

Triggering User-Defined Events

This document covers the following topics:

- Introduction
 - Passing Parameters to the Dialog
-

Introduction

Aside from standard events, such as before-open, you may define user-defined events for dialogs. User-defined events are useful whenever it is necessary for one dialog to cause an action to occur in another dialog.

A user-defined event occurs whenever you have specified a `SEND EVENT` statement in dialog A with the name of a user-defined event in the target dialog B. This target dialog B for which you wish to trigger the user-defined event must already be active. You can activate dialog B by using the `OPEN DIALOG` statement. If you do not issue the `OPEN DIALOG` statement first, the `SEND EVENT` statement will cause a runtime error.

You can define your own events for dialogs by choosing the **New** button in the **Events** dialog event handler menu or from the dialog's context menu. Enter any name for your newly-defined event and specify the corresponding event section. It is recommended that this name begin with "#" to distinguish your event from predefined events.

During execution of an event handler, the `SEND EVENT` statement triggers the user-defined event handler in a different dialog. After this user-defined event handler has been executed, control will be returned to the previous dialog, whose execution will resume at the statement following the `SEND EVENT` statement. This can be compared to a `CALLNAT` statement that causes a subprogram to be executed.

Similar to the `OPEN DIALOG` statement, parameters may be passed to the dialog. In order to pass parameters selectively (*PARAMETERS-clause*), you have to specify the name of the dialog in addition to the identifier of the dialog (*operand2*).

The `SEND EVENT` statement must not trigger an event in a dialog that is about to process an event. This is the case, for example, when dialog A sends an event to dialog B and the event handler in dialog B sends an event to dialog A which has not yet finished its event handling. A similar case is when dialog A opens dialog B and the before-open or after-open event contains a `SEND EVENT` back to dialog A.

To trigger a user-defined event, you specify the following syntax:

```
SEND EVENT operand1 TO [DIALOG-ID] operand2
  [ [ WITH operand3... USING [DIALOG] 'dialog-name' WITH PARAMETRS-clause ] ]
```

Operands

Operand1 is the name of the event to be sent.

Operand2 is the identifier of the dialog receiving the user-defined event and must be defined with format/length I4. You can retrieve this identifier, for example, by querying the value of #DLG\$PARENT.CLIENT-DATA.

Passing Parameters to the Dialog

It is possible to pass parameters to the dialog receiving the user event.

As *operand3* you specify the parameters which are passed to the dialog.

With the *PARAMETERS-clause*, parameters may be passed selectively.

PARAMETERS-clause

PARAMETERS [*parameter-name = operand3*]**_ END-PARAMETERS**

Anmerkung:

You may only use the *PARAMETERS-clause* if the target dialog is cataloged.

Dialog-name is the name of the dialog receiving the user-defined event.

When you use only *operand3* to pass parameters, it might look like this:

```
/* The following parameters are defined in the dialog's
/* parameter data area:
1 #DLG-PARM1 (A10)
1 #DLG-PARM2 (A10)
1 #DLG-PARM3 (A10)
1 #DLG-PARM4 (A10)
/* When sending the user-defined event, pass the operands #MYPARM1 'MYPARM2' to
the parameters #DLG-PARM1 and #DLG-PARM2:
SEND EVENT 'MYEVENT' TO #DLG$DIA-ID WITH #MYPARM1 'MYPARM2'
```

When you use the *PARAMETERS-clause*, the user-defined event might look like this:

```
/* The following parameters are defined in the dialog's
/* parameter data area:
1 #DLG-PARM1 (A10)
1 #DLG-PARM2 (A10)
1 #DLG-PARM3 (A10)
1 #DLG-PARM4 (A10)
/* When sending the user-defined event, the operand #MYPARM2 is passed to the
/* parameter #DLG-PARM2 and the operand 'MYPARM3' is passed to the parameter
/* #DLG-PARM3:
SEND EVENT 'MYEVENT' TO #DLG$DIA-ID
  USING DIALOG 'MYDIALOG'
  WITH PARAMETERS
    #DLG-PARM3='MYPARM3'
    #DLG-PARM2=#MYPARM2
  END-PARAMETERS
```

To avoid format/length conflicts between operands passed and their parameter definitions, see the BY VALUE option of the DEFINE DATA statement in the *Statements* documentation.