GRC Tools™ version 5.0
DLL COM component manual
Version 2006-3-1

GRC Database Information
Postbus 15213
1001 ME AMSTERDAM
The Netherlands
graham@grcdi.nl
http://www.grcdi.nl

End-User Licence Agreement & Disclaimer

You are granted a single-user licence to use **GRC Tools™**.

The structure and organization of this software are the property of GRC Database Information. You may not make, have made or permit to be made copies of this software, the documentation or any portions thereof except a sole copy for backup purposes. You agree not to modify, adapt, translate, reverse engineer, de-compile, disassemble or create derivative works based on this software

The lookup tables delivered with this software and any delivered in any later versions of this software or delivered separately to later versions of this software remain the property of GRC Database Information. These lookup tables may not be altered in any way. These tables may not be copied, printed or reproduced in any way except through the use of such options within **GRC Tools™**, nor may they be used or incorporated in any software products or in any other media external to this program.

GRC Database Information retains title, intellectual rights and ownership of this software, the media upon which it is recorded, and all subsequent copies of the software, regardless of the form or media or whichever form the original and other copies may exist.

This program and it associated tables and documentation are provided as is. GRC Database Information makes no warranty that **GRC Tools™** will meet your requirements and that the operation of the software will be uninterrupted or error-free. No warrantees or guarantees of any kind are offered. GRC Database Information is not responsible for any problems or damage caused by the software that may result from its use or misuse. This includes, but is not limited to, computer hardware, computer software, operating systems and any computer- or computing-accessories. You agree to hold GRC Database Information and/or any persons associated with the creation of **GRC Tools™** harmless for any problems arising from the use of the software.

About GRC Tools™

Many companies now have address databases containing addresses from many different countries, and many of these are increasingly being used for cross-border mailings. Up to now, however, there have been problems in using and outputting this data given the differences in language, format and expectations within and between these different countries.

GRC Tools™ is designed to resolve many of these problems. Based on many hundreds of man-hours of work with, and analysis of, international databases, this software is a painless way of assuring accuracy and consistency in address databases for different countries.

GRC Tools™ works on two premises - that data should be **consistent**, and that data should be **accurate**. Though address formats differ to such an extent between countries that consistency throughout a database are said to be achievable only at the expense of accuracy, **GRC Tools™** ensures consistency and accuracy on a country and/or language region level.

Achieving **consistency** on a country/language level enables you as a database manager to make your data more accessible (easier to find) as you know the format in which the data will be stored. It enables macro-changes to data such as telephone numbers to be made without excess hassle and, very importantly, it will dramatically increase your ability to locate and weed out duplicates, thus saving potentially enormous amounts of money in printing and mailing costs, and increasing the productivity of the database.

By working on a country/language level, consistency need no longer be achieved at the expense of **accuracy**. Nothing is more irritating to the person to whom you are mailing than being addressed in the wrong language, or being addressed in the wrong way. These are highly emotional points for many people and can dramatically reduce the value of any mailing that you do. Furthermore, format the address in the wrong way, and the mailing may never reach its intended recipient and be returned as undeliverable.

GRC Tools™ ensures consistency and accuracy without losing flexibility.

GRC Tools™ is a set of modules which gather data from lookup tables and change your data file according to what it finds there. These lookup tables have been formulated, based on a great deal of research, to make the data accurate on the basis of country or language area.

! **You are strongly advised to read the section on each module before running it to understand fully what it will do to your data.**

GRC Tools™ has been thoroughly tested. However, as a highly flexible program unexpected problems can always occur. As needs and requirements differ greatly between users, and modules may be run in combinations not expected by the program, there is a small chance that changes may be made that you do not expect or do not want.

! **It is highly advisable to back-up any data files over which you wish to run GRC Tools™. Remember that GRC Tools™ changes data. The author disclaims any responsibility for damaged data caused by the use or misuse of this program**.

Should you have any comments or suggestions for improvements to **GRC Tools™**, we would be delighted to hear of them.

! Countries have been chosen for inclusion in **GRC Tools™** on a pragmatic basis, and on postal and address

requirements. The list excludes countries with a claim to independence or which are *ipso facto* independent and includes some dependent territories. The parameters for inclusion have been that in terms of address systems a clear distinction can be made in some respect for the country or territory concerned.

Technical Requirements

GRC Tools™ requires:

- An IBM compatible computer with 486 50 MHz processor or higher
- Mouse
- 10 MB RAM
- DOS version 3.1 or higher
- Microsoft Windows version 95 or higher, or Microsoft Windows NT 3.51 or higher.
- 50 MB of disk space

For **GRC Tools™** , for each data string being processed, you must know:

- a country code

Installation

The **GRCTools™** DLL may be installed in any directory. However, the installation directory MUST have a daughter directory call *lu* which contains the lookup table data.

The files XICRCORE.DLL and MSVCRT.DLL should be written to your c:\windows\system directory. MSVCRT.DLL is a C++ runtime file, so if a newer version of this file already exists in your c:\windows \system directory, you do not need to replace it.

Summary of the modules

GRC Tools™ works on standardising names and addresses on a modular basis. Below is a summary of the modules, in the order in which they should be run for optimal quality, and the type of data upon which they work:

Data processing

Process	*String contents*
Split string	Any alpha-numeric string
Trim leading spaces	Any alpha-numeric string
Trim final string	Any alpha-numeric string
Remove non-numeric characters	Any alpha-numeric string
Remove punctuation	Any alpha-numeric string
Remove double spaces	Any alpha-numeric string
Remove postal code country code	Postal code
Parse thoroughfare type	Any alpha-numeric string
Format postal code	Postal code
Assess postal code validity	Postal code
Write corrected postal codes	Postal code
Locate postal codes	Postal code, any alpha-numeric string likely to contain a stray postal code
Locate postal codes (incl. format)	Postal code, any alpha-numeric string likely to contain a stray postal code
Assign language regions	Uses postal code and street address data
Remove accents	Any alpha-numeric string
Remove quotation marks	Any alpha-numeric string
Add apostrophes (French)	Any alpha-numeric string
To upper case	Any alpha-numeric string
To mixed case	Any alpha-numeric string
To mixed case - address	Street address data
To mixed case - other	Any alpha-numeric string, not street address
Move articles	Company name
Standardize “and” strings	Company name
Standardize abbreviations	Company name
Standardise abbreviations - names	Personal name
Standardize company types	Company name
Standardize department strings	Department data
Parse post office box numbers	Any alpha-numeric string which could contain stray postbox numbers, usually street address fields
Standardize thoroughfare strings	Street address data
Move/parse building numbers	Street address data
Parse house number suffix	House number data
Add/remove commas	Street address data
House number/letter format	House number/street address data
Parse/standardize sorting code	Any alpha-numeric string which could contain stray sorting codes, usually postal town fields

Parse/Standardise place names	Any alpha-numeric address string likely to contain a settlement name
Parse/assign provinces	Any alpha-numeric string which could contain stray province names
Assign provinces/regions	Uses postal code data
Assign regions	Uses postal code data
Parse/standardize forms of address	Any alpha-numeric string which could contain stray forms of address data, usually personal name fields
Standardize job titles	Job title

Country codes

The **GRCTools** DLL requires a country code to be sent to its modules along with the string to be processed. This country code must be the three-letter internal **GRCTools** code, as listed below in the final column.

Country name	*ISO 2*	*ISO 3*	***GRCTools country code***
Afganistan	AF	AFG	**AFG**
Albania	AL	ALB	**ALB**
Algeria	DZ	DZA	**ALG**
American Samoa	AS	ASM	**AMS**
Andorra	AD	AND	**AND**
Angola	AO	AGO	**ANG**
Anguilla	AI	AIA	**ANU**
Antigua & Barbuda	AG	ATG	**ANT**
Argentina	AR	ARG	**ARG**
Armenia	AM	ARM	**ARM**
Aruba	AW	ABW	**ARU**
Australia	AU	AUS	**AST**
Austria	AT	AUT	**AUS**
Azerbaijan	AZ	AZE	**AZE**
Bahamas	BS	BHS	**BAH**
Bahrain	BH	BHR	**BAR**
Bangladesh	BD	BGD	**BAN**
Barbados	BB	BRB	**BAB**
Belarus	BY	BLR	**BEO**
Belgium	BE	BEL	**BEL**
Belize	BZ	BLZ	**BEI**
Benin	BJ	BEN	**BEN**
Bermuda	BM	BMU	**BER**
Bhutan	BT	BTN	**BHU**
Bolivia	BO	BOL	**BOL**
Bosnia-Herzegovina	BA	BIH	**BOS**
Botswana	BW	BWA	**BOT**
Brazil	BR	BRA	**BRA**
British Virgin Islands	VG	VGB	**BVI**
Brunei Darussalam	BN	BRN	**BRU**
Bulgaria	BG	BGR	**BUL**
Burkina Faso	BF	BFA	**BUK**
Burma	MM	MMR	**BUR**
Burundi	BI	BDI	**BUU**
Cambodia	KH	KHM	**CAM**
Cameroon	CM	CMR	**CAE**
Canada	CA	CAN	**CAN**
Cape Verde Islands	CV	CPV	**CAP**
Cayman Islands	KY	CYM	**CAY**
Central African Republic	CF	CAF	**CEN**
Chad	TD	TCD	**CHA**
Chile	CL	CHL	**CHI**
China	CN	CHN	**CHN**
Christmas Island	CX	CXR	**CHR**
Cocos (Keeling) Islands	CC	CCK	**COC**
Colombia	CO	COL	**CLO**
Comoros	KM	COM	**COM**
Congo (Brazzaville)	CG	COG	**CNG**

Congo (Kinshasa)	ZR	ZAR	**ZAI**
Cook Islands	CK	COK	**COO**
Costa Rica	CR	CRI	**COS**
Croatia	HR	HRV	**CRO**
Cuba	CU	CUB	**CUB**
Cyprus	CY	CYP	**CYP**
Czech Republic	CZ	CZE	**CZE**
Denmark	DK	DNK	**DEN**
Djibouti	DJ	DJI	**DJI**
Dominica	DM	DMA	**DOI**
Dominican Republic	DO	DOM	**DOM**
East Timor	TL	TLS	**ETI**
Ecuador	EC	ECU	**ECU**
Egypt	EG	EGY	**EGY**
El Salvador	SV	SLV	**ELS**
Equatorial Guinea	GQ	GNQ	**EQA**
Eritrea	ER	ERI	**ERI**
Estonia	EE	EST	**EST**
Ethiopia	ET	ETH	**ETH**
Faeroe Islands	FO	FRO	**FAE**
Falkland Islands	FK	FLK	**FAL**
Fiji	FJ	FJI	**FIJ**
Finland	FI	FIN	**FIN**
France	FR	FRA	**FRA**
French Guiana	GF	GUF	**FGU**
French Polynesia	PF	PYF	**FPO**
Gabon	GA	GAB	**GAB**
Gambia, The	GM	GMB	**GAM**
Georgia	GE	GEO	**GEO**
Germany	DE	DEU	**GER**
Ghana	GH	GHA	**GHA**
Gibralter	GI	GIB	**GIB**
Greece	GR	GRC	**GRE**
Greenland	GL	GRL	**GRN**
Grenada	GD	GRD	**GRA**
Guadeloupe	GP	GLP	**GUD**
Guam	GU	GUM	**GUM**
Guatemala	GT	GTM	**GUA**
Guernsey	GG		**GUE**
Guinea	GN	GIN	**GUI**
Guinea-Bissau	GW	GNB	**GUB**
Guyana	GY	GUY	**GUY**
Haiti	HT	HTI	**HAI**
Holy See	VA	VAT	**VAT**
Honduras	HN	HND	**HON**
Hong Kong	HK	HKG	**HOK**
Hungary	HU	HUN	**HUN**
Iceland	IS	ISL	**ICE**
India	IN	IND	**IND**
Indonesia	ID	IDN	**INO**
Iran	IR	IRN	**IRA**
Iraq	IQ	IRQ	**IRQ**
Ireland	IE	IRL	**IRE**
Isle of Man	IM		**ISL**
Israel	IL	ISR	**ISR**
Italy	IT	ITA	**ITA**
Ivory Coast	CI	CIV	**IVO**

Jamaica	JM	JAM	**JAM**
Japan	JP	JPN	**JAP**
Jersey	JE		**JER**
Jordan	JO	JOR	**JOR**
Kazakhstan	KZ	KAZ	**KAZ**
Kenya	KE	KEN	**KEN**
Kiribati	KI	KIR	**KII**
Kuwait	KW	KWT	**KUW**
Kyrgyzstan	KG	KGZ	**KIR**
Laos	LA	LAO	**LAO**
Latvia	LV	LVA	**LAT**
Lebanon	LB	LBN	**LEB**
Lesotho	LS	LSO	**LES**
Liberia	LR	LBR	**LIR**
Libya	LY	LBY	**LIB**
Liechtenstein	LI	LIE	**LIE**
Lithuania	LT	LTU	**LIT**
Luxembourg	LU	LUX	**LUX**
Macau	MO	MAC	**MCA**
Macedonia	MK	MKD	**MCE**
Madagascar	MG	MDG	**MAD**
Malawi	MW	MWI	**MAW**
Malaysia	MY	MYS	**MAA**
Maldives	MV	MDV	**MAV**
Mali	ML	MLI	**MAI**
Malta	MT	MLT	**MAL**
Marshall Islands	MH	MHL	**MAR**
Martinique	MQ	MTQ	**MAN**
Mauritania	MR	MRT	**MAU**
Mauritius	MU	MUS	**MAT**
Mayotte	YT	MYT	**MAY**
Mexico	MX	MEX	**MEX**
Micronesia	FM	FSM	**MIC**
Moldova	MD	MDA	**MOL**
Monaco	MC	MCO	**MON**
Mongolia	MN	MNG	**MOG**
Montenegro			**MON**
Montserrat	MS	MSR	**MOT**
Morocco	MA	MAR	**MOR**
Mozambique	MZ	MOZ	**MOZ**
Namibia	NA	NAM	**NAM**
Nauru	NR	NRU	**NAU**
Nepal	NP	NPL	**NEP**
Netherlands Antilles	AN	ANT	**NAN**
Netherlands, The	NL	NLD	**NET**
New Caledonia	NC	NCL	**NCA**
New Zealand	NZ	NZL	**NEW**
Nicaragua	NI	NIC	**NIC**
Niger	NE	NER	**NIE**
Nigeria	NG	NGA	**NIG**
Niue	NU	NIU	**NIU**
Norfolk Island	NF	NFK	**NOF**
North Korea	KP	PRK	**NKO**
Northern Mariana Islands	MP	MNP	**NMI**
Norway	NO	NOR	**NOR**
Oman	OM	OMN	**OMA**
Pakistan	PK	PAK	**PAK**

Palau	PW	PLW	**PAL**
Panama	PA	PAN	**PAN**
Papua New Guinea	PG	PNG	**PAP**
Paraguay	PY	PRY	**PAR**
Peru	PE	PER	**PER**
Philippines	PH	PHL	**PHI**
Pitcairn Islands	PN	PCN	**PIT**
Poland	PL	POL	**POL**
Portugal	PT	PRT	**POR**
Puerto Rico	PR	PRI	**PUE**
Qatar	QA	QAT	**QAT**
Romania	RO	ROM	**ROM**
Russia	RU	RUS	**RUS**
Rwanda	RW	RWA	**RWA**
Réunion	RE	REU	**REU**
Saint Christopher & Nevis	KN	KNA	**STC**
Saint Helena	SH	SHN	**STH**
Saint Lucia	LC	LCA	**STL**
Saint Pierre & Miquelon	PM	SPM	**SPM**
Saint Vincent & The Grenadines	VC	VCT	**STV**
Samoa	WS	WSM	**WSM**
San Marino	SM	SMR	**SAN**
Sao Tome & Principe	ST	STP	**SAO**
Saudi Arabia	SA	SAU	**SAU**
Senegal	SN	SEN	**SEN**
Serbia	CS	SCG	**YUG**
Seychelles	SC	SYC	**SEY**
Sierra Leone	SL	SLE	**SIE**
Singapore	SG	SGP	**SIN**
Slovakia	SK	SVK	**SLO**
Slovenia	SI	SVN	**SLV**
Solomon Islands	SB	SLB	**SOL**
Somalia	SO	SOM	**SOM**
Somaliland			**SOA**
South Africa	ZA	ZAF	**SAF**
South Korea	KR	KOR	**SKO**
Spain	ES	ESP	**SPA**
Sri Lanka	LK	LKA	**SRI**
Sudan	SD	SDN	**SUD**
Suriname	SR	SUR	**SUR**
Swaziland	SZ	SWZ	**SWA**
Sweden	SE	SWE	**SWE**
Switzerland	CH	CHE	**SWI**
Syria	SY	SYR	**SYR**
Taiwan	TW	TWN	**TAI**
Tajikistan	TJ	TJK	**TAJ**
Tanzania	TZ	TZA	**TAN**
Thailand	TH	THA	**THA**
Togo	TG	TGO	**TOG**
Tokelau	TK	TKL	**TOK**
Tonga	TO	TON	**TON**
Trinidad & Tobago	TT	TTO	**TRI**
Tunisia	TN	TUN	**TUN**
Turkey	TR	TUR	**TUR**
Turkmenistan	TM	TKM	**TUK**
Turks & Caicos Islands	TC	TCA	**TCI**
Tuvalu	TV	TUV	**TUV**

Uganda	UG	UGA	**UGA**
Ukraine	UA	UKA	**UKR**
United Arab Emirates	AE	ARE	**UAE**
United Kingdom	GB	GBR	**UNI**
United States	US	USA	**USA**
United States Virgin Islands	VI	VIR	**VIR**
Uruguay	UY	URY	**URU**
Uzbekistan	UZ	UZB	**UZB**
Vanuatu	VU	VUT	**VAN**
Venezuela	VE	VEN	**VEN**
Vietnam	VN	VNM	**VIE**
Wallis & Futuna	WF	WLF	**WAL**
Western Sahara	EH	ESH	**WSH**
Yemen	YE	YEM	**YEM**
Zambia	ZM	ZMB	**ZAM**
Zimbabwe	ZW	ZWE	**ZIM**

Countries have been listed in English. For the sake of clarity, please note that:

- *Kampuchea* is listed as *Burma*
- *Khmer* is listed as *Cambodia*
- *Congo* (north of the river Congo) is listed as *Congo - Brazzaville*
- *The Democratic Republic of Congo* (south of the river Congo, ex-Zaire) is listed as *Congo - Kinshasa*

Split string

Purpose: to split a field string at a given sub-string.

This is a general process that can be run on any character field. It splits a field at a specified sub-string. The data before the given string stays in the specified field, the data after is moved to a new field.

Example: **Station House High Street**, split after **House** produces **Station House** and **High Street**

Information required: For each string sent, specify the sub-string at which the split is to be made (include leading and/or trailing spaces, as required; note: casing is taken into account); and whether the split is to be made before or after the given sub-string

Parameters:

Sent string	The string to split
Sub-string	The string at which to split the sent string
Split point	0=split at back of sub-string; 1=split at front of sub-string

Returns

GRCReturnedstring	The sent string without the section split off from it
GRCSplitString	The string split from the sent string

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.splitstr("Station House High Street",' House ',0,
@GRCSplitString)
? GRCReturnedString
? GRCSplitString
```

Returns: Station House
High Street

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.splitstr("Station House High Street",' House ',1,
@GRCSplitString)
? GRCReturnedString
? GRCSplitString
```

Returns: Station
House High Street

Trim leading spaces

Purpose: remove spaces at the front of a string within a field

This is a general process that can be run on any character field. It removes leading spaces from the string within a field.

Example: “ **Station House, High Street”** becomes “**Station House, High Street”**

Parameters:

Sent string	The string from which to remove leading spaces

Returns

GRCReturnedstring	The sent string with leading spaces trimmed off

Example code (Visual Foxpro)

```
close all
clear all
clear
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.trimleft('      - 17 Saint Helens Avenue')
? 'GRCReturnedString='+GRCReturnedString
```

Returns: `- 17 Saint Helens Avenue`

Trim final string

Purpose: to remove a user-defined character or string at the final position(s) of the string within a field, if it occurs there.

This is a general process that can be run on any character field. It removes a user-defined character or string at the final position(s) of the string within a field, if it occurs there.

Example: **Station House High Street /** , remove /, produces **Station House High Street .**

Information required: For each chosen field, specify the character or string to be removed.

Parameters:

Sent string	The string from which to remove leading spaces
String to remove	The string to remove from the sent string

Returns

GRCReturnedstring	The sent string with the given string to remove removed

Example code (Visual Foxpro)

```
close all
clear all
clear
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.trimlast('17 Saint Helens Avenue,',',')
? 'GRCReturnedString='+GRCReturnedString
```

Returns: `17 Saint Helens Avenue`

Remove non-numeric characters

Purpose: to remove non-numeric characters from the user-specified fields.

This is a general process that can be run on any character string intended only to hold numeric data. It should be the first process to be run over address data for those countries where the postal code consists only of numbers, as the postal code is required for **GRC Tools™** in its correct format (without addition code or punctuation) for many other processes.

This process also removes spaces. As spacing may be needed in the postal codes of certain countries (e.g. 999 99 in Sweden) this spacing should be added after this process has been run and before running other **GRC Tools™** processes that require the postal code.

Example: **23.45.67.89** becomes **23456789**

Information required: hyphens (-) are often found in telephone numbers in certain countries like Germany, where they indicate a direct-dialled internal extension number. In other countries it may indicate a number of telephone lines - **33-37** for example, may indicate that lines ending in 33, 34, 35, 35 and 37 belong to this company. **GRC Tools™** gives you the choice of removing or retaining hyphens for each field chosen.

Parameters:

Sent string	The string from which to remove non-numeric characters
Hyphen status	0=retain hyphens; 1=remove hyphens

Returns

GRCReturnedstring	The sent string without non-numeric characters (including or excluding hyphens)

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.nonnum(“(1234)-5678-90”,0)
? GRCReturnedString
```

Returns: 1234-5678-90

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.nonnum(“(1234)-5678-90”,1)
? GRCReturnedString
```

Returns: 1234567890

Remove punctuation

Purpose: to remove punctuation from the user-specified fields.

This is a general process that can be run on any character string. It removes punctuation from the field.

Example: **I.C.B. Watford - Bakers Ltd.** becomes **I C B Watford Bakers Ltd**

Parameters:

Sent string	The string from which to remove punctuation marks
Country code	GRCTools country code

Returns

GRCReturnedstring	The sent string without punctuation marks

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.punc(“L-o,n.d*o:n;--)())”,'UNI')
? GRCReturnedString
```

Returns: London

Remove double spaces

Purpose: to remove double spaces from the user-specified fields.

This is a general process that can be run on any character field. It removes double spaces from the field.

Example: **John W. Smith** becomes **John W. Smith**

Parameters:

Sent string	The string from which to remove double spaces

Returns

GRCReturnedstring	The sent string without double spaces

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.dspace(“London     NW1   4WW”)
? GRCReturnedString
```

Returns:
GRCReturnedString: London NW1 4WW

Locate postal codes
Locate postal codes (incl. format)

Purpose*:** to search for postal codes in the sent string. The process ***Locate postal codes searches for postal codes that are correctly formatted, whilst ***Locate postal codes (incl. format)*** searches also for postal codes that may be incorrectly formatted (where a postal code would normally contain a space or a hyphen). Thus, the former process for The United Kingdom will locate the postal code *TW1 1AA*, whilst the latter will locate *TW1 1AA and TW11AA*.

Where a postal code is preceded by a postal country code (e.g. **CH**-1726) then the postal country code is also parsed with the postal code. This country code can be stripped from the postal code using the process *Remove postal code country code*. Postal codes followed by certain punctuation marks are also recognised and parsed.

For the **United Kingdom** only, you have the option of allowing abbreviated forms of postal codes. Especially in London, addresses are often given with only the outward (first section) of the postal code, e.g. **SW1** instead of **SW1 2AB**. Though these postal codes are technically incomplete, this option allows you to search also for these abbreviated postal codes.

Parameters:

Sent string	The string from which to parse the postal code
Country code	The GRCTools country code
Allow short codes	For The United Kingdom only, 1=allow short codes, 2=do not allow short postal codes

Returns

GRCReturnedstring	Sent string, unchanged
m.GRCReturnedPC	The parsed postal code
m.GRCReturnedPosn	The position in the sent string where the first character of the found postal code was located
m.GRCNewString	Sent string without the parsed postal code

Example code (Visual Foxpro)

```
m.GRCReturnedPc = ""
m.GRCReturnedPosn=0
m.GRCNewString=""
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.mpc('High Street, SW1  ','UNI',1,@m.GRCReturnedPc,
@m.GRCReturnedPosn, @m.GRCNewString)
```
**!* NOTE: To run "Locate postal codes (incl. format)" run otemp.incpc*
```
? GRCReturnedString
? m.GRCReturnedPc
? m.GRCReturnedPosn
? m.GRCNewString
```

Returns:
```
GRCReturnedString     High Street, SW1
m.GRCReturnedPc       SW1
m.GRCReturnedPosn     14
m.GRCNewString        High Street,
```

Remove postal code country code

Purpose: to remove the postal country sorting code which sometimes precedes a postal code.

This process can be run on characters fields containing postal code data.

Example: **GB-TW1 1AA** becomes **TW1 1AA**
B-1000 becomes **1000**
CH1017 becomes **1017**

Parameters:

Sent string	The string from which to parse the postal code
Country code	The GRCTools country code

Returns

GRCReturnedstring	Sent string without the postal code country code

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.remcod('DK- 1000 COPENHAGEN,'DEN')? GRCReturnedString
```

Returns:
```
GRCReturnedString     1000 COPENHAGEN
```

Parse thoroughfare types

***Purpose*:** to parse strings indicating thoroughfare types and other address type strings (“Street”, “strasse”, “rue”, “house”, “apartment”, “zone” etc.) to user-defined fields.

***Example*: R. de Paris** becomes **rue|de Paris**; **Kölnstr.** becomes **Köln|straße** etc.

***Information required*:** For each string sent, **GRC Tools™** needs to know whether to leave the thoroughfare type unchanged, to write it in its standardised form, or in its fullest form. For example, if the string being processed is *HIGH STR*, then its unchanged form is *STR*, its standardised form is *ST* and its full form is *STREET*.

If you choose to standardise the thoroughfare types or write them in their fullest form, then you may choose the case in which to write the data.

Finally, you need to specify at which end of the string **GRC Tools™** should start looking for the thoroughfare type.

Parameters:

Sent string	The string containing the postal code
Country code	The GRCTools country code
Standardisation form	1=unchanged, 2=standardised, 3=in full
Search end	‘Back’=Back, ‘Front’=Front
Case to return the data	1=Upper, 2=Mixed

Returns

GRCReturnedstring	Unchanged string as sent for processing
GRCBeforeString	String before found thoroughfare string
GRCToString	Found thoroughfare string
GRCAfterString	String found after thoroughfare string

Example code (Visual Foxpro)

```
GRCBeforeString=""
GRCAfterString=""
GRCToString=""
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.parsetfar("Berliner Str",'GER',2,'Back',2,
@m.GRCBeforeString, @m.GRCToString, @m.GRCAfterString)
?"Unchanged string = "+ GRCReturnedString
?"Before thoroughfare string string = "+ GRCBeforeString
?"Thoroughfare string = "+ GRCToString
?"After thoroughfare string = "+ GRCAfterString
```

Returns:

```
GRCReturnedString     Berliner Str
GRCBeforeString       Berliner
GRCToString           Straße
GRCAfterString
```

```
GRCBeforeString=""
GRCAfterString=""
GRCToString=""
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.parsetfar("rue de Paris",'FRA',2,'Front',1,
@m.GRCBeforeString, @m.GRCToString, @m.GRCAfterString)
?"Unchanged string = "+ GRCReturnedString
```

```
?"Before thoroughfare string string = "+ GRCBeforeString
?"Thoroughfare string = "+ GRCToString
?"After thoroughfare string = "+ GRCAfterString
```

Returns:

```
GRCReturnedString     rue de Paris
GRCBeforeString
GRCToString           RUE
GRCAfterString        de Paris
```

Format postal code

Purpose: to add spaces or hyphens to postal codes which require them.

This process can be run on characters fields containing postal code data.

Example: **10100** becomes **10 100**
TW11AA becomes **TW1 1AA**

Parameters:

Sent string	The string containing the postal code
Country code	The GRCTools country code

Returns

GRCReturnedstring	Sent string with a postal code in corrected format

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.fpc('1018VV','NET')
? GRCReturnedString
```

Returns:
```
GRCReturnedString     1018 VV
```

Assess postal code validity

***Purpose*:** to check the validity of postal codes by analysing them for correct length, correct character type, the presence of invalid characters and the presence of foreign postal codes.

This process requires the postal code to be written in the sent string in the correct format and without the addition of country codes such as **F-75000** or **GB-W1A 4ZA**. I.e. these postal codes need to be written **75000** and **W1A 4ZA.**

For the **United Kingdom** only, you have the option of allowing abbreviated forms of postal codes. In London particularly, addresses are often given with only the outward (first section) of the postal code, e.g. **SW1** instead of **SW1 2AB**. Though these postal codes are technically incomplete, this option allows you not to mark these abbreviated postal codes as incorrect.

Parameters:

Sent string	The string containing the postal code
Country code	The GRCTools country code
UK short postal codes	1=allow short postal codes for the UK; 2=do not allow short postal codes for the UK
Returned message	Returned message (empty if postal code is correct)

Returns

GRCReturnedString	Sent string, unchanged
m.GRCReturnedMessage	Message containing errors in postal code

Example code (Visual Foxpro)

```
m.GRCReturnedMessage= ""
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.pc('1000VV',NET,2,@m.GRCReturnedMessage)
? GRCReturnedString
? m.GRCReturnedMessage
```

Returns:

```
GRCReturnedString     1000VV
m.GRCReturnedMessage  Postal code must be 7 digits long; Digit 5 must be a SPACE;
Digit 7 must be a CAPITAL LETTER
```

Assign Language Regions

***Purpose*:** To assign a language to each address within the database file specified in multi-lingual countries.

Although it is often sufficient to know the country in which an address is located in order for **GRC Tools™** to run its processes, in certain countries where more than one language is spoken, there is a clash between certain aspects of addressing between each language, and therefore **GRC Tools™** needs to work on the basis of language areas. This process assigns language areas. Please note that this language area assignment neither relates to the language spoken by the person/people living or working at the address, nor the language that you might want to send information in. It specifies either the predominant language of the area where the address is situated, or, for bilingual areas or for addresses without a postal code, the language in which the address is written.

***Information required*:** For each country, this process needs to know:

- the field to which the language code is to be written.
- the fields containing the street address (maximum two - one may be left empty).
- the field containing the postal code

Parameters:

Postal code	The postal code
Street address1	The first street address string
Street address 2	A second street address string
Country code	The GRCTools country code
Returned language code	3-letter lower case ISO 639-2 language code. eng=English, fra=French, deu=German, ita=Italian, nld=Dutch, bil=Bilingual

Returns

GRCLanguageRegion	3-letter lower case ISO 639-2 language code. eng=English, fra=French, deu=German, ita=Italian, nld=Dutch

Example code (Visual Foxpro)

```
GRCLanguageRegion=''
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.lregions('1000','Parijs straat','','BEL',@GRCLanguageRegion)
?GRCLanguageRegion
```

Returns:

```
GRCLanguageRegion     nld
```

Remove Quotation Marks

***Purpose*:** to remove paired quotation marks from the user-specified field.

***Example*: "Dun Roamin"** becomes **Dun Roamin**

This process searches for, and removes, double quotation marks.

Single quotation marks, found in addresses as apostrophes, are not removed.

If more than two double quotation marks are located within the same field, the process will not make changes to the field but will write the data to a file.

Parameters:

Sent string	The string to be processed

Returns

GRCReturnedString	The sent string without the quotation marks.
GRCAttentionRequired	0=String is OK; 1=quotation marks couldn't be automatically removed – attention is required

Example code (Visual Foxpro)

```
GRCReturnedString=''
GRCAttentionRequired=0
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.quotat(["Dunroamin""], @GRCAttentionRequired)
? GRCReturnedString
? GRCAttentionRequired
```

Returns:

```
GRCReturnedString     "Dunroamin""
GRCAttentionRequired  1
```

Example code (Visual Foxpro)

```
GRCReturnedString=''
GRCAttentionRequired=0
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.quotat(["Dunroamin'"], @GRCAttentionRequired)
? GRCReturnedString
? GRCAttentionRequired
```

Returns:

```
GRCReturnedString     Dunroamin
GRCAttentionRequired  0
```

Add Apostrophes (French)

***Purpose*:** To add omitted apostrophes into data strings.

This process searches for stand-alone occurrences of the letters ‘L’ and ‘D’ (in both upper- and mixed-cases) and, if followed by a vowel, ‘H’ or ‘Y’, an apostrophe is added.

Example: **L HUYSMAN D AMIENS** becomes **L’HUYSMAN D’AMIENS**

! Although this process cannot differentiate between an aspirated and unaspirated aitch (‘H’), the aitches found preceded by a stand-alone occurrence of a ‘D’ or ‘L’ are assumed to be unaspirated and therefore requiring the apostrophe. Note also that, except where occurring in the first 2 places of the string, the ‘L’ or ‘D’ preceded by another standalone character are assumed to be part of an abbreviation and are not given apostrophes. Thus **S A R L AMIENS** will not become **S A R L’AMIENS**.

Parameters:

Sent string	The string to be processed

Returns

GRCReturnedString	The sent string with added apostrophes.

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.apost("L avenue de l hotel")
?GRCReturnedString
```

Returns:
```
GRCReturnedString     L’avenue de l’hotel
```

To upper case

***Purpose*:** to correctly translate the contents of the user-defined field from mixed- or lower-case to upper case.

This process checks also each accented character in order to assign a correct upper-case equivalent, depending on the country or language concerned.

***Example*:** **Rue du Récif** in France becomes **RUE DU RECIF**
Kölnstraße in Germany becomes **KÖLNSTRASSE**

There is a clash in the casing rules for Belgium between Dutch/German and French; and in Switzerland between German, French and Italian. For this reason, this process handles each language area differently. It is therefore necessary to have already written the language areas to the database table using the procedure ***Assign language region*** (unless the language region is already coded). When running this process for Belgium or Switzerland, you are required to specify the language region.

Parameters:

Sent string	The string to be processed
Country code	Country code
Language region	ISO 639-2 language code. eng=English, fra=French, deu=German, ita=Italian, nld=Dutch, bil=Bilingual

Returns

GRCReturnedString	The sent string in upper case

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.ucase("gérant ",'SWI','fra')
?GRCReturnedString
```

Returns:
```
GRCReturnedString     GÉRANT
```

To mixed case
To mixed case - addresses
To mixed case - other

***Purpose*:** to correctly translate the contents of the user-defined field(s) from upper case to mixed (upper and lower) case.

The process ***To mixed case*** can be used for all fields. However, certain countries require two processes. These countries are those where there are *single character* prepositions within the language. For example, in French, a freestanding *A* could be an abbreviation but might also be the preposition *A* meaning *TO*. As the computer cannot distinguish these differences in usage, the process ***To mixed case - cddresses*** is used for data where abbreviations are unlikely to occur, such as the street address, and this assumes that free-standing characters such as *A* are prepositions and puts them into lower case. The process ***To mixed case - other*** assumes that free-standing characters such as *A* are abbreviations and leaves them in upper case.

! Running this process over street addresses for countries where are single letter indicates a proposition, such as **a** in France, will result in letters which are part of the house number being put into lower case. Thus in France, **21 A** would become **21 a** whilst **21 B** would remain unchanged. If this is undesirable, run the process ***To mixed case - other*** rather than ***To mixed case - addresses*** over the street address.

Abbreviations which are written with neither spaces nor commas in the field(s) defined cannot be distinguished by the process from other words, and will also be made into mixed case, i.e.

BBC Transmission Of Programmes Plc will become
Bbc Transmission of Programmes PLC

This does not happen if abbreviations are consistently written with full stops or spaces, i.e. *B.B.C.* or *B B C*.

If the data in the field to be processed is already in mixed case, and you wish only to correct the casing, you can choose the *Respect original casing* option.

Choosing this option prevents the process from lower-casing the original data during processing, so

BBC Transmission Of Programmes Plc will become
BBC Transmission of Programmes PLC

This option can <u>only</u> be used usefully if the data being processed is already in mixed case.

There is a clash in the casing rules for Belgium between Dutch/German and French; in Canada between French and English; and in Switzerland between German, French and Italian. For this reason, this process handles each language area differently. It is therefore necessary to have already assigned the language areas using the procedure ***Assign language region*** (or that the language areas already coded). When running a mixed-case process for Belgium, Canada or Switzerland, you are required to specify the language region.

Parameters:

Sent string	The string to be processed
Country code	Country code
Respect original casing	1=original casing is respected (i.e. NOT made first into lower case), 0= original casing ignored
Language region	ISO 639-2 language code. eng=English, fra=French, deu=German, ita=Italian, nld=Dutch, bil=Bilingual

Returns

GRCReturnedString	The sent string in mixed case

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.lcase("Überkoeln IM GMBH ",'SWI',0,'deu')
```
! Note: oTemp.lcase runs To Mixed Case and To Mixed Case Address. OTemp.lcase2 runs To Mixed Case Other
```
? GRCReturnedString
```

Returns:
```
GRCReturnedString     Überkoeln im GmbH
```

Move Articles

***Purpose*:** to move articles (“the”, “a”, “an”) from one end of a string to the other, whilst standardising string format.

***Example*: Baker’s Dozen, The** or **Baker’s Dozen (The)** become **The Baker’s Dozen**
or **The Baker’s Dozen** becomes **Baker’s Dozen, The** or **Baker’s Dozen (The)**

- For moving of articles from the *FRONT* to the *BACK* of a string, this process is unable to distinguish between single-lettered articles (such as “a” in English) and abbreviations/prepositions/conjunctions. These occurrences are therefore not changed by this process. For example, **A Baker’s Dozen** remains unchanged. However, these articles can be moved from the *BACK* to the *FRONT* of the address provided they are preceded by a comma or are in brackets. Thus **Baker’s Dozen (A)** will become **A Baker’s Dozen**.
- This process will only move the article from *BACK* to *FRONT* if it is the last string found on the line (to prevent articles which are part of sub-clauses within a field, such as **Ground Floor, The Smithson Building** from being affected).
- When an article is found by the process and a move has been made, the process looks no further - it is assumed that only one article per field will be found.
- Articles moved are put into a standard format, as also are those already in place. I.e., if you are moving the article to the back, with a comma, those articles at the back between brackets will be altered to be at the back with a comma.

Parameters:

Sent string	The string to be processed
Country code	Country code
End of string to move article FROM	1=from back to front; 0=from front to back
Case to write data	1=UPPER CASE; 0=Mixed case
Include comma	If moving from front to back, 1=include a comma; 0 =include brackets

Returns

GRCReturnedString	The sent string with the moved article

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.article("Big Book Shop, The  ",'UNI',1,0,0)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     The Big Book Shop
```

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.article("The Big Book Shop  ",'UNI',0,0,0)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Big Book Shop (The)
```

Standardize “and” strings

***Purpose*:** To standardize the different forms of the word AND (and symbol equivalents) in the local language.

***Example*: C. and A., C. + A.** etc. becomes **C. & A.**

Care must be taken when using this process for countries where the word “and” consists of a single character, such as Italy (“e”) and Spain (“y”). Whereas in most countries the lookup tables have been set to standardize the word “and” into the ampersand symbol (“&”), this is not the case for these countries as it is not possible to distinguish between the different uses of this single letter (as the word “and”, as an abbreviation, etc.). For these countries therefore, the ampersand is standardised into the letter rather than the other way around.

For Switzerland there is a clash between Italian- and French/German-speaking areas. For Switzerland, therefore it is necessary to have assigned the language areas using the procedure **Assign Language Region** (or have the language areas already coded).

Parameters:

Sent string	The string to be processed
Country code	Country code
Language region	ISO 639-2 language code. eng=English, fra=French, deu=German, ita=Italian, nld=Dutch, bil=Bilingual
Case to write data	1=UPPER CASE; 0=Mixed case

Returns

GRCReturnedString	The sent string with the standardised article

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.andchng("Schmidt und Braun GmbH",'SWI','ger',1)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Schmidt U. Braun GmbH
```

Standardize abbreviations
Standardize abbreviations - names

***Purpose*:** to standardize the format of abbreviations and acronyms so that they either all have full stops or all have spaces.

***Example*: A B C D** becomes **A.B.C.D.**; **A.B. C.** becomes **A.B.C.**
or **A.B.C.D.** becomes **A B C D**; **A B. C** becomes **A B C**

This process cannot recognize abbreviations which have been entered into the database without either spaces or full stops, such as **BBC**. As certain countries have prepositions or conjunctions ('and') which themselves consist of a single character, standalone characters are not given a full stop with the process ***Standardize abbreviations*.** Thus **SILVIO Y DONNA SERVICIOS DE PUBLICIDAD SA** remains unchanged. However, the process ***Standardize abbreviations- names*** does give a full stop to standalone characters, and should be run over fields containing personal names, so that initials can be correctly formatted. With this process, **Mr T Smith** would become **Mr T. Smith**.

Parameters:

Sent string	The string to be processed
Delimiter	1=make spaces into full stops; 0=make full stops into spaces

Returns

GRCReturnedString	The sent string with the standardised article

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.acronym("B B C & S.A. U  B  C  CO. K G",0)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     B B C & S A  U B C  CO. K G
```

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.acronym("B B C & S.A. U  B  C  CO. K G",0)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     B.B.C. & S.A. U.B.C. CO. K.G.
```

Standardize company types

***Purpose*:** to standardize company type indications (Ltd, PLC, SA etc.) and other commonly found words and abbreviations in company names.

***Example*: Société de la Cité S.A.** becomes **Ste. de la Cité SA; Burnleys Bank Limited** becomes **Burnleys Bank Ltd**

Parameters:

Sent string	The string to be processed
Country code	Country code
Case	Case in which to write the standardised form: 1=UPPER CASE; 0=Mixed case

Returns

GRCReturnedString	The sent string with the standardised company type

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.comptype("The Bookshop gmbh Limited",'GER',0)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     The Bookshop GmbH Ltd
```

Standardize department strings

***Purpose*:** to standardize department strings.

***Example*: Libr. Acquisitions Dept.** becomes **Library Acquisitions Department; Afd. Inkoop** becomes **Afdeling Inkoop**

Parameters:

Sent string	The string to be processed
Country code	Country code
Case	Case in which to write the standardised form: 1=UPPER CASE; 0=Mixed case

Returns

GRCReturnedString	The sent string with the standardised department string

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.dept("Mgmt Support, Account Dept.",'UNI',0)
?GRCReturnedString
```

Returns:

```
GRCReturnedString     Management Support, Account Department
```

Parse post office box numbers

***Purpose*:** to identify post office box numbers within street address fields and to move these numbers to a new field, with or without the correct, standardised local-language word for “post office box”.

This process works by identifying strings in the user-defined address field which indicate a post office box number. The number(s) following this string are written to a user-defined field (the process stops taking data from the string once anything other than a number or space is found). If required, the correct local-language form of the word “post office box” is also written to this return string. The remaining street address data is stripped of any remaining preceding punctuation (.,-;: or /). Once a string is found, the process stops checking - each address is assumed to have only one post office box number.

***Example*: Postbox 17, Ooststraat 21** in the Netherlands will be split into two fields: **Postbus 17** and **Ooststraat 21**.

post office boxes which commence with a single letter (e.g. *Locked Bag E142*) or those which end in one or two letters (e.g. *G.P.O. Box 128A* or *P.O. Box 1DT*) are also recognized and parsed. Apparent post office box strings which are not followed by a number are not parsed.

Parameters:

Sent string	The string to be processed
Country code	Country code
post office box string	1=include full localised post office box string in returned string; 0=return only post office box number
Casing	Write data in: 1=UPPER CASE; 0=Mixed case
`m.GRCReturnedPbox`	Returned postbox string

Returns

GRCReturnedString	The sent string without the parsed post office box number
m.GRCReturnedPbox	The parsed postal code information

Example code (Visual Foxpro)

```
m.GRCReturnedPbox = ""
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.pbox("Po Box 17,  high street",'UNI',1,1,@m.GRCReturnedPbox)
&& postal code string returned
? m.GRCReturnedPbox
? GRCReturnedString
```

Returns:

```
GRCReturnedString     high street
m.GRCReturnedPbox     P.O. BOX 17
```

Standardize thoroughfare strings

***Purpose*:** to standardize strings indicating thoroughfare types and other address type strings (“Street”, “strasse”, “rue”, “house”, “apartment”, “zone” etc.) within the user-defined field.

***Example*: R. de Paris** becomes **rue de Paris**; **Kölnstr.** becomes **Kölnstraße** etc.

***Information required*:** For each field chosen, **GRC Tools™** needs to know whether the data should be written in a *standardized* form or in *full*. By default, **GRC Tools™** writes the data in its long form. However, in some cases a single abbreviation can have more than one long form. For example, *ST* in English might be an abbreviation of *SAINT* or *STREET*; *PTE* in Dutch might be a normal word ending or the abbreviation of *PORTE*, and so on. In these cases, all occurrences are *standardized* to the short form. Thus:

SAINT HELENS ST
GEDEMPTE PORTE

will become

ST HELENS ST
GEDEMPTE PTE

Choosing the option *full* will retain the long form when it already exists, and leave the short forms untouched when found. Thus:

SAINT HELENS ST
GEDEMPTE PORTE

will become

SAINT HELENS ST
GEDEMPTE PORTE

Standardize will produce more standardized but less accurate results; *full* will produce less standardized but more accurate results.

The option is also provided to change *all* thoroughfare types within the user-specified field, or *only one*. For example, if the string contains the Italian address **V V Emmanual 12**, then changing *all* occurrences will change this to **via via Emmanual 12**. Changing *only one* occurrence will change it correctly to **via V Emmanual 12**. When *One Occurrence Only* is chosen, **GRC Tools™** will stop looking for thoroughfare types once the first is found, even when this thoroughfare type is already correctly written. Thus **via V Emmanual 12** will remain unchanged - the **V** will not be altered. *All Occurrences* is useful when there is more than one thoroughfare type in a field, e.g. **31 Railroad Terr., 17 High St.** When *One Occurrence Only* is chosen, then you must specify which end of the address string to start searching for the thoroughfare type (or choose *Accept lookup table defaults* to use the correct searching end for that country). In Italy, for example, thoroughfare types are usually at the front of the address (***via* V Emmanual**), whilst they are at the back for countries like the United Kingdom (**High *Street***). If *All Occurrences* is chosen, the address end is setting is ignored.

Note that choosing *One occurrence only* produces more accurate but less standardized data than choosing *All occurrences*.

Parameters:

Sent string	The string to be processed
Country code	Country code
Standard or full	1=use standardised alternative; 0=use full alternative
String end to search	'Back'=look for the thoroughfare string from back of sent string; 'Front'=look for thoroughfare string from front of sent string
Casing	Write data in: 1=UPPER CASE; 0=Mixed case

Returns

GRCReturnedString	The sent string without the parsed postbox number

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.tfare("Berliner Str.",'GER',0,'Front',0,0)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Berliner Straße
```

Move/parse building Numbers

***Purpose*:** to move house numbers from the front to the back of the street address string or vice versa, allowing commas to be added as desired after or before the house number; or to move the house number to a new field.

***Example*: 27 Wilhelmstrasse** becomes **Wilhelmstrasse 27** in Germany; **rue d'Amiens 27** becomes **27, rue d'Amiens** in France.

The process checks the first 10 digits for numeric characters if the house number is being moved from the front; and, because much information can follow house numbers situated at the end of street address strings, all but the first character if the house number is checked when a move is requested from the back. For this reason, the process of moving house numbers from the back can take slightly longer than the process of moving them from the front.

***Information required*: GRC Tools™** needs to know whether to move the data within the street address strings (from front to back or vice versa) or to a new field.

If *Within Street Address* is chosen, **GRC Tools™** needs to know whether the house number should be moved from the front to the back or vice versa, or whether lookup table defaults are to be accepted. You must also specify whether commas should be added after (if the number is being moved to the front of the address string) or before (if the number is being moved to the back of the address string) the house number.

! If the house number structure for a country is complicated, such as for Spanish- or Portuguese-speaking countries, it is better to move the house number to a new field than to move it within the field. This prevents the "shuffling" of data within a field.

! Though the option to add commas is allowed for every country for the sake of flexibility, it is incorrect to add commas to the addresses of some countries. Note also that commas are added/removed for numbers moved during this process, but not for house numbers already correctly positioned. Commas can be added to/removed from these numbers using the process **Add/Remove Commas**.

If *To new field* is chosen, **GRC Tools™** needs the name of the field to which to move the number, and whether to start the search from the front of the address string or the back, or whether lookup table defaults are to be accepted.

Numbers often contain a suffix, for example *bis* in *23 bis rue de Paris*. You may specify a field to which to move this suffix. If you do not specify a field, the suffix will be moved to the same field as the house number
Furthermore, if moving numbers from the back of a street address, a string may occur after the house number and its suffix, as in this case: *Bahnhofstrasse 25 C Zimmer 3*. To enable you to print this data out in the correct order after running this process, you may also specify a field to which to move the string after the house number. In this case, the fields would be filled with:

Bahnhofstrasse
25 C
Zimmer 3

If no field is specified, the data after the house number and suffix is retained in the original field.

! A street address can contain a whole set of numbers, such as dates. **GRC Tools™** attempts only to identify the house number. Checking from the end of the address which is most likely to contain the house number greatly increases its accuracy.

Addresses in Switzerland have a different format depending on the language region in which the address is situated. It is therefore necessary to specify the language region.

Parameters:

Sent string	The string to be processed

Country code	Country code
Language region	ISO639-2 language code for Switzerland: fra=French, deu=German, ita=Italian
Move number to where	1=numbers to be moved within string; 2=number to be moved to a new field
Comma requirements	1=comma required; 2=no comma required; 0=accept lookup table defaults
String end to start search	‘Back’=starts looking for a number at the right-hand side of the string; ‘Front’=starts looking for a number at the left-hand side of the string; a space (‘ ‘)=accept lookup table defaults
Returned string without house number	Returned string without the house number
House number	String containing house number if number is to be moved to a new string
House number suffix	String containing recognisable house number suffixes
House number suffix suffix	String containing non-recognisable house number suffixes and/or the string found after a house number suffix

Returns

GRCReturnedString	Corrected string if moved within strinf or unchanged sent data if moved to a new string.
m.GRCAddressString	Sent string without house number
m.GRCNumberString	House number including recognisable suffix
m.GRCSuffixString	all data in sent string after house number
m.GRCPostSuffixString	String containing non-recognisable house number suffixes and/or the string found after a house number suffix

```
GRCReturnedString =oTemp.snumber("19 ter High street ",'FRA','Front',2,1,' ',@m.GRCAddressString,
@m.GRCNumberString, @m.GRCSuffixString, @m.GRCPostSuffixString)
```

Example code (Visual Foxpro)

```
m.GRCAddressString=" " && Note SPACE not EMPTY
m.GRCNumberString=" " && Note SPACE not EMPTY
m.GRCSuffixString=" " && Note SPACE not EMPTY
m.GRCPostSuffixString=" " && Note SPACE not EMPTY
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString =oTemp.snumber("19 ter High street ",'FRA','Front',2,1,' ',
@m.GRCAddressString, @m.GRCNumberString, @m.GRCSuffixString, @m.GRCPostSuffixString)
?GRCReturnedString
?m.GRCAddressString
?m.GRCNumberString
?m.GRCSuffixString
?m.GRCPostSuffixString
```

Returns:

```
GRCReturnedString           19 ter High Street
m.GRCAddressString          ter High Street
m.GRCNumberString           19
m.GRCSuffixString
m.GRCPostSuffixString       ter
```

```
GRCReturnedString =oTemp.snumber("High street 17, Unit 5",'UNI','Back',1,2,'  ,
@m.GRCAddressString, @m.GRCNumberString, @m.GRCSuffixString, @m.GRCPostSuffixString)
```

Returns:

```
GRCReturnedString
m.GRCAddressString          High Street Unit 5
```

```
m.GRCNumberString           17
m.GRCSuffixString
m.GRCPostSuffixString       17 High Street Unit 5
```

```
GRCReturnedString =oTemp.snumber("17 High street, Unit 5",'UNI','Back',1,2,' ',
@m.GRCAddressString, @m.GRCNumberString, @m.GRCSuffixString, @m.GRCPostSuffixString)
```

Returns:

```
GRCReturnedString
m.GRCAddressString
m.GRCNumberString
m.GRCSuffixString
m.GRCPostSuffixString       17 High Street, Unit 5
```

Parse Building Number Suffix

***Purpose*:** to parse the part of the house number from the first non-numeric character to a new field.

***Example*: 27-29** becomes **27** and **–29; 1d, 7th floor** becomes **1** and **d, 7th floor.**

Though not all countries use addressing systems that include house numbers, this process can be run on data from any country for the sake of maximum flexibility.

***Information required*:** For each field chosen, **GRC Tools™** needs to know to which field to move the house number suffix.

Parameters:

Sent string	The string to be processed
Country code	Country code
`GRCSuffix`	The variable returned containing the house number suffix
`GRCSuffixSuffix`	The variable returned containing the string following the house number suffix

Returns

GRCReturnedString	The sent string without the house number suffix
`GRCSuffix`	The house number suffix
`GRCSuffixSuffix`	The string following the house number suffix

Example code (Visual Foxpro)

```
GRCSuffix=' '
GRCSuffixSuffix=' '
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.nrsuffix("UNIT 19/B,bis   ",'UNI', @GRCSuffix,
@GRCSuffixSuffix)
&& string to be processed; country code;
&& variable to write suffix to;  variable to write suffix suffix to
?'GRCReturnedString='+GRCReturnedString
?'GRCSuffix=' + GRCSuffix
?'GRCSuffixSuffix='+GRCSuffixSuffix
```

Returns:

```
GRCReturnedString     UNIT 19
GRCSuffix             /B
GRCSuffixSuffix       bis
```

Add/remove Commas

***Purpose*:** to standardize house number/street address formats by either adding commas after (if the number is at the front of the address string) or before (if it is at the end) the house number; or by removing the commas from these places.

This process checks the first 8 characters of the street address string if the house number is at the front of the string, or the last 8 if it at the end of the string. This process takes into account not just the position of the house number, but also its position relative to other strings, such as stand-alone characters (assumed to be part of the house number) or other strings which should not be split from the house number with a comma. Spacing is also corrected. This process is best illustrated using examples:

***Examples*:**

House number at front, add commas:

1,A	becomes	**1, A**
1 ABC	becomes	**1, ABC**
1 A ABC	becomes	**1 A, ABC**
1A ABC	becomes	**1A, ABC**
1 A,ABC	becomes	**1A, ABC**
1 BIS ABC	becomes	**1 BIS, ABC**

House number at back, add commas:

,1	becomes	**, 1**
ABC 1	becomes	**ABC, 1**

House number at front, remove commas:

1,A	becomes	**1 A**
1, ABC	becomes	**1 ABC**
1 A,ABC	becomes	**1 A ABC**
1A, ABC	becomes	**1A ABC**
1 A, ABC	becomes	**1A ABC**
1 BIS, ABC	becomes	**1 BIS ABC**

House number at back, remove commas:

ABC,1	becomes	**ABC 1**
ABC, 1	becomes	**ABC 1**

***Information required*:** For each country specified, **GRC Tools™** needs to know whether commas are to be added or removed, and at which end of the address to look for the house numbers.

! Though the option to add commas is allowed for most countries for the sake of flexibility, it is incorrect to add commas to the addresses of some countries.

Addresses in Switzerland have a different format depending on the language region in which the address is situated. It is therefore necessary to specify the end of the string in which the number is currently written and whether commas should be added or removed per language region, along with the name of the field where the language code is written and the codes which have been used (assigned, if not already existing, using ***Assign language region***).

Parameters:

Sent string	The string to be processed
Country code	Country code
`Language code`	For Switzerland, the ISO 639-2 language code (deu=German, fra=French, ita=Italian)

Comma action	1=add comma; 2=remove comma

Returns

GRCReturnedString	The sent string with added or removed comma

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.commas("Unit 17 B St Helens Avenue","UNI",'ita',2)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Unit 17 B St Helens Avenue
```

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.commas("Unit 17 B St Helens Avenue","UNI",'ita',1)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Unit 17 B, St Helens Avenue
```

House number/letter format

***Purpose*:** to split or concatenate house numbers and their associated single characters.

***Example*: 1 A London Road** becomes **1A London Road**; or **1A London Road** becomes **1 A London Road**

This process checks the first 8 characters of the street address string if the house number is at the front of the string, or the last 8 if it is at the end of the string. It searches for all numbers followed by a single character followed in its turn by a space or a comma.

***Information required*:** For each country specified, **GRC Tools™** needs to know at which end of the street address string the house number is situated, and whether the house numbers and their associated letters need to be concatenated (**1 A** to **1A**) or split (**1A** to **1 A**).

Addresses in Switzerland have a different format depending on the language region in which the address is situated. It is therefore necessary to specify these preferences per language region, along with the name of the field where the language code is written (by the process ***Assign language region*** if not already existing) and the codes which have been used.

Parameters:

Sent string	The string to be processed
Country code	Country code
`Language code`	For Switzerland, the ISO 639-2 language code (deu=German, fra=French, ita=Italian)
`Number action`	1=split the house number from its suffix; 2=concatenate the house number to its suffix

Returns

GRCReturnedString	The sent string with altered house number

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.nsplit("Unit 17B St Helens Avenue","UNI",'ita',1)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Unit 17 B St Helens Avenue
```

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.nsplit("Unit 17 B St Helens Avenue","UNI",'ita',2)
?GRCReturnedString
```

Returns:
```
GRCReturnedString     Unit 17B St Helens Avenue
```

Parse/standardize sorting code

***Purpose*:** to identify numbers and strings indicating sorting codes and to move these codes into a new field, standardizing the format at the same time. For certain countries, sorting code may be assigned if not found within the user-specified field.

In many countries, a sorting code is added, usually after the town name, to allow the post office of the relevant country to send the mail to the correct office for sorting. This code can consist of a string (e.g. **Paris Cédex**), a number (e.g. **Oslo 1**), a combination of the two (e.g. **Lille Cédex 2**), or the name of a province (e.g. **Milano (MI)**).

***Example*: Lille Cedex 2** becomes **Lille** and **Cédex 2**.

Sorting codes are defined in this program as any valid string written directly after, and on the same line as, the city name in an address, and used by postal services as a sorting code, though this may be a province or region name, for example.

The process searches first for strings indicating a sorting code and then searches for any numeric characters at the end of the field specified and adds these to the sorting code written. The string indicating sorting code must be preceded by and followed by at least one space in the user-defined field to be recognized.

***Information required*:** For each string sent, **GRC Tools™** needs to know if the data in the field is to be written in upper or mixed (upper and lower) case.

In certain countries the sorting code, if not found elsewhere in the address, can be assigned on the basis of the postal code. For this reason, the postal code must be specified.

In countries where the sorting code is an administrative district, such as Spain and Italy, many of the names of the administrative districts are the same as those of a city within it. To prevent the city name itself from being mistaken for a province name and being moved to a different field, the lookup tables used will not contain the city name as it would be written as a city. Thus for Italy, the lookup table may contain **MI**, the correct sorting code abbreviation for the province of Milano, or **(Milano)** as the city name will never be found between brackets, but not **Milano**.

Parameters:

Sent string	The string to be processed
Country code	Country code
`Casing of output string`	0=Write sorting code in mixed case; 1=write sorting code in UPPER case.
`Postal code`	Postal code
`GRCSortingCode`	Returned variable containing sorting code.

Returns

GRCReturnedString	The sent string with altered house number
GRCSortingCode	Returned variable containing sorting code.

Example code (Visual Foxpro)

```
GRCSortingCode=''
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.sortcode("MILANO (MI)","ITA",0,'21000',@GRCSortingCode)
?GRCReturnedString
?GRCSortingCode
```

Returns:

```
GRCReturnedString     MILANO
GRCSortingCode        MI
```

Parse/standardize place names

***Purpose*:** to locate, parse and standardize settlement names, replacing foreign or minority language forms of a town name, names incorrectly spelt or including typos, transcribed letters, incorrect diacritical marks and so on with the standardized local language equivalents.

***Example*:**

LONDNO	
16 High Street, London	
Londres	
Station House,London	

becomes

	London
16 High Street	**London**
	London
Station House	**London**

To be found for parsing, a settlement name must be on its own in a string or separated from the other string data with a comma. Any trailing commas after parsing are removed. A correctly formatted postal code is required.

! Most lookup tables are shipped with **GRC Tools™** structured to change foreign words for town names into the local- language equivalent only. However, for countries where a number of languages exists, the rules used for the lookup tables are as follows.

- *Belgium*: language areas are fixed by law - the names are altered to the local-language name as defined by law. In mixed language areas, the majority language name is used. Brussels, which is purely bilingual, is excluded from this process.
- *Finland*: the town name assigned (Finnish or Swedish) is that of the majority of speakers in the town concerned.
- *France*: in all cases, the French name is assigned.
- *Greenland*: the Greenlandic rather than the Danish name is assigned.
- *Ireland*: in *An Ghaeltacht* (Irish-speaking areas) the Irish name is assigned. In all other cases, the English name is assigned.
- *Italy*: the Italian name is assigned in all cases.
- *Malta*: the Maltese name is assigned in all cases.
- *The Netherlands*: the Dutch names are assigned in all cases. *Den Haag* is changed to *'s-Gravenhage*; *Den Bosch* is changed to *'s-Hertogenbosch*.
- *Spain*: Catalan/Valencian names are assigned for towns within the provinces of Catalonia, the Balearic Islands and Valencia. Basque names are assigned for towns within the province of País-Vasco. Galician names are assigned for towns within the province of Galicia. In Navarra, town names are assigned (Basque or Castilian) on the basis of the majority language within each town. In all other cases, Castilian is assigned.
- *Switzerland*: Town names are assigned on the basis of majority language within each town. The bilingual town of Biel/Bienne becomes in all cases *Biel/Bienne*.

! The lookup tables used by **GRC Tools™** for this process are very large. **GRC Tools™** may appear to hang for some time at 0% and 100% during processing. This is normal - avoid interrupting the program at these points.

***Information required*:** For each string sent chosen, **GRC Tools™** needs to know in which case the corrected version should be written.

Settlement names can be parsed using two search methods – exact string searching and fuzzy matching. Fuzzy matching is less accurate than exact string matching, so it should be used with caution. Using fuzzy matching increases the number of settlements parsed. If you choose to use fuzzy matching, this is always done as well as, and after, exact string matching. It is never done instead of exact string matching.

This process will only work for each town within the postal code area defined within the lookup table. This prevents address components with similar forms to settlement names being incorrectly parsed/standardised. For this reason the process requires knowing in which field the postal code is situated. The postal code should be in its correct format without punctuation or other codes such as country sorting codes (e.g. **GB-**).

Parameters:

Sent string	The string to be processed
Country code	Country code
`Postal code`	Postal code
`Fuzzy logic choice`	.f.=do not use fuzzy logic; .t.=use fuzzy logic
`Case of parsed/standardised data`	1=UPPER case; 0=mixed case
`GRCParsedPlace`	Returned parsed place name
`GRCStdPlace`	Returned standardised place name

Returns

GRCReturnedString	The sent string without place name if parsing was successful
`GRCParsedPlace`	Parsed and standardised place name
`GRCStdPlace`	Sent string with the place name standardised

Example code (Visual Foxpro)

```
GRCParsedPlace=' '
GRCStdPlace=' '
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.parsetown("High Street, Railway Station, THE
HAGUE","NET",'2511',.f.,0,@GRCParsedPlace,@GRCStdPlace)
?'GRCReturnedString='+GRCReturnedString
?'GRCParsedPlace='+GRCParsedPlace
?' GRCStdPlace ='+GRCStdPlace
```

Returns:

```
GRCReturnedString     High Street, Railway Station
GRCParsedPlace        ‘s-Gravenhage
GRCStdPlace     High Street, Railway Station, ‘s-Gravenhage
```

Parse/assign provinces
Assign provinces/regions
Assign regions

***Purpose*:** to identify strings indicating administrative districts (provinces, counties, states, cantons etc.) and to move these strings into a new field, standardizing the format at the same time. For certain countries, province and region name may be assigned if not found within the user-specified field.
Parse/assign provinces is used for countries where provinces are used in addresses and where provinces which are not found can be assigned on the basis of postal code.
Assign provinces/regions is used where province and/or region name can be assigned on the basis of postal code.
Assign regions is used where region names can be assigned on the basis of postal code.

***Example*: Milano (MILANO)** becomes **Milano** and **MI** and the region name **Lombardia** is assigned.

The process searches first for strings indicating an administrative district. The string indicating administrative district must be preceded and followed by at least one space in the field specified to be recognized. Should no administrative district be found, and postal codes are available within the lookup table, province names (and in certain cases region names) are assigned.

***Information required*:** For each string sent, **GRC Tools™** needs to know if the province name is to be written in upper or mixed (upper and lower) case; and the variable to which to move/assign the province name.

When ***Assign provinces/regions*** is chosen, **GRC Tools™** needs to know the case in which the province/region data is to be written and the variable to which this data is to be moved (if this field already has data in it, it will not be overwritten). Also required is the string containing the postal code. The postal code should be contained in this field in the correct format, without being preceded by a country postal sorting code (**GB-** etc.).

For certain countries, it is possible to assign the province/region on the basis of postal code. **GRC Tools™** therefore requires a string containing the postal code; the name of the variable to which the region name is to be written, and the case in which it is to be written.

In countries such as Spain and Italy many of the names of the administrative districts are the same as those of a city within it. To prevent the city name itself from being mistaken for a province name and being moved to a different field, the lookup table does not contain the city name as it would be written as a city. Thus for Italy, the lookup table may contain **(Milano)** as the city name will never be found between brackets, but not **Milano**; **Oxon** and **Oxfordshire** but not **Oxford** for the United Kingdom and so on. Equally, if the name of one region is the same as part of the name of another (for example *Lothian* and *East Lothian*, *Yorkshire* and *West Yorkshire* and so on), the shorter form has not been added to the lookup table to prevent the longer form from being incorrectly split, leaving, for example, *West* in one field and moving only the *Yorkshire*.

Avoid running this process with strings containing street addresses. Region names which are part of street addresses, as in *27 Avon Road* will be moved.

- **France**: *Département* names are never found in addresses. These can therefore be assigned on the basis of postal code, they are not otherwise listed in the lookup tables.
- **Italy**: Provinces and regions strings are searched for, and, if not found, provinces and regions are assigned on the basis of postal code. By default the region names are written in full. If the correct abbreviations (used as a sorting code) are required, use the process **Move/Correct Sorting Code**
- **Spain** Provinces and regions strings are searched for, and, if not found, provinces and regions are assigned on the basis of postal code. By default the region names are written in full. If they are required between brackets (used as a sorting code),

use the process **Move/Correct Sorting Code**

- **Switzerland**: Canton names are often found in addresses but should not be used. They are searched for but cannot be assigned if none is found.

PARSE/ASSIGN PROVINCES

Parameters:

Sent string	The string to be processed
Country code	Country code
`Postal code`	Postal code
`Case of parsed/standardised PROVINCE data`	1=UPPER case; 0=mixed case
`Case of parsed/standardised REGION data`	1=UPPER case; 0=mixed case
`GRCProvince`	Returned parsed province name
`GRCRegion`	Returned assigned region name

Returns

GRCReturnedString	The sent string without province name if parsing was successful
`GRCProvince`	Returned parsed province name
`GRCRegion`	Returned assigned region name

Example code (Visual Foxpro)

```
GRCProvince=' '
GRCRegion=' '
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.regions("Milan (MI)","ITA",'20000',1,0,@GRCProvince,
@GRCRegion)
?'GRCReturnedString='+GRCReturnedString
?'GRCProvince='+GRCProvince
?'GRCRegion='+GRCRegion
```

Returns:

```
GRCReturnedString     Milan
GRCProvince           MI
GRCRegion  Lombardia
```

ASSIGN PROVINCES/REGIONS

Parameters:

Sent string	Should be empty (a space)
Country code	Country code
`Postal code`	Postal code
`Case of parsed/standardised PROVINCE data`	1=UPPER case; 0=mixed case
`Case of parsed/standardised REGION data`	1=UPPER case; 0=mixed case
`GRCProvince`	Returned assigned province name
`GRCRegion`	Returned assigned region name

Returns

GRCReturnedString	Returned assigned province name
GRCProvince	Returned assigned province name
GRCRegion	Returned assigned region name

Example code (Visual Foxpro)

```
GRCReturnedString=' '
GRCProvince=' '
GRCRegion=' '
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.reg2("","ITA",'20000',1,0,@GRCProvince,@GRCRegion)
?'GRCReturnedString='+GRCReturnedString
?'GRCProvince='+GRCProvince
?'GRCRegion='+GRCRegion
```

Returns:

```
GRCReturnedString     MI
GRCProvince           MI
GRCRegion  Lombardia
```

ASSIGN REGIONS

Parameters:

Sent string	Should be empty (a space)
Country code	Country code
Postal code	Postal code
Case of parsed/standardised REGION data	1=UPPER case; 0=mixed case
GRCRegion	Returned assigned region name

Returns

GRCReturnedString	Returned assigned region name
GRCRegion	Returned assigned region name

Example code (Visual Foxpro)

```
GRCReturnedString=' '
GRCRegion=' '
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.reg3("","ITA",'20000',1,@GRCRegion)
?'GRCReturnedString='+GRCReturnedString
?'GRCRegion='+GRCRegion
```

Returns:

```
GRCReturnedString     LOMBARDIA
GRCRegion  LOMBARDIA
```

Parse/standardize forms of address

***Purpose*:** to identify strings indication forms of address and to move these to a new variable, make the strings consistent at the same time.

***Example*: Mister Smith** becomes **Mr** and **Smith**; **Mons. Dupont** becomes **M.** and **Dupont**.

***Information required*:** For each string sent, **GRC Tools™** needs to know if the form of address is to be written in upper or mixed (upper and lower) case.

You must then specify the variable to which to move the form of address, if found.

When a salutation can consist of a single letter, for example **M** to indicate **Monsieur** in French-speaking countries, then this has not been added to the lookup table as **GRC Tools™** cannot distinguish between **M Dupont** where the **M** indicates, for example, **Michel** and **M Dupont** where the **M** indicates **Monsieur**.

Parameters:

Sent string	String to be processed
Country code	Country code
`Case of parsed/standardised form of address`	1=UPPER case; 0=mixed case
`GRCSalutation`	Returned parsed/standardised form of address

Returns

GRCReturnedString	Returned string with form of address removed
`GRCSalutation`	Returned parsed/standardised form of address

Example code (Visual Foxpro)

```
GRCSalutation=' '
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.salut("Dhr. G.R. Rhind","NET",0,@GRCSalutation)
?'GRCReturnedString='+GRCReturnedString
?'GRCSalutation='+GRCSalutation
```

Returns:

```
GRCReturnedString     G.R. Rhind
GRCSalutation   Dhr
```

Standardize job titles

***Purpose*:** to standardize titles names into their fullest correct equivalent.

***Example*: C.E.O.** becomes **Chief Executive Officer**; **Mging Dir.** becomes **Managing director**

***Information required*:** For each string sent, **GRC Tools™** needs to know in which case to write the correct job title data.

Note: this process writes the job title in its most expanded correct form. Thus, **C.E.O**. (English) becomes **Chief Executive Officer**; **Dir. Technique** (French) remains **Dir. Technique** because it is not know if the job title owner is male (**Directeur**) or female (**Directrice**). This being so, the job titles in the database can be very long – the longest currently in the table is 120 characters long.

Parameters:

Sent string	String to be processed
Country code	Country code
`Case of standardised job title`	1=UPPER case; 2=mixed case

Returns

GRCReturnedString	Returned standardised job title

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.jobtitle('BUS MGR','USA',2)?
'GRCReturnedString='+GRCReturnedString
```

Returns:

```
GRCReturnedString     Business Manager
```

Remove Accents

***Purpose*:** to replace accented characters in the user-defined field with the correct non-accented equivalents.

***Information required*:** there is a clash in equivalent characters in Belgium between German and Dutch/French; in Switzerland between German, French and Italian; and in Canada between French and English. For this reason, this process handles each language area differently. It is therefore necessary to send the language code in these cases.

For each string sent for processing, **GRC Tools™** needs to know which case the data is in. Though it may seem self-evident that an upper-case accented-letter is replaced by an equivalent upper-case letter without accent, for some languages an accented character may be replaced by more than one letter and the casing of these letters needs to be assigned. For example, the German **Ü** is replaced by **Ue** at the start of a mixed-case word and by **UE** at the start of an upper case string.

Parameters:

Sent string	String to be processed
Country code	Country code
Language region code	language code for Switzerland, Belgium or Canada: fra=French, deu=German, ita=Italian, nld=Dutch, eng-English
Case of data being processed	1=UPPER case; 0=mixed case

Returns

GRCReturnedString	Returned string with removed/replaced accents

Example code (Visual Foxpro)

```
oTemp=CREATEOBJECT("grctools.grctools")
GRCReturnedString=oTemp.remacc("ÄBCĐÉFGHÏJKLMNÒPQRSTÜVWXYZÖ",'SWI','deu',0)
?'GRCReturnedString='+GRCReturnedString
```

Returns:

```
GRCReturnedString     AeBCĐÉFGHÏJKLMNÒPQRSTUeVWXYZOe
```

GRCTools 5 DLL Manual