Representing Keys and Shortcuts

Go Up to Using the RTL in Multi-Device Applications

This page explains the different ways that you may use to represent keys or shortcuts (combinations of keys).

Representing Keys

To represent a single key, you may use either a code or a string. The following tables show a complete list of codes and strings that you may use to represent each key.

The following tables do not show a string representation for some keys because the UI framework is not responsible for the string representation of those keys. Instead, the operating system where your application is running provides the string representation of those keys. The following tables do not show those string representations provided by the operating system because they may vary; for example, Windows provides locale-specific key names (https://msdn.microsoft.com/en-us/library/windows/desktop/ms646300).

Representing Letter Keys

Contents

Key	Code Representation	String Representation
	Constant (System.UITypes)	String
A	vkA	A
B	vkB	B
C	vkC	C
D	vkD	D
E	vkE	E
F	vkF	F
G	vkG	G
H	vkH	H
I	vkI	I
J	vkJ	J
K	vkK	K
L	vkL	L
M	vkM	M
N	vkN	N
O	vkO	O

See Also

P	vkP	P
Q	vkQ	Q
R	vkR	R
S	vkS	S
T	vkT	T
U	vkU	U
V	vkV	V
W	vkW	W
X	vkX	X
Y	vkY	Y
Z	vkZ	Z

Representing Digit Keys

Key	Code Representation	String Representation
	Constant (System.UITypes)	String
0	vk0	0
1	vk1	1
2	vk2	2
3	vk3	3
4	vk4	4
5	vk5	5
6	vk6	6
7	vk7	7
8	vk8	8
9	vk9	9

Representing Symbol Keys

Key	Code Representation	String Representation
	Constant (System.UITypes)	String
Backslash	vkBackslash	\
Bracket (Opening)	vkLeftBracket	[
Bracket (Closing)	vkRightBracket	]
Comma	vkComma	,
Equal sign	vkEqual	=
Minus sign	vkMinus	-
Period	vkPeriod	.
Quote	vkQuote	'
Semicolon	vkSemicolon	;
Slash	vkSlash	/
Tilde	vkTilde	~

Representing Editing Keys

Key	Code Representation	String Representation	
	Constant (System.UITypes)	Constant (FMX.Consts, Vcl.Consts)	String
BACKSPACE	vkBack	SmkcBkSp	BkSp
DELETE	vkDelete	SmkcDel	Del
ENTER RETURN	vkReturn	SmkcEnter	Enter
INSERT	vkInsert	SmkcIns	Ins
SPACE BAR	vkSpace	SmkcSpace	Space
TAB	vkTab	SmkcTab	Tab

Representing Navigation Keys

Key	Code Representation	String Representation	
	Constant (System.UITypes)	Constant (FMX.Consts, Vcl.Consts)	String
ESC (Escape)	vkEscape	SmkcEsc	Esc
PGUP (Page up)	vkPrior	SmkcPgUp	PgUp
PGDN (Page down)	vkNext	SmkcPgDn	PgDn
END	vkEnd	SmkcEnd	End
HOME	vkHome	SmkcHome	Home
← (Left arrow)	vkLeft	SmkcLeft	Left
↑ (Up arrow)	vkUp	SmkcUp	Up
→ (Right arrow)	vkRight	SmkcRight	Right
↓ (Down arrow)	vkDown	SmkcDown	Down

Representing Modifier Keys

Key	Code Representation	String Representation	
	System.Classes Constant	Constant (FMX.Consts, Vcl.Consts)	String
ALT	scAlt	SmkcAlt	Alt+
CMD	scCommand	SmkcCmd	Cmd+
CTRL (Control)	scCtrl	SmkcCtrl	Ctrl+
			^
SHIFT	scShift	SmkcShift	Shift+

Most computers running Windows do not have a CMD key, just as most computers running OS X do not have a CTRL key. Mind this when you configure the shortcuts of your application. You may need to define your shortcuts at run time and use conditional compilation to set the right shortcut for each platform. For example:

Delphi:

```
{$IF defined(MSWINDOWS)}
    MenuItem1.ShortCut := TextToShortCut('Ctrl+N');
{$ELSEIF defined(MACOS) and not defined(IOS)}
    MenuItem1.ShortCut := TextToShortCut('Cmd+N');
{$ENDIF}
```

C++:

```
#if defined(_WIN32)
    MenuItem1->ShortCut = TextToShortCut("Ctrl+N");
#elif defined(__APPLE__) && (defined(__i386__) or defined(__x86_64__))
    MenuItem1->ShortCut = TextToShortCut("Cmd+N");
#endif
```

You may also represent modifier keys as regular keys using the following constants and strings:

Key	Code Representation	String Representation	
	Constant (System.UITypes)	Constant (FMX.Consts)	String
ALT	vkMenu		Alt
ALT (Left)	vkLMenu	SmkcLMenu	Left Alt
ALT (Right)	vkRMenu	SmkcRMenu	Right Alt
CTRL (Control)	vkControl		Ctrl
CTRL (Left Control)	vkLControl	SmkcLControl	Left Ctrl
CTRL (Right Control)	vkRControl	SmkcRControl	Right Ctrl
SHIFT	vkShift		Shift
SHIFT (Left)	vkLShift	SmkcLShift	Left Shift
SHIFT (Right)	vkRShift	SmkcRShift	Right Shift
WINDOWS (Left)	vkLWin	SmkcLWin	Left Win
WINDOWS (Right)	vkRWin	SmkcRWin	Right Win

Representing Function Keys

Key	Code Representation	String Representation
	Constant (System.UITypes)	String
F1	vkF1	F1
F2	vkF2	F2
F3	vkF3	F3
F4	vkF4	F4
F5	vkF5	F5
F6	vkF6	F6
F7	vkF7	F7
F8	vkF8	F8
F9	vkF9	F9
F10	vkF10	F10
F11	vkF11	F11
F12	vkF12	F12
F13	vkF13	F13
F14	vkF14	F14
F15	vkF15	F15

F16	vkF16	F16
F17	vkF17	F17
F18	vkF18	F18
F19	vkF19	F19
F20	vkF20	F20
F21	vkF21	F21
F22	vkF22	F22
F23	vkF23	F23
F24	vkF24	F24

Representing Lock Keys

Key	Code Representation	String Representation	
	Constant (System.UITypes)	Constant (FMX.Consts)	String
CAPS LOCK	vkCapital	SmkcCapital	Caps Lock
NUM LOCK	vkNumLock	SmkcNumLock	Num Lock
SCROLL LOCK	vkScroll	SmkcScroll	Scroll Lock

Representing Numeric Pad Keys

Key	Code Representation	String Representation
	Constant (System.UITypes)	String
0	vkNumpad0	Num 0
1	vkNumpad1	Num 1
2	vkNumpad2	Num 2
3	vkNumpad3	Num 3
4	vkNumpad4	Num 4
5	vkNumpad5	Num 5
6	vkNumpad6	Num 6
7	vkNumpad7	Num 7
8	vkNumpad8	Num 8
9	vkNumpad9	Num 9
*	vkMultiply	Num *
+	vkAdd	Num +
,	vkSeparator	Num ,

-	vkSubtract	Num -
.	vkDecimal	Num .
/	vkDivide	Num /

Representing Multimedia Keys

Key	Code Representation	String Representation	
	Constant (System.UITypes)	Constant (FMX.Consts)	String
Browser: Back	vkBrowserBack	SmkcBrowserBack	BrowserBack
Browser: Forward	vkBrowserForward		
Browser: Refresh	vkBrowserRefresh		
Browser: Stop	vkBrowserStop		
Browser: Search	vkBrowserSearch		
Browser: Favorites	vkBrowserFavorites		
Browser: Home	vkBrowserHome		
Camera	vkCamera	SmkcCamera	Camera
Launch: Email	vkLaunchMail		
Launch: Media Select	vkLaunchMediaSelect		
Launch: App 1	vkLaunchApp1		
Launch: App 2	vkLaunchApp2		
Media: Next track	vkMediaNextTrack		
Media: Previous track	vkMediaPrevTrack		
Media: Stop	vkMediaStop		

Media: Play/Pause	`vkMediaPlayPause`		
Volume: Mute	`vkVolumeMute`		
Volume: Down	`vkVolumeDown`		
Volume: Up	`vkVolumeUp`		

Representing Buttons

Button	Code Representation
	Constant (System.UITypes)
Left mouse button	`vkLButton`
Right mouse button	`vkRButton`
Middle mouse button	`vkMButton`
X1 mouse button	`vkXButton1`
X2 mouse button	`vkXButton2`

Representing Language Input Keys

Key	Code Representation
IME Kana mode	`vkKana`
IME Hangul mode	`vkHangul`
IME Junja mode	`vkJunja`
IME final mode	`vkFinal`
IME Hanja mode	`vkHanja`
IME Kanji mode	`vkKanji`
IME convert	`vkConvert`
IME nonconvert	`vkNonConvert`
IME accept	`vkAccept`
IME mode change request	`vkModeChange`
IME process	`vkProcessKey`

Representing Other Keys

Key	Code Representation	String Representation	
	Constant (System.UITypes)	Constant (FMX.Consts)	String
ATTN	vkAttn		
BACK	vkHardwareBack	SmkcHardwareBack	HardwareBack
BREAK	vkCancel	SmkcCancel	Break
CLEAR	vkClear	SmkcClear	Clear
CRSEL	vkCrsel		
ERASE EOF	vkErEof		
EXECUTE	vkExecute		
EXSEL	vkExsel		
HELP	vkHelp		
Line feed	vkLineFeed		
MENU	vkApps	SmkcApps	Application
OEM 102 Either the angle bracket key or the backslash key on the RT 102-key keyboard.	vkOem102	SmkcOem102	OEM \
OEM-specific Clear key	vkOemClear		
OEM-specific Ico 00	vkIco00		

OEM-specific Ico Clear	vkIcoClear		
OEM-specific Ico Help	vkIcoHelp		
PA1	vkPA1		
Packet Used to pass Unicode characters as if they were keystrokes.	vkPacket		
Paragraph	vkPara	SmkcPara	Paragraph
PAUSE	vkPause	SmkcPause	Pause
PLAY	vkPlay		
PRINT	vkPrint		
PRINT SCREEN	vkSnapshot		
Reserved	vkNoname		
SELECT (http://edward.oconnor.cx/2002/07/the-select-key)	vkSelect		
SLEEP	vkSleep		
ZOOM	vkZoom		
No key	vkNone		

Representing Shortcuts

The following sections explain how to represent a shortcut as either an instance of TShortCut or a string.

Representing Shortcuts as Instances of TShortCut

To define a shortcut as an instance of System.Classes.TShortCut, join the code of a regular key and the codes of modifier keys using the OR bitwise operator. For example:

Delphi:

```
Shortcut := vkZ or scShift or scCtrl;
```

C++:

```
Shortcut = vkZ | scShift | scCtrl;
```

Representing Shortcuts Using Strings

To define a shortcut using a string, you can simply concatenate the keys, and place the main key at the end. These are some examples of strings that represent shortcuts:

- ^P (`CTRL + P`)
- Shift+Ctrl+Z (`SHIFT + CTRL + Z`)

Specifying Accelerator Keys

On Windows platforms, the **Accelerator keys** enable the user to access controls using only the keyboard input. For example, by pressing `Alt+ <accelerator_letter>` key combination. An accelerator key is shown as an underlined letter in the **Caption** or **Text** property of your control. To specify an accelerator key in your code, precede an `<accelerator_letter>` with an ampersand `&` character in the caption or text property of your control. The letter after the ampersand appears underlined in the caption/text on the control. For example, to set the **S** character as an accelerator key for a **Save** button, type `&Save` in the text property. On some controls specified `<accelerator_letter>` s appear underlined only when the `ALT` key is pressed.

> **Note**: This feature only works on Windows platforms. When run on other platforms, the ampersand is stripped from the text.

Since an ampersand `&` character in a caption/text property is handled as an instruction to underline the next letter, therefore, a single ampersand character is not shown in the caption/text of a corresponding control. To show a single ampersand `&` character in a caption/text specify two ampersand `&&` characters.

Switching Between Code and String Representation

Given an instance of System.Classes.TShortCut that represents either a key or a shortcut, you may use the following methods to obtain the string representation of that key or shortcut:

- FMX.Menus.IFMXMenuService.ShortCutToText
- Vcl.Menus.ShortCutToText

Conversely, given the string representation of a key or shortcut, you may use the following methods to obtain an instance of TShortCut that represents that key or shortcut:

- FMX.Menus.IFMXMenuService.TextToShortCut
- Vcl.Menus.TextToShortCut

Using these functions you can easily set the shortcut of a menu item or some other control at run time using the string representation of the shortcut instead of its code representation, which makes your code more readable:

Delphi:

```
MenuItem1.ShortCut := TextToShortCut('Cmd+N');
```

C++:

```
MenuItem1->ShortCut = TextToShortCut("Cmd+N");
```

See Also

- Specifying Accelerator Keys and Keyboard Shortcuts
- Virtual-Key Codes (https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731) (MSDN)
