Reference Guide

PhoneObject

32/64 BIT

Multiplatform

MELISSA DATA®

Phone Object

Reference Guide

Melissa Data Corporation

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
Internet: www.MelissaData.com

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

Document Code: DQTAPIPORG
Revision Number: 03102013.09

Dear Developer,

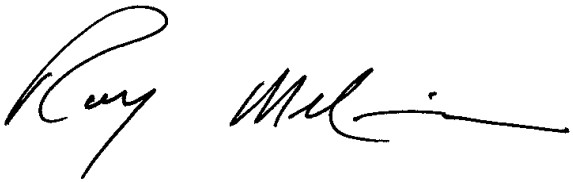
I would like to take this opportunity to thank you for your interest in Melissa Data products and introduce you to the company.

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,



Raymond F. Melissa

President/CEO

Contents

Introduction

Phone Object

Phone Object allows Web sites and custom applications to verify phone numbers down to 7 and 10 digits, update area codes, and append data about the phone number.

Use Phone Object to:

- Verify U.S. or Canadian phone numbers down to 7 or 10 digits.
- Update the area code if it changed in the last year.
- Append data on the telephone line, distinguishing between landlines, wireless numbers, or Voice Over IP (VOIP).
- Append data on the telephone owner, distinguishing between residential, business, or home office numbers.
- Parse the phone number into its components.
- Correct wrong or missing area codes (ZIP Code required).
- Check the distance from the phone number to the ZIP Code.

Global Phone Object

Global Phone Object can be used to verify, correct, and append data about a phone number from over 230 countries and territories.

Use the Global Phone Object to:

- Verify and append country dialing codes, international exit codes, national prefix, and more.
- Append data on the telephone line, distinguishing between landlines, wireless numbers, or Voice Over IP (VOIP).
- Append geographical information on the telephone line such as latitude, longitude, administrative area, and language.
- Parse the phone number into its components

Entering Your 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 Phone 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 Phone 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. The Phone and Global Phone 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.

Phone Object also used to employ its own environment variable, mdPhone_LICENSE. The MD_LICENSE variable is shared across the entire Melissa Data product line of programming tools. Phone 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 Phone Object

Verifying a Phone Number

1. Create an instance of Phone Object.
```
Set phonPtr as New Instance of PhoneCheck
```

2. Initialize the data files.
```
CALL Initialize WITH DataPath RETURNING Result
IF Result <> 0 Then
  CALL GetInitializeErrorString RETURNING ErrorString
   PRINT "Error: " & ErrorString
ENDIF
```

3. Pass a string containing a phone number and, optionally, a string containing a ZIP Code to the Lookup function. If the call is successful, use some or all of the functions shown below to retrieve the results.
```
CALL Lookup WITH Phone, ZIP
CALL GetResults RETURNING ResultCodes
Process ResultCodes
CALL GetAreaCode RETURNING AreaCode
CALL GetPrefix RETURNING Prefix
CALL GetSuffix RETURNING Suffix
CALL GetExtension RETURNING Extension
CALL GetCity RETURNING City
CALL GetState RETURNING State
CALL GetCountyName RETURNING CountyName
CALL GetCountyFips RETURNING CountyFips
CALL GetCountryCode RETURNING CountryCode
CALL GetMsa Returning MSACode
CALL GetPmsa RETURNING PMSACode
CALL GetLatitude RETURNING Latitude
CALL GetLongitude RETURNING Longitude
CALL GetDistance RETURNING Distance
CALL GetTimeZone RETURNING TimeZone
CALL GetTimeZoneCode RETURNING TimeZoneCode
```

Updating an Area Code after a Split

To update an area code after an area code split, use the CorrectAreaCode function.

```
CALL CorrectAreaCode WITH Phone,Zip RETURNING Result
IF Result is TRUE THEN
  CALL GetNewAreaCode RETURNING NewAreaCode
  CALL GetAreaCode RETURNING AreaCode
ELSE
  CALL GetResults RETURNING ResultCode
  Print ResultCode
ENDIF
```

Using Global Phone Object

Verifying an International Phone Number

1. Create an instance of the Global Phone Object.

   ```
   Set globalPhonePtr as New Instance of GlobalPhone
   ```

2. Initialize the data files

   ```
   Call Initialize WITH DataPath RETURNING Result
   IF Result <> 0 then
     CALL GetInitializeErrorString RETURNING ErrorString
       PRINT "Error: " & ErrorString
       ENDIF
   ```

3. Pass a string containing a phone number, and optionally, a string containing a Country name and/or a Country of Origin to the Lookup function. If the call is successful, use some or all of the functions shown below to retrieve the results.

   ```
   CALL Lookup WITH Phone, PhoneCountry, CountryOfOrigin
   CALL GetResults RETURNING ResultCodes
   ```

   Process ResultCodes

   ```
   CALL GetLocality RETURNING Locality
   CALL GetCountry RETURNING Country
   CALL GetCountryCode RETURNING CountryCode
   CALL GetDST RETURNING DaylightSavingsTime (Y/N)
   CALL GetLanguage RETURNING Language
   ```

4. If the input phone matches to more than one locality or administrative area, call the LookupNext function. If the function returns true, the next possible result will populate the return methods.

```
WHILE LookupNext RETURNS TRUE
CALL GetResults RETURNING ResultCodes
```

Process ResultCodes

```
CALL GetLocality RETURNING Locality
CALL GetCountry RETURNING Country
CALL GetCountryCode RETURNING CountryCode
CALL GetDST RETURNING DaylightSavingsTime (Y/N)
CALL GetLanguage RETURNING Language
```

For a full list of all functions returning data, begin with “Retrieve Status Information” on page 27

Phone Object Functions

Initialize Phone Object

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

SetLicenseString

The License String is a software key (supplied by the developer) that unlocks the full functionality of Phone Object.

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 the license string has not been set, Phone Object will operate in a demonstration mode (limited to Nevada Area Codes) and will return the string "DEMO" after the GetBuildNumber function.

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). The SetLicenseString function will return a FALSE Boolean value if the license string provided is incorrect.

Syntax	BooleanValue = object->SetLicenseString(LicenseString);
C	IntegerValue = mdPhoneSetLicenseString(object,LicenseString);
COM	BooleanValue = object.SetLicenseString(LicenseString)

Initialize

The Initialize function opens the needed data file and prepares the Phone Object for use.

If the function returns any value other than 0, call the GetInitializeErrorString function to retrieve the cause of the failure.

Input Parameters

The Initialize function has one parameter:

- **DataPath** - A String that contains the path to the location of the mdPhone.dat, mdPhone.idx, mdAddr.dat and NPA.TXT file.

Return Value

The Initialize function returns an integer value of 0 if successful.

Syntax	IntegerValue = object->Initialize(StringValue_DataPath);
C	IntegerValue = mdPhoneInitialize(object, StringValue_DataPath);
COM	IntegerValue = object.Initialize(StringValue_DataPath)

GetInitializeErrorString

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

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

Syntax	StringValue = object->GetInitializeErrorString();
C	StringValue = mdPhoneGetInitializeErrorString(object);
COM	StringValue = object.GetInitializeErrorString

GetBuildNumber

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

The word “DEMO” will be reported after the build number if no license string is provided, or if an incorrect license string is entered.

Input Parameters

None.

Return Value

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

Syntax	StringValue = object->GetBuildNumber();
C	StringValue = mdPhoneGetBuildNumber(object);
COM	StringValue = object.GetBuildNumber

GetDatabaseDate

The GetDatabaseDate function returns a date value that represents the date of the phone data files.

If the GetDatabaseDate function is called before the Initialize function is called and the data files are not in the same directory as the PHONEOBJ.DLL, this function will return a date outside the normal range of the system date, such as 1969 or 1899, depending on the system.

Input Parameters

None.

Return Value

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

Syntax	StringValue = object->GetDatabaseDate();
C	StringValue = mdPhoneGetDatabaseDate(object);
COM	DateTime = object.GetDatabaseDate

GetLicenseExpirationDate

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

License strings issued by Melissa Data are valid for 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->GetLicenseExpirationDate();
C	StringValue = mdPhoneGetLicenseExpirationDate(object);
COM	DateTime = object.GetLicenseExpirationDate

Look Up Phone Numbers

These two functions look up a phone number, checking either for geographic data or an area code that was updated due to an area code split.

Lookup

The Lookup function verifies the submitted phone number down to the first 7 or 10 digits and returns geographic information for the submitted phone number.

On a FALSE return, the state and country code fields might still be populated if the area code is valid. Use the GetResults function to determine if the problem is a prefix error. If this is the case, you can use the GetState and GetCountryCode functions.

The return values of following functions are set by the Lookup function:

GetResults	GetAreaCode	GetPrefix
GetSuffix	GetExtension	GetCity
GetState	GetCountyName	GetCountyFips
GetCountryCode	GetDistance	GetLatitude
GetLongitude	GetMsa	GetPmsa
GetTimeZone	GetTimeZoneCode	

Input Parameters

- **Phone** - A 24-character (maximum) string containing the phone number.
  - Phone numbers from outside the U.S. and Canada will be not be validated.
  - Toll free numbers will be parsed, but not verified.
  - Phone Object will recognize a leading country code from U.S. and Canadian phone numbers (i.e. a “1”) and strip that digit before parsing the phone number.

- Any extensions (any consecutive digits after first ten) will be parsed out and returned by the GetExtension function.

- **ZIPCode** - A five-character (optional) ZIP Code. If the ZIP Code is provided, the GetDistance function will contain the distance between the area code/prefix and the ZIP Code.
  - If your programming language does not support optional parameters, you should pass an empty string ("") to the function.
  - The ZIPCode parameter only accepts U.S. ZIP codes and not Canadian Postal Codes.
  - Because of phone number portability, the number returned by the GetDistance function may not be accurate.

Return Value

The Lookup function returns a value of (1) TRUE if the area code/prefix combination is valid and a value of (0) FALSE if it is invalid.

After calling the Lookup function, check the return values of the GetResults function. This will indicate the level of match, the type of phone number verified, and the cause of any errors.

Syntax	BooleanValue = object->Lookup(Phone[, ZIPCode]);
C	IntegerValue = mdPhoneLookup(object, Phone, ZIPCode);
COM	BooleanValue = object.Lookup(Phone[, ZIPCode])

CorrectAreaCode

The CorrectAreaCode function will return the correct area code for a phone number and ZIP Code combination to the NewAreaCode. The CorrectAreaCode function will also populate the values returned by GetAreaCode and GetResults functions.

On a FALSE return, the CorrectAreaCode function will set the GetResults function to indicate the cause of failure.

Input Parameters

The CorrectAreaCode function has these parameters:

- **PhoneNum** - A 24 character (maximum) string containing the phone number.
- **ZIPCode** - A five-character VARIANT containing the ZIP Code.

Return Value

The CorrectAreaCode function returns a Boolean value of TRUE if the function call was successful, FALSE if there was an error.

Syntax	BooleanValue = object->CorrectAreaCode(PhoneNum, ZIPCode);
C	IntegerValue = mdPhoneCorrectAreaCode(object, PhoneNum, ZIPCode);
COM	BooleanValue = object.CorrectAreaCode(PhoneNum, ZIPCode)

Retrieve Status Information

These function retrieve information about the most recent calls to the Lookup and CorrectAreaCode functions. The GetErrorCode and GetStatusCode functions have been deprecated in favor of the GetResults function.

GetResults

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

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

The function returns one or more Results Codes in a comma-delimited list. For a list of these Results Codes, see Results Codes on page 34.

Syntax	StringValue = object->GetResults();
C	StringValue = mdPhoneGetResults(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 = mdPhoneGetResultCodeDescription(object, StringValue_ResultCode, int);

Retrieve Phone Number Data

These functions retrieve information about the phone number passed via the last call to the Lookup function.

GetAreaCode

This function returns the area code of the phone number string that was passed to the CorrectAreaCode or Lookup function.

The GetAreaCode function returns a three-character string value set by a call to the CorrectAreaCode or Lookup function.

If there are not enough digits for the area code, this function will be empty. If the area code/prefix combination has been split, the new area code will be in the GetNewAreaCode function. If a ZIP Code was also passed in and corrections were made to the area code, these corrections will appear in the GetNewAreaCode function. If the CorrectAreaCode or Lookup function has not been called, or the call resulted in an error, this function will be blank.

Syntax	StringValue = object->GetAreaCode();
C	StringValue = mdPhoneGetAreaCode(object);
COM	StringValue = object.AreaCode

GetNewAreaCode

This function returns the new area code if the phone number passed to the CorrectAreaCode function has undergone an area code split.

The GetNewAreaCode function returns a three-character string value after a call to the CorrectAreaCode function.

Example: “714”

The GetNewAreaCode function will return a corrected or updated area code based on the phone number string passed into the CorrectAreaCode function.

An area code is corrected when it appears to be invalid for the ZIP Code passed in. Phone Object will change the area code based on the distance between valid area code/prefix combinations and the ZIP Code.

An updated area code is a new area code that is based on the input of an area code/prefix combination that has been split. If no new area code was found, then the GetNewAreaCode function will return the current area code.

If the CorrectAreaCode function has not been called, or resulted in an error, this function will return a null value.

Syntax	StringValue = object->GetNewAreaCode();
C	StringValue = mdPhoneGetNewAreaCode(object);
COM	StringValue = object.NewAreaCode

GetPrefix

This function returns the prefix (the first three digits after the area code) of a phone number passed to the CorrectAreaCode function or the Lookup function.

The GetPrefix function returns a three-character string value after a call to the CorrectAreaCode function or the Lookup function.

If the CorrectAreaCode function or Lookup function has not been called, or resulted in an error, this function will be blank.

Syntax	StringValue = object->GetPrefix();
C	StringValue = mdPhoneGetPrefix(object);
COM	StringValue = object.Prefix

GetSuffix

This function returns the suffix (last four digits) of the phone number passed to the Lookup function.

The GetSuffix function returns a four-character string value after a call to the Lookup function.

If the Lookup function has not been called, or resulted in an error, the return value of the GetSuffix function will be a blank string.

Syntax	StringValue = object->GetSuffix();
C	StringValue = mdPhoneGetSuffix(object);
COM	StringValue = object.Suffix

GetExtension

This function returns the extension (if any) of the phone number passed to the CorrectAreaCode function or the Lookup function.

The GetExtension function returns a maximum 10-character string value set after a call to the CorrectAreaCode function or the Lookup function.

If the CorrectAreaCode function or the Lookup function has not been called, or resulted in an error, this function will be blank.

Syntax	StringValue = object->GetExtension();
C	StringValue = mdPhoneGetExtension(object);
COM	StringValue = object.Extension

Retrieve GeoCode Data

The following functions retrieve geographic data about the phone number submitted to the Lookup function.

GetCity

This function returns the city name associated with the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetCity();
C	StringValue = mdPhoneGetCity(object);
COM	StringValue = object.City

GetState

This function returns the state abbreviation associated with the area code passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetState();
C	StringValue = mdPhoneGetState(object);
COM	StringValue = object.State

GetCountyName

This function returns the name of the county associated with the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetCountyName();
C	StringValue = mdPhoneGetCountyName(object);
COM	StringValue = object.CountyName

GetCountyFips

This function returns the five-digit county FIPS code associated with the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

The GetCountyFips function returns a five-digit string value after a successful call to the Lookup function.

The Federal Information Processing Standard (FIPS) is a five-digit code defined by the U.S. Census Bureau. The first two digits are the state code and the last three indicate the county within the state.

For example: “06037” is the County FIPS for Los Angeles, CA (“06” is the state code for California and “037” is the county code for Los Angeles).

Syntax	StringValue = object->GetCountyFips();
C	StringValue = mdPhoneGetCountyFips(object);
COM	StringValue = object.CountyFips

GetCountryCode

This function returns the two-character abbreviation that indicates the country of origin for the phone number passed to the Lookup function.

Phone Object includes data for American and Canadian area codes. The two-character abbreviation "US" is returned for area codes in the United States, and the two-character abbreviation "CA" is returned for Canadian area codes.

Syntax	StringValue = object->GetCountryCode();
C	StringValue = mdPhoneGetCountryCode(object);
COM	StringValue = object.CountryCode

GetDistance

This function returns the distance in miles from the area code/prefix wire center to the ZIP Code centroid, if a ZIP Code has been passed to the Lookup function in addition to a phone number.

The GetDistance function is a five-character (maximum) string value set by a call to the Lookup function.

The GetDistance function returns the distance between the centroids of the ZIP Code and wire center of the area code/prefix number passed into the Lookup function.

Because of phone number portability, the number returned by the GetDistance function may not be accurate.

If the Lookup function has not been called, resulted in an error, or no ZIP Code was passed in, this function will return "9999."

Syntax	StringValue = object->GetDistance();
C	StringValue = mdPhoneGetDistance(object);
COM	StringValue = object.Distance

GetLatitude

This function returns the latitude of the NPA/NXX wire center for the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Latitude is the geographic coordinate of the NPA/NXX wire center measured in degrees north or south of the equator. (NPA/NXX = area code/prefix).

The GetLatitude function returns a seven-character (max) string value after a call to the Lookup function. It is accurate to four decimal places. Phone numbers within the United States will always return a positive number.

If the Lookup function has not been called, or resulted in an error, this function will return 0.0.

Syntax	StringValue = object->GetLatitude();
C	StringValue = mdPhoneGetLatitude(object);
COM	StringValue = object.Latitude

GetLongitude

This function returns the longitude of the NPA/NXX wire center for the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Longitude is the geographic coordinate of the NPA/NXX wire center measured in degrees east or west of the Greenwich Meridian (NPA/NXX = area code/prefix).

The GetLongitude function returns a nine-character (max) string value after a call to the Lookup function. It is accurate to four decimal places and the negative sign is used to indicate a longitude in the United States.

If the Lookup function has not been called, or resulted in an error, this function will return 0.0.

Syntax	StringValue = object->GetLongitude();
C	StringValue = mdPhoneGetLongitude(object);
COM	StringValue = object.Longitude

GetMSA

This function returns the Metropolitan Statistical Area (MSA) number associated with the phone number passed to the Lookup function.

The Office of Management and Budget defines the Metropolitan Statistical Area (MSA). An MSA consists of one or more counties forming a large population with adjacent communities and having a high degree of social and economic integration.

The GetMsa function returns a four-digit value after a call to the Lookup function.

If the Lookup function has not been called, or resulted in an error, this function's return value will be blank.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetMSA();
C	StringValue = mdPhoneGetMSA(object);
COM	StringValue = object.MSA

GetPMSA

This function returns the Primary Metropolitan Statistical Area (PMSA) number associated with the phone number passed to the Lookup function.

The Office of Management and Budget defines the Primary Metropolitan Statistical Area (PMSA) for regions that contain a population of more than one million.

The GetPmsa function returns a four-digit string value after a call to the Lookup function.

Example: "5495" or "0"

If the Lookup function has not been called, or resulted in an error, this function value will be blank.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetPMSA();
C	StringValue = mdPhoneGetPMSA(object);
COM	StringValue = object.PMSA

GetTimeZone

This function returns a string describing the time zone of the phone number passed to the Lookup function.

All Melissa Data products express time zones in UTC (Coordinated Universal Time).

The GetTimeZone function returns a string value describing the time zone where the phone number is located. The function will return one of the following strings:

"Atlantic Time"	"Hawaii Time"
"Eastern Time"	"Samoa Time"
"Central Time"	"Marshall Islands Time"
"Mountain Time"	"Guam Time"
"Pacific Time"	"Palau Time"
"Alaska Time"	

If the input phone number cannot be corrected by the Lookup function, this string will be blank.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetTimeZone();
C	StringValue = mdPhoneGetTimeZone(object);
COM	StringValue = object.TimeZone

GetTimeZoneCode

This function returns a 1 or 2-digit code representing the time zone associated with the phone number passed to the Lookup function.

Following are the possible values for the TimeZoneCode function:

Code	Zone
4	Atlantic Time
5	Eastern Time
6	Central Time
7	Mountain Time
8	Pacific Time
9	Alaska Time

Code	Zone
10	Hawaii Time
11	Samoa Time
13	Marshall Island Time
14	Guam Time
15	Palau Time

If the input phone number cannot be corrected with the Lookup function, the return value will be blank.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetTimeZoneCode();
C	StringValue = mdPhoneGetTimeZoneCode(object);
COM	StringValue = object.TimeZoneCode

Computation Functions

The following functions do not require the data files to be initialized, but they can be used with the GetLatitude and GetLongitude functions to calculate distances and bearing between two records.

ComputeBearing

The ComputeBearing function returns a Bearing in degrees (-0 to 360) representing the compass direction from point 1 to point 2.

You do not have to call the Initialize function before calling the ComputeBearing function.

Input Parameters

This function accepts four double-precision floating point numbers.

- **lat1** - latitude for point 1 [In Degrees (90 to –90)]
- **long1** - longitude for point 1 [In Degrees (180 to –180)]
- **lat2** - latitude for point 2 [In Degrees (90 to –90)]
- **long2** - longitude for point 2 [In Degrees (180 to –180)]

Return Value

The ComputeBearing function returns a double-precision floating point bearing based on input latitudes and longitudes.

Syntax	DoubleFloat = object->ComputeBearing(lat1, long1, lat2, long2);
C	DoubleFloat = mdPhoneComputeBearing(object, lat1, long1, lat2, long2);
COM	DoubleFloat = object.ComputeBearing(lat1, long1, lat2, long2)

ComputeDistance

The ComputeDistance function returns a straight-line distance in miles between point 1 and point 2.

You do not have to call the Initialize function before calling the ComputeDistance function.

Input Parameters

The function accepts four double-precision floating point numbers.

- **latitude 1** - latitude for point 1 [In Degrees (90 to –90)]
- **longitude 1** - longitude for point 1 [In Degrees (180 to –180)]
- **latitude 2** - latitude for point 2 [In Degrees (90 to –90)]
- **longitude 2** - longitude for point 2 [In Degrees (180 to –180)]

Return Value

The ComputeDistance function returns a double-precision floating point value representing the distance between two points based on the input latitudes and longitudes.

Syntax	DoubleFloat = object->ComputeDistance(lat1, long1, lat2, long2);
C	DoubleFloat = mdPhoneComputeDistance(object, lat1, long1, lat2, long2);
COM	DoubleFloat = object.ComputeDistance(lat1, long1, lat2, long2)

Global Phone Object Functions

Initialize Global Phone Object

SetLicenseString

The License String is a software key (supplied by the developer) that unlocks the full functionality of Phone Object.

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 the license string has not been set, Phone Object will operate in a demonstration mode (limited to Nevada Area Codes) and will return the string "DEMO" after the GetBuildNumber function.

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). The SetLicenseString function will return a FALSE Boolean value if the license string provided is incorrect.

Syntax	BooleanValue = object->SetLicenseString(LicenseString);
C	IntegerValue = mdGlobalPhoneSetLicenseString(object,LicenseString);

Initialize

The Initialize function opens the needed data file and prepares the Global Phone Object for use.

If the function returns any value other than 0, call the GetInitializeErrorString function to retrieve the cause of the failure.

Input Parameters

The Initialize function has one parameter:

- **DataPath** - A String that contains the path to the location of the mdPhone.dat, mdPhone.idx, mdAddr.dat and NPA.TXT file.

Return Value

The Initialize function returns an integer value of 0 if successful.

Syntax	IntegerValue = object->Initialize(StringValue_DataPath);
C	IntegerValue = mdPhoneInitialize(object, StringValue_DataPath);

GetInitializeErrorString

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

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

Syntax	StringValue = object->GetInitializeErrorString();
C	StringValue = mdGlobalPhoneGetInitializeErrorString(object);

GetBuildNumber

The GetBuildNumber function returns the current development release build number of the Global Phone Object.

The word "DEMO" will be reported after the build number if no license string is provided, or if an incorrect license string is entered.

Input Parameters

None.

Return Value

The GetBuildNumber function returns the current development release build number of Global Phone Object.

Syntax	StringValue = object->GetBuildNumber();
C	StringValue = mdGlobalPhoneGetBuildNumber(object);

GetDatabaseDate

The GetDatabaseDate function returns a date value that represents the date of the Global Phone data files.

If the GetDatabaseDate function is called before the Initialize function is called and the data files are not in the same directory as the PHONEOBJ.DLL, this function will return a date outside the normal range of the system date, such as 1969 or 1899, depending on the system.

Input Parameters

None.

Return Value

The GetDatabaseDate function returns a value that represents the date of the phone data files. The standard object returns a string value.

Syntax	StringValue = object->GetDatabaseDate();
C	StringValue = mdGlobalPhoneGetDatabaseDate(object);

GetLicenseExpirationDate

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 standard object returns a string value.

Syntax	StringValue = object->GetLicenseExpirationDate();
C	StringValue = mdGlobalPhoneGetLicenseExpirationDate(object);

Look Up Global Phone Numbers

Lookup

The Lookup function verifies the submitted phone number and returns the detected geographic information.

Input Parameters

This function accepts three inputs: phone number, country, and country of origin.

- **Phone Number** - (Required) the phone number to be verified
- **Phone Country** - the suspected country of the phone number
- **Country of Origin** - the country from where the verification is being done. If the Country of Origin differs from the Phone Country, then the outputted phone number will be changed to a callable format from the country of origin.

Syntax	BooleanValue = object->Lookup(StringValue_Phone, StringValue_PhoneCounty, StringValue_CountryOrigin);
C	IntegerValue = mdGlobalPhoneLookup(StringValue_Phone, StringValue_PhoneCounty, StringValue_CountryOrigin);

LookupNext

The LookupNext function returns the next suggestion, if any, based on the originally submitted number.

This function can be called after a successful call to the Lookup function and can be repeated until the function returns an integer value of 0, indicating that no further suggestions are available.

If an integer value of 1 is returned, then the suggested phone number will be used to populate the return values of the same functions as the Lookup function.

Syntax	BooleanValue = object->LookupNext(StringValue_Phone, StringValue_PhoneCounty, StringValue_CountryOrigin);
C	IntegerValue = mdGlobalPhoneLookupNext(StringValue_Phone, StringValue_PhoneCounty, StringValue_CountryOrigin);

Retrieve Status Information

GetResults

This function returns a comma-delimited string of four-character codes which detail the level of matching found for the current phone number and any errors that occurred during the last call to the Lookup or Lookup Next function.

The GetRestuls function returns one or more Results Codes in a comma-delimited list. For a list of these Results Codes, see "Results Codes" on page 34.

Syntax	StringValue = object->GetResults();
C	StringValue = mdGlobalPhoneGetResults(object);

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 = mdGlobalPhoneGetResultCodeDescription(object, StringValue_ResultCode, int);

Retrieve Phone Number Data

GetInternationalPrefix

This function returns the international prefix (international exit code) needed to call a number outside of the dialing country.

If the country and country of origin differ, this function will return the digit(s) requred to be dialed before the country code.

Syntax	StringValue = object->GetInternationalPrefix();
C	StringValue = mdGlobalPhoneGetInternationalPrefix(object);

GetNationalDestinationCode

This function returns the national destination code of the inputted phone number.

The national destination code can be a single or combination of decimal digits (not including any prefix) that identifies a numbering area within a country (or group of countries) and/or network/ services.

Syntax	StringValue = object->GetNationalDestinationCode();
C	StringValue = mdGlobalPhoneGetNationalDestinationCode(object);

GetNationPrefix

This function returns the nation prefix of the inputted phone number.

The national prefix is the digit, or combination of digits, which must be dialed before an area (city) code when calling a number from within the same country but outside the numbering area.

For example: 1-949-589-5200 where 1 is the national prefix.

Syntax	StringValue = object->GetNationPrefix();
C	StringValue = mdGlobalPhoneGetNationPrefix(object);

GetPhoneNumber

This function returns the standardized phone number after a successful call to the Lookup or LookupNext function.

The phone number will return in varying formats depending on the inputs from the Lookup function.

If the inputted country and country of origin are the same, GetPhoneNumber will return the National Destination Code (NDC) + Subscriber Number (SN). For Example:

Input Phone	Input Country	Country of Origin	Phone Returned
495-728-5802	RU	RU	495-728-5802
+7 495-728-5802	RU	RU	495-728-5802
+1 495-728-5802	RU	RU	495-728-5802

If the inputted country and country of origin differ, GetPhoneNumber will return a leading '+' followed by the Country Code + NDC + SN. The leading '+' represents the International Prefix that is required to dial a number outside of the caller's country. For Example:

Input Phone	Input Country	Country of Origin	Phone Returned
495-728-5802	RU	US	+7 495-728-5802
+7 495-728-5802	RU	US	+7 495-728-5802
+1 495-728-5802	RU	US	+7 495-728-5802

If a country is entered and a country of origin is left blank or not passed in, GetPhoneNumber will match the format of the inputted phone number. For Example:

Input Phone	Input Country	Country of Origin	Phone Returned
495-728-5802	RU	<Blank>	495-728-5802
+7 495-728-5802	RU	<Blank>	+7 495-728-5802
+1 495-728-5802	RU	<Blank>	+7 495-728-5802

Syntax	StringValue = object->GetPhoneNumber();
C	StringValue = mdGlobalPhoneGetPhoneNumber(object);

Retrieve GeoCode Data

GetLocality

This function returns the locality (city) associated with the phone number passed to the Lookup function.

Because of phone portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetLocality();
C	StringValue = mdGlobalPhoneGetLocality(object);

GetCountry

This function returns the name of the country for the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetCountry();
C	StringValue = mdGlobalPhoneGetCountry(object);

GetCountryCode

This function returns the country code equivalent for the detected country of the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetCountryCode();
C	StringValue = mdGlobalPhoneGetCountryCode(object);

GetDST

This function returns a string designating whether the detected geographical location of the phone number observes daylight saving time.

Daylight saving time is the practice of advancing clocks (typically an hour) near the beginning of spring and adjusted backward in autumn.

GetDST will return a 'Y' for yes and 'N' for no.

Syntax	StringValue = object->GetDST();
C	StringValue = mdGlobalPhoneGetDST(object);

GetLanguage

This function returns the predominant language of the phone's detected geographical location.

The GetLanguage function will return the written out language name. For example, "French".

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetLanguage();
C	StringValue = mdGlobalPhoneGetLanguage(object);

GetLatitude

This function returns the latitude of the geographically identifiable service area of the exchange.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Latitude is the geographic coordinate of the locale, city, municipality, or other geographically identifiable service area of the exchange measured in degrees north or south of the equator.

The GetLatitude function returns a, at most, 22-character string value after a call to the Lookup function.

If the Lookup function has not been called, or resulted in an error, this function will return 0.0.

Syntax	StringValue = object->GetLatitude();
C	StringValue = mdGlobalPhoneGetLatitude(object);

GetLongitude

This function returns the longitude of the geographically identifiable service area of the exchange.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Longitude is the geographic coordinate of the locale, city, municipality, or other geographically identifiable service area of the exchange measured in degrees east or west of the Greenwich Meridian.

The GetLatitude function returns a, at most, 22-character string value after a call to the Lookup function.

If the Lookup function has not been called, or resulted in an error, this function will return 0.0.

Syntax	StringValue = object->GetLongitude();
C	StringValue = mdGlobalPhoneGetLongitude(object);

GetAdministrativeArea

This function returns the administrative area associated with the phone number passed to the Lookup function.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetAdministrativeArea();
C	StringValue = mdGlobalPhoneGetAdministrativeArea(object);

GetSubscriberNumber

This function returns the subscriber number associated with the phone number passed to the Lookup function.

The subscriber number contains significant leading digits that further define the local exchange area and/or service.

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetSubscriberNumber();
C	StringValue = mdGlobalPhoneGetSubscriberNumber(object);

GetUTC

This function returns the universal time code for the time zone associated with the phone number passed to the Lookup function.

The GetUTC function returns the time zone specified with the format: +/- hh:mm .

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax	StringValue = object->GetUTC();
C	StringValue = mdGlobalPhoneGetUTC(object);

Appendix

Results Codes

Phone Status Codes

Code	Short Description	Long Description
PS01	10-Digit Match	The first 10-digits of the phone number have been verified as valid.
PS02	7-Digit Match	The first 7-digits of the phone number has been verified as valid.
PS03	Corrected Area Code	NewAreaCode contains corrected area code that was changed according to the postal code it falls into.
PS04	Demo Mode	Demo mode is active and the phone number is outside the range of phone numbers allowed by the Demo.
PS05	Database Expired	The phone database is expired. Please update your data.
PS06	Updated Area Code	The area code was changed due to an area code split. The updated code is located within NewAreaCode.
PS07	Cellular Line	The exchange type of the phone number indicates the number is a cellular number.
PS08	Land Line	The exchange type of the phone number indicates the number is a land line number.
PS09	VOIP Line	The exchange type of the phone number indicates the number is a VOIP number.
PS10	Residential Number	The phone number belongs to a residence.
PS11	Business Number	The phone number belongs to a business.
PS12	SOHO Number	The phone number belongs to a small office or home office.
PS13	Toll Free Number	The phone number is a toll free number.
PS14	Special Number	This is a phone number with premium service, data, internet access, etc. which all incur a higher charge rate.

Phone Change Codes

Code	Short Description	Long Description
PC01	Country Changed	The country was added or changed to correspond to the phone number.

Phone Error Codes

Code	Short Description	Long Description
PE01	Bad Area Code	The area code does not exist in our database or contains non-numbers.
PE02	Blank Phone Number	The phone number is blank.
PE03	Bad Phone Number	The phone number has too many or too few digits.
PE04	Multiple Match	Two or more possible area codes are available as a fix and their distance is too close to choose one over the other.
PE05	Bad Prefix	The phone prefix does not exist in our database.
PE06	Bad Postal Code	The input postal code is invalid.
PE08	No Country Input	The input country is blank and the phone number has no '+' sign.
PE09	Out of Range Suffix	The subscriber's phone number suffix is out of range.
PE10	Invalid Input Country	The input country in not valid.