

# Executing Standardized Procedures

For procedures frequently needed in event-driven applications, the following is available:

- a set of PROCESS GUI statement actions and
- a set of NGU-prefixed subprograms and dialogs in library SYSTEM.

Examples for frequently needed procedures are starting up a message box, reading the lines entered into an edit area control, or dynamically creating dialog elements.

For your convenience, the local data areas NGULKEY1 and NGULFCT1 are automatically included in the list of local data areas used by any new dialog.

- NGULFCT1 is necessary to use the NGU-prefixed subprograms and dialogs;
- NGULKEY1 lists reserved keywords to be used in any event-handler code. This enables you to refer to certain attribute values by the more meaningful keyword rather than by the numeric IDs. It also enables you to use meaningful dialog element names as parameters.

For more information on the PROCESS GUI statement actions, subprograms and dialogs available, and on the parameters that can be passed, refer to the Dialog Component Reference documentation.

## PROCESS GUI Statement

```

PROCESS GUI ACTION action-name WITH { operand1...
                                     {PARAMETERS-clause}
                                     [GIVING operand2 ]

```

Operand	Possible Structure C S A G N	Possible Formats A N P I F B D T L C	Reference Permitted	Dynamic Definition
Operand1	X X X	X X X X X X X X X X	X	
Operand2	X	X X X	X	

The PROCESS GUI statement is used to perform an action. An action in this context is a procedure frequently needed in event-driven applications.

As *action-name*, you specify the name of the action to be invoked.

As *operand1*, you specify the parameter(s) to be passed to the action. The parameters are passed in the sequence in which they are specified.

For the action "ADD", you can also pass parameters by name (instead of position); to do so, you use the *PARAMETERS-clause*:

```
PARAMETERS [parameter-name =operand1 ]_ END-PARAMETERS
```

This clause can only be used for the action "ADD", not for any other action.

As *operand2*, you can specify a field to receive the response code from the invoked action after the action has been performed.

[Back to Event-Driven Programming Techniques.](#)