Processing of Date Information

This chapter covers various aspects concerning the handling of date information in Natural applications.

The following topics are covered:

- Edit Masks for Date Fields and Date System Variables
- Default Edit Mask for Date - DTFORM Parameter
- Date Format for Alphanumeric Representation - DF Parameter
- Date Format for Output - DFOUT Parameter
- Date Format for Stack - DFSTACK Parameter
- Year Sliding Window - YSLW Parameter
- Combinations of DFSTACK and YSLW
- Year Fixed Window
- Date Format for Default Page Title - DFTITLE Parameter

Edit Masks for Date Fields and Date System Variables

If you wish the value of a date field to be output in a specific representation, you usually specify an edit mask for the field. With an edit mask, you determine character by character what the output is to look like.

If you wish to use the current date in a specific representation, you need not define a date field and specify an edit mask for it; instead you can simply use a *date system variable*. Natural provides various date system variables, which contain the current date in different representations. Some of these representations contain a 2-digit year component, some a 4-digit year component.

For more information and a list of all date system variables, see the *System Variables* documentation.

Default Edit Mask for Date - DTFORM Parameter

The profile parameter `DTFORM` determines the default format used for dates as part of the default title on Natural reports, for date constants and for date input.

This date format determines the sequence of the day, month and year components of a date, as well as the delimiter characters to be used between these components.

Possible `DTFORM` settings are:

Setting	Date Format *	Example
DTFORM=I	*yyyy-mm-dd*	2005-12-31
DTFORM=G	*dd.mm.yyyy*	31.12.2005
DTFORM=E	*dd/mm/yyyy*	31/12/2005
DTFORM=U	*mm/dd/yyyy*	12/31/2005

* *dd* = day, *mm* = month, *yyyy* = year.

The DTFORM parameter can be set in the Natural parameter module/file or dynamically when Natural is invoked. By default, DTFORM=I applies.

Date Format for Alphanumeric Representation - DF Parameter

If an edit mask is specified, the representation of the field value is determined by the edit mask. If no edit mask is specified, the representation of the field value is determined by the session parameter DF in combination with the profile parameter DTFORM.

With the DF parameter, you can choose one of the following date representations:

DF=S	8-byte representation with 2-digit year component and delimiters (*yy-mm-dd*).
DF=I	8-byte representation with 4-digit year component without delimiters (*yyyymmdd*).
DF=L	10-byte representation with 4-digit year component and delimiters (*yyyy-mm-dd*).

For each representation, the sequence of the day, month and year components, and the delimiter characters used, are determined by the DTFORM parameter.

By default, DF=S applies (except for INPUT statements; see below).

The session parameter DF is evaluated at compilation.

It can be specified with the following statements:

- FORMAT,
- INPUT, DISPLAY, WRITE and PRINT at statement and element (field) level,
- MOVE, COMPRESS, STACK, RUN and FETCH at element (field) level.

When specified in one of these statements, the DF parameter applies to the following:

Statement	Effect of DF parameter
DISPLAY, WRITE, PRINT	When the value of a date variable is output with one of these statements, the value is converted to an alphanumeric representation before it is output. The DF parameter determines which representation is used.
MOVE, COMPRESS	When the value of a date variable is transferred to an alphanumeric field with a MOVE or COMPRESS statement, the value is converted to an alphanumeric representation before it is transferred. The DF parameter determines which representation is used.
STACK, RUN, FETCH	When the value of a date variable is placed on the stack, it is converted to alphanumeric representation before it is placed on the stack. The DF parameter determines which representation is used. The same applies when a date variable is specified as a parameter in a FETCH or RUN statement (as these parameters are also passed via the stack).
INPUT	When a data variable is used in an INPUT statement, the DF parameter determines how a value must be entered in the field. However, when a date variable for which *no* DF parameter is specified is used in an INPUT statement, the date can be entered either with a 2-digit year component and delimiters or with a 4-digit year component and no delimiters. In this case, too, the sequence of the day, month and year components, and the delimiter characters to be used, are determined by the DTFORM parameter.

Note:
With DF=S, only 2 digits are provided for the year information; this means that if a date value contained the century, this information would be lost during the conversion. To retain the century information, you set DF=I or DF=L.

Examples of DF Parameter with WRITE Statements

These examples assume that DTFORM=G applies.

```
/* DF=S (default)
WRITE *DATX   /* Output has this format: dd.mm.yy
END

FORMAT DF=I
WRITE *DATX   /* Output has this format: ddmmyyyy
END

FORMAT DF=L
WRITE *DATX   /* Output has this format: dd.mm.yyyy
END
```

Example of DF Parameter with MOVE Statement

This example assumes that DTFORM=E applies.

```
DEFINE DATA LOCAL
  1 #DATE  (D) INIT <D'31/12/2005'>
  1 #ALPHA (A10)
END-DEFINE
...
MOVE #DATE        TO #ALPHA /* Result: #ALPHA contains 31/12/05
MOVE #DATE (DF=I) TO #ALPHA /* Result: #ALPHA contains 31122005
MOVE #DATE (DF=L) TO #ALPHA /* Result: #ALPHA contains 31/12/2005
...
```

Example of DF Parameter with STACK Statement

This example assumes that DTFORM=I applies.

```
DEFINE DATA LOCAL
  1 #DATE  (D) INIT <D'2005-12-31'>
  1 #ALPHA1(A10)
  1 #ALPHA2(A10)
  1 #ALPHA3(A10)
END-DEFINE
...
STACK TOP DATA #DATE (DF=S) #DATE (DF=I) #DATE (DF=L)
...
INPUT #ALPHA1 #ALPHA2 #ALPHA3
...
/* Result: #ALPHA1 contains 05-12-31
/*         #ALPHA2 contains 20051231
/*         #ALPHA3 contains 2005-12-31
...
```

Example of DF Parameter with INPUT Statement

This example assumes that DTFORM=I applies.

```
DEFINE DATA LOCAL
  1 #DATE1 (D)
  1 #DATE2 (D)
  1 #DATE3 (D)
  1 #DATE4 (D)
END-DEFINE
...
INPUT #DATE1 (DF=S) /* Input must have this format: yy-mm-dd
      #DATE2 (DF=I) /* Input must have this format: yyyymmdd
      #DATE3 (DF=L) /* Input must have this format: yyyy-mm-dd
      #DATE4        /* Input must have this format: yy-mm-dd or yyyymmdd
...
```

Date Format for Output - DFOUT Parameter

The session/profile parameter DFOUT only applies to date fields in INPUT, DISPLAY, WRITE and PRINT statements for which no edit mask is specified, and for which no DF parameter applies.

For date fields which are displayed by INPUT, DISPLAY, PRINT and WRITE statements and for which neither an edit mask is specified nor a DF parameter applies, the profile/session parameter DFOUT determines the format in which the field values are displayed.

Possible DFOUT settings are:

DFOUT=S	Date variables are displayed with a 2-digit year component, and delimiters as determined by the DTFORM parameter (*yy-mm-dd*).
DFOUT=I	Date variables are displayed with a 4-digit year component and no delimiters (*yyyymmdd*).

By default, DFOUT=S applies. For either DFOUT setting, the sequence of the day, month and year components in the date values is determined by the DTFORM parameter.

The lengths of the date fields are not affected by the DFOUT setting, as either date value representation fits into an 8-byte field.

The DFOUT parameter can be set in the Natural parameter module/file, dynamically when Natural is invoked, or with the system command GLOBALS. It is evaluated at runtime.

Example:

This example assumes that DTFORM=I applies.

```
DEFINE DATA LOCAL
1 #DATE  (D) INIT <D'2005-12-31'>
END-DEFINE
...
WRITE #DATE        /* Output if DFOUT=S is set ...: 05-12-31
                   /* Output if DFOUT=I is set ...: 20051231
WRITE #DATE (DF=L) /* Output (regardless of DFOUT): 2005-12-31
...
```

Date Format for Stack - DFSTACK Parameter

The session/profile parameter DFSTACK only applies to date fields used in STACK, FETCH and RUN statements for which no DF parameter has been specified.

The DFSTACK parameter determines the format in which the values of date variables are placed on the stack via a STACK, RUN or FETCH statement.

Possible DFSTACK settings are:

DFSTACK=S	Date variables are placed on the stack with a 2-digit year component, and delimiters as determined by the profile parameter DTFORM (*yy-mm-dd*).
DFSTACK=C	Same as DFSTACK=S. However, a change in the century will be intercepted at runtime.
DFSTACK=I	Date variables are placed on the stack with a 4-digit year component and no delimiters (*yyyymmdd*).

By default, DFSTACK=S applies. DFSTACK=S means that when a date value is placed on the stack, it is placed there without the century information (which is lost). When the value is then read from the stack and placed into another date variable, the century is either assumed to be the current one or determined by the setting of the YSLW parameter (see below). This might lead to the century being different from that of the original date value; however, Natural would not issue any error in this case.

DFSTACK=C works the same as DFSTACK=S in that a date value is placed on the stack without the century information. However, if the value is read from the stack and the resulting century is different from that of the original date value (either because of the YSLW parameter, or the original century not being the current one), Natural issues a runtime error.

Note:
This runtime error is already issued at the time when the value is placed on the stack.

DFSTACK=I allows you to place a date value on the stack in a length of 8 bytes without losing the century information.

The DFSTACK parameter can be set in the Natural parameter module/file, dynamically when Natural is invoked, or with the system command GLOBALS. It is evaluated at runtime.

Example:

This example assumes that DTFORM=I and YSLW=0 apply.

```
DEFINE DATA LOCAL
  1 #DATE  (D) INIT <D'2005-12-31'>
  1 #ALPHA1(A8)
  1 #ALPHA2(A10)
END-DEFINE
...
STACK TOP DATA #DATE #DATE (DF=L)
...
INPUT #ALPHA1 #ALPHA2
...
/* Result if DFSTACK=S or =C is set: #ALPHA1 contains 05-12-31
/* Result if DFSTACK=I is set .....: #ALPHA1 contains 20051231
/* Result (regardless of DFSTACK) .: #ALPHA2 contains 2005-12-31
...
```

Year Sliding Window - YSLW Parameter

The profile parameter YSLW allows you determine the century of a 2-digit year value.

The YSLW parameter can be set in the Natural parameter module/file or dynamically when Natural is invoked. It is evaluated at runtime when an alphanumeric date value with a 2-digit year component is moved into a date variable. This applies to data values which are:

- used with the mathematical function VAL*(field)*,
- used with the IS(D) option in a logical condition,
- read from the stack as input data, or

- entered in an input field as input data.

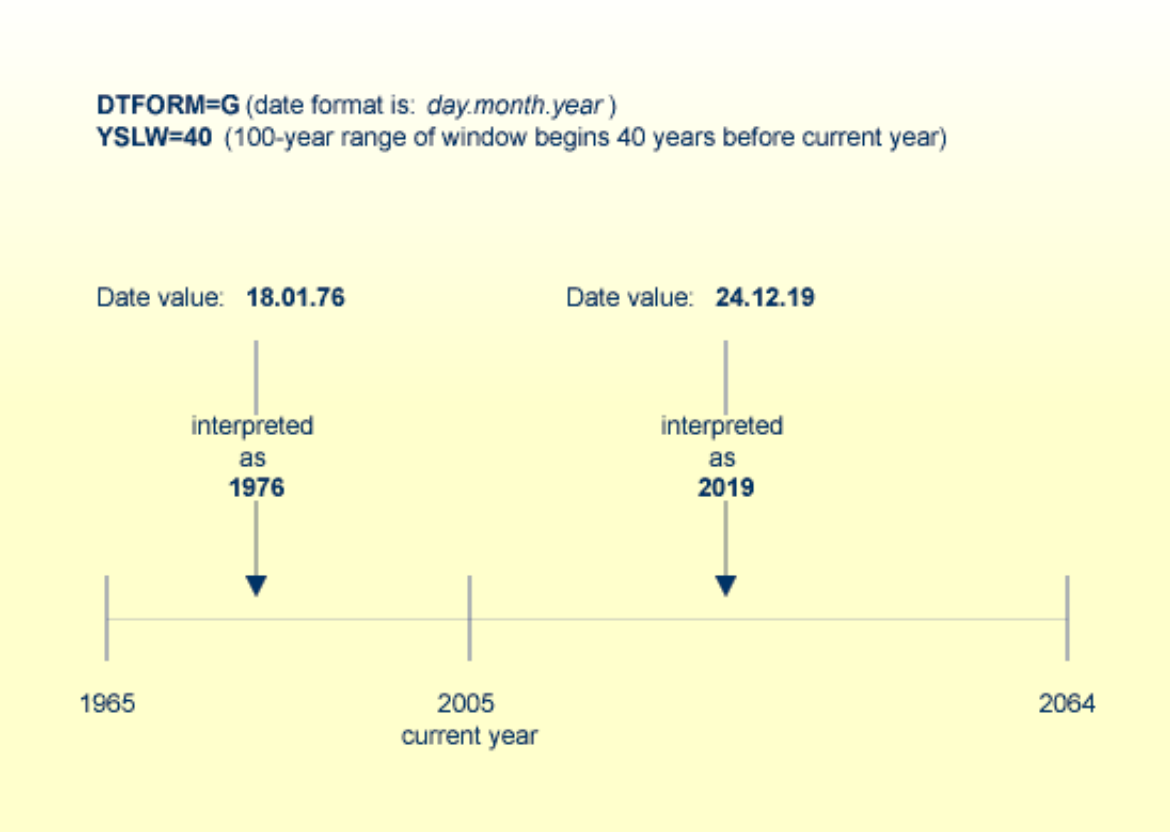
The YSLW parameter determines the range of years covered by a so-called "year sliding window". The sliding-window mechanism assumes a date with a 2-digit year to be within a "window" of 100 years. Within these 100 years, every 2-digit year value can be uniquely related to a specific century.

With the YSLW parameter, you determine how many years in the past that 100-year range is to begin: The YSLW value is subtracted from the current year to determine the first year of the window range.

Possible values of the YSLW parameter are 0 to 99. The default value is YSLW=0, which means that no sliding-window mechanism is used; that is, a date with a 2-digit year is assumed to be in the current century.

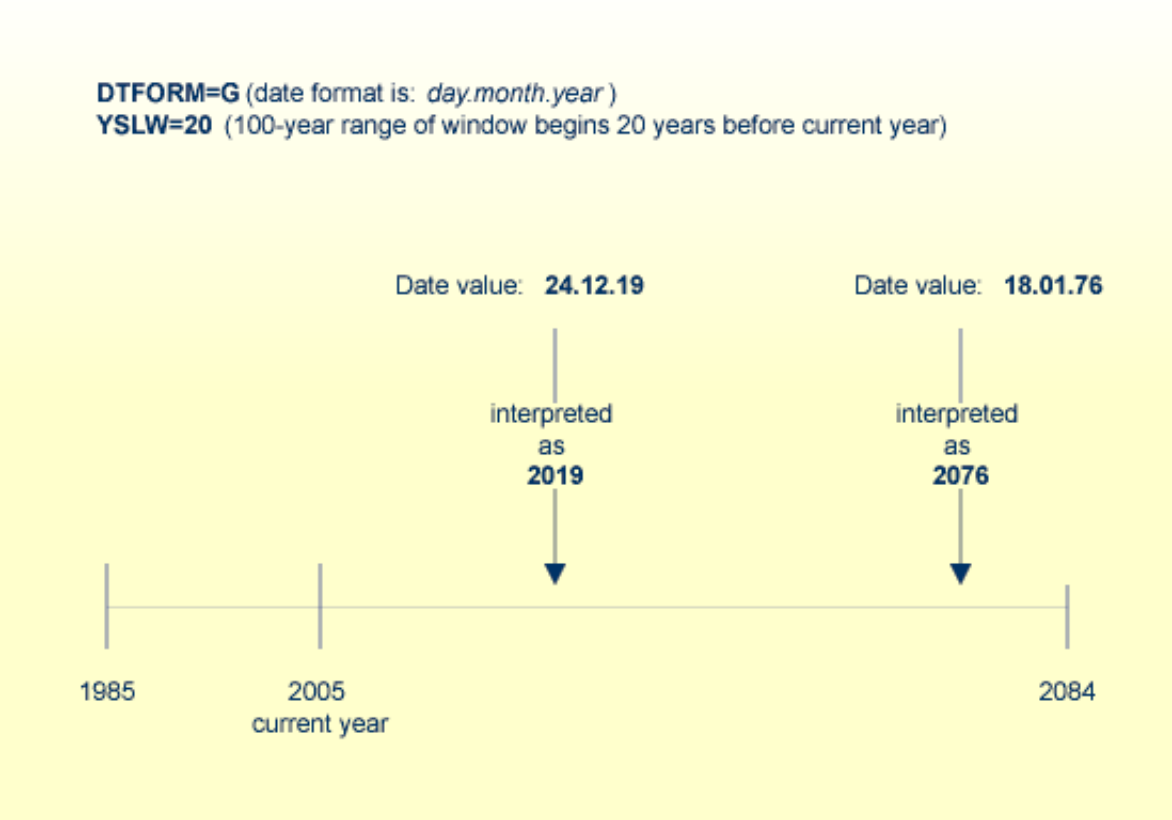
Example 1:

If the current year is 2005 and you specify YSLW=40, the sliding window will cover the years 1965 to 2064. A 2-digit year value *nn* from 65 to 99 is interpreted accordingly as 19*nn*, while a 2-digit year value *nn* from 00 to 64 is interpreted as 20*nn*.



Example 2:

If the current year is 2005 and you specify YSLW=20, the sliding window will cover the years 1985 to 2084. A 2-digit year value *nn* from 85 to 99 is interpreted accordingly as 19*nn*, while a 2-digit year value *nn* from 00 to 84 is interpreted as 20*nn*.



Combinations of DFSTACK and YSLW

The following examples illustrate the effects of using various combinations of the parameters DFSTACK and YSLW.

Note:
All these examples assume that DTFORM=I applies.

Example 1:

This example assumes the current year to be 2005, and that the parameter settings DFSTACK=S (default) and YSLW=20 apply.

```
DEFINE DATA LOCAL
  1 #DATE1 (D) INIT <D'1956-12-31'>
  1 #DATE2 (D)
END-DEFINE
...
STACK TOP DATA #DATE1 /* century information is lost (year 56 is stacked)
...
INPUT #DATE2          /* year sliding window determines 56 to be 2056
...
/* Result: #DATE2 contains 2056-12-31
           even if #DATE1 is set to <D'2156-12-31'>
```

In this case, the year sliding window is not set appropriately, so that the century information is (inadvertently) changed.

Example 2:

This example assumes the current year to be 2005, and that the parameter settings DFSTACK=S (default) and YSLW=60 apply.

```
DEFINE DATA LOCAL
  1 #DATE1 (D) INIT <D'1956-12-31'>
  1 #DATE2 (D)
END-DEFINE
...
STACK TOP DATA #DATE1 /* century information is lost (year 56 is stacked)
...
INPUT #DATE2          /* year sliding window determines 56 to be 1956
...
/* Result: #DATE2 contains 1956-12-31
           even if #DATE1 is set to <D'2056-12-31'>
```

In this case, the year sliding window is set appropriately, so that the original century information is correctly restored.

Example 3:

This example assumes the current year to be 2005, and that the parameter settings DFSTACK=C and YSLW=0 (default) apply.

```
DEFINE DATA LOCAL
  1 #DATE1 (D) INIT <D'1956-12-31'>
  1 #DATE2 (D)
END-DEFINE
...
STACK TOP DATA #DATE1 /* century information is lost (year 56 is stacked)
...
INPUT #DATE2          /* 56 is assumed to be in current century -> 1956
...
/* Result: RUNTIME ERROR (UNINTENDED CENTURY CHANGE)
```

In this case, the century information is (inadvertently) changed. However, this change is intercepted by the DFSTACK=C setting.

Example 4:

This example assumes the current year to be 2005, and that the parameter settings DFSTACK=C and YSLW=60 (default) apply.

```
DEFINE DATA LOCAL
  1 #DATE1 (D) INIT <D'2056-12-31'>
  1 #DATE2 (D)
END-DEFINE
...
STACK TOP DATA #DATE1 /* century information is lost (year 56 is stacked)
...
INPUT #DATE2          /* year sliding window determines 56 to be 1956
...
/* Result: RUNTIME ERROR (UNINTENDED CENTURY CHANGE)
```

In this case, the century information is changed due to the year sliding window. However, this change is intercepted by the DFSTACK=C setting.

Year Fixed Window

For information on this topic, see the description of the profile parameter YSLW.

Date Format for Default Page Title - DFTITLE Parameter

The session/profile parameter DFTITLE determines the format of the date in a default page title (as output with a DISPLAY, WRITE or PRINT statement).

DFTITLE=S	The date is output with a 2-digit year component and delimiters (*yy-mm-dd*).
DFTITLE=L	The date is output with a 4-digit year component and delimiters (*yyyy-mm-dd*).
DFTITLE=I	The date is output with a 4-digit year component and no delimiters (*yyyymmdd*).

For each of these output formats, the sequence of the day, month and year components, and the delimiter characters used, are determined by the DTFORM parameter.

The DFTITLE parameter can be set in the Natural parameter module/file, dynamically when Natural is invoked, or with the system command GLOBALS. It is evaluated at runtime.

Example:

This example assumes that DTFORM=I applies.

```
WRITE 'HELLO'
END
/*
/* Date in page title if DFTITLE=S is set ...: 05-10-31
/* Date in page title if DFTITLE=L is set ...: 2005-10-31
/* Date in page title if DFTITLE=I is set ...: 20051031
```

Note:
The DFTITLE parameter has no effect on a user-defined page title as specified with a WRITE TITLE statement.