| ppt | application/vnd.ms-powerpoint |
| zip | application/zip |
| gz | application/gzip |
| json | application/json |
| js | application/javascript |
| ps | application/postscript |
| xml | application/xml |
| htm, html | text/html |
| mp3 | audio/mpeg |
| | application/octet-stream |

HTML

Now that we have looked back on the various enhancements made to 4D Internet Commands, let's focus on a more practical subject: What is the current best practice with regards to sending an email in HTML format?

At the most basic level, sending an email in HTML is no different to sending an email in plain text. You just set the "Content-Type" header to "text/html" and pass the HTML source text as the body. The plugin does not have a specific API to set the "Content-Type", but the generic command *SMTP_AddHeader* can be used instead.

```
$error:=SMTP_AddHeader ($smtpId;"Content-Type";"text/html; charset=utf-8";0)
```

The process has been simplified in **v14.0**. The "text/html" content type can now be set with the *Body* and *QuickSend* commands.

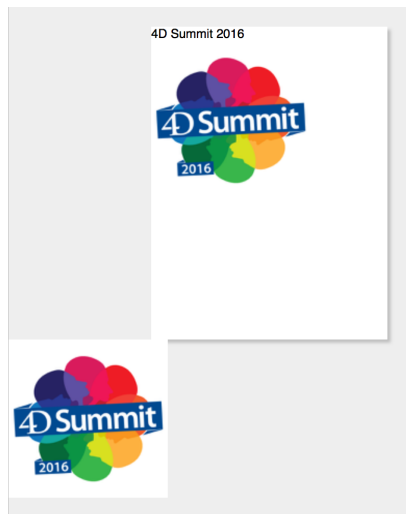
```
$error:=SMTP_Body ($smtpId;$body;4)
```

The above is effectively an abbreviation of the following:

```
$error:=SMTP_AddHeader ($smtpId;"Content-Type";"text/html; charset=utf-8";0)
$error:=SMTP_Body ($smtpId;$body)
```

Today, many email servers automatically change the MIME type to the [RFC1521](#) "multipart/alternative" type and add a plain text version of the message on top of the original HTML body. Although this is helpful for email client that prefer the message as plain text, the conversion could be unreliable and there is possibility that the intent of the email is misrepresented or lost. Unfortunately, a standard use of 4D Internet Commands will always result in a "multipart/mixed" content-type, in which you can have either a message in plain text or HTML, but not both.

As mentioned earlier, you can easily send an email with embedded images, thanks to the "Content-ID" option added to *Attachment* command. However, you might have noticed that the recipient email client displays a copy of the attachment at the end of the message, in addition to the one rendered in context.



This is due to the fact that the MIME type is "multipart/mixed", as opposed to "multipart/related", in which case Apple email client will display the associated files again at the end of the body.

SUMMARY OF ISSUES

So far we have identified a couple of problems with the default behaviour of 4D Internet Commands when sending an email in HTML format:

1. The body must be plain or HTML. The server may add a plain text version of the message, but we have no control over the exact rendering.
2. Though image files can be embedded and referenced in the HTML using a content ID, we can't prevent it being displayed twice in the recipients inbox.

One way to overcome these issues is to construct the MIME message yourself.

MIME

You may not have thought of it this way, but the plugin is actually performing two distinct tasks when you use its SMTP command suite; first it will construct a MIME representation of your message, and then it submits that message to an email server following the SMTP protocol. The first part is pure string manipulation, as opposed to the latter which is a network communication task. Could it be possible to do the first half without using the plugin?

The sample application contains a set of methods to procedurally construct your own MIME. It is capable of producing the following type of emails:

- Plain text
- HTML
- HTML with plain text alternative
- Each of the above, but with attachments

You can also set the MIME type to "multipart/related" instead of "multipart/mixed", and specify whether an attachment's content disposition should be "inline" as opposed to "attachment".

An example of using the MIME constructor methods:

```
$charset="utf-8"
$message=MIME_New ("multipart/related";$charset)

MIME_ADD_HEADER ($message;"From";"MIYAKO <keisuke.miyako@4d.com>")
MIME_ADD_HEADER ($message;"To";"MIYAKO <keisuke.miyako@4d.com>")
MIME_ADD_HEADER ($message;"Subject";"test_"+Generate UUID)

$image=Get_resource_image ("summit.png")

MIME_ADD_PART ($message;->$image;"logo.png";"image/png";$charset;"img1";"inline")

$plain="plain message"
$html="<html><body><strong>html</strong>"+\
  "<img src=\"cid:img1\" />message</body></html>"

MIME_ADD_ALTERNATIVE_PART ($message;$plain;$html;$charset)

$MIME=MIME_Export_to_variable ($message)
```

Now that we have our own raw message, we can use the new *QuickSend* options to pass a MIME directly. The feature was added in **14R5** primarily to simplify the process of sending a [4D Write Pro](#) document, but it can be used with a MIME created using any method. Just keep in mind that the standard SMTP headers (from, to, date, subject) should be left out for the plugin to complete.

14R5:

| | | | | v12.1 | v13.2 | 14.0 | 14R5 |
|--|-----------|---|--------------------|-------|-------|------|------|
| | QuickSend | 8 | MIME/STARTTLS | | | | ○ |
| | | 9 | MIME/SMTP over SSL | | | | ○ |

Notice how we have effectively overcome an intrinsic limitation of *QuickSend*; we can now send attachments!

BEYOND 4D INTERNET COMMANDS

The 4D Internet Commands plugin is designed to help developers send emails without having to directly deal with the underlying technologies such as MIME SMTP or SSL. Still, as demonstrated in the previous section, it is possible to take over the MIME part to some degree. Technically, you can even take over the SMTP part, by using the low-level TCP commands and implement SMTP yourself. However, that should generally be considered as a last resort.

One features not supported by 4D Internet Commands is [OAuth 2.0](#), which is now the standard authentication method for external applications to access Gmail.

If you find a certain aspect of the plugin to be unsatisfactory, you might want to evaluate the available alternate methods before rolling out your own SMTP client. After all, 4D Internet Commands is only a free utility plugin.

- 4D PDF Printer <http://www.node.de/indexplug.htm>

The HTML email feature is offered as a freeware (Windows)

- Networking ToolKit <https://www.pluggers.nl/product/ntk-plugin/>

Professional SMTP functionality

The sample application contains an example of using cURL to send emails from 4D.

CONCLUSION

In this presentation we have explored the major updates made in **4D Internet Commands**. Major changes happened with the versions 12.1, 13.2, 14.0, 14R4 and 14R5. To recap on the revision history:

| version | | QuickSend | Send |
|---------|--|-----------|------|
| 12.1 | SMTP over SSL (without authentication) | ○ | ○ |
| | SMTP over SSL (with authentication) | | ○ |
| 13.2 | SMTP over SSL (with authentication) | ○ | ○ |
| | STARTTLS | ○ | ○ |
| | Set content type as text/html | ○ | ○ |
| 14.0 | Set message ID | | ○ |
| | Unicode support | ○ | ○ |
| 14R4 | Set content ID | | ○ |
| 14R5 | Send MIME directly | ○ | |

As general rule, the *QuickSend* command should be used whenever possible, to keep the code simple. Although the commands doesn't support attachments, you can effectively overcome the limitation by constructing the MIME procedurally. Doing so comes with the added bonus of having more control over the how an attachment image is displayed by the recipient's email client. Some advanced features, such as OAuth 2.0, are currently not support by the plugin. If that becomes a problem, you would have to look beyond 4D Internet Commands or roll out your own.