Reference Guide

EmailObject

32/64 BIT

Multiplatform

MELISSA DATA®

Email Object

Reference Guide

Copyright

Trademarks

MELISSA DATA CORPORATION
22382 Avenida Empresa
Rancho Santa Margarita, CA 92688-2112

Phone: 1-800-MELISSA (1-800-635-4772)
Fax: 949-589-5211

E-mail: info@MelissaData.com
Web site: www.MelissaData.com

For the latest version of this Reference Guide, visit **http://www.MelissaData.com/tech/emailobject.htm**.

Document Code: EMORFG
Revision Number: 111006.244
Last Update: October 6, 2011

Dear Developer,

I would like to take this opportunity to introduce you to Melissa Data Corp. Founded in 1985, Melissa Data provides data quality solutions, with emphasis on address and phone verification, postal encoding, and data enhancements.

We are a leading provider of cost-effective solutions for achieving the highest level of data quality for lifetime value. A powerful line of software, databases, components, and services afford our customers the flexibility to cleanse and update contact information using almost any language, platform, and media for point-of-entry or batch processing.

This manual will guide you through the functions of our easy-to-use programming tools. Your feedback is important to me, so please don't hesitate to email your comments or suggestions to ray@MelissaData.com.

I look forward to hearing from you.

Best Wishes,

[illegible]

Raymond F. Melissa
President

Table of Contents

Email Object

Email Object will allow your Web sites and custom applications to update email addresses in your database files while verifying and correcting misspelled domain names.

You can use Email Object to:

- Verify and optionally correct syntax errors in an address, checking for illegal characters and extra “@” characters.
- Verify and optionally correct top-level domain names.
- Check for, and optionally correct, misspelled domain names.
- Check for, and optionally correct, domains that have changed.
- Standardize casing in email addresses.

Entering Your Email Object License

The license string is a software key that unlocks the functionality of the component. Without this key, the object does not function. You set the license string using an environment variable called MD_LICENSE. If you are just trying out Email Object and have a demo license, you can use the environment variable MD_LICENSE_DEMO for this purpose. This avoids conflicts or confusion if you already have active subscriptions to other Melissa Data object products.

In earlier versions of Email Object, you would set this value with a call to the **SetLicenseString** function. Using an environment variable makes it much easier to update the license string without having to edit and re-compile the application.

It used to be necessary, even when employing an environment variable, to call the **SetLicenseString** function without passing the license string value. This is longer true. Email Object will still recognize the **SetLicenseString** function, but you should eventually remove any reference to it from your code.

Windows

Windows users can set environment variables by doing the following:

1. Select **Start > Settings**, and then click **Control Panel**.
2. Double-click **System**, and then click the **Advanced** tab.
3. Click **Environment Variables**, and then select either *System Variables* or *Variables for the user X*.
4. Click **New**.
5. Enter “MD_LICENSE” in the *Variable Name* box.
6. Enter the license string in the *Variable Value* box, and then click **OK**.

Please remember that these settings take effect only upon start of the program. It may be necessary to quit and restart the application to incorporate the changes.

Linux/Solaris/HP-UX/AIX

Unix-based OS users can simply set the license string via the following (use the actual license string, instead):

```
export MD_LICENSE=A1B2C3D4E5
```

If this setting is placed in the .profile, remember to restart the shell.

Email Object also used to employ its own environment variable, MDEMAIL_LICENSE. The MD_LICENSE variable is shared across the entire Melissa Data product line of programming tools. Email Object will still use the old license variable for the time being, but you should transition to using MD_LICENSE as soon as possible.

Using Email Object

1. Create an instance of Email Object.

```
SET eMailPtr As New Instance of EmailCheck
```

2. Set the data path, then initialize the instance.

```
CALL SetPathToEmailFiles WITH EmailDataPath
CALL InitializeDataFiles RETURNING Results
IF Results Is 0 THEN
   CALL GetBuildNumber RETURING BuildNumber
   CALL GetDatabaseDate RETURNING DatabaseDate
   CALL GetDatabaseExpirationDate RETURNING DBExpireDate
   CALL GetLicenseStringExpirationDate RETURNING LicExpire
ELSE
   CALL GetInitializeErrorString RETURNING ErrorString
   PRINT ErrorString
ENDIF
```

3. Set the five Email Object options that control the degree to which it will correct and update the submitted email address. This sample enables all of the options.

```
CALL SetStandardizeCasing WITH TRUE
CALL SetCorrectSyntax WITH TRUE
CALL SetUpdateDomain WITH TRUE
CALL SetDatabaseLookup WITH TRUE
CALL SetMXLookup WITH TRUE
```

4. Call the **VerifyEmail** function. The sample email address will introduce a few errors so the results will demonstrate the sort of updating that Email Object is capable of.

```
EmailAddress Is "John@Doe@800Nail.con"
CALL VerifyEmail WITH EmailAddress
CALL GetResults RETURNING ResultsCode
```

5. After calling the **VerifyEmail** function, check the results code to make sure that there was no problem with the verification processes. Otherwise, read the object functions to get the verified, corrected, and standardized e-mail address.

```
IF ResultsCode CONTAINS "ES01" THEN
   CALL GetMailBoxName RETURNING Mailbox
   CALL GetDomainName RETURNING Domain
   CALL GetTopLevelDomain RETURNING TLD
   CALL GetEmailAddress RETURNING EmailAddress
```

```
ELSE
   PROCESS ResultsCode FOR ERROR INFORMATION
ENDIF
```

The original email address, **John@Doe@800Nail.con**, should be returned as **johndoe@800mail.com**. In short, the extra “@” was removed by syntax checker. “800nail” has been replaced by the correct domain name, “800mail.” The incorrect top-level domain, “.con,” is now “.com.” Finally, the email address has been converted to all lowercase.

The **GetResults** function would return the result codes ES10, ES11, and ES12 for those changes, respectively.

Email Object Functions

Initialize Email Object

These functions initialize Email Object and connect it to its data files.

Set Email Object Options

These functions enable or disable the various options of Email Object.

Verify the Email Address

The VerifyEmail function passes the submitted email address through Email Object's verification routines.

Retrieve the Status Information

The following functions return general information about the success or failure of the last call to the VerifyEmail function.

Retrieve the Email Address Data

The following functions return the detailed information on the last email address passed to the VerifyEmail function.

Initialize Email Object

These functions initialize Email Object and connect it to its data files.

SetPathToEmailFiles

This function passes a string value containing the file path to the data files used by Email Object.

Remarks

This function must be called prior to calling the **InitializeDataFiles** function. The parameter must contain a valid path to the directory that contains the files `mdDomain.dat, mdEmail.ini`, and `mdEmailConfig.ini` files.

Syntax

```
object->SetPathToEmailFiles(DataPath);
```

C

```
mdEmailSetPathToEmailFiles(object, char*);
```

COM

```
object.PathToEmailFiles = StringValue
```

SetLicenseString

This function sets the license string required to enable Email Object's complete functionality.

Remarks

The license string is included with the documentation you received. If you have not purchased a license, call Melissa Data toll free at 1-800-MELISSA (1-800-635-4772) or send an email to sales@MelissaData.com.

The license string is normally set using an environment variable, either MD_LICENSE or MD_LICENSE_DEMO. Calling SetLicenseString is an alternative method for setting the license string, but applications developed for a production environment should only use the environment variable.

When using an environment variable, it is not necessary to call the **SetLicenseString** function.

For more information on setting the environment variable, see page 2 of this guide.

If a valid license string is not set, Email Object will not operate.

Input Parameters

The **SetLicenseString** function has one parameter:

LicenseString	A string value representing the software license key.

Return Value

The **SetLicenseString** function returns a Boolean value of 0 (FALSE) or 1 (TRUE). It will return a FALSE Boolean value if the license string provided is incorrect.

Syntax

```
BooleanValue = object->SetLicenseString(LicenseString);
```

C

```
IntegerValue = mdEmailSetLicenseString(object,LicenseString);
```

COM

```
BooleanValue = object.SetLicenseString(LicenseString)
```

InitializeDataFiles

The **InitializeDataFiles** function opens the required data files and prepares Email Object for use.

Remarks

Before calling this function, the **SetPathToEmailFiles** function must have been called.

Check the return value of the **GetInitializeErrorString** function to retrieve the result of the initialization call. Any result other than “No Error” means the initialization failed for some reason.

Return Value

This function returns a value of the enumerated date type ProgramStatus. If the initialization is successful, this value will be zero. If any other value is returned, check the **GetInitializeErrorString** function to see the reason for the error.

Syntax

```
ProgramStatus = object->InitializeDataFiles();
```

C

```
ProgramStatus = mdEmailInitializeDataFiles(object);
```

COM

```
ProgramStatus = object.InitializeDataFiles
```

GetInitializeErrorString

This function returns a descriptive string to describe the error from the **InitializeDataFiles** function.

Remarks

The **GetInitializeErrorString** function returns a string describing the error caused when the **InitializeDataFiles** function fails.

Syntax

```
StringValue = object->GetInitializeErrorString();
```

C

```
StringValue = mdEmailGetInitializeErrorString(object);
```

COM

```
StringValue = object.GetInitializeErrorString
```

GetBuildNumber

The **GetBuildNumber** function returns the current development release build number of Email Object.

Remarks

Input Parameters

None.

Return Value

The **GetBuildNumber** function returns the current development release build number of the Email Object.

Syntax

```
StringValue = object->GetBuildNumber();
```

C

```
StringValue = mdEmailGetBuildNumber(object);
```

COM

```
StringValue =object.GetBuildNumber
```

GetDatabaseDate

The **GetDatabaseDate** function returns a date value that represents the date of your Email data files.

Remarks

Input Parameters

None.

Return Value

The **GetDatabaseDate** function returns a value that represents the date of your Email data files. The COM object returns a date value, while the standard object returns a string value.

Syntax

```
StringValue = object->GetDatabaseDate();
```

C

```
StringValue = mdEmailGetDatabaseDate(object);
```

COM

```
DateTime = object.GetDatabaseDate
```

GetDatabaseExpirationDate

This function returns a date value containing the expiration date for the current mdDomain.dat file.

Remarks

If this date has passed, Email Object will not initialize. You will need to update your copy to continue using it. As long as your subscription is still current, you should receive an undated copy of Email Object before your current database expires.

Return Value

This function returns a value containing the expiration date of the current mdDomain.dat file. The COM object returns a date value, while the standard object returns a string value.

Syntax

```
StringValue = object->GetDatabaseExpirationDate();
```

C

```
StringValue = mdEmailGetDatabaseExpirationDate(object);
```

COM

```
DateTime = object.GetDatabaseExpirationDate
```

GetLicenseStringExpirationDate

This function returns a date value corresponding to the date when the current license string expires.

Remarks

License strings issued by Melissa Data are valid a certain period of time. This function returns the date after which the current license string is no longer valid.

The COM object returns a date value, while the standard object returns a string value.

Syntax

```
StringValue = object->GetLicenseStringExpirationDate();
```

C

```
StringValue = mdEmailGetLicenseStringExpirationDate(object);
```

COM

```
DateTime = object.GetLicenseStringExpirationDate
```

Set Email Object Options

These functions enable or disable the various options of Email Object.

SetCorrectSyntax

This function enables or disables the syntax correction functionality of Email Object.

Remarks

If the Boolean value passed to this function is TRUE, Email Object will attempt to correct the syntax of the email address passed to the **VerifyEmail** function. This process includes:

- Removal of any illegal characters from the address. This would include excess "@" characters.
- Correction of misspelled domain names. For example, "yaho.com" would be replaced by "yahoo.com."
- Correction of misspelled top-level domain names. For example, ".con" would be replaced with ".com."

The CorrectSyntax feature is enabled by default.

Syntax

```
object->SetCorrectSyntax(bool);
```

C

```
mdEmailSetCorrectSyntax(object, bool);
```

COM

```
object.CorrectSyntax = Boolean
```

SetDatabaseLookup

This function enables or disables the database lookup validation function of Email Object.

Remarks

If the Boolean value passed to this function is TRUE, Email Object will attempt to validate the email address passed to the **VerifyEmail** function using its own database of known valid and invalid domain names. Domain names that appear neither on the known valid nor invalid domain lists will return a status of “U” for “unverified.”

The Database Lookup feature is enabled by default.

Syntax

```
object->SetDatabaseLookup(bool);
```

C

```
mdEmailSetDatabaseLookup(object, bool);
```

COM

```
object.DatabaseLookup = Boolean
```

SetMXLookup

This function enables or disables the DNS lookup validation function of Email Object.

Remarks

If the Boolean value passed to this function is TRUE, Email Object will attempt to validate the email address passed to the **VerifyEmail** function by attempting to locate an MX (Mail Exchange) record or an A (Address Name) record for the domain on a DNS server. This is slower than a database lookup but potentially more accurate if the domain name is either obscure or new.

The results of this check will set the return value of the **GetResults** function. Domain names that cannot be verified by an MX Lookup will return a code for “bad email address.”

The MX Lookup feature is disabled by default, so it must be explicitly enabled for this feature to be used.

Syntax

```
object->SetMXLookup(bool);
```

C

```
mdEmailSetMXLookup(object, bool);
```

COM

```
object.MXLookup = Boolean
```

SetStandardizeCasing

This function enables or disables the standardize casing function of Email Object.

Remarks

If the Boolean value passed to this function is TRUE, Email Object will set the email address passed to the **VerifyEmail** function to all lowercase letters. For example, “JSmith@MelissaData.com” would become “jsmith@melissadata.com.”

The Standardize Casing feature is enabled by default.

Syntax

```
object->SetStandardizeCasing(bool);
```

C

```
mdEmailSetStandardizeCasing(object, bool);
```

COM

```
object.StandardizeCasing = Boolean
```

SetUpdateDomain

This function enables or disables the domain update functionality of Email Object.

Remarks

If the Boolean value passed to this function is set to TRUE, Email Object will attempt to update the domain name of the email address. One domain name can replace another in cases such as a change in corporate ownership. For example, the domain of subscribers to the @Home cable internet service was switched from “home.com” to “cox.net.”

The Update Domain feature is enabled by default.

Syntax

```
object->SetUpdateDomain(bool);
```

C

```
mdEmailSetUpdateDomain(object, bool);
```

COM

```
object.UpdateDomain = Boolean
```

Verify the Email Address

The **VerifyEmail** function passes the submitted email address through Email Object's verification routines.

VerifyEmail

Launches Email Object's verification and correction routines on the submitted email address.

Remarks

The **VerifyEmail** function performs some or all of the following processes on the submitted email addresses, based on the settings passed to the **SetCorrectSyntax**, **SetUpdateDomain**, **SetDatabaseLookup**, **SetMXLookup**, and **SetStandardizeCasing** functions.

Correct Syntax

Removes illegal characters and corrects misspelled domain names and top-level domain names. This function can be enabled or disabled using the **SetCorrectSyntax** function.

Update Domain Name

If one domain name has been superceded by another for any reason, VerifyEmail will replace the old name with the new name, if the value passed to the **SetUpdateDomain** function is TRUE.

Lookup Domain Name

Email object can attempt to validate the domain name of an email address in two ways. One is to check the name against Email Object's own database of known valid and invalid domain names, assuming the **SetDatabaseLookup** function is set to TRUE.

The other is to perform an MX lookup via a DNS server, assuming the value passed to the **SetMxLookup** function is TRUE.

Check the value returned by the **GetResults** function to see the results of these attempts to verify the validity of the domain name.

Standardize Casing

Email Object can also switch all alphabetic characters in the email address to lower case, assuming that the value passed to the **SetStandardizeCasing** function is TRUE.

Result Codes

Check the output of the **GetResults** function (page 19) after calling VerifyEmail. Result codes indicate the validity of the submitted email address, the reason for any errors, and any changes made to correct the submitted address.

Configuration Files

Email Object’s syntax correction, domain updating, and domain verification routines can be updated via the mdEmailConfig.ini file. This is a user-modifiable text file that can be used to override the information found in Email Object’s own database, if necessary.

See *Modifying Settings for Email Object* on page 28 for more information on modifying this file.

Input Parameters

This function has one input parameter, a string containing the complete email address that you wish to verify and, optionally, correct and update.

Return Value

This function returns TRUE if the email address, including any corrections, is valid. Returns FALSE if not.

Syntax

```
BooleanValue = object->VerifyEmail(StringValue);
```

C

```
IntegerValue = mdEmailVerifyEmail(object, char*);
```

COM

```
BooleanValue = object.VerifyEmail(StringValue)
```

Retrieve the Status Information

The following functions return general information about the success or failure of the last call to the **VerifyEmail** function.

GetResults

This function returns a comma-delimited string of four-character codes which detail the level of matching found for the current email address and any errors that occurred during the last call to the **VerifyEmail** function.

Remarks

The **GetResults** function is intended to replace the **GetStatusCode** and **GetErrorCode** functions, providing a single source of information about the last **VerifyEmail** call and eliminating the need to call multiple functions to determine if a particular email address was verified.

The function returns one or more of the following codes in a comma-delimited list.

Code	Status/Error	Description
ES01	Valid Email Domain	The domain name of the submitted email address was confirmed as valid by either the DatabaseLookup or MXLookup
ES02	Invalid Email Domain	The domain name of the submitted email address was either not located by MXLookup or was located on the list of invalid domains.
ES03	Unverified Email Domain	The domain name of the submitted email address was not confirmed as valid by either DatabaseLookup or MXLookup, but was not found on the list of invalid domain names.
ES04	Mobile Email Address	The domain name of the submitted email was identified as a mobile email address, classified as not deliverable by FCC regulations.
ES10	Syntax Was Changed	The syntax of the submitted email address was changed.
ES11	Top Level Domain Changed.	The top level domain of the submitted email address was changed.
ES12	Domain Changed (Spelling)	The domain of the submitted email address was corrected for spelling.
ES13	Domain Changed (Update)	The domain of the submitted email address was updated due to a domain name change.
EE01	Syntax Error	There is a syntax error in the submitted email address.

Code	Status/Error	Description
EE02	Top Level Domain Not Found	The top level domain of the submitted email address was not found.
EE03	Mail Server Not Found	The mail server (domain) of the submitted email address was not found.
EE04	Invalid Mailbox Name	An invalid mailbox name was detected (IE: noreply). To configure invalid mailbox names, review mdEmailConfig.ini.
EE05	Email Object Not Initialized	Email Object was not initialized properly. Please review your code and data files.

Syntax

```
StringValue = object->GetResults();
```

C

```
StringValue = mdEmailGetResults(object);
```

COM

```
StringValue = object.Results
```

GetStatusCode (Deprecated)

This function returns the status code after a call to the **VerifyEmail** function.

This function has been deprecated. You should use the **GetResults** function instead. See page 19 for documentation on this function.

Remarks

The **GetStatusCode** function returns a one-character string value set by a call to the **VerifyEmail** function.

Possible return values from the **GetStatusCode** function are:

Code	Definition
V	Verified email address. The domain name of the submitted email address was confirmed as valid by either the DatabaseLookup or MXLookup.
U	Unverified email address. The domain name of the submitted email address was not confirmed as valid by either DatabaseLookup, but was not found on the list of invalid domain names.
X	Bad email address. The domain name of the submitted email address was either not located by MXLookup or was located on the list of invalid domains.

Code	Definition
empty	Verification not performed. This usually results when an error has prevented a successful call to the **VerifyEmail** function.

An “X” value would be returned if neither the **SetDatabaseLookup** nor the **SetMXLookup** functions are set to TRUE and the submitted email address contains syntax errors that were not corrected.

A “V” value would be returned if neither the **SetDatabaseLookup** nor the **SetMXLookup** functions are set to TRUE and the submitted email address contains no syntax errors.

Syntax

```
StringValue = object->GetStatusCode();
```

C

```
StringValue = mdEmailGetStatusCode(object);
```

COM

```
StringValue = object.StatusCode
```

GetErrorCode (Deprecated)

This function returns the error code after an unsuccessful call to the **VerifyEmail** function.

This function has been deprecated. You should use the **GetResults** function instead. See page 19 for documentation on this function.

Remarks

The **GetErrorCode** function returns a one-character string value set by an unsuccessful call to the **VerifyEmail** function.

Possible return values from the **GetErrorCode** function are:

Code	Definition
Empty	No error
I	Email Object is not initialized
S	Syntax error
T	Top Level domain not found
M	Mail server not found

Code	Definition
B	Invalid Mailbox

The "S" code (syntax error) will be returned if:

- There is a syntax error in the submitted email address and the Correct Syntax feature is disabled.
- There is an error that Email Object cannot correct, such as a missing mail box name, domain or top-level domain or no "@" character.

If the **VerifyEmail** function has not been called, or resulted in no error, this function will be empty.

Syntax

```
StringValue = object->GetErrorCode();
```

C

```
StringValue = mdEmailGetErrorCode(object);
```

COM

```
StringValue = object.ErrorCode
```

GetErrorString (Deprecated)

This function returns a description of any errors after an unsuccessful call to the **VerifyEmail** function.

This function has been deprecated. You should use the **GetResults** function instead. See page 19 for documentation on this function.

Remarks

The **GetErrorString** function is a string value set by an unsuccessful call to the **VerifyEmail** function.

Possible return values from the **ErrorString** function are:

Code	Definition
Empty	No error
I	"Email Object is not initialized"
S	"Syntax error"
T	"Top Level Domain not found"

Code	Definition
M	“Mail server not found”
B	“Invalid Mailbox”

If the **VerifyEmail** function has not been called, or resulted in no error, this function will be empty.

Syntax

```
StringValue = object->GetErrorString();
```

C

```
StringValue =mdEmailGetErrorString(object);
```

COM

```
StringValue = object.ErrorString
```

GetChangeCode (Deprecated)

Deprecated. This function returns a single integer value which indicates what changes, if any, have been performed on the submitted email address.

This function has been deprecated. You should use the **GetResults** function instead. See page 19 for documentation on this function.

Remarks

The **GetChangeCode** function returns a single integer value to indicate if the submitted email address has been changed and what changes have been made.

To use this function’s value, use an `AND` operator to compare the value to the table below.

Integer	Indicates	Bit
0	No change	
1	Syntax changed	1
2	Top-level domain changed	2
4	Domain changed (spelling correction)	3
8	Domain changed (updated domain)	4

For example, if the expression `object.ChangeCode AND 8` evaluates to `TRUE`, that means the domain name has changed due to the domain itself being updated. See *Using Email Object* on page 3 for a more detailed example.

Syntax

```
IntegerValue = object->GetChangeCode();
```

C

```
IntegerValue = mdEmailGetChangeCode(object);
```

COM

```
IntegerValue = object.ChangeCode
```

Retrieve the Email Address Data

The following functions return the detailed information on the last email address passed to the **VerifyEmail** function.

GetMailBoxName

This function returns the mailbox or user name portion of the email address passed to the **VerifyEmail** function, including any changes or corrections that have been made by Email Object.

Remarks

This function returns a string value containing the mailbox name or user name portion of the corrected email address (all characters that precede the “@” character). If the final address is “jsmith@melissadata.com,” this function just returns “jsmith.”

Syntax

C++

```
StringValue = object->GetMailBoxName();
```

C

```
StringValue = mdEmailGetMailBoxName(object);
```

COM

```
StringValue = object.MailBoxName
```

GetDomainName

This function returns the domain name portion of the email address passed to the **VerifyEmail** function, excluding the top-level domain, including any changes or corrections that have been made by Email Object. To get the full domain name, combine the results of this function with a “.” character and the return value of the **GetTopLevelDomain** function.

Remarks

This function returns a string value containing the domain name portion of the corrected email address. For example, it returns all characters that come after the “@”

character, not including the top-level domain, such as ".com." If the final address is "jsmith@melissadata.com," this function just returns "melissadata."

Syntax

```
StringValue = object->GetDomainName();
```

C

```
StringValue = mdEmailGetDomainName(object);
```

COM

```
StringValue = object.DomainName
```

GetTopLevelDomain

This function returns the top-level domain name portion of the email address passed to the **VerifyEmail** function, including any changes or corrections that have been made by Email Object.

Remarks

This function returns a string value containing the top-level domain name portion of the corrected email address, such as "com." If the final address is "jsmith@melissadata.com," this function just returns "com."

Syntax

```
StringValue = object->TopLevelDomain();
```

C

```
StringValue = mdEmailGetTopLevelDomain(object);
```

COM

```
StringValue = object.TopLevelDomain
```

GetTopLevelDomainDescription

This function returns the long-form description of the top-level domain name portion of the email address passed to the **VerifyEmail** function. This description is found and can be modified in the .ini files. See *Modifying Settings for Email Object* on page 28.

Remarks

This function returns a string value containing the long form description of the top-level domain name portion of the corrected email address.

Syntax

```
StringValue = object->TopLevelDomainDescription();
```

C

```
StringValue = mdEmailGetTopLevelDomainDescription(object);
```

COM

```
StringValue = object.TopLevelDomainDescription
```

GetEmailAddress

This function returns the email address passed to the **VerifyEmail** function, including any changes or corrections that have been made by Email Object.

Remarks

This function returns a string value containing the corrected version of the email address passed to **VerifyEmail** function.

If the **VerifyEmail** function returns a FALSE result, indicating an error, this function will contain the original text string passed to that function.

Syntax

```
StringValue = object->GetEmailAddress();
```

C

```
StringValue = mdEmailGetEmailAddress(object);
```

COM

```
StringValue = object.EmailAddress
```

Modifying Settings for Email Object

Email Object uses a database of valid domain names and top-level domains, plus corrections for common misspellings of popular domain names, updated every two months.

You can add to or override this information by modifying a file called **mdEmailConfig.ini**. Changes to this file override any settings found in the default database and are not overwritten when Email Object is updated.

The file contains several sections, one for each type of change that Email Object can perform.

To add any item to the list, add a line beginning with a "+" followed by the new item. To remove an item from the database, use the "-" character.

These changes do not make any permanent changes to the default database.

Comment lines can be added. Just begin the line with the "#" character.

Top Level Domains

This section begins with the line containing "[TLD]."

Optionally, you can add a space, followed by a description of the TLD. For example:

```
+.new A New Top Level Domain
```

To remove a top-level domain:

```
-.old
```

Valid Domains

This section begins with the line containing "[valid domains]." These domains will be assumed to be good and no MX Lookup will be performed if that option is enabled.

Adding a valid domain:

```
+validdomain.com
```

Removing a valid domain:

```
-validdomain.com
```

Invalid Domains

This section begins with the line containing "[invalid domains]." These domains will be assumed to be invalid and all email addresses with these domains will be considered invalid.

Adding an invalid domain:

```
+invaliddomain.com
```

Removing an invalid domain:

```
-invaliddomain.com
```

Domain Updates

This section begins with the line containing “[updates].” All old domains found in this section will be updated to the new, correct domain name.

Adding a domain update:

```
+olddomain.com->newdomain.com
```

Removing a domain update:

```
-olddomain.com->newdomain.com
```

Top Level Domain Misspellings

This section begins with the line containing “[TLD misspellings].” All TLDs matching the misspelled forms will be replaced with the corrected spelling.

Adding a TLD misspelling:

```
+bad->good
```

Removing a TLD misspelling:

```
-bad->good
```

Domain Misspellings

This section begins with the line containing “[domain misspellings].” All domains matching the misspelled forms will be replaced with the corrected spelling.

Adding a domain misspelling:

```
+800nails.com->800mail.com
```

Removing a domain misspelling:

```
-800nails.com->800mail.com
```

Invalid Mailboxes

This section begins with the line containing “[invalid mailboxes].” These mailboxes will return an error of “invalid mailbox”.

Adding an invalid mailbox:

```
+NoReply
```

Removing an invalid mailbox:

```
-NoReply
```