



# SMTP Client Deux

## Developer Documentation v1.2.0b01

©1998-2002 Deep Sky Technologies, Inc. All Rights Reserved.  
Published and Distributed Worldwide by Deep Sky Technologies, Inc.

Deep Sky Technologies, Inc.  
P.O. Box 6897  
Vero Beach, FL 32961-6897  
772.794.9494



## Software License and Limited Warranty

Please read this license carefully before using the software. By using the software, you agree to become bound by the terms of this agreement, which includes the software license and warranty disclaimer (collectively referred to herein as the "agreement"). This agreement constitutes the complete agreement between you and Deep Sky Technologies, Inc. If you do not agree to the terms of this agreement, do not use the software and promptly destroy all copies in your possession, physical and electronically.

**1. Ownership of Software:** The enclosed manual and computer programs ("Software") were developed and are copyrighted by Deep Sky Technologies, Inc. ("DSTi") and are licensed, not sold, to you by DSTi for use under the following terms, and DSTi reserves any rights not expressly granted to you. DSTi retains ownership of all copies of the Software itself. Neither the manual nor the Software may be copied in whole or in part except as explicitly stated below.

**2. License:** DSTi, as Licensor, grants to you, the Licensee, a non-exclusive, non-transferable right to use this Software subject to the terms of the license as described below:

- a. You may make backup copies of the Software for your use provided they bear the DSTi copyright notice.
- b. You may use this Software in an unlimited number of custom or commercial databases or applications created by the original licensee. No additional product license or royalty is required.

**3. Restrictions:** You may not distribute copies of the Software to others (except as an integral part of a database or application within the terms of this License) or electronically transfer the Software from one computer to another over a network. You may distribute copies of the Software as an integral part of a development shell or non-compiled commercial database as long as the DSTi copyright notices and documentation remain intact with the distribution. The Software contains trade secrets and to protect them you may not decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human perceivable form. You may not modify, adapt, translate, rent, lease, loan or resell for profit the software or any part thereof.

**4. Termination:** This license is effective until terminated. This license will terminate immediately without notice from DSTi if you fail to comply with any of its provisions. Upon termination you must

destroy the Software and all copies thereof, and you may terminate this license at any time by doing so.

**5. Update Policy:** DSTi may create, from time to time, updated versions of the Software. At its option, DSTi will make such updates available to the Licensee.

**6. Warranty Disclaimer:** The software is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. DSTi does not warrant, guarantee, or make any representations regarding the use, or the results of the use, of the software or written materials in the terms of correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of the software is assumed by the Licensee. If the software or written materials are defective you, and not DSTi or its dealers, distributors, agents, or employees, assume the entire cost of all necessary servicing, repair or correction. No oral or written information or advice given by DSTi, its dealers, distributors, agents, or employees shall create a warranty or in any way increase the scope of this warranty, and you may not rely on such information or advice. This warranty gives you specific legal rights. You may have other rights, which vary from state to state.

**7. Governing Law:** This agreement shall be governed by the laws of the State of Florida.

## Copyrights and Trademarks

All trade names referenced in this document are the trademark or registered trademark of their respective holder.

BASh, TCP Deux, SMTP Client Deux, POP3 Deux, FTP Client Deux, HTTP Client Deux, eTrans, TCP Server Deux, HTTP Server Deux, and HTTP Log Deux are copyright Deep Sky Technologies, Inc.

4th Dimension, ACI, ACI US, 4D Compiler, 4D, 4D Server, 4D Client, and 4D Insider are trademarks of 4D, Inc.

Macintosh and MacOS are trademarks of Apple Computer, Inc.

Windows is a trademark of Microsoft Corporation.

## Preface

The SMTP Client Deux component is designed to work in conjunction with many other components. Specifically, the SMTP Client Deux component requires that the BASH component, available for free from DSTi, and TCP Deux component both be installed already in your database structure file.

Make certain that you view the compatibility matrix for components available to make certain you are using compatible versions of the different components required. There is a compatibility matrix available in this manual; the most recent compatibility matrix is available on the Deep Sky Technologies, Inc., web site.

## Acknowledgements

The creation of the SMTP Client Deux component is not directly attributable to any single person. Particular pieces of functionality within the SMTP Client Deux component may be from the direct knowledge and experience of certain developers, but the overall concept and construction of the SMTP Client Deux component has come from all of the developers at Deep Sky Technologies, Inc.

Tweaks and additions to the SMTP Client Deux component have resulted from the training and development efforts of James A. Crate. Mr. Crate's experience in many different programming environments has provided refreshing insights into the overall structure and organization of the core routines at DSTi, the same core routines which are available in the BASH and SMTP Client Deux components.

Finally, I, Steven G. Willis, might have had something to do with the creation of the SMTP Client Deux component...

## Features

SMTP Client Deux is a 4th Dimension component that provides a set of SMTP routines for sending email from 4D. SMTP Client Deux works on top of TCP Deux, providing a common code set for sending email that will work with all of the major TCP plugins available currently for 4D (4D Internet command v6.7.x, Internet ToolKit v2.0.x, and Internet ToolKit v2.5.x).

SMTP Client Deux is made available for free. Of course, it does require the purchase of TCP Deux to work properly. But, with TCP Deux, SMTP Client Deux provides then full email sending functionality without any of plugin specific drawbacks of any single TCP plugin.

The single greatest feature of SMTP Client Deux is that it does not use the SMTP layer within 4D Internet Commands. Over the years, many developers have encountered different bugs and anomalies within the SMTP implementation of 4D Internet Commands. SMTP Client Deux uses the routines within the TCP Deux component package for all TCP communications, which themselves only use the TCP layer within 4D Internet Commands when that plugin is in use. So, most if not all errors which 4D developers experience when sending email with 4D Internet Commands are removed when using SMTP Client Deux.

As well, SMTP Client Deux provides a full and complete implementation of the SMTP protocol. So, for those developers using either version of Internet ToolKit and experiencing difficulty setting up their own email sending routines, there is no longer any need to go through this hassle. SMTP Client Deux provides you with successive email sending routines so that you can choose the implementation that works best for your email sending needs.

SMTP Client Deux includes full compatibility with SMTP AUTH, authenticated login for SMTP servers. When enabled, SMTP Client Deux will choose the most secure means supported by the SMTP server to login. This includes CRAM-MD5, NTLM, Login, and Plain SMTP AUTH mechanisms. There is currently no other solution available in 4D that provides complete support for SMTP AUTH.

All of the code within the SMTP Client Deux component is ready to use immediately. There are no compiler issues to be concerned about; all of the variables are typed in compiler methods for even the most stringent 4D developers. Just install the component in your structure and start calling the methods within it. It is really that simple!

## System Requirements

The SMTP Client Deux component is compatible with both Macintosh and Windows installations of 4th Dimension.

Since it is a component, it does require at least version 6.7 of 4th Dimension or above, including 4D Insider v6.7 or above for installation.

Other than the normal hardware and software requirements for your version of 4th Dimension, there are no other minimum requirements for proper use of this software.



## Support

Support is provided for the SMTP Client Deux component free of charge for all currently licensed users. Included support services provided for all currently licensed users encompasses all of the online support services available through the DSTi web site (email, FAQ, messaging, etc.). Check the DSTi web site for current direct support options available; we are always working to offer more resources for your needs.

Contact information, including email address(es), phone number(s), and a Contact Us request form, for Deep Sky Technologies, Inc., can be found on the DSTi web site located at:

<http://www.deepskytech.com/>

If there are terms or conventions which you find difficult to understand in relation to the SMTP Client Deux component or TCP protocols and servers in general, feel free to contact Deep Sky Technologies, Inc., support. We will be more than happy to help you in any way we reasonably can. And, only through your questions do we know what subjects to include in future versions of this manual.

## Components

A component groups various 4D objects (tables, project methods, forms, menu bars, variables, etc.) representing one or more additional functions. Developing a 4D component providing electronic mail functionality is one such example. A component is autonomous and must be able to be installed in any 4D structure.

Components are defined, generated, and installed with the help of 4D Insider. The component definition is based on the cross referencing analysis performed by 4D Insider (target objects and source objects).

Unlike libraries and groups, components embed the idea of security of objects that they compose. During the development phase of the component, each object is attributed an access type, "Public", "Protected" or "Private". This attribute determines whether each object will be visible or modifiable in 4th Dimension and in 4D Insider once the component is installed within a 4D database.

## Installing and Updating SMTP Client Deux

Installing SMTP Client Deux or updating an existing version of SMTP Client Deux within a 4D database is performed using 4D Insider. The activity primarily consists of installing the SMTP Client Deux component in a database structure opened with 4D Insider (installing the SMTP Client Deux component in a library is not supported at this time).

4D Insider will manage possible conflict issues within the installation and will inform you as they are detected. Though, with the naming conventions used within the SMTP Client Deux component and the limited number of object names, conflicts should be very rare.

To install or update the SMTP Client Deux component, follow these very simple steps:

**Open the uncompiled structure that you wish to install SMTP Client Deux into using 4D Insider.**

**Choose the "Install/Update..." command in the "Components" menu.**

A standard open file dialog box will appear.

**Select the SMTP Client Deux component file and click on the "Open" button.**

4D Insider parses the SMTP Client Deux component and prepares to integrate it with your open database. 4D Insider will detect if the operation is an installation or an update of the SMTP Client Deux component.

In the event of a new installation, all SMTP Client Deux objects are installed.

In the event of an update, 4D Insider compares the version numbers of both the currently installing SMTP Client Deux component and the already installed SMTP Client Deux component. If the date of the "new" component is older than the already installed component, a dialog box will alert you, allowing you to then "Continue" or "Cancel" the update.

4D Insider replaces old objects with newer objects within the SMTP Client Deux component and adds new objects from the new SMTP Client Deux component. 4D Insider takes into account "public" objects having been modified by you (e.g.

"\_ERROR" methods) and will prompt you to either save or replace them. If any other conflicts arise from the installation or update of the SMTP Client Deux component, 4D Insider will prompt you with an appropriate dialog box.

### **Save the database in 4D Insider.**

### **Place a copy of the Affix SMTPcd document in the 4DX folder.**

The Affix SMTPcd document contains essential data and resources for many of the methods within the SMTP Client Deux component. For many of the methods within the SMTP Client Deux component to function properly, the Affix SMTPcd document must be in the 4DX folder for the current structure.

On Macintosh, the Affix SMTPcd document is entitled **Affix\_SMTPc\_Deux.4DX** (under 4D v6.8.x, there is a carbonized version entitled **Affix\_SMTPc\_Deux.4CX**) and is located in the Mac4DX directory in the SMTP Client Deux component's archive. The document should be copied into the Mac4DX folder of your current structure. If the SMTP Client Deux component is going to be used in all of your 4D projects, the Affix SMTPcd document can instead be placed within the Mac4DX folder within the 4D folder of your system.

On Windows, the Affix SMTPcd document is actually two documents: **Affix\_SMTPc\_Deux.4DX** and **Affix\_SMTPc\_Deux.RSR**. These two documents correspond to the data fork and the resource fork of the Affix SMTPcd document used on Macintosh. These documents are located in the Win4DX directory in the SMTP Client Deux component's archive. These documents should be copied into the Win4DX folder of your current structure. If the SMTP Client Deux component is going to be used in all of your 4D projects, the Affix SMTPcd document can instead be placed within the Win4DX folder within the 4D folder of your system.

For client/server installations in cross-platform environments, both the Macintosh and Windows versions of the Affix SMTPcd document should be installed.

### **Call the method *INIT\_SMTPcd* early in the *On Startup* and *On Server Startup* database methods.**

To initialize the SMTP Client Deux component in your code, place a call to the method ***INIT\_SMTPcd*** early in your ***On Startup*** and ***On Server Startup*** database methods. However, this method must be after the calls to ***INIT\_BASH*** and ***INIT\_TCPd***.

Details about the ***INIT\_SMTPcd*** method can be found in the module and method documentation, below.

The SMTP Client Deux component is now installed/updated in your database and is listed on the "Components" page of the 4D explorer.

## Managing Installation Conflicts

On very rare occasions, when the SMTP Client Deux component is installed or updated in your 4D database, several questions and conflicts may arise. In the event of an update, 4D Insider will detect that you have modified one or more "Public" objects in SMTP Client Deux after the initial installation. Or, one or more objects of the same type and of the same name may already exist in your database and in the SMTP Client Deux component.

4D Insider detects and solves these conflicts during installation:

### Modified public objects (updates only)

In this case, 4D Insider alerts you by a dialog box, allowing you to choose an update mode:

Replace the object

Replace all objects

Do not replace the object

Stop installation

### Name conflicts

In this case, 4D Insider stops the SMTP Client Deux installation process, alerts you through a dialog box and saves the

list of objects in conflict. This list is stored as a text file in the 4D database folder.

Naming conflicts between logical objects, such as variables, are managed by 4D Insider, in a manner that allows database compilation and avoids conflicts between SMTP Client Deux and other 4D components.

It may be necessary to rename certain objects in your database or in other components in order to be able to install the SMTP Client Deux component.

If any naming conflicts do occur between SMTP Client Deux and other 4D components, please notify Deep Sky Technologies, Inc., immediately.

### Affix SMTPcd Document

The Affix SMTPcd document (entitled **Affix\_SMTPc\_Deux.4DX** on Macintosh; entitled **Affix\_SMTPc\_Deux.4DX** and **Affix\_SMTPc\_Deux.RSR** on Windows; under 4D v6.8.x, there is a carbonized version entitled **Affix\_SMTPc\_Deux.4CX**)) contains essential data and resources for many of the methods within the SMTP Client Deux component. For many of the methods within the SMTP Client Deux component to function properly, the Affix SMTPcd document must be in the 4DX folder for the current structure. For distributed and installed versions of your 4D projects, the Affix SMTPcd document must be available in the 4DX folder for the SMTP Client Deux methods to continue to function properly.

**Note:** it is best to consider the Affix SMTPcd document another plug-in within your 4D project. Though there no actual plug-in calls available within the Affix SMTPcd document, it does contain data and resources essential to the operation of the SMTP Client Deux methods. The SMTP Client Deux component has been designed to find the Affix SMTPcd document correctly in all environments (any platform, any 4DX folder, single user or clients/server, etc.). Since the Affix SMTPcd document is configured similarly to plug-ins, 4D and the SMTP Client Deux component will automatically manage the document for you in all of the possible installations of a 4D project.

## 4D v6.8.x

With the availability of 4D v6.8.x, the complete 4th Dimension environment is now fully carbonized. 4D as a carbonized application allows for a single set of tools to function on either MacOS X or MacOS v9.x using CarbonLib.

With the release of 4D v6.8.x, there is now a new plugin architecture available for third party developers. As well, there are some changes which have been made in the actual plugin hierarchy and naming conventions. There have also been changes made to the component architecture within the 4D product line. Reading the release notes for 4D v6.8.x is a great source of information regarding these changes.

SMTP Client Deux is currently available with compatibility for 4D v6.8.x. This comes in the form of new affix documents for use with 4D v6.8.x. The SMTP Client Deux archive contains both 4D v6.7.x and 4D v6.8.x compatible versions of the component and affix documents.

When updating an existing database from 4D v6.7.x to 4D v6.8.x, we have found that installed components will no longer update properly. The first time that a component is updated after upgrading a 4D structure, the component must first be removed from the structure before installing the new version of the component. We have not seen any problems with doing this other than the extra step required to remove the component using 4D Insider.

It is also important that you install the correct affix documents for your environment. Whether you are running under MacOS X or MacOS 9 is not a factor. Rather, whether you are using 4D v6.7.x or 4D v6.8.x is the determining factor. Copy the appropriate documents for your current version of 4D from the component archive for use in your 4D structure. The differences between the 4D v6.7.x and 4D v6.8.x affix documents is simple to determine; the names of the documents compatible with 4D v6.7.x end in ".4DX" and the names of the documents compatible with 4D v6.8.x end in ".4CX".

Since 4D v6.8.x is still currently in beta, it is best to report any issues with components immediately to Deep Sky Technologies, Inc. We can research the issue very quickly and notify 4D, Inc./4D SA of any compatibility and functional problems quickly so that the final release of 4D v6.8.x is as bug free as possible.

## Uninstalling SMTP Client Deux

4D Insider allows you to uninstall the SMTP Client Deux component from your 4D database.

To uninstall SMTP Client Deux from your 4D database:

**Using 4D Insider, open your database containing the copy of SMTP Client Deux to be uninstalled.**

**In the "Main" listing window, select the SMTP Client Deux component.**

**Consider again how great the SMTP Client Deux component is and make certain that you will really no longer need it in your 4D database.**

**Select the "Uninstall..." command in the "Components" menu.**

This command is only active when a component is installed in the database. A dialog box appears allowing you to confirm or cancel the operation. If you uncertain about the previous step then the cancel option is probably your best choice at this time.

**Click "OK" to validate the operation.**

**Remove the Affix SMTPcd document from your 4DX folder.**

**Remove the call to the method *INIT\_SMTPcd* from your *On Startup* and *On Server Startup* database methods.**

All objects from the SMTP Client Deux component are deleted from your 4D database. Obviously, you are now very sad to no longer have the SMTP Client Deux component in your 4D database. Crying is allowed...



## SMTP Client Deux Conventions

Throughout this manual, and all other documentation and supporting materials, included with the SMTP Client Deux component package, there are different core knowledge which is essential to know and understand. With this knowledge, basically concerning the conventions used on TCP networks and conventions used within the SMTP Client Deux component, you will be able to more easily and efficiently utilize the functionality available within this software package.

If there are other terms or conventions which you find difficult to understand in relation to the SMTP Client Deux component or TCP networks in general, feel free to contact Deep Sky Technologies, Inc., support. We will be more than happy to help you in any way we reasonably can. And, only through your questions do we know what subjects to include in future versions of this manual.

## Basics of SMTP

SMTP, the Simple Mail Transfer Protocol, is an Internet standard for sending email. SMTP is used to not only to begin the sending process, that of transferring an email message from a regular email client to an email server, but SMTP is also used to deliver email from an originating email server to a destination email server. In other words, the exact same protocol that is used to start the sending of an email is also used to transfer email from place to place on the Internet.

The procedure for implementing SMTP provides a mechanism for the originator and destination servers and addresses to be specified independent of the actual message contents. It is important to understand that this mechanism for specifying the originator and destination data is independent of the actual email content; and, the email content includes not just the message and subject being sent, but the headers lines which included with the email message. Because of this disparity, it is a simple matter for the source and destination information to be reasonably masked from the message which is actually delivered, hence the complexity of preventing spam (unsolicited email messages).

This section tries to provide the basic information needed to understand the different features of any SMTP system. In no way is this explanation complete. But, it should be enough to provide most 4D implementations of SMTP with more than enough details to do whatever is reasonably needed.

For more information regarding the SMTP protocol, it is recommended that the reader refer to RFCs 821 (SMTP), 822 (SMTP Headers), 2554 (SMTP AUTH), and 2822 (Text Message Formats). You can download any of these RFCs by following these links:

[http://www.deepskytech.com/rfcs/rfc\\_0821.txt](http://www.deepskytech.com/rfcs/rfc_0821.txt)

[http://www.deepskytech.com/rfcs/rfc\\_0822.txt](http://www.deepskytech.com/rfcs/rfc_0822.txt)

[http://www.deepskytech.com/rfcs/rfc\\_2554.txt](http://www.deepskytech.com/rfcs/rfc_2554.txt)

[http://www.deepskytech.com/rfcs/rfc\\_2822.txt](http://www.deepskytech.com/rfcs/rfc_2822.txt)

## Mail Header Lines

Within the contents of an SMTP message, commonly called an email message, there are standard header lines used as indicators for routing, recipient, sender, priority, content details, and message details. These header lines have a specific format that for most purposes consists of a named value pair delimited by a colon. Values which are exceptionally long can wrap to multiple lines (a new line being defined as a carriage return followed by a linefeed) by subsequent lines beginning with a space; often, these lines are called "folded" lines.

Some mail header lines are added by a mail server when it routes a message. Most email header lines are the responsibility of the originating sender host to include correctly.

Within SMTP Client Deux, the setting of the actual header lines are handled internally by the methods ***SMTPcd\_Send\_Message\_Simple*** and ***SMTPcd\_Send\_Message\_Attachment***. Only when greater control over the actual contents of the message being sent should the method ***SMTPcd\_Send\_Message\_Complex*** be needed by the developer. And, when using this method, the developer has almost complete control when using SMTP Client Deux to set all of the header lines within a message.

SMTP Client Deux includes a set of constants, SMTPcd\_Header\_Codes, for specifying the commonly supported header line types. It is important to format the data for each header line correctly. RFC 822 is the best source of information for the actual format of the different header lines. RFC 822 can be downloaded from:

[http://www.deepskytech.com/rfcs/rfc\\_0822.txt](http://www.deepskytech.com/rfcs/rfc_0822.txt)

Often, SMTP server will not restrict messages that contain improperly formatted header lines. But, this can cause problems for the recipients of messages as it may make it difficult if not impossible to read the message.

## Formatted Recipients

For header lines that specify recipients, or senders, of an SMTP message, there is a particular format for the values which is worth going over. These header lines include the following:

### Constant

SMTPcd\_FROM\_Type  
 SMTPcd\_TO\_Type  
 SMTPcd\_CC\_Type  
 SMTPcd\_BCC\_Type  
 SMTPcd\_Reply\_To\_Type  
 SMTPcd\_Sender\_Type  
 SMTPcd\_Resent\_Reply\_To\_Type  
 SMTPcd\_Resent\_From\_Type  
 SMTPcd\_Resent\_Sender\_Type  
 SMTPcd\_Resent\_To\_Type  
 SMTPcd\_Resent\_CC\_Type  
 SMTPcd\_Resent\_BCC\_Type

For these header lines, a recipient, whether that be an account receiving the message, a sender, or another account referenced, will consists of multiple pieces of information. For all practical purposes, these pieces of information will be the recipients name and recipients email address.

For instance, if a message were to be sent to Steven G. Willis at Deep Sky Technologies, Inc., then the recipient's name would be "Steven G. Willis" and the recipient's email address would be "sgwillis@deep-skytech.com".

The formatting for these values in a header line should always be the recipient name in double quotations followed by a space, and then the recipient address in angled brackets. So, a properly formatted recipient for the above example would be:

"Steven G. Willis" <sgwillis@deepskytech.com>

Multiple recipients within a single header line should be delimited by a comma and a space. Though, for values cooresponding to the SMTPcd\_FROM\_Type, SMTPcd\_TO\_Type, SMTPcd\_CC\_Type, and

SMTPcd\_BCC\_Type in the parameter arrays for the method ***SMTPcd\_Send\_Message\_Complex***, only one formatted address should be specified for each array element.

There are methods within SMTP Client Deux to help with the creation and parsing of formatted recipient values. See the section documenting the methods ***SMTPcd\_Get\_Recipient\_Formatted***, ***SMTPcd\_Extract\_EmailAddress***, and ***SMTPcd\_Extract\_FullName*** for more details on these methods.

## Recipients Versus Header Lines

As mentioned previously, the actually originating and destination address of an SMTP message is set separately from the values which are set into the header lines of a message. Spammers often take advantage of this fact to send spam to email accounts; and, it is not uncommon for a recipient of such spam to wonder how they received it when they are not even listed as a recipient in the headers of the message.

For practical reasons, SMTP Client Deux does **not** discriminate between these two; in other words, even when using the method ***SMTPcd\_Send\_Message\_Complex***, the recipients of an SMTP message are pulled directly from the headers lines for TO:, CC:, and BCC: when the message is actually send by SMTP Client Deux. Of course, BCCed addresses are not included within the headers of a message that is being sent, but this does not preclude the need for at least one value TO: recipient to be included within the headers of an SMTP message sent with SMTP Client Deux.

## Attachments

When attachments are sent with an SMTP message, the attachment is actually encoded and formatted in a particular way within the body of the message. The encoding of the attachment assures for the proper transmission of the document within the constraints of transport restrictions; universally, base64 encoding of an attachment is adequate to assure the proper deliver of any attachment type to any recipient program. The formatting of the encoded attachment will included information pertaining to the encoding type used, the original document title, and the type of content of the attachment.

SMTP Client Deux can send attachments with the methods ***SMTPcd\_Send\_Message\_Attachment*** and ***SMTPcd\_Send\_Message\_Complex***. But, for each method, the handling of attachments is different.

When using the method ***SMTPcd\_Send\_Message\_Attachment***, SMTP Client Deux will handle internally the encoding and formatting of the attachment when sending the message. All that is required is that the document contents, document title, and document content type be passed in as parameters. SMTP Client Deux will handle the details of properly encoding and formatted the attachment for inclusion with the message.

When using the method ***SMTPcd\_Send\_Message\_Complex***, it is the responsibility of the developer to properly encode and format the attachment. A utility method, ***SMTPcd\_Get\_Attachment\_w\_Header***, is included with SMTP Client Deux; this method will take as parameters the contents of the document, the document title, and the document content type and will format the data for passing to the method ***SMTPcd\_Send\_Message\_Complex***. If the developer wishes to provide more detailed or alternate encoding or formatting for attachments sent using ***SMTPcd\_Send\_Message\_Complex***, then it is the responsibility of the developer to properly encode and format the attachment for inclusion with an SMTP message.

## Content Types

When sending an SMTP message, the content type of the message body must be specified. Using the methods ***SMTPcd\_Send\_Message\_Simple*** and ***SMTPcd\_Send\_Message\_Attachment***, this is handled automatically by SMTP Client Deux. But, when using the method ***SMTPcd\_Send\_Message\_Complex***, it is up to the developer to set the content type of the message body.

Common values for content type include:

```
text/plain
text/html
image/gif
image/jpeg
application/pdf
application/octet-stream
application/x-macbinary
application/x-stuffit
multipart/mixed
```

Depending on the contents of the message being sent, different content types should be used. Most commonly, for an SMTP message, either text/plain or multipart/mixed are used; the former is for a standard text message and the latter is for a message that contains multiple parts (e.g. attachments).

Recently, text/html is becoming more and more popular, whereby an SMTP message consists solely of "HTML email" in the body of the message. It is recommended though that if the goal is to send HTML email, then the HTML should be sent as an attachment with the HTML message and the raw text of the message be included for those recipients that do not support the viewing of HTML email.

For SMTP messages which have a content type of multipart/mixed, the different "parts" of the message body must be separate by a boundary. This boundary is an indicator for the receiving software to properly delimit the different sections of the message body. The boundary is usually formatted within the value of the content type header as:

```
multipart/mixed; boundary="boundary_here"
```

An example header line within an SMTP message for a content type with an attachment might look like:

```
Content-Type: multipart/mixed; boundary="-----
e44A07500B354C3BB0e42090"
```



For attachments included with an SMTP message, a content type must be provided also in the headers of the attachment. This is used to indicate the content type of the encoding attachments which follows. Obviously, an attachment with a content type of multipart/mixed makes little sense at this point within the header of the attachment.

## SMTP Client Deux Constants

There are a few custom constants included with the SMTP Client Deux component package. These constants are grouped into convenient constant groups for easier referencing and organization.

Where appropriate, it is highly recommended that the custom constants included with the SMTP Client Deux component be utilized within your code; this will considerably simplify future feature enhancements to the core code within SMTP Client Deux.

## SMTPcd\_Header\_Codes

The SMTPcd\_Header\_Codes constants group contains one constant for each of the common SMTP header lines tags supported within the SMTP Client Deux component package. The following is a listing of the constants, their values, and the text of the actual header line in email corresponding to each, within the SMTPcd\_Header\_Codes constant group:

Constant	Value	Text of Header Line
SMTPcd_FROM_Type	1	From
SMTPcd_TO_Type	2	To
SMTPcd_CC_Type	3	CC
SMTPcd_BCC_Type	4	BCC
SMTPcd_Reply_To_Type	5	Reply-To
SMTPcd_Sender_Type	6	Sender
SMTPcd_Resent_Reply_To_Type	7	Resent-Reply-To
SMTPcd_Resent_From_Type	8	Resent-From
SMTPcd_Resent_Sender_Type	9	Resent-Sender
SMTPcd_Orig_Date_Type	10	Orig-Date
SMTPcd_Date_Type	11	Date
SMTPcd_Resent_Date_Type	12	Resent-Date
SMTPcd_Resent_To_Type	13	Resent-To
SMTPcd_Resent_CC_Type	14	Resent-CC
SMTPcd_Resent_BCC_Type	15	Resent-BCC
SMTPcd_In_Reply_To_Type	16	In-Reply-To
SMTPcd_References_Type	17	References
SMTPcd_Keywords_Type	18	Keywords
SMTPcd_Subject_Type	19	Subject
SMTPcd_Comments_Type	20	Comments
SMTPcd_encrypted_Type	21	encrypted
SMTPcd_Mime_Ver_Type	22	Mime-Version
SMTPcd_Content_Type_Type	23	Content-Type
SMTPcd_X_Mailer_Type	24	X-Mailer

In general, these constants will probably not be used within your coding with the SMTP Client Deux component. The only exception to this is if you use the SMTPcd method ***SMTPcd\_Send\_Message\_Complex*** in which case you will be using these values directly.

## SMTPcd\_Charsets

The SMTPcd\_Charsets constants group contains one constant for each of the supported characters set encoding types supported for sending email with automatically within SMTP Client Deux. The following is a listing of the constants and their values within the SMTPcd\_Charsets constants group:

Constant	Value
SMTPcd_Charset_Roman	1
SMTPcd_Charset_Japanese	2
SMTPcd_Charset_Chinese	3

All of the email sending routines within SMTP Client Deux support an optional parameter to specify the character set to use for sending email. SMTP Client Deux will send content as Roman if not specified otherwise in the optional parameter for each of the email sending routines.

If a call is made to an SMTP Client Deux email sending routine and the optional character set parameter is set to Japanese, SMTP Client Deux will automatically handle encoding of different sections of the email. Specifically, SMTP Client Deux will assume that values for the subject, the body, and all target email addresses are passed as Shift-JIS values. SMTP Client Deux will automatically convert these values to the standard JIS format for inclusion in the email to be sent. Formatting and encoding are handled internally within the SMTP Client Deux component for the character set within the email to be sent.

If a call is made to an SMTP Client Deux email sending routine and the optional character set parameter is set to Chinese, SMTP Client Deux will automatically handle encoding of different sections of the email. SMTP Client Deux will automatically convert the subject, body, and target email addresses values to the quoted printable format for inclusion in the email to be sent. Formatting and encoding are handled internally within the SMTP Client Deux component for the specified character set within the email to be sent.

**Note:** the SMTPcd\_Charsets constants group was added within SMTP Client Deux v1.2.0.

## Modules

All of the code within the SMTP Client Deux component is organized into modules. Each module is designated by a three (3) to five (5) character module prefix. All of the module prefixes are used within the name of every object within the module (methods names, variable names, semaphore names, etc.). This allows for the easy identification of any object within the SMTP Client Deux component.

each module contains a set of methods which can be used to easily implement a web server once the SMTP Client Deux component is installed. Method names all begin with the module prefix followed by an underscore ("\_") characters. The remainder of the method name then describes the function of the method.

## SMTPcd Module

---

### ENV\_Get\_SMTPcd\_HardName\_Long

*ENV\_Get\_SMTPcd\_HardName\_Long* => Long Hard Name

*ENV\_Get\_SMTPcd\_HardName\_Long*  
=> Long Hard Name : Text

	Parameter	Type	Description
	<i>Long Hard Name</i>	Text	Full, hard coded name of SMTP Client Deux component including versioning information

The method **ENV\_Get\_SMTPcd\_HardName\_Long** returns the full, hard coded name of the SMTP Client Deux component, including versioning information.

*Long Hard Name* is the full, hard coded name of the SMTP Client Deux component. As of this release, this will always return the value "Affix\_SMTPc\_Deux\_v1.2.0".

**Note:** this method was added as of SMTP Client Deux v1.1.0.

---

### ENV\_Get\_SMTPcd\_HardName\_Short

*ENV\_Get\_SMTPcd\_HardName\_Short* => Short Hard Name

*ENV\_Get\_SMTPcd\_HardName\_Short*  
=> Short Hard Name : Text

	Parameter	Type	Description
	<i>Short Hard Name</i>	Text	Short hard coded name of SMTP Client Deux component; does not include versioning information

The method **ENV\_Get\_SMTPcd\_HardName\_Short** returns a shortened, hard coded name of the SMTP Client Deux component.

*Short Hard Name* is the shortened, hard coded name of the SMTP Client Deux component. As of this release, this will always return the value "Affix\_SMTPc\_Deux".

**Note:** this method was added as of SMTP Client Deux v1.1.0.

---

## **INIT\_SMTPcd**

*INIT\_SMTPcd*

*INIT\_SMTPcd*

	Parameter	Type	Description
	<i>none</i>	n/a	n/a

The method ***INIT\_SMTPcd*** initialises the SMTP Client Deux component. A single call to this method should be made early in the ***On Startup*** and ***On Server Startup*** database methods in your 4D application. Make certain the call to this method follows the initialization call to the BASH and TCP Deux components but before any other calls to the SMTP Client Deux component package.

---

## **SMTPcd\_ERROR**

***SMTPcd\_ERROR*** ( *SMTPcd Error Number; Special Error Text; Calling Method Name* )

***SMTPcd\_ERROR***

```
(
    -> SMTPcd Error Number : Longint
    -> Special Error Text : Text
    -> Calling Method Name : Text
)
```

	Parameter	Type	Description
	<i>SMTPcd Error Number</i>	Longint	Internal SMTPcd error number



	Parameter	Type	Description
	<i>Special Error Text</i>	Text	Special text to describe the exact error instance
	<i>Calling Method Name</i>	Text	Name of the method that the error condition occurred in

The method **SMTPcd\_ERROR** acts as a callback method from within the SMTPcd module for errors that may occur. Any time an error condition is detected within the SMTPcd module, a call to the method **SMTPcd\_ERROR** is made.

The internal *SMTPcd Error Number* is passed to this method as the first parameter. The *Special Error Text* parameter will contain any relevant error text which is specific to the error which occurred. It is not uncommon for the *Special Error Text* value to be empty. The *Calling Method Name* will always contain the name of the SMTPcd method which call the **SMTPcd\_ERROR** method.

The **SMTPcd\_ERROR** method has been implemented as a source for a consistent interface and/or error tracking mechanism to be available while using the SMTP Client Deux component. This method can be modified to suit the needs of the database in which the SMTP Client Deux component has been installed.

---

## SMTPcd\_Extract\_EmailAddress

**SMTPcd\_Extract\_EmailAddress** ( *Formatted Recipient* ) => *Email Address*

```
SMTPcd_Extract_EmailAddress
(
    -> Formatted Recipient : Text
)
=> Email Address : Text
```

	Parameter	Type	Description
	<i>Formatted Recipient</i>	Text	Formatted recipient
	<i>Email Address</i>	Text	Full name of recipient

The method ***SMTPcd\_Extract\_EmailAddress*** will extract the email address within a properly formatted recipient. See the **Formatted Recipients** section of this manual for details on properly formatted recipients for use in the header of email being sent.

*Formatted Recipient* is the formatted recipient to extract a value from.

*Email Address* is email address extracted from *Formatted Recipient*.

---

## SMTPcd\_Extract\_FullName

*SMTPcd\_Extract\_FullName* ( *Formatted Recipient* ) => *Full Name*

```
SMTPcd_Extract_FullName
(
    -> Formatted Recipient : Text
)
=> Full Name : Text
```

	Parameter	Type	Description
	<i>Formatted Recipient</i>	Text	Formatted recipient
	<i>Full Name</i>	Text	Full name of recipient

The method ***SMTPcd\_Extract\_FullName*** will extract the full name within a properly formatted recipient. See the **Formatted Recipients** section of this manual for details on properly formatted recipients for use in the header of email being sent.

*Formatted Recipient* is the formatted recipient to extract a value from.

*Full Name* is full name extracted from *Formatted Recipient*.

---

## SMTPcd\_Get\_Attachment\_w\_Header

**SMTPcd\_Get\_Attachment\_w\_Header** ( *Referenced Attachment* ; *Attachment Filename* ; *Content Type* { ; *Character Set Code* } ) => *Error Code*

**SMTPcd\_Get\_Attachment\_w\_Header**

```
(
    -> Referenced Attachment : Pointer
    -> Attachment Filename : Text
    -> Content Type : Text
    { -> Character Set Code : Longint }
)
=> Error Code : Longint
```

	Parameter	Type	Description
	<i>Referenced Attachment</i>	Pointer	Referenced BLOB containing encoded document to include as document within formatted attachment
	<i>Attachment Filename</i>	Text	Full filename of attachment formatted for sending through email
	<i>Content Type</i>	Text	Content type of attachment being formatted for sending through email
	<i>Character Set Code</i>	Longint	Optional parameter to indicate the character set to use to encode attachment wrapper for sending thru email
	<i>Error Code</i>	Longint	error code returned from method for formatting attachment for sending through email

The method **SMTPcd\_Get\_Attachment\_w\_Header** will encode and format a document for properly sending through email. This routine should only be used in conjunction with the SMTP Client Deux method **SMTPcd\_Send\_Message\_Complex**.

*Referenced Attachment* is a pointer to a BLOB containing a document to include as an attachment within the formatted attachment being created for sending through email. The contents of the referenced BLOB will be Base64 encoded (a standard encoding scheme for email attachments) for sending through email.

The resulting, formatted attachment will be the value referenced by *Referenced Attachment* after this method is successfully called.

*Attachment Filename* is the full document name of the attachment which is being formatted for sending through email.

*Content Type* is the type of content of the attachment being formatted for sending through email. Common values for *Content type* include:

```
text/plain
text/html
image/gif
image/jpeg
application/pdf
application/octet-stream
application/x-macbinary
application/x-stuffit
```

See the **Content Types** section of this manual, above, for more information on content types.

*Character Set Code* is an optional parameter to indicate the character set to use for wrapping the attachment for inclusion as an attachment in email. Set the documentation for the constants group SMTPcd\_Charsets for details on the acceptable values for this parameter.

*Error Code* is the error code returned from calling this method for formatting an attachment for sending through email. The following tables lists all of the values returned from this method in *Error Code*:

Value	Description
0	no error, formatting done successfully
- 1	referenced attachment is empty
- 2	filename is empty
- 3	content type is empty

---

## SMTPcd\_Get\_Recipient\_Formatted

**SMTPcd\_Get\_Recipient\_Formatted** ( *Full Name ; Email Address* ) => *Formatted Recipient*

**SMTPcd\_Get\_Recipient\_Formatted**  
 (  
     -> *Full Name* : Text  
     -> *Email Address* : Text  
 )  
 => *Formatted Recipient* : Text

	Parameter	Type	Description
	<i>Full Name</i>	Text	Full name of recipient
	<i>Email Address</i>	Text	email address of recipient
	<i>Formatted Recipient</i>	Text	Formatted recipient

The method **SMTPcd\_Get\_Recipient\_Formatted** will properly format a recipient line for inclusion in an email being sent. See the **Formatted Recipients** section of this manual for details on properly formatted recipients for use in the header of email being sent.

*Full Name* is full name of the recipient being formatted. It is valid to have an empty value for this parameter.

*Email Address* is the email address of the recipient being formatted

*Formatted Recipient* is the formatted recipient for the parameters specified in calling this method.

---

## SMTPcd\_Send\_HTML\_x

**SMTPcd\_Send\_HTML\_x** ( *From Formatted Address ; To Formatted Address ; SMTP AUTH Enabled ; Username ; Password ; Subject ; Plain Text Body ; HTML Body ; SMTP Server Domain Name ; Reply-To Formatted Address { ; Character Set Code* } ) => *Error Code*

**SMTPcd\_Send\_HTML\_x**  
 (  
     -> *From Formatted Address* : Text  
     -> *To Formatted Address* : Text

```

-> SMTP AUTH Enabled : Longint
-> Username : Text
-> Password : Text
-> Subject : Text
-> Plain Text Body : Text
-> HTML Body : Text
-> SMTP Server Domain Name : Text
-> Reply-To Formatted Address : Text
{ -> Character Set Code : Longint }
)
=> Error Code : Longint

```

	Parameter	Type	Description
	<i>From Formatted Address</i>	Text	Formatted address to send email from
	<i>To Formatted Address</i>	Text	Formatted address to send email to
	<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
	<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
	<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
	<i>Subject</i>	Text	Subject of email
	<i>Plain Text Body</i>	Text	Plain text contents of email
	<i>HTML Body</i>	Text	HTML contents of email
	<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
	<i>Reply-To Formatted Address</i>	Text	Formatted address to include to email as a reply-to address
	<i>Character Set Code</i>	Longint	Optional parameter to indicate the character set to use to encode attachment wrapper for sending thru email
	<i>Error Code</i>	Longint	error code returned from method for sending email

The method ***SMTPcd\_Send\_HTML\_x*** will send a single email using the parameters provided. This method provides a simple and easy means to send HTML email using SMTP Client Deux, providing basic parameters and options for the email to be sent.

*From Formatted Address* is the formatted address to use to send the email from. Due to limitations in the SMTP protocol, the formatted address listed as the From: need not be an actual email address; but , it is strongly considered to be "bad form" to send email with a From: line that is invalid.

*To Formatted Address* is the formatted address to send the email to and to include in the To: line as a recipient of the email.

*SMTP AUTH Enabled* is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

*Username* is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Password* is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Subject* is the subject line of the email to be sent. There are no prohibitions against sending an email with an empty Subject: line, though it is considered to be "bad form" to do so.

*Plain Text Body* is the full plain text body of the email to be sent. There are no prohibitions against sending an HTML email with no plain text content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefore be used only for textual bodies of 32000 bytes or less.

*HTML Body* is the full HTML body of the email to be sent. There are no prohibitions against sending an HTML email with no HTML content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefore be used only for HTML bodies of 32000 bytes or less.

*SMTP Server Domain Name* is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified in a text format.

*Reply-To Formatted Address* is the formatted address to use in an optional Reply-To: line in the email to be sent. Pass in NULL for this parameter if no Reply-To: line is wanted. A Reply-To: line is optional in email and not all email clients support the Reply-To: line when replying to an email received.

*Character Set Code* is an optional parameter to indicate the character set to use for wrapping the attachment for inclusion as an attachment in email. Set the documentation for the constants group SMTPcd\_Charsets for details on the acceptable values for this parameter.

*Error Code* is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.

**Note:** this method was added in SMTP Client Deux v1.1.0.

## SMTPcd\_Send\_HTML\_z

*SMTPcd\_Send\_HTML\_z* ( *From Formatted Address* ; *Referenced To Formatted Addresses* ; *Referenced CC Formatted Addresses* ; *Referenced BCC Formatted Addresses* ; *SMTP AUTH Enabled* ; *Username* ; *Password* ; *Subject* ; *Plain Text Body* ; *HTML Body* ; *SMTP Server Domain Name* ; *Reply-To Formatted Address* { ; *Character Set Code* } ) => *Error Code*

### *SMTPcd\_Send\_HTML\_z*

(

- > *From Formatted Address* : Text
- > *Referenced To Formatted Addresses* : Pointer
- > *Referenced CC Formatted Addresses* : Pointer
- > *Referenced BCC Formatted Addresses* : Pointer
- > *SMTP AUTH Enabled* : Longint
- > *Username* : Text
- > *Password* : Text



```

-> Subject : Text
-> Plain Text Body : Text
-> HTML Body : Pointer
-> SMTP Server Domain Name : Text
-> Reply-To Formatted Address : Text
{ -> Character Set Code : Longint }
)
=> Error Code : Longint

```

	Parameter	Type	Description
	<i>From Formatted Address</i>	Text	Formatted address to send email from
	<i>Referenced To Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for To: line
	<i>Referenced CC Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for CC: line
	<i>Referenced BCC Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for blind carbon copy recipients of email
	<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
	<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
	<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
	<i>Subject</i>	Text	Subject of email
	<i>Plain Text Body</i>	Text	Plain Text contents of email
	<i>HTML Body</i>	Pointer	Referenced HTML contents of email
	<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
	<i>Reply-To Formatted Address</i>	Text	Formatted address to include to email as a reply-to address
	<i>Character Set Code</i>	Longint	Optional parameter to indicate the character set to use to encode attachment wrapper for sending thru email
	<i>Error Code</i>	Longint	error code returned from method for sending email

The method ***SMTPcd\_Send\_HTML\_z*** will send a single HTML email using the parameters provided. This method is similar

to the method ***SMTPcd\_Send\_HTML\_x***, but allows for larger HTML bodies to be sent, as well as the capability to specify multiple recipient addresses.

*From Formatted Address* is the formatted address to use to send the email from. Due to limitations in the SMTP protocol, the formatted address listed as the From: need not be an actual email address; however, it is strongly considered to be "bad form" to send email with a From: line that is invalid.

*Referenced To Formatted Addresses* is a pointer to an array of formatted addresses to use in the To: line of the email and to include as recipients of the email. Pass in NULL for this parameter if no To: addresses are wanted.

*Referenced CC Formatted Addresses* is a pointer to an array of formatted addresses to use in the CC: line of the email and to include as recipients of the email. Pass in NULL for this parameter if no CC: addresses are wanted.

*Referenced BCC Formatted Addresses* is a pointer to an array of formatted addresses to include as recipients of the email. BCCed formatted addresses will not be listed in the header of the email as being recipients of any kind. Pass in NULL for this parameter if no BCC: addresses are wanted.

*SMTP AUTH Enabled* is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

*Username* is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Password* is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Subject* is the subject line of the email to be sent. There are no prohibitions against sending an email with an empty Subject: line, though it is considered to be "bad form" to do so.

*Plain Text Body* is the full plain text body of the email to be sent. There are no prohibitions against sending an HTML email with no plain text content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefore be used only for textual bodies of 32000 bytes or less.

*HTML Body* is a reference to a BLOB containing the full HTML body of the email to be sent. There are no prohibitions against sending an HTML email with no HTML content, though it is considered to be "bad form" to do so. Since *HTML Body* is a BLOB, it is not subject to the common 4D restrictions for text variables.

*SMTP Server Domain Name* is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified as a text dotted IP address.

*Reply-To Formatted Address* is the formatted address to use in an optional Reply-To: line in the email to be sent. Pass in NULL for this parameter if no Reply-To: line is wanted. A Reply-To: line is optional in email and not all email clients support the Reply-To: line when replying to an email received.

*Character Set Code* is an optional parameter to indicate the character set to use for wrapping the attachment for inclusion as an attachment in email. Set the documentation for the constants group SMTPcd\_Charsets for details on the acceptable values for this parameter.

*Error Code* is the error code returned from calling this method for sending the email. If the sending of the email was successful, *error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.

---

## SMTPcd\_Send\_Message\_Attachment

**SMTPcd\_Send\_Message\_Attachment** ( *From Formatted Address ; Referenced To Formatted Addresses ; Referenced CC Formatted Addresses ; Referenced BCC Formatted Addresses ; SMTP AUTH Enabled ; Username ; Password ; Subject ; Body ; SMTP Server Domain Name ; Reply-To Formatted Address ; Referenced Unencoded Attachment ; Attachment Filename ; Attachment Content Type { ; Character Set Code }* ) =>Error Code

### **SMTPcd\_Send\_Message\_Attachment**

```
(
-> From Formatted Address : Text
-> Referenced To Formatted Addresses : Pointer
-> Referenced CC Formatted Addresses : Pointer
-> Referenced BCC Formatted Addresses : Pointer
-> SMTP AUTH Enabled : Longint
-> Username : Text
-> Password : Text
-> Subject : Text
-> Body : Text
-> SMTP Server Domain Name : Text
-> Reply-To Formatted Address : Text
-> Referenced Unencoded Attachment : Pointer
-> Attachment Filename : Text
-> Attachment Content Type : Text
{ -> Character Set Code : Longint }
)
=> Error Code : Longint
```

	<b>Parameter</b>	<b>Type</b>	<b>Description</b>
	<i>From Formatted Address</i>	Text	Formatted address to send email from
	<i>Referenced To Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for To: line
	<i>Referenced CC Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for CC: line
	<i>Referenced BCC Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for blind carbon copy recipients of email
	<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
	<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH

	Parameter	Type	Description
	<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
	<i>Subject</i>	Text	Subject of email
	<i>Body</i>	Text	Contents of email
	<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
	<i>Reply-To Formatted Address</i>	Text	Formatted address to include to email as a reply-to address
	<i>Referenced Unencoded Attachment</i>	Pointer	Referenced BLOB containing unencoded document to include as attachment with email being sent
	<i>Attachment Filename</i>	Text	Full filename of attachment being sent with email
	<i>Attachment Content Type</i>	Text	Content type of attachment being sent with email
	<i>Character Set Code</i>	Longint	Optional parameter to indicate the character set to use to encode attachment wrapper for sending thru email
	<i>Error Code</i>	Longint	error code returned from method for sending email

The method ***SMTPcd\_Send\_Message\_Attachment*** will send a single email using the parameters provided. This method is the simplest and easiest means to send email with an attachment using SMTP Client Deux, providing basic parameters and options for the email to be sent.

*From Formatted Address* is the formatted address to use to send the email from. Due to limitations in the SMTP protocol, the formatted address listed as the From: need not be an actual email address; however, it is strongly considered to be "bad form" to send email with a From: line that is invalid.

*Referenced To Formatted Addresses* is a pointer to an array of formatted addresses to use in the To: line of the email and to include as recipients of the email. Pass in NULL for this parameter if no To: addresses are wanted.

*Referenced CC Formatted Addresses* is a pointer to an array of formatted addresses to use in the CC: line of the email and to include as recipients of the email. Pass in NULL for this parameter if no CC: addresses are wanted.

*Referenced BCC Formatted Addresses* is a pointer to an array of formatted addresses to include as recipients of the email. BCCed formatted addresses will not be listed in the header of the email as being recipients of any kind. Pass in NULL for this parameter if no BCC: addresses are wanted.

*SMTP AUTH Enabled* is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

*Username* is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Password* is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Subject* is the subject line of the email to be sent. There are no prohibitions against sending an email with an empty Subject: line, though it is considered to be "bad form" to do so.

*Body* is the full body of the email to be sent. There are no prohibitions against sending an email with no content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefore be used only for textual bodies of 32000 bytes or less.

*SMTP Server Domain Name* is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified as a text dotted IP address.

*Reply-To Formatted Address* is the formatted address to use in an optional Reply-To: line in the email to be sent. Pass in NULL for this parameter if no Reply-To: line is wanted. A Reply-To: line is optional in email and not all email clients support the Reply-To: line when replying to an email received.

*Referenced Unencoded Attachment* is a pointer to a BLOB containing the document to include as an attachment with the email to be sent. The contents of the referenced BLOB will be Base64 encoded (a standard encoding scheme for email attachments) for sending and included in the email in a standard format for reception. If no attachment is to be sent with the email then this parameter can be set to NULL.

*Attachment Filename* is the full document name of the attachment which is being sent with the email. If no attachment is being sent then this parameter can be set to NULL.

*Attachment Content Type* is the type of content of the attachment being sent with the email. Common values for content type include:

- text/plain
- text/html
- image/gif
- image/jpeg
- application/pdf
- application/octet-stream
- application/x-macbinary
- application/x-stuffit

If no attachment is being sent with the email then this parameter can be set to NULL.

See the **Content Types** section of this manual for more information on content types.

*Character Set Code* is an optional parameter to indicate the character set to use for wrapping the attachment for inclusion as an attachment in email. Set the documentation for the constants group SMTPcd\_Charssets for details on the acceptable values for this parameter.

*Error Code* is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.

## SMTPcd\_Send\_Message\_Complex

**SMTPcd\_Send\_Message\_Complex** ( *Referenced Mail Header IDs* ; *Referenced Mail Header Values* ; *SMTP Server Domain Name* ; *SMTP AUTH Enabled* ; *Username* ; *Password* ; *Referenced Body* ; *Content Type* ; *Referenced Array of Referenced Attachments* { ; *Character Set Code* } ) => Error Code

### **SMTPcd\_Send\_Message\_Complex**

```
(
    -> Referenced Mail Header IDs : Pointer
    -> Referenced Mail Header Values : Pointer
    -> SMTP Server Domain Name : Text
    -> SMTP AUTH Enabled : Longint
    -> Username : Text
    -> Password : Text
    -> Referenced Body : Pointer
    -> Content Type : Text
    -> Referenced Array of Referenced Attachments : Pointer
    { -> Character Set Code : Longint }
)
=> Error Code : Longint
```

	Parameter	Type	Description
	<i>Referenced Mail Header IDs</i>	Pointer	Pointer to longint array containing mail header ID values corresponding to <i>Referenced Mail Header Values</i> , below
	<i>Referenced Mail Header Values</i>	Pointer	Pointer to text array containing mail header values corresponding to <i>Referenced Mail Header IDs</i> , above
	<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
	<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
	<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
	<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
	<i>Referenced Body</i>	Pointer	Pointer to BLOB containing body of email to be sent
	<i>Content Type</i>	Text	Content type of body of email being sent



	Parameter	Type	Description
	<i>Referenced Array of Referenced Attachments</i>	Pointer	Pointer to an array of pointers referenced formatted attachments to be sent with the email
	<i>Character Set Code</i>	Longint	Optional parameter to indicate the character set to use to encode attachment wrapper for sending thru email
	<i>Error Code</i>	Longint	error code returned from method for sending email

The method ***SMTPcd\_Send\_Message\_Complex*** will send an email with the parameter specified. This method is the most complex, and most flexible, means to send email using SMTP Client Deux, providing parameters and options for all areas of the email to be sent.

*Referenced Mail Header IDs* is a pointer to a longint array containing the email header IDs corresponding to different email header lines. The list of accepted email header IDs are listed in the SMTPcd\_Header\_Codes section of the manual. The array referenced by this parameter must have the same number of elements and correspond by index for values as that of the *Referenced Mail Header Values* parameter.

*Referenced Mail Header Values* is a pointer to a text array containing the email header values corresponding to different email header lines. The array referenced by this parameter must have the same number of elements and correspond by index for IDs as that of the *Referenced Mail ID Values* parameter.

All values in these arrays which exist for To:, CC:, and BCC: lines will be recipients of the email being sent. Of course, values in these arrays for BCC: recipients will not be listed in the actual headers of the email being sent. Values for each header line must be properly formatted. Specifically, the formatting of To:, CC:, BCC:, and Reply-To: values can be easily accomplished by using the ***SMTPcd\_Get\_Recipient\_Formatted*** SMTP Client Deux method. Make certain there is a separate entry (array element) for each formatted address that is to be a To:, CC:, or BCC: in the email message to be sent.

*SMTP Server Domain Name* is the full domain name of the SMTP server to use to send the email. This can be the IP

address of the SMTP server, if needed; merely make certain in this case that the IP address is specified in a text format.

*SMTP AUTH Enabled* is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

*Username* is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Password* is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Referenced Body* is a pointer to a BLOB containing the body of the email to be sent. The body of an email need not be just text, though it is commonly considered "bad form" to send an email with a body of any other content type. There are no restrictions at all, either for size or content type, for the body of the email to be sent when using this method. The contents of *Referenced Body* will be used directly as the content of the email to be sent.

*Content Type* is the type of content of the body of the email being sent. Common values for content type include:

- text/plain
- text/html
- image/gif
- image/jpeg
- application/pdf
- application/octet-stream
- application/x-macbinary
- application/x-stuffit
- multipart/mixed

If NULL is passed as the value for this parameter, the body content type will assume to be "text/plain" and the sent will reflect this assumption.

See the **Content Types** section of the manual for more information on content types.

*Referenced Array of Referenced Attachments* is a pointer to an array of pointers which reference BLOBs containing the attachments to send with the email. Each attachment in the doubly referenced BLOBs must be encoded and formatted already for sending. Use the SMTP Client Deux method ***SMTPcd\_Get\_Attachment\_w\_Header*** to properly format attachments for sending in an email. Passing NULL for this parameter will indicate that no attachments are to be sent with the email.

*Character Set Code* is an optional parameter to indicate the character set to use for wrapping the attachment for inclusion as an attachment in email. Set the documentation for the constants group SMTPcd\_Charsets for details on the acceptable values for this parameter.

*Error Code* is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.

## SMTPcd\_Send\_Message\_Simple

***SMTPcd\_Send\_Message\_Simple*** ( *From Formatted Address ; To Formatted Address ; SMTP AUTH Enabled ; Username ; Password ; Subject ; Body ; SMTP Server Domain Name ; Reply-To Formatted Address { ; Character Set Code }* ) => *Error Code*

### ***SMTPcd\_Send\_Message\_Simple***

```
(
    -> From Formatted Address : Text
    -> To Formatted Address : Text
    -> SMTP AUTH Enabled : Longint
    -> Username : Text
    -> Password : Text
    -> Subject : Text
    -> Body : Text
    -> SMTP Server Domain Name : Text
    -> Reply-To Formatted Address : Text
```

```

    { -> Character Set Code : Longint }
)
=> Error Code : Longint

```

	Parameter	Type	Description
	<i>From Formatted Address</i>	Text	Formatted address to send email from
	<i>To Formatted Address</i>	Text	Formatted address to send email to
	<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
	<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
	<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
	<i>Subject</i>	Text	Subject of email
	<i>Body</i>	Text	Contents of email
	<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
	<i>Reply-To Formatted Address</i>	Text	Formatted address to include to email as a reply-to address
	<i>Character Set Code</i>	Longint	Optional parameter to indicate the character set to use to encode attachment wrapper for sending thru email
	<i>Error Code</i>	Longint	error code returned from method for sending email

The method ***SMTPcd\_Send\_Message\_Simple*** will send a single email using the parameters provided. This method is the simplest and easiest means to send email using SMTP Client Deux, providing basic parameters and options for the email to be sent.

*From Formatted Address* is the formatted address to use to send the email from. Due to limitations in the SMTP protocol, the formatted address listed as the From: need not be an actual email address; but , it is strongly considered to be "bad form" to send email with a From: line that is invalid.

*To Formatted Address* is the formatted address to send the email to and to include in the To: line as a recipient of the email.

*SMTP AUTH Enabled* is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

*Username* is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Password* is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH enabled* parameter, then the value of this parameter is ignored.

*Subject* is the subject line of the email to be sent. There are no prohibitions against sending an email with an empty Subject: line, though it is considered to be "bad form" to do so.

*Body* is the full body of the email to be sent. There are no prohibitions against sending an email with no content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefor be used only for textual bodies of 32000 bytes or less.

*SMTP Server Domain Name* is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified in a text format.

*Reply-To Formatted Address* is the formatted address to use in an optional Reply-To: line in the email to be sent. Pass in NULL for this parameter if no Reply-To: line is wanted. A Reply-To: line is optional in email and not all email clients support the Reply-To: line when replying to an email received.

*Character Set Code* is an optional parameter to indicate the character set to use for wrapping the attachment for inclusion as an attachment in email. Set the documentation for the constants group SMTPcd\_Charsets for details on the acceptable values for this parameter.

*Error Code* is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.

## Version History

The following is a brief version history of the SMTP Client Deux component. It details release notes, bug fixes, and changes for each version publicly available.

## SMTP Client Deux v1.2.0b01

*released Tuesday, December 10th, 2002*

### Changes:

Changed name of component from SMTP Deux to SMTP Client Deux. Also changed all methods names, variable names, and constants names to reflect the SMTPcd indicator instead of the previous SMTPd indicator.

Added check for minimum installed version of BASH to be at least v1.8.1.

Corrected resource ID conflicts within platforms of SMTPcd Affix documents; made certain all affix documents coincide with the new plugin ID for SMTPcd (29606).

Made Affix document on Mac under 4D v67x a true plugin stub.

Corrected issue with bug in 4D v68x on Windows in which resource names can cause conflicts.

Corrected bug in ***ENV\_Get\_SMTPcd\_RF\_FullPath*** method in which it would scan the system 4DX folder even if the Affix document was found in the 4DX folder next to the structure.

Added SMTPcd\_Charsets constants group to Affix document.

Added optional charsets code parameter to methods:

***SMTPcd\_Get\_Attachment\_w\_Header***  
***SMTPcd\_Send\_HTML\_x***  
***SMTPcd\_Send\_HTML\_z***  
***SMTPcd\_Send\_Message\_Attachment***  
***SMTPcd\_Send\_Message\_Complex***  
***SMTPcd\_Send\_Message\_Simple***

Added support for character set encoding of body, subject, and addresses through send message handler hierarchy of routines; for Japanese, the content, subject and addresses are automatically converted from ShiftJIS to JIS and base64 encoded; for Chinese, the content, subject and addresses are automatically quoted printable encoded.



## SMTP Client Deux v1.1.0

*released 20020330*

### Changes:

SMTP Client Deux now requires BASH 1.6.3 or above.

Added a new constant to allow for the Content-transfer-encoding type header to be set.

Fixed a bug where an error could be generated when trying to set the content type.

Changed action when SMTP AUTH is enabled to default to using the LOGIN methodology when the server responds with an unknown or undetermined support for AUTH; it used to return an error in these cases.

Changed the name of the Windows affix document from "Afx\_SMTPcd" to "Affix\_SMTP\_Deux.4DX".

Added support for new 4D plugin architecture under 4D v6.8.x in which plugins can be located next to the current application in any environment.

Created a carbon affix document entitled "Affix\_SMTP\_Deux.4CX".

Added support for NTLM authentication; this method of authentication is preferred by Microsoft products.

Added routines for easily sending messages with HTML content.

New protected methods:

***SMTPcd\_Send\_HTML\_x***

***SMTPcd\_Send\_HTML\_z***

## SMTP Client Deux v1.0.0

*released 20010906*

### Changes:

Updated the component to require TCP Deux v1.0.1 or above.

Added the methods ***ENV\_Get\_SMTPcd\_HardName\_Long*** and ***ENV\_Get\_SMTPcd\_HardName\_Short***.

# SMTP Client Deux v1.0.0b01

*released 20010518*

## **Changes:**

First public beta release of the component.

## Errors

The listing of error codes and conditions is obviously a continuously updated process. With practically every new version of SMTP Client Deux, the error codes and conditions can and do change. Though it has been a long time coming, we are now documenting much of the error conditions that can occur in SMTP Client Deux.

Different methods in the SMTP Client Deux component can generate errors when used incorrectly or when used under the wrong circumstances. When an error condition is encountered, the methods within the BASH component will call the “SMTPcd\_ERROR” method. The “SMTPcd\_ERROR” method is documented above. The first parameter sent to the “SMTPcd\_ERROR” method is the error code identifying the unique error condition that occurred.

With future versions of the SMTP Client Deux component (and other components to come), the management and handling of error conditions will become much better documented, much clearer to understand, and easier to handle in your code. For now, though, reading this manual thoroughly is the best single source of understanding for the error conditions that can arise in using the SMTP Client Deux component.

## Error Codes

When the “SMTPcd\_ERROR” method is called in the SMTP Client Deux component, the first parameter is always an error code. This error code is always a 7 digit integer indicating the unique error condition which was detected in the code.

These 7 digit numbers actually consist of two pieces for uniquely identifying the error code and condition. The first five digits is an internal code for the module which an error occurred within. The last two digits identifies the unique error condition from within a module that has been detected.

The following is a quick listing of all of the error codes, grouped by the module which the error codes are from. Each error code includes the textual description of the error condition.

### **SMTPcd: 29031 series**

2903101	Timed out receiving data from SMTP server.
2903102	Error received from SMTP server.
2903103	TCP error occurred during communication with SMTP server.
2903104	Index into the Mail Header Types stack is out of range.
2903105	Unable to load Mail Headers Type stack from resource.
2903106	Failed to establish TCP connection to SMTP server.
2903107	Failed to establish reliable TCP connection to SMTP server.
2903108	Failed to receive welcome response from SMTP server.
2903109	Failed to send starting message boundary to SMTP server.
2903110	Failed to send the content-type to SMTP server.
2903111	Failed to send beginning boundary information to SMTP server.
2903112	Failed to send the message content to SMTP server.
2903113	Failed to send message attachment(s) properly.
2903114	Failed to send ending message boundary to SMTP server.
2903115	Failed to send message attachment boundary to SMTP server.
2903116	Failed to send message attachment to SMTP server.
2903117	Failed to send MAIL FROM command to SMTP server.
2903118	SMTP server failed to recognize MAIL FROM command.
2903119	Mail Header ID array is not a longint array.
2903120	Mail Header Values array is not a text array.
2903121	Message content is not a BLOB.
2903122	Mail Headers stack is not well formed.
2903123	SMTP server domain name not specified.
2903124	Addressee not formatted correctly.
2903125	No recipients specified to send email message to.
2903126	Failed to send HELO command to SMTP server.
2903127	Failed to receive response to HELO command from SMTP server.
2903128	Failed to send MAIL command to SMTP server.
2903129	Failed to send the RCPT TO command to SMTP server.
2903130	Failed to send the DATA command to SMTP server.
2903131	Failed to receive response to DATA command from SMTP server.
2903132	Failed to send the message header to SMTP server.
2903133	Failed to send the end of message identifier to SMTP server.
2903134	Failed to receive confirmation of the end of message identifier from SMTP server.
2903135	Failed to send the QUIT command to SMTP server.
2903136	Failed to receive response to QUIT command from SMTP server.

2903137 Failed to receive response to RCPT TO command from SMTP server.  
2903138 Attachment to encode is empty.  
2903139 Filename of attachment not specified.  
2903140 Content type of attachment not specified.  
2903141 Email address is empty.  
2903142 Failed to send EHLO command to SMTP server.  
2903143 Failed to receive response to eHLO command from SMTP server.  
2903144 Failure to get authorized to SMTP server.  
2903145 SMTP server does not support authorization techniques (CRAM-MD5, PLAIN, or LOGIN).  
2903146 Failure to send SMTP password.  
2903147 Failure to send SMTP user name.  
2903148 Failure to receive reply from AUTH LOGIN command.  
2903149 Failure to send the AUTH LOGIN command.  
2903150 Failure to send the AUTH PLAIN command.  
2903151 Failure to send the CRAM-MD5 authorization.  
2903152 Failure to receive reply from AUTH CRAM-MD5 command.  
2903153 Failure to send the AUTH CRAM-MD5 command.  
2903154 The version of TCPd needs to be 1.0.1 or greater.  
2903155 The hard name of TCPd is incorrect.  
2903156 Failure to send the AUTH NTLM command.  
2903157 Failure to receive reply from AUTH NTLM command.  
2903158 Failure to send NTLM challenge.  
2903159 Failure to receive NTLM server response.  
2903160 Failure to send the AUTH NTLM authorization.  
2903161 The version of BASH needs to be 1.8.1 or greater.

## Method Listing

The following is a listing of all of the methods within the SMTP Client Deux component (COMPILER methods are not included in this listing), including all private methods which are not directly available in the API when the SMTP Client Deux component is installed. Following each method is a list of the error codes which each method can generate.

```

ENV_Get_SMTPcd_HardName_Long
ENV_Get_SMTPcd_HardName_Short
ENV_Get_SMTPcd_RF_FullPath
INIT_SMTPcd
RES_Open_SMTPcd
SMTPcd_AUTH_to_Server
    2903126
    2903127
    2903142
    2903143
    2903144
    2903145
    2903146
    2903147
    2903148
    2903149
    2903150
    2903151
    2903152
    2903153
    2903156
    2903157
    2903158
    2903159
    2903160
SMTPcd_Check_ResponseCode
SMTPcd_Confirm_Reply_f_Server
SMTPcd_Encode_ISO2022_x
SMTPcd_Encode_ISO2022_z
SMTPcd_ERROR
SMTPcd_Extract_EmailAddress
SMTPcd_Extract_FullName
SMTPcd_Generate_DES_Key_x
SMTPcd_Get_Attachment_w_Header
    2903138
    2903139
    2903140
SMTPcd_Get_AuthorizationType
SMTPcd_Get_Charset_Name
SMTPcd_Get_LANManager_Hash
SMTPcd_Get_MH_Type_by_Index
    2903104
SMTPcd_Get_NTLM_Auth_Response
SMTPcd_Get_NTLM_Challenge
SMTPcd_Get_NTLM_ResponseKey
SMTPcd_Get_Recipient_Formatted
    2903141
SMTPcd_Get_Reply_from_Server
    2903101

```

2903102  
2903103  
SMTPcd\_INIT  
2903105  
2903154  
2903155  
2903161  
SMTPcd\_INIT\_P  
SMTPcd\_MH\_Add\_Recipients  
SMTPcd\_MH\_Add\_to\_Stack  
SMTPcd\_MH\_Clear\_Stack  
SMTPcd\_Open\_Connection  
2903106  
2903107  
2903108  
SMTPcd\_Send\_Content  
2903109  
2903110  
2903111  
2903112  
2903113  
2903114  
2903115  
2903116  
SMTPcd\_Send\_Header  
SMTPcd\_Send\_HTML\_x  
SMTPcd\_Send\_HTML\_z  
SMTPcd\_Send\_MAIL\_Command  
2903117  
2903118  
SMTPcd\_Send\_Message\_Attachment  
SMTPcd\_Send\_Message\_Complex  
2903119  
2903120  
2903121  
2903122  
2903123  
2903124  
2903125  
SMTPcd\_Send\_Message\_Handler  
2903112  
2903128  
2903129  
2903130  
2903131  
2903132  
2903133  
2903134  
2903135  
2903136  
2903144  
SMTPcd\_Send\_Message\_Simple  
SMTPcd\_Send\_RCPT\_Command  
2903124  
2903125  
2903129  
2903137