

---

## **AIMMS Function Reference - Dialog Functions**

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit [www.aimms.com](http://www.aimms.com)

# Dialog Functions

AIMMS supports the following functions for simple interaction with the end user.

- DialogAsk
- DialogError
- DialogGetColor
- DialogGetDate
- DialogGetElement
- DialogGetElementByData
- DialogGetElementByText
- DialogGetNumber
- DialogGetPassword
- DialogGetString
- DialogMessage
- DialogProgress
- StatusMessage

---

## DialogAsk

The procedure `DialogAsk` displays a small dialog box containing a message and two or three buttons. Usually these buttons are an **OK** and **Cancel**, or **Yes**, **No** and **Cancel**, but they can contain any text you want. The procedure returns the number of the button that is pressed by the user.

```
DialogAsk(  
    message,      ! (input) string expression  
    button1,     ! (input) string expression  
    button2,     ! (input) string expression  
    [button3]    ! (optional) string expression  
    [title]      ! (optional) title of dialog box  
)
```

### Arguments:

#### *message*

A scalar string expression containing the text you want to display in the dialog box.

#### *button1*

A scalar string expression containing the text of the first button.

#### *button2*

A scalar string expression containing the text of the second button.

#### *button3 (optional)*

A scalar string expression containing the text of the third button. If this argument is omitted then the dialog box will only show two buttons.

#### *title*

A scalar string expression containing the text that you want to appear in the title of the dialog box.

### Return value:

The procedure returns the number of the button that is pressed: 1 for the first button, 2 for the second button or 3 for the third button.

### Remarks:

If the user presses the **Esc** key, or closes the dialog box via the [x] in the top right corner, then this is interpreted as pressing the last button in the dialog box (which is usually the **Cancel** button).

### See also:

The procedures `DialogMessage`, `DialogError`.

---

## DialogError

The procedure `DialogError` displays a small dialog box containing a specified error message and an **OK** button. The execution will be halted until the user presses the **OK** button.

```
DialogError(  
    message,      ! (input) string expression  
    [title]      ! (optional) title of dialog box  
)
```

### Arguments:

*message*

A scalar string expression containing the text you want to display in the dialog box.

*title*

A scalar string expression containing the text that you want to appear in the title of the dialog box.

### Remarks:

The procedures `DialogMessage` and `DialogError` only differ in the icon that is displayed at the left side of the dialog box.

### See also:

The procedures `DialogMessage`, `DialogAsk`, `DialogProgress`.

---

## DialogGetColor

The procedure `DialogGetColor` displays a standard Windows color selection dialog box. The procedure returns the color (RGB values) selected by the user.

```
DialogGetColor(  
    r,          ! (input/output) scalar numerical parameter  
    g,          ! (input/output) scalar numerical parameter  
    b          ! (input/output) scalar numerical parameter  
)
```

### Arguments:

*r*

A scalar numerical parameter containing the red value of the selected color.

*g*

A scalar numerical parameter containing the green value of the selected color.

*b*

A scalar numerical parameter containing the blue value of the selected color.

### Return value:

The procedure returns 1 if the user completed the color selection dialog box successfully, or 0 otherwise.

---

## DialogGetDate

The procedure `DialogGetDate` displays a standard Windows date selection dialog box. The procedure returns the date (in the specified format) selected by the user.

```
DialogGetDate(  
    title,          ! (input) string expression  
    format,        ! (input) string expression  
    date,          ! (input/output) scalar string parameter  
    [nr_rows,]    ! (optional) integer expression  
    [nr_columns]  ! (optional) integer expression  
)
```

### Arguments:

#### *title*

A scalar string expression containing the text you want to display in the title of the dialog box.

#### *format*

A scalar string expression containing the date format of the *date* argument.

#### *date*

A scalar string parameter in which the selected date is returned according to the date format specified in *format*.

#### *nr\_rows* (optional)

A scalar integer expression in the range 1, ..., 3 containing the number of rows to be displayed in the date selection dialog box.

#### *nr\_columns* (optional)

A scalar integer expression in the range 1, ..., 4 containing the number of columns to be displayed in the date selection dialog box.

### Return value:

The procedure returns 1 if the user completed the date selection dialog box successfully, or 0 otherwise.

### Remarks:

If the *date* argument contains a valid date according to the format specified in *date-format*, AIMMS will set the initial date in the date selection dialog box equal to the specified date.

### See also:

The date format specification components are discussed in full detail in Section [31.7.1](#) of the Language Reference.

---

## DialogGetElementByData

The procedure `DialogGetElementByData` is an extension of the procedure `DialogGetElementByText`. Instead of only showing a list box with only a single string per element, this procedure allows you to show a list box with multiple columns of text per element. The text that is displayed in each column is specified via a 2-dimensional string parameter. The first dimension of this parameter corresponds to the rows of the list box, the second dimension corresponds to the column in the listbox.

```
DialogGetElementByData(  
    title,          ! (input) string expression  
    reference,     ! (input/output) scalar element parameter  
    element_data   ! (input) 2-dimensional string parameter  
)
```

### Arguments:

#### *title*

A scalar string expression containing the text you want to display as title of the dialog box.

#### *reference*

A scalar element parameter. When creating the dialog box, the range set of this parameter is used to fill the list with elements, and the current value of the element parameter will be initially selected. On return, this parameter will refer to the selected element.

#### *element\_data*

A 2-dimensional string parameter. The first index in its domain should matches the range set of the element parameter *reference*, the second index defines the number of columns that are shown. Instead of the element names, the dialog box will display multiple columns of text derived from this parameter.

### Return value:

The procedure returns 1 if the user has pressed the **OK** button, and 0 if he has pressed the **Cancel** button.

### See also:

The procedures `DialogGetElement`, `DialogGetElementByText`.

---

## DialogGetElement

The procedure `DialogGetElement` displays a dialog box in which the user can select an element from a list of set elements.

```
DialogGetElement(  
    title,          ! (input) string expression  
    reference      ! (input/output) scalar element parameter  
)
```

### Arguments:

#### *title*

A scalar string expression containing the text you want to display as title of the dialog box.

#### *reference*

A scalar element parameter. When creating the dialog box, the range set of this parameter is used to fill the list with elements, and the current value of the element parameter will be initially selected. On return, this parameter will refer to the selected element.

### Return value:

The procedure returns 1 if the user has pressed the **OK** button, and 0 if he has pressed the **Cancel** button.

### See also:

The procedures [DialogGetElementByText](#), [DialogGetElementByData](#), [DialogGetNumber](#).

---

## DialogGetElementByText

The procedure `DialogGetElementByText` displays a dialog box in which the user can select an element from a set. However, other than `DialogGetElement`, this procedure does not show a list of element names but a list of strings, which are given as a separate argument to the procedure.

```
DialogGetElementText(  
    message,      ! (input) string expression  
    reference,    ! (input/output) scalar element parameter  
    element_text  ! (input) 1-dimensional string parameter  
)
```

### Arguments:

#### *message*

A scalar string expression containing the text you want to display as title of the dialog box.

#### *reference*

A scalar element parameter. When creating the dialog box, the range set of this parameter is used to fill the list with elements, and the current value of the element parameter will be initially selected. On return, this parameter will refer to the selected element.

#### *element\_text*

A 1-dimensional string parameter, with a domain that matches the range set of the element parameter *reference*. Instead of the element names, the dialog box will display the corresponding strings of this parameter.

### Return value:

The procedure returns 1 if the user has pressed the **OK** button, and 0 if he has pressed the **Cancel** button.

### See also:

The procedures `DialogGetElement`, `DialogGetElementByData`.

---

## DialogGetNumber

The procedure `DialogGetNumber` displays a small dialog box in which the user can enter a single numerical value. The dialog box remains on the screen (and thus halts the execution) until the user presses either the **OK** or the **Cancel** button.

```
DialogGetNumber(  
    message,      ! (input) string expression  
    reference,    ! (input/output) scalar numerical identifier  
    [decimals,]  ! (optional) integer  
    [title]      ! (optional) string expression  
)
```

### Arguments:

#### *message*

A scalar string expression containing the text you want to display in front of the edit field.

#### *reference*

A scalar identifier. When creating the dialog box, its value is used to fill the edit field. After the user presses the **OK** button, the edited value is returned through this argument.

#### *decimals*

A integer expression to indicate the number of decimals that is displayed initially.

#### *title*

A scalar string expression containing the text that you want to appear in the title of the dialog box.

### Return value:

The procedure returns 1 if the user has pressed the **OK** button, and 0 if he has pressed the **Cancel** button.

### See also:

The procedures `DialogGetString`, `DialogGetElement`.

---

## DialogGetPassword

The procedure `DialogGetPassword` displays a small dialog box in which the user can enter a password string. In the dialog box the string is presented by a sequence of asterisks. The dialog box remains on the screen (and thus halts the execution) until the user presses either the **OK** or the **Cancel** button.

```
DialogGetPassword(  
    message,      ! (input) string expression  
    password,    ! (input/output) scalar string parameter  
    [title]      ! (optional) string expression  
)
```

### Arguments:

#### *message*

A scalar string expression containing the text you want to display in front of the edit field.

#### *password*

A scalar string valued identifier containing the password. When creating the dialog box, its value is used to fill the edit field. After the user presses the **OK** button, the edited password string is returned through this argument.

#### *title*

A scalar string expression containing the text that you want to appear in the title of the dialog box.

### Return value:

The procedure returns 1 if the user has pressed the **OK** button, and 0 if he has pressed the **Cancel** button.

### See also:

The procedure [DialogGetString](#).

---

## DialogGetString

The procedure `DialogGetString` displays a small dialog in which the user can enter a text string. The dialog remains on the screen (and thus halts the execution) until the user presses either the **OK** or the **Cancel** button.

```
DialogGetString(  
    message,      ! (input) string expression  
    reference,    ! (input/output) scalar string parameter  
    [title]      ! (optional) string expression  
)
```

### Arguments:

#### *message*

A scalar string expression containing the text you want to display in front of the edit field.

#### *reference*

A scalar string valued identifier. When creating the dialog, its value is used to fill the edit field. After the user presses the **OK** button, the edited string is returned through this argument.

#### *title*

A scalar string expression containing the text that you want to appear in the title of the dialog box.

### Return value:

The procedure returns 1 if the user has pressed the **OK** button, and 0 if he has pressed the **Cancel** button.

### See also:

The procedures `DialogGetNumber`, `DialogGetPassword`, `DialogGetElement`.

---

## DialogMessage

The procedure `DialogMessage` displays a small dialog box containing a specified informational message and an **OK** button. The execution will be halted until the user presses the **OK** button.

```
DialogMessage(  
  message,      ! (input) string expression  
  [title]      ! (optional) string expression  
)
```

### Arguments:

*message*

A scalar string expression containing the text you want to display in the dialog box.

*title*

A scalar string expression containing the text that you want to appear in the title of the dialog box.

### Remarks:

The procedures `DialogMessage` and `DialogError` only differ in the icon that is displayed at the left side of the dialog box

### See also:

The procedures `DialogError`, `DialogAsk`.

---

## DialogProgress

The procedure `DialogProgress` displays a small dialog box containing a specified message and a progress bar that can indicate how much of a specific task has already been processed. This dialog box will not halt the execution, and you can call the procedure sequentially during a timely task to change either the displayed message or the length of the progress bar.

```
DialogProgress(  
    message,          ! (input) string expression  
    [percentage]     ! (optional) integer expression  
)
```

### Arguments:

*message*

A scalar string expression containing the text you want to display in the dialog box.

*percentage (optional)*

A scalar value between 0 and 100. It is used to set the length of the progress bar at the bottom of the dialog box. If this argument is omitted then the progress bar is not displayed.

### Remarks:

The progress dialog box does not adjust the length of the progress bar itself, so you must do it yourself by sequentially calling the procedure with an increasing percentage. The progress dialog box is automatically removed from the screen if the execution terminates. If you want to remove the dialog box yourself, then you should call `DialogProgress` with an empty message string: `DialogProgress("")`.

### See also:

The procedures `DialogMessage`, `DialogError`, `DialogAsk`.

---

## StatusMessage

With the procedure `StatusMessage` you can display a short message in the status bar at the bottom of the AIMMS window.

```
StatusMessage(  
    message      ! (input) string expression  
)
```

### Arguments:

*message*

A scalar string expression containing the text you want to display in the status bar.

### Remarks:

If you have set the status bar to be hidden (via the project options), then the message will not be visible to the user.

### See also:

The procedures [DialogMessage](#), [DialogProgress](#).