

Net
Address Microsoft for .net
4.0

Address Validation & Standardization Component

The Software Company, Inc.
www.SoftwareCompany.com

The Software Software is our middle name
Company™

NetAddress for .NET allows you to quickly and easily build address verification, standardization and parsing into your custom applications. Accept addresses free-form and let NetAddress do the rest. Each address will be standardized, split into USPS standard components then graded for overall completeness and accuracy. Plus, no USPS database subscription is required.

NetAddress can process all types of addresses including Suite Numbers and City/State/Zip. A special Address_Quality flag is returned each time an address is processed allowing you to easily identify questionable addresses before they enter your system. Also returned is the Address_Type flag indicating the type of address being processed: Street, Military, PO Box, Rural Route, Highway Contract, General Delivery or Suite giving you flexibility in their handling. And, for a more appealing presentation, let NetAddress set the proper capitalization.

As a bonus, when you combine NetAddress with our NetGender product, you can handle even the impossible task of identifying data that has been entered free-form, where the names, addresses and C/S/Z “float” from field to field. You’ll always be certain of what data you’re working with.

NetAddress is being used by government agencies from coast-to-coast. Some states now require that addresses be stored in standardized and parsed format to facilitate address matching from agency to agency.

Benefits

- Save \$\$\$ on Postage by Eliminating Incomplete or Non-Addresses
- Catch Data-Input Errors Before They Enter Your Database
- Standardize Addresses for Faster Processing
- Unlimited Processing Volume - No Recurring Update Charges
- FREE Upgrades for a Full Year

Features

- Addresses are Standardized to USPS Recommended Abbreviations
- Proper Case Conversion for More Attractive Data Presentation
- Royalty-Free Runtime
- Easily Separate Name, Street Address and City/State/Zip
- Designed for Use with All .NET Compatible Programming Languages

NetAddress for .NET starts by carefully identifying each individual address component based on its context. Intuitive algorithms examine the results and a selection is made of the most complete and correct data. If needed, format corrections are made and the USPS recommended abbreviations are applied. The `Address_Quality` flag is then set to indicate how complete and correct the address is. Finally, the standardized address components are returned to your application along with a complete and cleansed composite address.

Strict conformity to USPS "[Postal Addressing Standards Publication 28](#)" ensures consistent standardization of every address. However, you can easily customize these settings for critical applications.

NetAddress for .NET is the *only* address verification and parsing software that can reliably find and extract a Street Address when it's surrounded by extraneous data. It can even separate Street Address from City/State/Zip when they're in the same field.

The success or failure of any parsing software is dependent on how well it can handle "dirty" addresses. These are addresses that have non-standard abbreviations or the address elements are run together such as APT6. NetAddress can handle these and more.

Examples

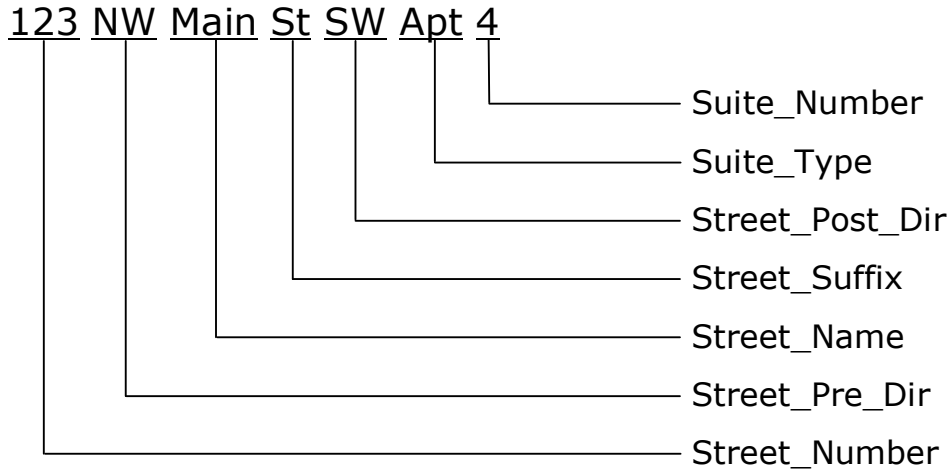
Address_In: IS DEPT 123NE MAIN ST#1
Address_Out: 123 NE Main St # 1

Address_In: Re: DOC#222 123 ADAMS BL AP5%DORIS
Address_Out: 123 Adams Blvd Apt 5

Address_In: AP2,123NE SOUTH STREET WEST%JANE
Address_Out: 123 NE South St W Apt 2

Address_In: 6TH FLOOR ONE BROADWAY ST,RE:LN-123456
Address_Out: 1 Broadway St Fl 6

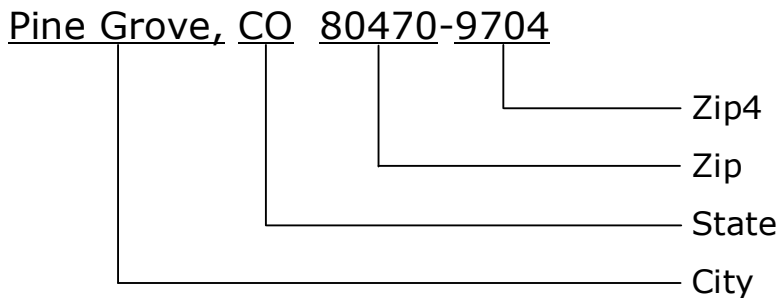
Street Address



Box Address



City/State/Zip



Address_In

Syntax: Address_In = String

Description:

Set this property to the address string to be processed.

When the “Parse” method is invoked, the Address_In string is standardized and corrected then placed into the Address_Out property. In addition, each element of the Address_In string is placed into the appropriate address component property.

CSZ_In

Syntax: CSZ_In = String

Description:

Set this property to the city, state and zip string (last line) of the address.

When the “Parse” method is invoked, the CSZ_In string is standardized and corrected then placed into the CSZ_Out property. In addition, each component of the CSZ_In property is placed into the appropriate City, State, Zip component property.

Setting CSZ_In to “USA” or “Canada” will force the appropriate interpretation of Address_In.

Canadian Addresses:

Municipality / Province / Postcode are synonymous with City / State / Zip respectively.

Address_CSZ_Combined (new in v4.0+)
(replaces obsolete “CSZ_InSameField” property)

Syntax: Address_CSZ_Combined = Boolean

Description:

Set this property to Boolean (True/False) to indicate whether or not the Address_In property also contains city/state/zip data. Set Address_CSZ_Combined to “True” to parse this extra data into the City, State and Zip properties. **Default is “False”.**

Address_CSZ_Delimiter (new in v4.0+)

Syntax: Address_CSZ_Delimiter = String

Description:

Set this property to an *optional* delimiter string to indicate where to separate the address from the CSZ when Address_CSZ_Combined is set to “True”. **Default is no delimiter.**

Street_Number_Suite_Combined (new in v4.0+)

Syntax: Street_Number_Suite_Combined = Boolean

Description:

Set this property to Boolean (True/False) to indicate whether or not the Street_Number property also contains a Suite_Number. Set Street_Number_Suite_Combined to “True” if you want hyphenated street numbers parsed into Street_Number and Suite_Number. **Default is “False”.**

USA: 10-100 Main Street = 10 Main St #100

CANADA: 10-100 Main Street = 100 Main St #10

Note:

If a suite number is present in the Address_In string, this setting will be overridden.

Numeric_Street_Conv

Syntax: Numeric_Street_Conv = Boolean

Description:

Set this property to Boolean (True/False) to indicate whether or not to convert a spelled out ordinal street name to an ordinal number. (“Third” converts to “3rd”, etc.) Set Numeric_Street_Conv to “True” if you want the street name converted. Numeric street names are spelled out only when there are duplicate street names within a postal delivery area and the only distinguishing factor is that one of them is spelled out. **Default is “False”.**

Numeric_Street_No_Ordinal (new in v4.0+)

Syntax: Numeric_Street_No_Ordinal = Boolean

Description:

Set this property to Boolean (True/False) to indicate whether or not to convert a numeric street name to an ordinal number. (“3” converts to “3rd”, etc.) Set Numeric_Street_No_Ordinal to “True” if you want to retain the original numeric street name. **Default is “False”.**

Output_Case (changed in v3.0+)

Syntax: Output_Case = “StringLiteral”

Description:

Set this property to “Upper”, “Mixed” or “None” to indicate your capitalization preference for the output address and its components. **Default is “None”.**

Reference_File_Path (changed in v3.0+)

Syntax: Reference_File_Path = String

Description:

Set this property to the full path and file name of the user-defined file containing the Street_Suffix and Suite_Type abbreviations. A standard set of abbreviations is supplied and installed in the NetAddress installation folder under the name: "NetAddress.ref". You can rename and relocate this file to any other folder as long as you set this property to the full path and file name. **Defaults are the NetAddress installation folder and then the folder of the invoking application. (AppDomain.CurrentDomain.BaseDirectory)**

See "Updating User Control Tables" later in this guide for instructions on customizing this file.

Static_Key_Name

Syntax: Static_Key_Name = String

Description:

Set this property to the name portion of the static key assignment or blank.

Static_Key

Syntax: Static_Key = String

Description:

Set this property to the key portion of the static key assignment or blank.

Address_Out (read only)

Syntax: String = Address_Out

Description:

After invoking the “Parse” method, this property will contain the standardized and corrected address string from the Address_In property including a suite number if present. If the Address_In string returns an Address_Quality of “Low”, this property will be blank.

Address_Out_Street (read only)

Syntax: String = Address_Out_Street

Description:

After invoking the “Parse” method, this property will contain the street portion of Address_Out.

Address_Out_Suite (read only)

Syntax: String = Address_Out_Suite

Description:

After invoking the “Parse” method, this property will contain the suite portion of Address_Out.

Street_Number (read only)

Syntax: String = Street_Number

Description:

After invoking the “Parse” method, this property is set to the primary address number component of Address_Out commonly referred to as house number, street number, civic number or range.

Street_Pre_Dir (read only)

Syntax: String = Street_Pre_Dir

Description:

After invoking the “Parse” method, this property is set to the Predirectional component of Address_Out. Values will be a valid directional (N, NE, S, SE, etc.) or blank.

Street_Name (read only)

Syntax: String = Street_Name

Description:

After invoking the “Parse” method, this property is set to the Street Name component of Address_Out. Value will be alphanumeric.

Street_Suffix (read only)

Syntax: String = Street_Suffix

Description:

After invoking the “Parse” method, this property is set to the Street Suffix component of Address_Out. Values will be a valid suffix (St, Ave, Rd, etc.) or blank.

Street_Post_Dir (read only)

Syntax: String = Street_Post_Dir

Description:

After invoking the “Parse” method, this property is set to the Postdirectional component of Address_Out. Values will be a valid directional (N, NE, S, SE, etc.) or blank.

Suite_Type (read only)

Syntax: String = Suite_Type

Description:

After invoking the “Parse” method, this property is set to the Suite Type component of Address_Out. Values will be only valid suite types (Apt, Suite, Unit, etc.) or blank.

Suite_Number (read only)

Syntax: String = Suite_Number

Description:

After invoking the “Parse” method, this property is set to the Suite Number component of Address_Out. Values will be alphanumeric suite number or blank.

Box_Type (read only)

Syntax: String = Box_Type

Description:

After invoking the “Parse” method, this property is set to the Box Type component of Address_Out. Values will be only valid box types (PO Box, RR, HC, etc.) or blank.

Box_Type_Number (read only)

Syntax: String = Box_Type_Number

Description:

After invoking the “Parse” method, this property is set to the Box Type Number component of Address_Out. Values will be alphanumeric box type number or blank.

Box (read only)

Syntax: String = Box

Description:

After invoking the "Parse" method, this property is set to the Box component of Address_Out. Value will be "Box" or blank.

Canadian Addresses:

Value may also be "Stn" and "RPO"

Box_Number (read only)

Syntax: String = Box_Number

Description:

After invoking the "Parse" method, this property is set to the Box Number component of Address_Out. Values will be alphanumeric box number or blank.

Canadian Addresses:

Value may also be Station name or Retail Postal Outlet (RPO) name.

Address_Quality (read only)

Syntax: String = Address_Quality

Description:

After invoking the "Parse" method, this property is set according to the completeness of the Input_Address. "Low" if no recognizable address is present. "Medium" if an address is present but is incomplete, such as a missing apartment number or street suffix. "High" is returned when a complete and technically correct address or suite is found.

If your addresses "float" from field to field you can easily determine which field contains the address through trial and error by examining Address_Quality and Address_Type after trying each field.

Address_Type (read only)

Syntax: String = Address_Type

Description:

After invoking the “Parse” method, this property is set to one of the following address types:

S	Street	(1 N Main St, 2 US Highway 285, etc.)
A	Suite	(Apt 1, Suite 2, etc.)
P	Post Office Box	(PO Box 1)
R	Rural Route	(RR 1 Box 2)
H	Highway Contract	(HC 1 Box 2)
G	General Delivery	(General Delivery, Gen Del, GD, etc.)
M	Military	(CMR 1 Box 2, etc.)
N	Not a valid address	Address_Quality will also be set to “Low”

Address_Leading_Data (read only)

Syntax: String = Address_Leading_Data

Description:

After invoking the “Parse” method, this property will contain all data that precedes the actual address. If the Address_In string returns an Address_Quality of “Low”, this property will be blank.

Address_Trailing_Data (read only)

Syntax: String = Address_Trailing_Data

Description:

After invoking the “Parse” method, this property will contain all data that follows the actual address. If the Address_In string returns an Address_Quality of “Low”, this property will be blank.

Address_Filtered_Data (read only)

Syntax: String = Address_Filtered_Data

Description:

After invoking the “Parse” method, this property will contain all data that was filtered out before processing according to the [FilterAddress] section of NetAddress.ref file.

CSZ_Out (read only)

Syntax: String = CSZ_Out

Description:

After invoking the “Parse” method, this property will contain the standardized city/state/zip (last line) from the CSZ_In property.

Canadian Addresses:

Municipality / Province / Postcode are synonymous with City / State / Zip respectively.

City (read only)

Syntax: String = City

Description:

After invoking the “Parse” method, this property is set to the City component of CSZ_Out.

Canadian Addresses:

Municipality will be placed in City.

State (read only)

Syntax: String = State

Description:

After invoking the “Parse” method, this property is set to the State component of CSZ_Out. Values will be only valid USPS state abbreviations (FL, AZ, CO, etc.) or blank.

Canadian Addresses:

Province abbreviation will be placed into State.

Zip (read only)

Syntax: String = Zip

Description:

After invoking the “Parse” method, this property is set to the Zip component of CSZ_Out. Values will be a 5-digit numeric zip code or blank.

Canadian Addresses:

Forward Sortation Area will be placed in Zip. This is the left segment of the postcode: A1A 1A1

Zip4 (read only)

Syntax: String = Zip4

Description:

After invoking the “Parse” method, this property is set to the Zip+4 component of CSZ_Out. Values will be a 4-digit numeric zip add-on code (sector/segment) or blank.

Canadian Addresses:

Local Delivery Unit will be placed in Zip4. This is the right segment of the postcode: A1A 1A1

Country (read only) (changed in v4.0+)

Syntax: String = Country

Description:

After invoking the “Parse” method, this property will contain the country identified by the input state/province. Values will be the ISO 3166 standardized country code: “US” (USA), “CA” (Canada) or blank.

State_FIPS (read only) (new in v4.0+)

Syntax: String = State_FIPS

Description:

After invoking the “Parse” method, this property is set to the Federal Information Processing Standard (FIPS) code for the state/province in which the address resides. Values will be a two-digit numeric string or blank.

CSZ_Quality (read only)

Syntax: String = CSZ_Quality

Description:

After invoking the “Parse” method, this property is set to ”Low”, “Medium” or “High” to indicate the probability that the CSZ_In property value is complete and correct.

CSZ_Filtered_Data (read only) (new in v4.0+)

Syntax: String = CSZ_Filtered_Data

Description:

After invoking the “Parse” method, this property will contain all data that was filtered out before processing according to the [FilterCityStateZip] section of NetAddress.ref file.

Return_Code (read only)

Syntax: String = Return_Code

Description:

After invoking the “Parse” method, this property is set to blank upon successful completion. Some common exceptions are listed below. Most exceptions usually occur on the first call to the “Parse” method. *This property should be examined on each return from NetAddress.*

Common Return Codes:

R35	Reference File Not Found (see Reference_File_Path property)
T00	Suffix Table Limit Reached (1024)
T01	Suite Type Table Limit Reached (256)
T02	Filter Table Limit Reached (128)
L00	Evaluation Period Expired
L01	Static Key Validation Failed (see Static_Key property)
L50 - L69	License Validation Failed

Clear

Syntax: NetAddress.Clear

Description:

When this method is invoked, all properties are set to null with the exception of Static_Key, Static_Key_Name and Reference_File_Path.

Parse

Syntax: NetAddress.Parse

Description:

When this method is invoked, the Address_In property is standardized and corrected then placed into the Address_Out property. In addition, each element of the Address_Out property is placed into the appropriate address component property and the Address_Quality and Address_Type flags are set. The Return_Code property is also set and should be checked after each invocation of the “Parse” method. *See “Return_Code” property.*

Updating User Control Tables

NetAddress.ref contains a complete list of street suffixes and suite types along with their abbreviations and full spellings. It is located in the “NetAddress” installation folder. Use Notepad or a similar text editor to edit the file. Detailed instructions on the format of the entries are contained within the file. This file can also be relocated. See *Reference_File_Path* property.

NetAddress allows you to specify which street suffixes and suite types are to be recognized as well as your preferred abbreviations.

The filter section of the table allows you to specify which, if any, characters, words or phrases are to be ignored during processing. All filters that were found in the Address_In property will be stored in the Address_Filtered_Data property.

- 1st Column: Common
- 2nd Column: Full Spelling
- 3rd Column: Abbreviation
- 4th Column: City Prefix Flag (Y/N) - When Address_CSZ_Combined is set to "True" and a street suffix is already present, treat this suffix as part of the city name.

[StreetSuffixUSA]

ST	Street	St	Y
STR	Street	St	N
STREET	Street	St	N

[SuiteType]

STE	Suite	Ste	Y
SUITE	Suite	Ste	Y

[FilterAddress]

C/O
ET AL

Note: Changing the “Full Spelling” column (column 2) is not recommended.

If Installation Doesn't Start Automatically:

- Select **Start > Run** from the Task Bar.
- Type CD-ROM drive letter followed by “:\NetAddress40.msi” and press enter.

In the folder “Program Files\The Software Company\NetAddress 4.0” you will find a sample program named: VBSample.vbproj. There is also a compiled version called VBSample.exe that you can run to demonstrate NetAddress.

Deploying Your Applications

Be sure to include the following in your deployment package:

NetAddress.dll – usually placed in the application folder
NetAddress.ref* – usually placed in the application folder

* NetAddress.ref can be relocated anywhere on the target machine as long as the full path and file name are specified in the Reference_File_Path property.

In addition to the above, there is a small runtime package that is installed into the Global Assembly Cache (GAC) of the target machine. There are two ways to do this:

Include the **MergeModuleRuntimeNET20.msm** with your installation package

- OR -

Execute **StandaloneRuntimeNET20.exe** for each new installation

There's an explanation of each at the following link:

http://www.SoftwareCompany.com/dotnet/dotnet_runtime.htm

This product is initially licensed for a period of 30 days or up to 1000 calls. It must be registered to continue using it after this evaluation period. Please contact us at:

Sales@SoftwareCompany.com

303/838-1223 (voice)

303/838-1224 (fax)