# NetGender for .NET 4.4

## Name Parsing and Gender Identification API User's Guide

**The Software Company, Inc.**
**www.SoftwareCompany.com**

Software is our middle name

# NetGender v4.4 for .NET

## Introduction

**NetGender for .NET** allows you to quickly and easily build name verification, parsing and gender determination into your custom applications. Accurately verify whether or not a particular field contains a valid individual or company name.

Accept names free-form and let NetGender automatically split each name into its standard components: Prefix, First, Middle, Last and Suffix no matter what the original format. Using a 200,000+ first and last name dictionary in combination with a 10,000+ term company dictionary, the gender is determined with unmatched precision. Easily updated tables control the entire process.

NetGender can process all styles of names including inverse, natural order, hyphenated and multi-part last names. Multiple names in the same field and companies can be easily separated providing you with powerful formatting control.

With the help of the "Name_Variants" property, you can easily identify obscure duplicate names such as "Bill", Billy", Will", "Willy" and "William".

## Benefits

- **200,000+ First and Last Name Dictionary** for Pinpoint Accuracy
- **Quickly Identify Incomplete or Incorrect Names** – before they enter your database
- **Automatic Name Style Identification** – standardize lists of various formats
- **Name Variants Property** - invaluable for finding common duplicates
- **Unlimited Processing Volume** – no recurring update charges
- **Free Upgrades** for a full year

## Features

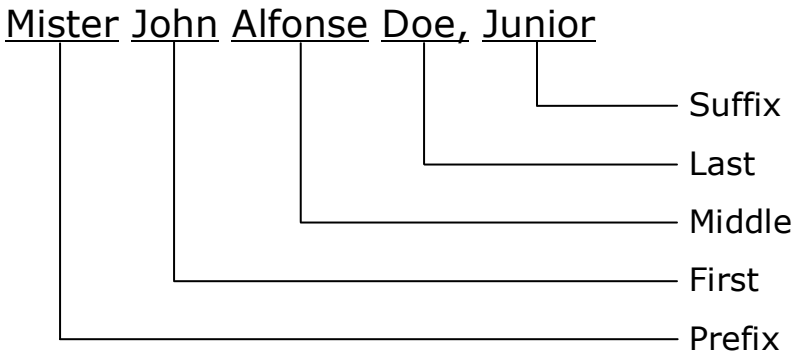
- Accurately separate name elements: Prefix, First, Middle, Last, Suffix
- Proper-case conversion for more attractive data presentation
- User-updatable tables so your applications never go out-of-date
- Royalty-free runtime
- Designed for use with all .NET compatible programming languages

**NetGender for .NET** starts by meticulously identifying each individual name element based on the Name_Style you've selected. Intuitive algorithms examine the results and a selection is made of the most complete and correct data. Next, the name elements are split according to Name_Style and the user-specified prefix/suffix abbreviations are applied. If not already coded by the gender-override table, the gender is now determined according to the 200,000+ name system gender table. The Name_Quality flag is then set to indicate how complete and correct the name is. Finally, the standardized name components are returned to your application along with a complete and cleansed composite name.

**Examples using "Variable" name style:**

| | | |
|---|---|---|
| Name_In | | DE LA ROSA, JUAN R OR MARIA |
| Name_Out | Male | Juan R De La Rosa |
| Name2_Out | Female | Maria De La Rosa |
| Name_In | | SMITH-LA BELLA JR, JOHN A & MARY B, PHD |
| Name_Out | Male | John A Smith-La Bella Jr |
| Name2_Out | Female | Mary B Smith-La Bella PhD |
| Name_In | | MARY MYTH C/O THE SOFTWARE COMPANY |
| Name_Out | Female | Mary Myth |
| Name2_Out | Company | The Software Company |
| Name_In | | O'LEARY-VAN HORN, DALE & MIKE |
| Name_Out | Neutral | Dale O'Leary-Van Horn |
| Name2_Out | Male | Mike O'Leary-Van Horn |
| Name_In | | BO JONES & WIFE BO PEEP |
| Name_Out | Male | Bo Jones |
| Name2_Out | Female | Bo Peep Jones |

## Name Elements




---

### A Word About Gender Coding

Not all gender identification systems are created the same. Most use a simple table of less than 20,000 first names. NetGender uses an extensive dictionary of more than 200,000 first and last names that were specially selected for their rich ethnic-diversity.

NetGender also uses a unique gender percentage factor. This factor is based on the proportion of males to females for a particular name allowing you to set the point at which certain names will be returned with a neutral gender. *See "Gender_Confidence" and "Gender_Percentage" properties.*

NetGender then applies the pre-programmed rules from a 10,000+ term company dictionary resulting in a high level of gender accuracy.

*This unique and powerful strategy produces unparalleled results.*

## Name_In

**Syntax:** Name_In = String

**Description:**
Set this property to the name string to be processed.

When the "Parse" method is invoked, the Name_In string is standardized and formatted then placed into the Name_Out property. In addition, each element of the Name_In string is placed into the corresponding name component property.

---

## Name_Style (changed in v3.0+)

**Syntax:** Name_Style = StringLiteral

**Description:**
Set this property to the style that most closely matches the format of the Name_In string. In the examples below, F=First, M=Middle, L=Last, &=generic delimiter, (…)=optional element.

| | | |
|---|---|---|
| FML | John A Doe | Names are in Natural Order |
| FML&FM | John A Doe & Jane A | Never a Last Name in Name 2 |
| FML&FM(L) | John A Doe & Jane A (Jones) | Sometimes a Last Name in Name 2 |
| FML&FML | John A Doe & Jane A Jones | Always a Last Name in Name 1 & 2 |
| FM(L)&FML | John A (Doe) & Jane A Jones | Sometimes a Last Name in Name 1 |
| FM(L)&FM(L) | John A (Doe) & Jane A (Jones) | Sometimes a Last Name in Name 1 or 2 |
| FM&FML | John A & Jane A Doe | Never a Last Name in Name 1 |
| LFM | Doe, John A | Names are in Inverse Order |
| LFM&FM | Doe, John A & Jane A | Never a Last Name in Name 2 |
| LFM&FM(L) | Doe, John A & Jane A (Jones) | Sometimes a Last Name in Name 2 |
| LFM&(L)FM | Doe, John A & (Jones)Jane A | Sometimes a Last Name in Name 2 |
| LFM&LFM | Doe, John A & Jones Jane A | Always a Last Name in Name 1 & 2 |
| VARIABLE | Inconsistent & Unknown | Names are mixed Natural & Inverse |
| SPLIT | Anything & Everything | Split at Name Connector only |
| COMPANY | Sanford & Son | Forces a company name check without splitting at the name connector |

The input names can be separated by various delimiters such as: "c/o", "&", "dba", "and", "attn", etc. The "&" above, generically represents these delimiters. The delimiters are user-defined. *See "Updating User Control Tables" later in this guide.* Parenthesis "(…)" indicate that this element may or may not be present in all of the input name strings. If two individuals or a company and an individual co-exist in the input name string, you should choose one of the multi-name styles such as: FML&FML, FML&FM, etc. for best results.

*The closer the style matches your data the better the results. Use "Variable" when the name format is inconsistent or unknown.*

## Name Style Examples - Natural Order

**FML** **Use when there is only one name in the input string.**
Name is split into First, Middle, Last:
**Name_In:** JOHN A DOE
**Name_Out:** John A Doe

**FML&FM** **Use when last name is *never* present in name two.**
Name two ALWAYS inherits its last name from Name one:
**Name_In:** JANE A DOE & JOHN A
**Name_Out:** Jane A Doe
**Name2_Out:** John A Doe

**FML&FM(L)** **Use when last name is *sometimes* present in name two.**
Name two keeps its own last name if present:
**Name_In:** JOHN A DOE & JANE A JONES
**Name_Out:** John A Doe
**Name2_Out:** Jane A Jones
Name two inherits its last name from Name one:
**Name_In:** JOHN A DOE & JANE A
**Name_Out:** John A Doe
**Name2_Out:** Jane A Doe

**FML&FML** **Use when a full name is *always* present in name one & two.**
Name one & two ALWAYS keep their own last names:
**Name_In:** JOHN DOE DBA THE SOFTWARE COMPANY
**Name_Out:** John Doe
**Name2_Out:** The Software Company

**FM(L)&FML** **Use when last name is *sometimes* present in name one.**
Name one keeps its own last name if present:
**Name_In:** JOHN A DOE & JANE A JONES
**Name_Out:** John A Doe
**Name2_Out:** Jane A Jones
Name one inherits its last name from Name two:
**Name_In:** JOHN A DOE & JANE A
**Name_Out:** John A Doe
**Name2_Out:** Jane A Doe

## Name Style Examples - Natural Order (cont.)

**FM(L)&FM(L) Use when last name is *sometimes* present in name one or two.**

Name one and two keep their own last names if present:

**Name_In:** JOHN A DOE & JANE A JONES
**Name_Out:** John A Doe
**Name2_Out:** Jane A Jones

Name one inherits its last name from Name two:

**Name_In:** JOHN A & JANE A JONES
**Name_Out:** John A Jones
**Name2_Out:** Jane A Jones

**FM&FML Use when last name is *never* present in name one.**

Name one ALWAYS inherits its last name from Name two:

**Name_In:** JANE A & JOHN DOE
**Name_Out:** Jane A Doe
**Name2_Out:** John Doe

## Name Style Examples - Inverse Order

**LFM** **Use when there is only one name in the input string.**
Name is split into First, Middle, Last:
**Name_In:** DOE, JOHN A
**Name_Out:** John A Doe

**LFM&FM** **Use when last name is *never* present in name two.**
Name two ALWAYS inherits its last name from Name one:
**Name_In:** DOE, JANE A & JOHN A
**Name_Out:** Jane A Doe
**Name2_Out:** John A Doe

**LFM&FM(L)** **Use when last name is *sometimes* present in name two.**
Name two keeps its own last name if present:
**Name_In:** DOE, JOHN A & JANE A JONES
**Name_Out:** John A Doe
**Name2_Out:** Jane A Jones
Name two inherits its last name from Name one:
**Name_In:** DOE, JOHN A & JANE A
**Name_Out:** John A Doe
**Name2_Out:** Jane A Doe

**LFM&(L)FM** **Use when last name is *sometimes* present in name two.**
Name two keeps its own last name if present:
**Name_In:** DOE, JOHN A & JONES, JANE A
**Name_Out:** John A Doe
**Name2_Out:** Jane A Jones
Name two inherits its last name from Name one:
**Name_In:** DOE, JOHN A & JANE A
**Name_Out:** John A Doe
**Name2_Out:** Jane A Doe

**LFM&LFM** **Use when a full name is *always* present in name one & two.**
Name one & two ALWAYS keep their own last names:
**Name_In:** DOE, JOHN DBA THE SOFTWARE COMPANY
**Name_Out:** John Doe
**Name2_Out:** The Software Company

## Name Style Examples - Other

**VARIABLE** **Use when the format of the name(s) is inconsistent or unknown.**
Style is automatically determined based on the format, punctuation and gender lookup results. Name one is FML, Name two is LFM:

**Name_In:** JOHN A DOE & JONES, JANE A
**Name_Out:** John A Doe
**Name2_Out:** Jane A Jones

*Also, see Variable_Default property*

**SPLIT** **Use when you want to split name one & two at the [NameConnector]**
Filters are removed, no other formatting takes place.
[NameConnector] of "c/o" was specified in the "NetGender.ref" file:

**Name_In:** DOE JOHN A c/o THE SOFTWARE COMPANY
**Name_Out:** Doe John A
**Name2_Out:** The Software Company

**COMPANY** **Use when company names also contain name connectors.**
Use "COMPANY" style to avoid confusion with company names that also contain Name Connectors. When [NameConnectors] such as "&" are specified in the user-defined "NetGender.ref" file, certain company names such as Dewey Cheatum & Howell can sometimes be confused with a compound individual name such as John & Mary Smith. If this is the case, use the COMPANY style first to identify the field as a company.

If an exact match on a company name is found in the "NetGender.ref" file or certain company keywords such as "ABC" are found in the system gender table, the Gender property is returned as "C" (company) and the Name_Quality property is returned as "High". A Name_Quality of "Medium" is returned when certain keywords such as "CLUB" are found in the system gender table indicating that this could possibly be either a company or individual.

## Variable_Default, Variable_Default2

**Syntax:** Variable_Default = StringLiteral
Variable_Default2 = StringLiteral

**Description:**
Set this property to "FML" (natural order) or "LFM" (inverse order) to indicate a default output format. When ambiguous name components make it impossible to determine the name format, NetGender will default to this setting. This setting is only active when the Name_Style is set to "Variable". **Default is "FML".**

**Note:**
As an alternative to the "VARIABLE" style, another very successful approach has been to try several different styles checking the Name_Quality and Gender properties after each parse. Start by calling the "Parse" method using "LFM" then "FML" styles. Next, try the "VARIABLE" style and then try each other style in order down the line. Lastly, try "SPLIT" style and then the default "LFM", "FML" and then "VARIABLE" again. After each parse check the Name_Quality and Gender properties and exit at the first sign of a good parse.

---

## Company_Check

**Syntax:** Company_Check = Boolean (True/False)

**Description:**
Set this property to Boolean (True/False) to indicate if an attempt should be made to identify the Name_In string as a company. Set Company_Check to False if your data has no company names. **Default is "False".**

---

## Name_Variant_Check (new v4.4+)

*(replaces obsolete "Nickname_Check" property)*

**Syntax:** Name_Variant_Check = Boolean (True/False)

**Description:**
Set this property to Boolean (True/False) to indicate if an attempt should be made to identify common name variants associated with Name and Name2. Set Name_Variant_Check to True if you want name variants returned in Name_Variants and Name2_Variants. **Default is "False".**

## Gender_Confidence

**Syntax:** Gender_Confidence = Integer

**Description:**
Set this property to a value between 51% and 100% representing the cutoff point below which a gender is considered neutral. Each name in the System Gender Table is encoded with a percentage from 51% to 100% based on the proportion of males to females for that particular name. After the name lookup, if a percentage is found that is below the "Gender_Confidence" level, NetGender will return a gender of Neutral for that name. A setting of 51% will force a gender of either Male or Female. **Default is 70%**.

**Neutral Gender Breakdown:**

| Gender_Confidence | # of Neutral | % of Gender Table | |
|---|---|---|---|
| 100% | 7,800 | 7.8% | |
| 95% | 6,400 | 6.4% | |
| 90% | 5,100 | 5.1% | |
| 80% | 3,900 | 3.9% | |
| 70% | 2,400 | 2.4% | Default Setting |
| 60% | 900 | .9% | |
| 51% | 0 | .0% | |

## Capitalization (new in v4.4+)
*(replaces obsolete "Output_Case" property)*

**Syntax:** Capitalization = StringLiteral

**Description:**
Set this property to "Upper", "Lower", "Mixed" or "None" to indicate your capitalization preference for the output name and its components. Use "None" when you want to retain the existing capitalization. **Default is "None".**

## Reference_File_Path, Gender_File_Path (changed in v3.0+)

**Syntax:** Reference_File_Path = String
Gender_File_Path = String

**Description:**
Set this property to the full path and file name of the following system and user files:

**NetGender.ref** is a user-defined file containing the tables for Prefix, Suffix, Filter and Connector identification as well as Gender and Name Variant Overrides.

**NetGender.gnd** is a system file that contains the main Genderization Control Tables.

A standard set of these files is supplied and installed in the NetGender installation folder under the names: "NetGender.ref" and "NetGender.gnd". You can rename and relocate these files to any other folder as long as you provide the full path and file name information in each respective path string. **Default path is first the folder of the invoking application: "AppDomain.CurrentDomain.BaseDirectory" then the NetGender installation folder.**

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

---

## Static_Key_Name (new in v3.0+)

**Syntax:** Static_Key_Name = String

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

---

## Static_Key (new in v3.0+)

**Syntax:** Static_Key = String

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

## Name_Out, Name2_Out (read only)

**Syntax:** String = Name_Out
String = Name2_Out

**Description:**
After invoking the "Parse" method, this property will contain the corrected name string, in natural order (FML), from the Name_In property.

---

## Name_Filtered_Data, Name2_Filtered_Data (read only)

**Syntax:** String = Name_Filtered_Data
String = Name2_Filtered_Data

**Description:**
After invoking the "Parse" method, this property will contain all data that was filtered out before processing according to the "[NameFilter]" section of NetGender.ref file.
*See "Updating User Control Tables" later in this guide for instructions on customizing this file.*

---

## Name_Quality, Name2_Quality (read only) (changed in v3.0+)

**Syntax:** String = Name_Quality
String = Name2_ Quality

**Description:**
After invoking the "Parse" method, this property is set to "Low", "Medium" or "High" to indicate how complete a name appears.

---

## Prefix, Prefix2 (read only)

**Syntax:** String = Prefix
String = Prefix2

**Description:**
After invoking the "Parse" method, this property is set to the name Prefix component of Name_In. Values will be valid prefixes (Mr, Mrs, etc.) or blank.

---

## First, First2 (read only)

**Syntax:** String = First
String = First2

**Description:**
After invoking the "Parse" method, this property is set to the First Name component of Name_In. Value will be an individual's first name or blank.

## Middle, Middle2 (read only)

**Syntax:** String = Middle
String = Middle 2

**Description:**
After invoking the "Parse" method, this property is set to the Middle Name component of Name_In. Value will be an individual's middle name or blank.

---

## Last, Last2 (read only)

**Syntax:** String = Last
String = Last2

**Description:**
After invoking the "Parse" method, this property is set to the Last Name component of Name_In. Value will be an individual's last name or blank.

---

## Suffix, Suffix2 (read only)

**Syntax:** String = Suffix
String = Suffix2

**Description:**
After invoking the "Parse" method, this property is set to the Suffix component of Name_In. Values will be valid suffixes (Jr, Sr, Esq, etc.) or blank.

---

## Company, Company2 (read only)

**Syntax:** String = Company
String = Company2

**Description:**
After invoking the "Parse" method, this property is set to the Company component of Name_In. Values will be a company name or blank.

---

## Gender, Gender2 (read only)

**Syntax:** String = Gender
String = Gender2

**Description:**
After invoking the "Parse" method, this property is set to the Gender indicated by the "First" property. Values will be "M" (male), "F" (female), "N" (neutral), "C" (company) or "U" (unknown). *See "Gender_Confidence" property.*

## Gender_Percentage, Gender2_Percentage (read only) (new v4.4+)

**Syntax:** Integer = Gender_Percentage
Integer = Gender2_Percentage

**Description:**
After invoking the "Parse" method, this property is set to the statistical gender percentage. Each name in the System Gender Table is encoded with a percentage from 51% to 100% based on the proportion of males to females for that particular name. For example, the name "Chris" returns a gender of "M" and a Gender_Percentage of 89%. This indicates that 89% of the persons with the name "Chris" are statistically male and 11% are female. *See "Gender_Confidence" and "Gender" properties.*

**Note:** This property is only valid when the Gender property returns a value of "M" or "F".

---

## Name_Variants, Name2_Variants (read only) (new v4.4+)
*(replaces obsolete "Nicknames" property)*

**Syntax:** String = Name_Variants
String = Name2_Variants

**Description:**
After invoking the "Parse" method, these properties are set to a list of common name variants associated with the names returned in First and First2. Values will be a list of one or more common name variants.

**Note:** Name variants are only returned when Name_Variant_Check property is set to True and the Gender property returns a value of "M" or "F".

---

### Accessing Individual Name Variants - VB Sample Code

```
Dim i As Integer, Name As String, Gender As String

cbNameVariants.Items.Clear()

For i = 1 To Len(NetGender.Name_Variants) Step 17
    Name = Mid(NetGender.Name_Variants, i, 16)
    Gender = Mid(NetGender.Name_Variants, i + 16, 1)
    cbNameVariants.Items.Add(Name & " " & Gender)
Next i
```

## Return_Code (read only) (new in v3.0+)

**Syntax:** String = Return_Code

**Description:**
After invoking the "Parse" method, this property is set to blank upon successful completion. Most exceptions occur on the first invocation. The most common ones are listed below. *This property should be examined on each return from NetGender.*

**Common Return Codes:**

| | |
|---|---|
| **G00** | Gender File Wrong Version |
| **G35** | Gender File Not Found *(see "Gender_File_Path" property)* |
| **R35** | Reference File Not Found *(see "Reference_File_Path" property)* |
| **S00** | Unrecognized Name Style *(see "Name_Style property)* |
| **T00** | Prefix/Suffix Table Limit Reached (1,024) |
| **T01** | Name Variant Override Table Limit Reached (256) |
| **T02** | Gender Override Table Limit Reached (1,024) |
| **T03** | Spelling Override Table Limit Reached (1,024) |
| **L00** | Evaluation Period Expired |
| **L01** | Static Key Validation Failed *(see "Static_Key" property)* |
| **L50 - L69** | Evaluation License Error |

## Clear

**Syntax:** NetGender.Clear

**Description:**
When this method is invoked, all properties are cleared with the exception of Static_Key, Static_Key_Name and Reference_File_Path.

---

## Parse

**Syntax:** NetGender.Parse

**Description:**
When this method is invoked, each element of the Name_In property will be inspected. Multiple names are separated, each name is gender coded and placed into the Name_Out property. Each individual element of the Name_In property will be placed into the appropriate name component property. And, if the Name_Variant_Check property is set to True, a list of common name variants is returned in Name_Variants and Name2_Variants. 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

**NetGender.ref** is a file containing the name standardization control tables. It is located by default in the "NetGender" installation folder. Use Notepad or a similar text editor to edit the contents. Detailed information on the format of the entries is contained within the file. This file can also be relocated. *See "Reference_File_Path" property*.

NetGender allows you to specify which Prefixes and Suffixes are to be recognized as well as your preferred abbreviations.

---

**Extensive tables are included. Below are a few examples:**

**[NamePrefix]**
M&M Mr. & Mrs.
M/M Mr. & Mrs.

**[NameSuffix]**
PHD Ph.D.
MANAGER Mgr.

**[NameConnector]**
&
C/O
GUARDIAN OF

**[NameFilter]**
ETAL
TRUST

**[CompanyOverride]**
C TED'S SHEDS

**[GenderOverride]**
M BILLY-JOE
F BOBBY-SUE
L FRUMP

**[NameVariantOverride]**
WOODY WOODIE WOODROW WOODFORD

**[SpellingOverride]**
diGenova
The UPS Store

### Deploying Your Applications

Be sure to include the following in your deployment package:

**NetGender.dll –** usually placed in the application folder or Global Assembly Cache (GAC)
**NetGender.ref –** usually placed in the application folder*
**NetGender.gnd –** usually placed in the application folder*

* "NetGender.ref" and "NetGender.gnd" reference files can be placed anywhere on the target machine as long as the full path to them is specified in the "Reference_File_Path" and "Gender_File_Path" properties.

---

In addition to the above, there is a common runtime that can be placed in the application folder or the Global Assembly Cache (GAC) of the target machine.

**Fujitsu.COBOL.dll**

---

### Evaluation License

The evaluation license is valid for a period of 30 days or up to 1,000 calls.

Sales@SoftwareCompany.com
Support@SoftwareCompany.com