FEDIT Help

FORMATTED EDIT (FEDIT) CUSTOM CONTROL is a powerful extension of the standard MS Windows Edit control. Designed as a superclass of the standard EDIT control, it provides all facilities needed to ensure that the data received by the program is correct. It relieves application programmer of the burden of verifying the validity of data, providing more time to concentrate on the business flow of the application. It gives total control of the data by embedding intelligence into Resource File (.RC). It comes as a Dynamic Link Library (DLL) and it is *installable* into the **Borland Resource Workshop** and **Microsoft Dialog Editor.** The DLL contains over 60 API functions and 20 messages that are used along with standard MS-Windows API procedures and messages to manipulate FEDIT control. When installed, FEDIT becomes an integral part of your Windows programming environment and provides new functionality to your applications.

Features

The following features are included in the FEDIT custom control:

- Support for 5 main **data styles**. You can set the desired behavior for all your fields by selecting the style and setting up a **Formatting Mask** string. The Formatting string can be changed dynamically by the application in order to display and accept data in a different format.
- Installable into **Microsoft Dialog Editor** and **Borland Resource Workshop**.
- Support for the *required* and *optional* fields. FEDIT allows an application programmer to specify which fields or even which parts of the field must be filled by a user.
- Support of the *insert* and *overwrite* editing modes.
- Support of the **clipboard** and reformatting values through clipboard.
- Support of the ***symbolic names*** for the controls.
- 19 ***date and time*** manipulation procedures.
- A set of formattting procedures for ***data conversion and string manipulation*** .
  Support of all standard MS-Windows EDIT control style features.

Next FEDIT features are described in this section:

Field validation
Using the clipboard
Symbolic names

Designing Dialog Boxes

There are two ways to create an FEDIT control in a dialog box:

1. By calling CreateWindow() function;
2. By using a template from a resource file (.RC). The resource file can be created with:

- **Microsoft Dialog Editor**
- **Borland Resource WorkshopTM**
- **A text editor**

Using Microsoft Dialog Editor

The Dialog Editor can automate the process of designing dialog boxes with FEDIT controls. You can see your FEDIT controls and test them as you design your dialog boxes. The Dialog Editor automatically creates resource scripts for you.

This chapter explains how to create, modify and test FEDIT controls using the Dialog Editor. For the complete description of the Dialog Editor, please refer to the Microsoft documentation.

To install FEDIT.DLL into Microsoft Dialog Editor:

1. Select Open Custom command from the FILE menu
2. Enter: <full_path>\FEDIT.DLL.
3. Press OK.

To create a new FEDIT control:

1. Create a new dialog or open an existing one.
2. Click the Custom Control (bottom right) tool on the Toolbox. If you have more then one custom control installed, the Select Custom Control dialog box appears.
3. From the Available Controls list box, select the FEDIT control.
4. Choose OK.
5. Position the cursor inside your dialog. Drop the control by clicking the mouse button.

To specify FEDIT control properties:

1. Double click the FEDIT control to open its Styles dialog box. You can also open the Styles dialog box for the currently selected control by pressing ENTER or choosing the Styles command from the Edit menu.

2. Select the desired format style and enter the format string. You can also enter the name and modify other attributes of the control.

3. Press OK button to accept changes or CANCEL to abort modification. You can obtain the on-line help by pressing the HELP button.

To see how the FEDIT control works, choose the Test Mode command from the Options menu. The FEDIT control becomes operational with a default or empty string in it. You can now type in the characters to check the control's behavior. To quit Test Mode, choose the Test Mode command from the Options menu, or press Alt+F4.

Using Borland Resource Workshop

The Resource Workshop can automate the process of designing dialog boxes with FEDIT controls. You can see your FEDIT controls and test them as you design your dialog boxes. The Resource Workshop automatically creates resource scripts for you.

This chapter explains how to create, modify and test FEDIT controls, using the Resource Workshop. For the complete description of the Resource Workshop , please refer to the Borland documentation.

To install FEDIT into Borland Resource Workshop:

1. Create a new dialog or select the existing one. The Resource Workshop brings up the Dialog Editor window.
2. Select the Install Control Library command form the Options menu.
3. Enter <full_path>\FEDIT.DLL
4. Press OK. A new item   (F) will appear in the Toolbox, that represents the FEDIT control class.

To create a new FEDIT control:

1. Create a new project or open an existing one. Create a new dialog or select the one you have. The Resource Workshop brings up the Dialog Editor window.
2. It is recommended to include the FEDIT.H file in all resource files, that define FEDIT custom controls.   If you did not add it to your project   before, then:
   - Select the Add to project ... command of the file menu.
   - Select H   c Header from the File type listbox.
   - Enter the <full_path>\FEDIT.H into the File name edit box.
   - Press OK.
3. Click the (F)   icon   int the Toolbox. The cursor will change its shape to the (F). Drag the cursor into the dialog box and release the mouse button to place the control into the dialog.

3a. You can also click the 'key' icon in the Toolbox, or select 'Custom...' item from the Options menu. The New custom control dialog box appears. Select FEDIT class and press OK button. Move you cursor into the dialog box. The cursor will change its shape to the cross. Click the left button to place the control into the dialog.

To specify FEDIT control properties:

1. Double click the FEDIT control to open it's Styles dialog box. You can also open the Styles dialog box for the currently selected control by pressing ENTER or choosing the Styles... command from the Control menu.

2. Select the desired format style and enter the format string. You can also enter the name and modify other attributes of the control.

3. Press OK button to accept changes or CANCEL to abort modification. You can obtain the on-line help by pressing the HELP button.

To see how the FEDIT control works, select the Test icon in the toolbox or choose the Test Dialog command from the Options menu. The FEDIT control becomes operational with a default or empty string in it. You can now type in the characters to check the control's behavior.

To quit Test Mode, press the Escape key, or choose the Test Dialog command from the Options menu, or press Alt+F4.

Using Text Editor

To specify a new FEDIT control for add to the dialog template in your resource file a new line in the next format:

CONTROL *format, id,* "FEDIT", *style, x, y, widht, height*

where

format	Specifies any valid FEDIT format string (See Data Styles and Formats topic. for more details) and a name for the control, enclosed in double quotation marks. A special character \206 is used to separate the format from the name.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
style	Specifies the control styles. This value can be a combination of the FEDIT class styles (see Data Styles and Formats topic), edit class styles (see the Edit-control styles topic in the MS-Windows manual) and the following styles: WS_TABSTOP, WS_GROUP, WS_VSCROLL, WS_HSCROLL, and WS_DISABLED. You can use the bitwise OR (|) operator to combine styles.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The height is in 1/8-character units.

Using the clipboard

FEDIT control allows user to copy, paste and reformat values using the clipboard.

Function	Hot key	Comment
Copy	<CNTL> <INS>	The textual value of the control is copied to the clipboard along with the internal information about the data style of the control.
Paste	<SHIFT><INS>	FEDIT will try to make the most sense out of data in the clipboard. If the value in the clipboard has the same data style as the receiving FEDIT control then this value is reformatted and placed in the control. Otherwise FEDIT attempts to paste text from the clipboard. The new value of the control is always correct for the data style of the receiving field, regardless of the clipboard content.

Using the Symbolic Names

The FEDIT control allows an application programmer to specify an additional character string, assosiated with the FEDIT control. The FEDIT Style Dialog Box in the Dialog Editor and the Resource Workshop provide a field called name. This name is joined together with the format string and placed into the controls window name. The format is separated from the name with the special character \206 (decimal 134). The Dialog Editor and the Resource Workshop pack and unpack these two strings for you. If you create the resource script using a text editor, you should place both the format and the name, separated by '\206', into the text part of the CONTROL statement.

You can specify any additional information into the name field and later access it from your application using **FM_GETNAME**and messages or **FxGetName** and **FxSetName** extended functions. For example you can specify the maximum number, the user can enter into the field and then compare the entered value with this number. In this case you will be able to modify your program even without recompiling it.

FEDIT provides a special function **FxDlgGetByName.** This function assumes that the value in the name field is a unique string within the dialog. It retrieves the control with the name, that matches the string provided as a parameter.

Every FEDIT control has extended flags. These flags are not used by the control and can be set, tested and cleared by the program. The initial value of these flags is zero. You can access these flags using **FM_GETFLAG** and **FM_SETFLAG** messages or **FxGetFlag** and **FxSetFlag** extended functions.

Field validation

Null values

FEDIT control class supports a special 'NULL' value. It can be set and tested using extended API functions **FxSetNull** and **FxIsNull** or **FM_SETNULL** and **FM_ISNULL** messages. The NULL value causes FEDIT to suppress output of the picture clause or default value and display an empty string. This value can be, but is not necessarily, used to represent a NULL value in the database. User can enter this value into the FEDIT control by pressing <CTRL-DEL> keys. You may also specify NULL value as a default value for the FEDIT control by selecting the radio button provided in the FEDIT Options dialog box or by specifying FS_INITNULL style.

If a FEDIT control with NULL value looses focus a special notification message FN_NULL is sent to its parent. A program can intercept this message to take appropriate action or just ignore it.

If data is required in a control then you can select a NOT NULL radio button in the FEDIT Options dialog box or specify FS_NONULL style. In this case FEDIT automatically checks the value of the control and doesn't allow user to remove focus from it until data is entered. This mechanism forces a dialog or a window with an empty control to be temporary system modal and thus should be used with caution.

See also:

FS_INITNULL style, FS_INITDFLT style, FS_NONULL style, FN_NULL notification message, FM_SETNULL message, FM_ISNULL message, FxDlgGetNull, FxDlgSetNull

Invalid values

In some cases FEDIT control cannot check the user input as it is being entered and needs the whole value to be entered before the full validation can take place. For example, when user enters 31 as the day of a month FEDIT can check it only after the month has been entered.

FEDIT control class recognizes the following values as the invalid ones :

- Invalid date/time - the date/time value that does not make sense;
- Alphanumeric string with no data in the required positions.

The value of the control can be checked by using **FxIsValid** API function or *all controls* of the dialog can be validated by a call to the **FxDlgGetInvalid** function.

If a FEDIT control with an invalid value looses focus the special notification message FN_INVALID is sent to its parent. A program can intercept this message to take appropriate action or just ignore it.

If the data must be validated at the time it has been entered and the control is about to loose its focus then you can select the VALIDATE radio button in the FEDIT Options dialog box or specify FS_VALIDATE style. In this case FEDIT *automatically* checks the validity of the input data and does not allow the focus to be removed from the control until a valid data is entered. This mechanism forces a dialog or a window with an invalid control to be temporary system modal and thus should be used with caution.

See also:
FS_VALIDATE style, FxIsValid, FxDlgGetInvalid

Custom control in the program

A program that uses a custom control DLL should load it before any control of that type is created. There are two ways to do it:

- **By using LoadLibrary() and FreeLibrary() API functions.**
  Specify "FEDIT.DLL" as an argument to the LoadLibrary() function.   It is sufficient to load library once at the beginning of the program and free it at the end. You can also load and free DLL every time you need it.
  For example:

  ```
  HANDLE hFeditLib = LoadLibrary("FEDIT.DLL");
  if(hFeditLib >= 32)
  {
          ...
      FreeLibrary(hFeditLib);
  }
  ```

- **By calling any of the extended API functions.** You should link your program with the import library - FEDIT.LIB to be able to call any of these functions. This is a preferred way since it does not require additional coding.   If you do not have any calls to the extended API functions in your program you can insert a call to the FxGetEditMode() anywhere in the program.

Data styles and formats

This section describes FEDIT data styles that are used to specify the data type, the level of validation and the initial value of a control. All FEDIT styles can be combined with any of the standard window and EDIT styles that are valid for a control. Only one FEDIT style from every group can be specified. Here is a brief description of   FEDIT data styles:

Group	Name	Comment
Data style	FS_ALPHANUM	Defines a control that represents an arbitrary formatted string.
	FS_NUMERIC	Defines a control that represents a double value.
	FS_DATE	Defines a control that represents date and/or time.
	FS_BOOLEAN	Defines a control that represents a boolean value.
	FS_EDIT	Defines a control that represents an unformatted string.
Initialization	FS_INITNULL	Creates a control with NULL initial value.
	FS_INITDFLT	Creates a control with default initial value.
Validation	FS_NONULL	Prevents control from loosing focus when it's value is NULL
	FS_VALIDATE	Prevents control from loosing focus when it's value is invalid.

There is a special Formatting Mask that must be created for every data style. Each Formatting mask uses its own set of special characters.

FS_ALPHANUM style

Use **FS_ALPHANUM** style to define a generic character string. This style is used to represent names, phone numbers, social security numbers, licence plates and other fixed alphanumeric strings.
The following formatting characters are used for **FS_ALPHANUM** style:

UPPERCASE:

'A'	Represents any alphabetic character, i.e. 'a'-'z' or 'A'-'Z' range.
'X'	Represents any printable character.
'Z'	Represents any alphanumeric character, this is a combination of 'A' and '9'.

LOWERCASE:

'a'	Same as 'A' for *required* input.
'x'	Same as 'X' for *required* input.
'z'	Same as 'Z' for *required* input.

'9'	Represents a digit (0 - 9).
'0'	Represents a *required* digit (0 - 9).

For example, a formatting mask for zip code may specified as "00000-9999". This format requires user to input first five digits of the zip code and optionally four more.

Note, that lowercase letters allow to input spaces, but **FxIsValid** API function returns FALSE in this case.

Other characters are considered delimiters and display only. Use FS_INITNULL to suppress display of delimiters when there is no data in the control. Delimiters are ignored by **FM_GETTEXT** and **FM_SETTEXT** messages.

See also:
FS_NUMERIC, FS_DATE, FS_BOOLEAN, **FS_EDIT, Field Validation**

Examples:

(999) 999-9999	Phone number
999-99-9999	Social security number
00000-9999	Zip code with first five digits required.

FS_NUMERIC style

Use **FS_NUMERIC** style to define a double value. This style is used to represent amounts, currency, percentage and other numbers with fixed decimal point.
The following formatting characters are used for **FS_NUMERIC** style:

'9'	Represents any digit, i.e. symbol from the '0'-'9' range. This character will be replaced with an actual digit or a space for leading zero.
'S'	Represents a sign. Whenever user presses plus/minus key FEDIT control will place the sign in the appropriate position.
','	Represents a comma delimiter. A comma is not shown if digits to the left of it are zeros.
'.'	Represents a decimal point. The decimal point separates the decimal part of the number from the whole one.

Other symbols are considered delimiters and are display only. Use FS_INITNULL to suppress display of delimiters when there is no data in the control. Delimiters are ignored by **FM_GETTEXT** and **FM_SETTEXT** messages.

See also:
FS_ALPHANUM, **FS_DATE**, **FS_BOOLEAN**, **FS_EDIT**

Examples:
999.99%
S9,999,999,999.9999
$9,999.99
S999,999,999£
{S999}

FS_DATE style

Use **FS_DATE** style todefine a date/time value.
The following formatting characters are used for **FS_DATE** style:

'mm'	Represents a month in a decimal form, e.g '08'.
'dd'	Represents a day of a month.
'yy'	Represents a year without century, e.g. '92' .
'yyyy'	Represents a year with century, e.g. '1992'.
'hh'	Represents an hour.
'xx'	Represents a minute.
'ss'	Represents a second.
'mmm'	Represents a month in a string form, e.g. 'aug'.
'www'	Represents a Day of the Week (e.g. 'tue'). The Day of the Week is set automatically according to the date value and cannot be modified by a user.
'am'	
'pm'	Represents a part of the day. When FEDIT finds any of these strings it represent time in the 12 hours mode.

You may specify a capital letter instead of a lowercase letter.
For decimal formats FEDIT allows to input space in the uppercase position or replace a character with a space when user deletes it. For example, 'Mm/Dd/yy' format suppresses leading zeros for the month and date parts of the date. For string formats the case of the formatting character specifies the case of the character displayed in the position. For example 'Mmm' format specifies a capitalized month name, e.g. 'Aug'. You can specify more characters in the string format todisplay the name of the month or day of the week, e.g. 'Mmmmmmmmm' will display as 'September'.

Other characters and substrings are considered delimiters and are display only.

See also:
FS_ALPHANUM, FS_NUMERIC, FS_BOOLEAN, FS_EDIT

Examples:
mm/dd/yy
Mmm dd, yyyy
dd-mm-yyyy (Www)
hh:nnam
DATE: MM/DD/YYYY   TIME: HH:NN:SS   DOW: Wwwwwwwww

FS_BOOLEAN style

Use **FS_BOOLEAN** style to define a boolean value.
The following formatting characters are used for **FS_BOOLEAN** style:

'Y'	Represents Y/N	(Y is default).
'YYY'	Represents YES/NO	(YES is default).
'N'	Represents Y/N	(N is default).
'NNN'	Represents YES/NO	(NO is default).
'T'	Represents T/F	(T is default).
'TTTTT'	Represents TRUE/FALSE	(TRUE is default).
'F'	Represents T/F	(F is default).
'FFFFF'	Represents TRUE/FALSE	(FALSE is default).

You can combine lowercase and uppercase characters to display the case of a letter in the same position

See also:
FS_ALPHANUM, **FS_DATE**, **FS_NUMERIC**, **FS_EDIT**

Examples:
'Yyy'
'N'
'Fffff'
't'

FS_EDIT style

Use **FS_EDIT** style to define unformatted input.   **FS_EDIT** style is generally used to represent notes, memos and other unformatted data. This style is a similar to the standard EDIT control and is included to provide consistent approach in handling all types of data. You can use all FEDIT API functions and messages with this data style.

See also:
FS_ALPHANUM, FS_DATE, FS_NUMERIC, FS_BOOLEAN

FS_INITNULL

Use **FS_INITNULL** style to initialize the value of control to NULL.

User can enter this value into the FEDIT control by pressing <CTRL-DEL> keys. You may also specify NULL value as a default value for the FEDIT control by selecting the radio button provided in the FEDIT Options dialog box.

If a FEDIT control with NULL value looses focus a special notification message FN_NULL is sent to its parent. A program can intercept this message to take appropriate action or just ignore it.

If data is required in a control then you can select a NOT NULL radio button in the FEDIT Options dialog box or specify FS_NONULL style. In this case FEDIT automatically checks the value of the control and doesn't allow user to remove focus from it until data is entered. This mechanism forces a dialog or a window with an empty control to be temporary system modal and thus should be used with caution.

See also:
Null values**, FS_INITDFLT**

FS_INITDFLT

Use **FS_INITDFLT** style to initialize the value with the default value for the control's style:

Style	**Default value**
FS_ALPHANUM	String with delimiters in the specified positions.
FS_NUMERIC	Zero.
FS_DATE	Current date/time.
FS_BOOLEAN	TRUE for the 'Y' and 'T' formats, FALSE otherwise.
FS_EDIT	Empty string.

See also:

FS_INITNULL

FS_VALIDATE

FEDIT control created with the **FS_VALIDATE** style keeps control in focus until a valid data is entered. The following values are considered invalid:

- Invalid date/time.
- Alphanumeric string with empty *required* positions.

Note:

FS_VALIDATE style forces a dialog or a window with an invalid control temporary into system modal and thus should be used with caution.

See also:

Invalid values, **FS_NONULL**, **FS_ALPHANUM**

FS_NONULL

FEDIT control created with the **FS_NONULL** style keeps control in focus until   some data is entered.

Note:

FS_NONULL style forces a dialog or a window with an empty control to be temporary system modal and thus should be used with caution.

See also:

Null values, **FS_VALIDATE**

Application Programmer Interface

Messages

FEDIT supports all EDIT standard messages and provides the following additional messages that are sent to FEDIT by SendMessage and SendDlgItemMessage functions:

Message	Description
FM_GETFORMAT	Get control format string.
FM_SETFORMAT	Set control format string.
FM_GETNAME	Get control string name.
FM_SETNAME	Set control string name.
FM_GETSTRING	Get control value as an unformatted string.
FM_SETSTRING	Set control value using unformatted string.
FM_GETDOUBLE	Get control value as a double number.
FM_SETDOUBLE	Set control value using a double number.
FM_GETLONG	Get control value as a long number.
FM_SETLONG	Set control value using a long number.
FM_GETTIME	Get control value as a **tm** structure.
FM_SETTIME	Set control value using **tm** structure.
FM_SETNULL	Set control value to NULL .
FM_ISNULL	Compare control value with NULL.
FM_GETFLAG	Get user-defined flag or group of flags.
FM_SETFLAG	Set user-defined flag or group of flags.

FEDIT also provides a set of notification messages that are sent to the FEDIT controls parent if a special event takes place:

Notification Message	Is sent when
FN_TOGGLEINS	User presses Insert key.
FN_DBLCLK	User double clicks on the control.
FN_ERROR	Control with invalid value looses the input focus.
FN_NULL	Control with NULL value looses the input focus.

Control management functions

The functions in this category provide format independent control input/output, type checking and control validation functions and macros.   You can use these procedures along with standard MS-Windows API functions. The following list briefly describes each custom control management function:

Function	Description
FxGetType	Returns data style of a control.
FxGetString	Retrieves a format independent string representation of the controls text.
FxSetString	Sets the text of a control using a format independent string value.
FxGetDouble	Translates the text of a control into a double value.
FxSetDouble	Sets the text of a control with the string representation of the double value.
FxGetStringAs	Retrieves the text of a control as a string in the provided format.
FxSetStringAs	Sets the text of a control using a string in the specified format.
FxGetTime	Retrieves the value of a control as a **tm** structure.
FxSetTime	Sets the text of a control using a **tm** structure.
FxGetEditMode	Returns the current editing mode (Insert/Overwrite).
FxSetEditMode	Sets the editing mode (Insert/Overwrite).
FxGetFormat	Retrieves the format of a control.
FxSetFormat	Sets the format of a control and reformats the controls value.
FxGetLong	Translates the text of a control into a long value.
FxSetLong	Sets the text of a control to a string representation of the long value.
FxGetFlag	Retrieves the flags assosiated with a control.
FxSetFlag	Sets the flags assosiated with a control.
FxGetName	Retrieve the name of a control.
FxSetName	Clears the value of a control by setting it to the NULL.
FxIsFEDIT	Determines whether a control belongs to the FEDIT class.
FxIsValid	Validates the value of a control.
FxIsBool macro	Determines whether a control has a FS_BOOL style.
FxIsDate macro	Determines whether a control has a FS_DATE style.
FxIsEdit macro	Determines whether a control has a FS_EDIT style.
FxIsNumeric macro	Determines whether a control has a FS_NUMERIC style.
FxIsString macro	Determines whether a control has a FS_ALPHANUM style.
FxWasModified macro	Determines whether the text of a control was modified by user.
FxClsModified macro	Clears the modification flag of a control.
FxSetModified macro	Sets the modification flag of a control.

Date and Time functions

The date and time functions allow you to convert date and/or time to a different formats. These procedures are using the next types:

Type	Description
LONG (**time_t**)	Represents a number of seconds elapsed since midnight, January 1, 1970. The negative value specifies the date before 01/01/1970. While this representation is used by the most compilers, some of them may use different starting date. Microsoft C/C++ ver 7.0 (only) uses midnight, December 31, 1899 for this purpose.
LDATE	Represents a number of days elapsed since January 1, 1970. The negative value is used to specify dates before 01/01/1970.
LTIME	Represents a number of seconds elapsed since midnight.
struct tm	This structure is defined in the TIME.H standard C/C++ header file and has next **int** fields:
tm_sec	Seconds
tm_min	Minutes
tm_hour	Hours (0-23)
tm_mday	Day of month (1-31)
tm_mon	Month (0-11)
tm_year	Year (current year minus 1900)
tm_wday	Day of week (0-6)
tm_yday	Day of year (0-365)
tm_isdist	Not used by these functions.

The current version of the FEDIT includes next date/time functions:

Routine	Description
FxTDate	Get **LDATE** value using **time_t**.
FxTTime	Get **LTIME** value using **time_t.**
FxTMake	Combine **LDATE** and **LTIME** into the **time_t** value.
FxTMon	Get month (0 - 11) by a **LDATE** value
FxTDay	Get day of month (1 - 31) by a **LDATE** value
FxTYear	Get year by a **LDATE** value
FxTDow	Get day of week (0 - 6) by a **LDATE** value
FxTMakeDate	Create **LDATE** value using day, month and year.
FxTHour	Get hour (0 - 23) by a **LTIME** value
FxTMin	Get minute (0 - 59) by a **LTIME** value
FxTSec	Get second (0 - 59) by a **LTIME** value
FxTMakeTime	Create **LTIME** value using hour, minute and second.
FxTttol	Convert **struct tm** value into the **LONG** (**time_t**) value.
FxTltot	Convert **LONG** (**time_t**) value into the **tm** structure.
FxTCMonth	Convert a month (0 - 11) into its string representation.
FxTDMonth	Convert string representation of a month into number (0 - 11)
FxTCDow	Convert a day of week (0 - 6) into its string representation.
FxTDDow	Convert string representation of a day of week into number (0 - 6)

FxTIsLeapYear	Check if year is a leap one.

Formatting Functions

FEDIT control library contains procedures and macros that provide the same format conversions that FEDIT control, but can be used independent of any control. By using these functions you can convert virtually any string into any other string, get string value or create a new string using a provided value.

Function	**Description**
FxFmtTranslate	Translates a string from one format into another.
FxFmtGetString	Retrieves a format independent value of the formatted string
FxFmtSetString	Creates a formatted string using a format independent value and a format.
FxFmtGetLong	Retrieves a long representation of the formatted string.
FxFmtSetLong	Creates a formatted string using a long representation and a format.

Dialog Box Functions

The functions in this section deal with a dialog box or a window rather then with a single control. The following list briefly describes each dialog function:

Function	Description
FxDlgGetByName	Returns the handle to the control by it's name.
FxDlgGetInvalid	Returns the handle of the first invalid control in the dialog or NULL, if there are no invalid controls.
FxDlgGetModified	Returns the handle of the first modified control in the dialog box or NULL, if there are no modified controls.
FxDlgClrModified	Clears modification flag for all FEDIT controls in the dialog box.
FxDlgSetNull	Clears all FEDIT controls in the dialog box by sending FM_SETNULL message to all of them.
FxDlgGetNull	Returns the handle of the first control with Null value in the dialog or NULL, if there are no empty controls.