PROCESS PAGE

This chapter covers the following topics:

- Function
- Syntax 1 - PROCESS PAGE
- Syntax 2 - PROCESS PAGE USING
- Syntax 3 - PROCESS PAGE UPDATE
- Syntax 4 - PROCESS PAGE MODAL
- Examples

Function

The PROCESS PAGE statement constitutes a general interface description to an external rendering engine, such as Natural for Ajax, thus linking the Natural internal data representation with an external data representation. Via this link, data and events, but no rendering information, are sent to and returned from an external, browser-based application.

For further information, refer to the *Natural for Ajax* documentation.

Syntax 1 - PROCESS PAGE

```
PROCESS PAGE [(parameter)] operand1
 [WITH PARAMETERS
  {[NAME] operand3 [VALUE] operand4 [(parameters)]} ...
 END-PARAMETERS]
 [GIVING operand11]
```

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Belongs to Function Group: *Screen Generation for Interactive Processing*

Syntax Description - Syntax 1

Syntax 1 of the PROCESS PAGE statement is normally only used inside a Natural adapter. An adapter is a Natural object that forms the interface between Natural application code and web page. It is automatically created/updated by Natural for Ajax when the layout is saved.

Note:

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1	C	S				A	U											yes	no
operand2		S	A												C			no	no
operand3	C	S				A	U											yes	no
operand4	C	S	A			A	U	N	P	I	F	B	D	T	L			yes	yes
operand5		S	A												C			no	no
operand11		S								I4								yes	yes

Syntax Element Description:

Syntax Element	Description
parameter	**Attribute Control Variable(s):** The parameter CV, enclosed within parentheses, may be specified to reference one or more attribute control variables as specified in *operand2*: (CV=*operand2*) See also *Logical Condition Criteria*, *MODIFIED Option - Check whether Field Content has been Modified* in the *Programming Guide*.
operand1	**External Page Layout Name:** *operand1* contains the name of the external page layout.
operand2	*operand2* contains the name of the attribute control variable, must be of format C and must be either a scalar or a single array occurrence.
operand3	**Name of Attribute Control Variable(s):** *operand3* contains the name(s) of the external data field(s) *operand4* will be transferred to/from.
operand4	**Name(s) of Data Field(s):** *operand4* contains the name(s) of the Natural data field(s) which will be transferred.

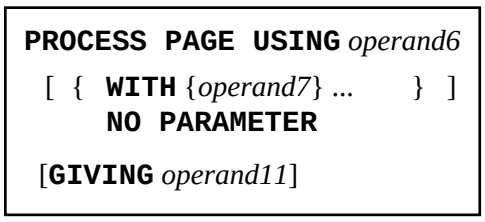
Syntax Element	Description
parameters	**Parameters:** One or more parameters, enclosed within parentheses, may be specified immediately after *operand4*: EM or EMU — Edit mask used during data transfer. For further information, see the session parameter EM in the *Parameter Reference*. For details on Unicode edit masks, see the session parameter EMU in the *Parameter Reference*. CV — The parameter CV, enclosed within parentheses, may be specified immediately after *operand4* to reference one or more attribute control variables as specified in *operand5*: (CV=*operand5*) See also *Logical Condition Criteria, MODIFIED Option - Check whether Field Content has been Modified* in the *Programming Guide*.

Syntax Element	Description
operand5	**Name of Attribute Control Variable:** *operand5* contains the name of the attribute control variable. The variable must be of format C. If *operand4* is a scalar or a single array occurrence, *operand5* must be • a scalar • or a single array occurrence. If *operand4* is the full range of an array of dimension 1, *operand5* must be • a scalar • or a single array occurrence • or the full range of an array of dimension 1 with the same size. If *operand4* is the full range of an array of dimension 2, *operand5* must be • a scalar • or a single array occurrence • or the full range of an array of dimension 2 with the same size in both dimensions • or the full range of an array of dimension 1 with the same size that *operand4* has in dimension 1. If *operand4* is the full range of an array of dimension 3, *operand5* must be • a scalar • or a single array occurrence • or the full range of an array of dimension 3 with the same size in all three dimensions • or the full range of an array of dimension 2 with the same size that *operand4* has in dimension 1 and 2 • or the full range of an array of dimension 1 with the same size that *operand4* has in dimension 1.
GIVING *operand11*	**GIVING Clause:** *operand11* contains the Natural error if the request could not be performed.

Example of an adapter which has been created by Natural for Ajax:

```
* PAGE1: PROTOTYPE       --- CREATED BY Natural for Ajax ---
* PROCESS PAGE USING 'XXXXXXXX' WITH
* INFOPAGENAME RESULT YOURNAME
DEFINE DATA PARAMETER
1 INFOPAGENAME (U) DYNAMIC
1 RESULT (U) DYNAMIC
1 YOURNAME (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE U'/njxdemos/helloworld' WITH
PARAMETERS
 NAME U'infopagename'
  VALUE INFOPAGENAME
 NAME U'result'
  VALUE RESULT
 NAME U'yourname'
  VALUE YOURNAME
END-PARAMETERS
*
*  TODO: Copy to your calling program and implement.
/*/*( DEFINE EVENT HANDLER
* DECIDE ON FIRST *PAGE-EVENT
*  VALUE U'nat:page.end'
*   /* Page closed.
*   IGNORE
*  VALUE U'onHelloWorld'
*   /* TODO: Implement event code.
*   PROCESS PAGE UPDATE FULL
*  NONE VALUE
*   /* Unhandled events.
*   PROCESS PAGE UPDATE
* END-DECIDE
/*/*) END-HANDLER
*
END
```

Syntax 2 - PROCESS PAGE USING



For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Belongs to Function Group: *Screen Generation for Interactive Processing*

Syntax Description - Syntax 2

This syntax is used to perform rich GUI input/output processing using an object of type adapter that has been generated from a page layout created with Natural for Ajax or a similar tool.

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand6	C	S				A												yes	no
operand7		S	A	G		A	U	N	P	I	F	B	D	T	L	C		yes	yes
operand11		S								I4								yes	yes

Syntax Element Description:

Syntax Element	Description
USING *operand6*	**Adapter Name:** Invokes an adapter definition which has been previously stored in a Natural system file. See also *Processing a Rich GUI Page - Adapter* in the *Programming Guide*. The adapter name (*operand4*) may be a 1 to 8 character alphanumeric constant or user-defined variable. If a variable is used, it must have been defined previously. The adapter name may contain an ampersand (&); at execution time, this character will be replaced by the current value of the Natural system variable *LANGUAGE. This feature is provided for historical reasons. If you need multi-lingual adapters, use the capability of the external rendering system (for example, Natural for Ajax). **Note:** New applications do not need the ampersand feature to be multilingual. Pages designed, for example, using Natural for Ajax, can hold multilingual information as part of the layout design. See *Multi Language Management* in the *Natural for Ajax* documentation.
operand7	**Field Specification:** A list of database fields and/or user-defined variables, all of which must have been defined previously. The fields must agree in number, sequence, format, length and (for arrays) number of occurrences with the fields in the referenced adapter; otherwise, an error occurs. When the content of a database field is modified as a result of PROCESS PAGE processing, only the value as contained in the data area is modified. In order to change the content of the database, appropriate database UPDATE/STORE statements must be used. See *PROCESS PAGE USING Fields Defined in the Program*.
NO PARAMETER	**NO PARAMETER Option:** See *PROCESS PAGE USING without Parameter List*.

Syntax Element	Description
`GIVING` *operand11*	**GIVING Clause:** *operand11* contains the Natural error if the request could not be performed. **Note:** The `GIVING` clause interrupts the common Natural error handling, if an error occurs while the adapter object is being activated or executed. Instead of back-tracking the Natural modules in order to find an `ON ERROR` clause, the Natural error code is passed to a variable (*operand9*) and execution is continued with the next statement.

PROCESS PAGE USING without Parameter List

The following requirements must be met when `PROCESS PAGE USING` is used without parameter list:

- The adapter name (*operand7*) must be specified as an alphanumeric constant (up to 8 characters).
- The adapter used in this manner must have been created prior to the compilation of the program which references the adapter.
- The names of the fields to be processed are taken dynamically from the adapter source definition at compilation time. The field names used in both program and adapter must be identical.
- All fields to be referenced in the `PROCESS PAGE` statement must be accessible at that point.
- In structured mode, fields must have been defined previously (database fields must be properly referenced to processing loops or views).
- When the page layout is changed, the programs using the adapter need not be recataloged. However, when array structures or names, formats/lengths of fields are changed, or fields are added/deleted in the adapter, the programs using the adapter must be recataloged.
- The adapter source must be available at program compilation; otherwise, the `PROCESS PAGE USING` statement cannot be compiled.

Note:
If you wish to compile the program even if the adapter is not yet available, specify `NO PARAMETER`. The `PROCESS PAGE USING` statement can then be compiled even if the adapter is not yet available.

PROCESS PAGE USING Fields Defined in the Program

By specifying the names of the fields to be processed within the program (*operand7*), it is possible to have the names of the fields in the program differ from the names of the fields in the adapter.

The sequence of fields in the program must match the sequence in the adapter. If you use Natural maps as adapter objects, note that the map editor sorts the fields as specified in the map in alphabetical order by field name. For more information, see the map editor description in your *Editors* documentation.

A check is made at execution time to ensure that the format and length of the fields as specified in the program match the fields as specified in the adapter. If both layouts do not agree, an error message is produced.

Syntax 3 - PROCESS PAGE UPDATE

PROCESS PAGE UPDATE [**FULL**] []
event-option[**GIVING** *operand11*]

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Belongs to Function Group: *Screen Generation for Interactive Processing*

Syntax Description - Syntax 3

The PROCESS PAGE UPDATE statement is used to return to and re-execute a PROCESS PAGE statement. It is generally used to return from event processing that the data input processing of the preceding PROCESS PAGE statement was incomplete.

Note:
No INPUT, WRITE, PRINT or DISPLAY statements may be executed between a PROCESS PAGE statement and its corresponding PROCESS PAGE UPDATE statement.

The PROCESS PAGE UPDATE statement, when executed, repositions the program status regarding subroutine, special condition and loop processing as it existed when the PROCESS PAGE statement was executed (as long as the status of the PROCESS PAGE statement is still active). If the loop was initiated after the execution of the PROCESS PAGE statement and the PROCESS PAGE UPDATE statement is within this loop, the loop will be discontinued and then restarted after the PROCESS PAGE statement has been reprocessed as a consequence of the PROCESS PAGE UPDATE statement.

If a hierarchy of subroutines was invoked after the execution of the PROCESS PAGE statement, and the PROCESS PAGE UPDATE is performed within a subroutine, Natural will trace back all subroutines automatically and reposition the program status to that of the PROCESS PAGE statement.

It is not possible, however, to have a PROCESS PAGE statement positioned within a loop, a subroutine or a special condition block, and then execute the PROCESS PAGE UPDATE statement when the status under which the PROCESS PAGE statement was executed has already been terminated. An error message will be produced and program execution terminated when this error condition is detected.

Operand Definition Table:

Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
operand11	S	I4	yes	yes

Syntax Element Description:

Syntax Element	Description
FULL	**FULL Option:** If you specify the FULL option in a PROCESS PAGE UPDATE statement, the corresponding PROCESS PAGE statement will be re-executed fully: ● With an ordinary PROCESS PAGE UPDATE statement (without FULL option), the contents of variables that were changed between the PROCESS PAGE and PROCESS PAGE UPDATE statement will not be displayed; that is, all variables on the screen will show the contents they had when the PROCESS PAGE statement was originally executed. ● With a PROCESS PAGE UPDATE FULL statement, all changes that have been made after the initial execution of the PROCESS PAGE statement will be applied to the PROCESS PAGE statement when it is re-executed; that is, all variables on the screen contain the values they had when the PROCESS PAGE UPDATE statement was executed.
event-option	**EVENT Option:** See *EVENT Option* below.
GIVING (*operand11*)	**GIVING Clause:** *operand11* contains the Natural error if the request could not be performed.

Example User Program Fragment:

```
PROCESS PAGE USING "HELLOW-A"
*
/*( DEFINE EVENT HANDLER
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
 VALUE U'onHelloWorld'
  COMPRESS "HELLO WORLD" YOURNAME INTO RESULT
  PROCESS PAGE UPDATE FULL
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
/*) END-HANDLER
```

EVENT Option

```
AND SEND EVENT operand8
 [WITH PARAMETERS
  {[NAME] operand9 [VALUE] operand10 [ { (EMU=value) } ]}...
                                       { (EM=value)  }
 END-PARAMETERS]
```

With this option, you can advise the external I/O system to run specific functions. These functions are part of the external I/O system or implement special functions regarding the output processing as setting of focus, displaying message boxes, etc.

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand8	C	S				A	U											yes	no
operand9	C	S				A	U											yes	no
operand10	C	S	A			A	U	N	P	I	F	B	D	T	L			yes	yes

Syntax Element Description:

Syntax Element	Description
`AND SEND EVENT` *operand8*	**Event Requested from the External I/O System:** Depending on the implementation of the external I/O system, events are available, refer to *Sending Events to the User Interface* in the *Natural for Ajax* documentation.
`WITH PARAMETERS`	**WITH PARAMETERS Clause:** With this clause, you can specify the following:
`NAME` *operand9*	**External Data Field Name:** *operand9* contains the external name of the data fields *operand10* will be transferred to/from.
`VALUE` *operand10*	**Natural Data Fields:** *operand10* contains the Natural data fields which will be transferred.
`EMU=` `EM=`	**Edit Mask:** Edit mask used during data transfer. For details on edit masks, see the session parameter `EM` in the *Parameter Reference*. For details on Unicode edit masks, see the session parameter `EMU` in the *Parameter Reference*.
`END-PARAMETERS`	**End of WITH PARAMETERS Clause:** The Natural reserved word `END-PARAMETERS` must be used to end the `WITH PARAMETERS` clause.

Syntax 4 - PROCESS PAGE MODAL

```
PROCESS PAGE MODAL
  statement ...
END-PROCESS
```

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statements: `ESCAPE` | `PROCESS PAGE`

Belongs to Function Group:

- Loop Execution

- Screen Generation for Interactive Processing

Syntax Description - Syntax 4

The PROCESS PAGE MODAL statement is used to initiate a processing block and to control the lifetime of a modal rich GUI window.

Note:
The PROCESS PAGE MODAL statement is not valid in batch mode.

When the PROCESS PAGE MODAL statement block is entered, data from Report 0 which is not displayed yet will be displayed first.

The system variable *PAGE-LEVEL is incremented and the opening of a modal page is prepared. The physical opening of the modal page will be performed with the next PROCESS PAGE USING *'adapter'* WITH statement.

Note:
No PRINT, WRITE, INPUT or DISPLAY statements referring to Report 0 may be executed between a PROCESS PAGE MODAL statement and its corresponding END-PROCESS statement.

Leaving the PROCESS PAGE MODAL statement block causes the following actions to be performed:

- if a modal page has been opened for this level, the modal page will be closed;
- the system variable *PAGE-LEVEL is decremented and the system variable *PAGE-EVENT is set back to the value it had before the statement block was entered.

Syntax Element Description:

Syntax Element	Description
statement	**Statement(s) to be Executed:** In place of *statement*, you must supply one or several suitable statements, depending on the situation. If you do not want to supply a specific statement, you may insert the IGNORE statement.
END-PROCESS	**End of PROCESS PAGE MODAL Statement:** The Natural reserved word END-PROCESS must be used to end the PROCESS PAGE MODAL statement.

Example:

```
* Name: First Demo/Open modal!
*
PROCESS PAGE USING "EMPTY-A"
*
/*( DEFINE EVENT HANDLER
DECIDE ON FIRST *PAGE-EVENT
  VALUE U'nat:page.end', U'onClose'
    /* Page closed.
    IGNORE
```

```
  VALUE U'onNextLevel'
    PROCESS PAGE MODAL
      FETCH RETURN "EMPTY-P"
    END-PROCESS
    PROCESS PAGE UPDATE
  NONE VALUE
    PROCESS PAGE UPDATE
END-DECIDE
/*) END-HANDLER
END
```

Examples

Further examples of using the PROCESS PAGE statement are contained in library SYSEXNJX.