Reference Guide

EmailObject

32/64 BIT

Multiplatform

MELISSA DATA®

Email Object

Reference Guide

Melissa Data Corporation

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
Internet: www.MelissaData.com

For the most recent version of this document, visit
http://www.melissadata.com/

Document Code: DQTAPIEORG
Revision Number: 23102013.17

Dear Developer,

I would like to take this opportunity to thank you for your interest in Melissa Data products and introduce you to the company.

Melissa Data has been a leading provider of data quality and address management solutions since 1985. Our data quality software, Cloud services, and data integration components verify, standardize, consolidate, enhance and update U.S., Canadian, and global contact data, including addresses, phone numbers, and email addresses, for improved communications and ROI. More than 5,000 companies rely on Melissa Data to gain and maintain a single, accurate and trusted view of critical information assets.

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 me at: Ray@MelissaData.com.

I look forward to hearing from you.

Best Wishes,

[illegible]

Raymond F. Melissa

President/CEO

Contents

Introduction

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 no longer true. Email Object will still recognize the SetLicenseString function, but you should eventually remove any reference to it from your code.

Windows

1. Windows users can set environment variables by doing the following:
2. Select Start > Settings, and then click Control Panel.

3. Double-click System, and then click the Advanced tab.
4. Click Environment Variables, and then select either System Variables or Variables for the user X.
5. Click New.
6. Enter "MD_LICENSE" in the Variable Name box.
7. 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. Neither legacy license strings nor environment variable MDEMAIL_LICENSE can be used to initialize the object. The new license structure is now required.

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 RETURNING 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 SetFuzzyLookup WITH TRUE
CALL SetWSLookup 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@@yajoo. 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@@yajoo. con’, should be returned as john@yahoo.com. In short, the extra “@” was removed by syntax checker. “yajoo” has been replaced by the correct domain name, “yahoo.” 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.

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.

SetPathToEmailFiles

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

It must be called prior to calling the InitializeDataFiles function. The parameter must contain a valid path to the directory that contains the files mdEmail.dat and mdEmail.cfg.

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.

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 the environment variable MD_LICENSE. 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 1.

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.

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.

This 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.

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.

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

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 date value containing the expiration date for the current mdEmail.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.

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.

SetCachePath

This function sets the path for the directory where Email Object will write temporary cache files. If this is not set and SetCacheUse is turned on, Email Object will place the cache files in the local folder (Unix) or the default temporary folder (Windows).

You may also set the path using CachePath in mdEmail.cfg. Please see page 21. This will override what is set by SetCachePath.

Syntax	object->SetCachePath(string);
C	mdEmailSetCachePath(object, string);

SetCacheUse

This function enables or disables the use of temporary cache files within Email Object. These cache files are used to store domains that have already been checked to speed up processing time. It is highly recommended that this feature it turned on.

There are also two other ways to turn off or on cache files:

You may also turn off the cache files by using the environment variable SUPPRESS_MDEMAIL_CACHE and setting its value to 1. This will override whatever is set by SetCacheUse and the CachePath setting.

You can set the SuppressCache setting in mdEmail.cfg. Please see page 21. This will override whatever is set by SetCacheUse.

The default value for this method is 1.

If the integer value passed to this function is 1, Email Object will use the cache files.

If the integer value passed to this function is 0, Email Object will not use the cache files.

Syntax	object->SetCacheUse(int);
C	mdEmailSetCacheUse(object, int);

SetCorrectSyntax

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

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 top-level domain names. For example, ".con" would be replaced with ".com."
- Mailbox syntax will be corrected to the RFC 5322 specification strict definitions. It allows for usage of special characters in some situations. For example: "very.unusual.@.unusual.com"@example.com is allowed because the additional '@' is surrounded by quotes.

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.

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 result code of "ES03" for "unverified."

The Database Lookup feature is enabled by default.

Syntax	object->SetDatabaseLookup(bool);
C	mdEmailSetDatabaseLookup(object, bool);
COM	object.DatabaseLookup = Boolean

SetFuzzyLookup

This function enables or disables Melissa Data's fuzzy lookup validation function of Email Object.

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 apply fuzzy matching algorithms to the input domain. This is slower than a database lookup but potentially more accurate if the domain name contains a common or transposed typo.

The results of this check will set the return value of the GetResults function. Domain names that cannot be verified by a Fuzzy Lookup will return a code for "Unverified email address."

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

Syntax	object->SetFuzzyLookup(bool);
C	mdEmailSetFuzzyLookup(object, bool);
COM	object.FuzzyLookup = Boolean

SetWSLookup

This function enables or disables the Melissa Data server lookup validation function of Email Object.

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 the domain from a compiled and continuously updated list of valid domains. This is slower than a database lookup but potentially more accurate if the domain name is either obscure, new, or no longer valid.

The results of this check will set the return value of the GetResults function. Domain names that cannot be verified by an WS Lookup will return a code for "Unverified email address."

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

Syntax	object->SetWSLookup(bool);
C	mdEmailSetWSLookup(object, bool);

COM	object.WSLookup = Boolean

SetMXLookup

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

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.

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.

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.

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, SetFuzzyLookup, SetWSLookup, 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 superseded 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 many ways. Assuming the SetDatabaseLookup function is set to TRUE, it will check the name against Email Object's own database of known valid and invalid domain names. The object will also attempt to apply a fuzzy algorithm to validate the domain, a Melissa Data web server of compiled domains, and an MX lookup via a DNS Server, if the following options are also set to TRUE - SetFuzzyLookup, SetWSLookup, and SetMXLookup respectively.

  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 overridden via the mdEmail.cfg file. This is a plain text file, which allows you to add, remove, or override some of Email Object's default mdEmail.dat tables.
- See Modifying Settings for Email Object on page 19 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.

This 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	Short Description	Long Description
ES01	Valid Email Domain	The email domain name was confirmed as valid by either the DatabaseLookup or MXLookup.
ES02	Invalid Email Domain	The email domain name was either not located by MXLookup or was located on the list of invalid domains.
ES03	Unverified Email Domain	The domain name was not confirmed as valid by either DatabaseLookup, but was not found on the list of invalid domain names.
ES04	Mobile Email Address	The domain name was identified as a mobile email address and classified as not deliverabled by the FCC.
ES05	Disposable Domain	The domain name of the submitted email was identified as a disposable domain. For example: DODGEIT.COM
ES06	Spamtrap Domain	The domain name of the submitted email was identified as a spamtrap domain. Mailing to one of these domains could result in the sender being blacklisted. For example: LAUNCH.COM.
ES10	Syntax 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.
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

GetResultCodeDescription

This function returns the description of the inputted Result Code. It can only be used through the Standard DLL.

It requires two values to be passed in, a Result Code and an enumerated option. If a string of Result Codes are inputted, only the first code will be used. The enumerated option will determine whether a short or long description will be returned.

ResultCdDescOp

Enumerated Value	Integer Value	Description
ResultCodeDescriptionLong	0	Returns a detailed description of the inputted result code.
ResultCodeDescriptionShort	1	Returns a brief description of the inputted result code.

Syntax	StringValue = object->GetResultCodeDescription(StringValue_ResultCode, ResultCdDescOpt);
C	StringValue = mdEmailGetResultCodeDescription(object, StringValue_ResultCode, int);

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 12 for documentation on this function.

This 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.
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 12 for documentation on this function.

This 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
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 "GetResults" on page 12 for documentation on this function.

This 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"
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 12 for documentation on this function.

This 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

Integer	Indicates	Bit
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 2 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.

This 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	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.

This 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.

This 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 in the mdEmail.dat data file and can be modified in the [Tld] section of the mdEmail.cfg file. See Modifying Settings for Email Object on page 19.

This 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.

This 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 mdEmail.cfg. 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, enter the new domain in all uppercase into the appropriate section (See the following descriptions for details about each section). 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 or ";" character.

Domain Status

Each domain can be assigned a status by a comma delimited append of the desired letter.

Letter	Status
V	Valid
C	Valid Mobile Domain
M	Invalid Domain (no MX Servers)
I	Invalid Domain (no DNS Entry)
H	Invalid Domain (Spamtrap)
D	Invalid Domain (Disposable)

Domain

This section begins with the line containing "[Domain]." This is a list of all domains.

Action	Syntax	Example
Add a domain:	`<DOMAIN>,<STATUS>`	`ADDEDDOMAIN.COM,C`
Remove a domain:	`-<DOMAIN>`	`-KNOWNSPAMMER.COM`

Domain Fuzzy

This section begins with the line containing "[DomainFuzzy]." These will be fuzzy searched if a match is not found in [Domain]. Entries added to this list should also be added to [Domain]. It is recommended that you only add V or C domains to this list, as you usually don't want to fuzzy match to invalid domain names.

Action	Syntax	Example
Add a domain:	`<DOMAIN>,<STATUS>`	`NEWDOMAIN.NET,V`

Domain Updates

This section begins with the line containing "[DomainUpdates]." These are also known as domain substitutions. This list can be used to update an older domain name to a newer one. No assumption is made regarding either the old or new domain name's status. If you add a new <update> domain here, you should also add it to [Domain] with an appropriate status.

Action	Syntax	Example
Add a domain:	`<DOMAIN>,<UPDATE>`	`OLDDOMAIN.COM,NEWDOMAIN.COM`

Spam Trap Mailboxes

This section begins with the line containing "[SpamTrapMailboxes]." These are mailboxes that generally are routed to a system administrator (ie, abuse, noreply).

Action	Syntax	Example
Add a domain:	`<MAILBOX>`	`STOPSPAMMINGUS`

Top Level Domains

This section begins with the line containing "[Tld]." These are top level domains and their descriptions.

Action	Syntax	Example
Add a domain:	`<TLD>,<Description>`	`SOS,International Distress Signal`

Top Level Domain Misspellings

This section begins with the line containing "[TldMisspelled]." These are common top level domain misspellings and their corrections.

Action	Syntax	Example
Add a domain:	<TLD>, <correction>	GOOB,gov

General Domains

This section begins with the line containing "[General]." These are general settings, and can be any of the following:

Syntax	Description
WebServiceURL=<url>	URL used for web service communications
WebServiceTimeout=<timeout>	Timeout used for web service communications (default is 20 sec)
WebServiceRetries=<retries>	Retry count used for web service communications (default is 3)
WSRetryInterval=<interval>	Waiting period between failed attempts (default is 30000 ticks, 30 sec)
WSCacheRefresh=<interval>	How often a cached result should be refreshed (default is 86400 sec, 24 hrs)
MXRetryInterval=<interval>	Waiting period between failed DNS attempts (default is 180000 ticks, 180 sec)
MXCacheRefresh=<interval>	How often a cached DNS result should be refreshed (default is 86400 sec, 24 hrs)
CachePath=<path>	The path to write the cache files. Please see page 7 for more details on cache files. This will override the value set by SetCachePath.
SuppressCache=<integer>	Integer value to turn on (0) or off (1) cache files. Please see page 7 for more details on cache files. This will override the value set by SetCacheUse but will not override the value set by the SUPPRESS_MDEMAIL_CACHE environment variable.