Nectar and year 2000 issues
Contents

Introduction, 3
  Basic assumptions, 3
  Data types for date items, 4

Problems with the year 2000, 5
  Handling leap years, 5
  Especially for dates in string format, 5
  Year items in string format, 5
  Date/time variables in Nectar Basic and C/API, 6
  Interpretation of double-digit years in older versions of
  Nectar, 7
  Interpretation of double-digit years in Nectar 6.1, 8
  Interpretation as current month and year, 9

Recommended actions, 10
  Upgrading to Nectar 6.1, 10
  To bear in mind when deciding not to upgrade, 11
  Recommended approach, 11
  Aids for year 2000 certification, 11
**Introduction**

It’s high time to get to grips with year 2000 issues in our computer systems. – *No problem since Nectar stores dates in the UNIX format* – might be your response. But this is not the whole truth. It won’t do to dismiss a review of year 2000 issues just because *one* software product is able to manage dates after the turn of the millennium.

In this document, we hope to make clear any issues that can arise during year 2000 certification of applications developed using the Nectar tool. Issues are considered in versions of Nectar from 4.0 and upwards.

**Basic assumptions**

In order to limit the number of typical cases in the document, we principally assume that date and time items have been defined in accordance with the following:

- Date and combined date-and-time fields must have a form field length which accommodates the editing masks for date and time defined in the application. When using the complete year mask (YYYY), however, date fields are allowed to be two characters shorter in order to display just the decade and year (YY). Time fields, too, can be shorter in order to display, for example, just hours and minutes.

- Furthermore, Nectar’s check function for date, date-and-time or just time is to be used (Define Item, Check function 2, 3 or 7).

- All date/time fields are to have free editing (Local Item data, Editing type 7 or 17).
Data types for date items

The following data types can be used for date items in Nectar:

– Date(c): Date in string format (character date)
– Date(i): Date in integer format (integer Unix date)
– Date(d): Date in double float format (double Unix date)
– Date(d): Date in double float format (SQL-date)
Problems with the year 2000

Handling leap years

All versions of Nectar interpret the year 2000 as a leap year. Irrespective of data type, it is possible to specify the date 29-02-2000, with the exception of the problems described in this document. All items that have a check function for dates in Nectar use the following rule to determine whether it is a leap year or not:

- The month of February has 29 days if and only if it is a leap year.
- Leap years occur during all years that are evenly divisible by 400 or evenly divisible by 4 and not evenly divisible by 100.

Especially for dates in string format

The classical year 2000 issue is storing dates, in the best COBOL style, in a six-character long string (DDMMYY). Here the millennium and century are not accommodated, which is why these items should be extended to at least 8 characters in the database, especially in the case of calculations, sorts and comparisons. Take the example if two dates are compared: Is 310100 031298 true?

Year items in string format

A hidden problem in many applications is the use of a string variable which only describes the year, e.g. YY. An item like
this lacks a check function for dates. In the case of calculations, sorts and comparisons, the problem is the same as in the paragraph above. The item must be extended to 4 characters and supplemented with two year-digits. Checks for reasonableness should be introduced during which input of the year is carried out.

The same rationale applies when you use a string variable which describes both the year and the month, e.g. MMYY.

**Date/time variables in Nectar Basic and C/API**

The incorrect handling of date/time variables in processing rules, triggers and in C/API programs (previously Lib*C) can lead to problems. These variables are updated each time a date or time item is handled in a form. In C/API, they are also updated after the most recent call of the standard functions ZNSDKON and ZNDSKON. The variables are limited by the definition for Unix dates, which is why the interval is from 14-12-1901 to 19-01-2038. If a date which is outside this interval is handled (date(c) or date(d)), the variables will not be initialised.

Examples of problems caused by the following cases not being known when the system was developed:

- The variable znyear acquires the value 100 in the year 2000. The year in question minus 1900. If the numerical value is converted into a string, and this is only expected to be two characters long, an error will result. If the string is truncated to two characters and another year is subtracted, the consequences will be accordingly (00-99 = -99).

- The variable znweek acquires the value 10001 on 03-01-2000. The year in question minus 1900 followed by the week number in question. Can lead to the same consequences as above.

This applies to all versions of Nectar.
Interpretation of double-digit years in older versions of Nectar

Items with a check function for date and date-and-time have an interpretation error of the year "00" in versions previous to 6.1. It is the section for screen handling (IOS) and the standard function ZNDSKON (conversion from date string to date in seconds) which interpret input of the year "00" incorrectly – if the 21st century is being referred to. Furthermore, the year "01" is interpreted as 1901, not 2001, as one might expect. When entering a date after the turn of the millennium, 310100, for example, will not be enough to describe January 31 in the year 2000. The date must be specified as 31012000, irrespective of the data type, in order not to be interpreted as 31-01-1970.

Examples (date mask DD-MM-YYYY):

<table>
<thead>
<tr>
<th>Data type</th>
<th>Input</th>
<th>Interpreted as</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary</td>
<td>101097</td>
<td>10-10-1997</td>
<td>If editing type 7. Blank if 17</td>
</tr>
<tr>
<td>- “ -</td>
<td>010100</td>
<td>01-01-1970</td>
<td>- ” -</td>
</tr>
<tr>
<td>- “ -</td>
<td>010170</td>
<td>01-01-1970</td>
<td>Year 00 interpreted as 1970</td>
</tr>
<tr>
<td>- “ -</td>
<td>310100</td>
<td>31-01-1970</td>
<td></td>
</tr>
<tr>
<td>- “ -</td>
<td>310170</td>
<td>31-01-1970</td>
<td></td>
</tr>
<tr>
<td>- “ -</td>
<td>31012000</td>
<td>31-01-2000</td>
<td>Year 2000 must be input as &quot;2000&quot;</td>
</tr>
<tr>
<td>Not Date(i)</td>
<td>010101</td>
<td>01-01-1901</td>
<td>Interpreted as 20th century</td>
</tr>
<tr>
<td>Date(i)</td>
<td>010101</td>
<td>Incorrect!</td>
<td>1901 is outside limit value for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date(i)</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>01012001</td>
<td>01-01-2001</td>
<td>Interpreted as 20th century</td>
</tr>
<tr>
<td>- “ -</td>
<td>010112</td>
<td>01-01-1912</td>
<td></td>
</tr>
<tr>
<td>- “ -</td>
<td>01012012</td>
<td>01-01-2012</td>
<td></td>
</tr>
</tbody>
</table>

The consequences of incorrect interpretations in older versions of Nectar can be minimized by verifying, in connection with year 2000 certification of the application, the following points:

- The date mask in the application must contain a complete year specification, YYYYY.
All instances of date items in forms must accommodate all of the new date mask (field length at least equal to the length of the date mask).

All instances of date-and-time items in forms must accommodate all of the date mask, all of the time mask plus one blank (field length at least equal to the sum of the length of the date mask, the length of the time mask and one blank).

All instances of date(c) items in tables must have a field length and a length in the database which accommodates at least the date mask (see Especially for dates in string format).

Users of the system must be informed/trained to always specify, during input of 2000 dates, the entire year completely – using four characters.

**Interpretation of double-digit years in Nectar 6.1**

The interpretation of double-digit years takes place smoothly in version 6.1. There, a double-digit year specification can be interpreted as an arbitrary predefined century. Alternatively, a double-digit year specification can be interpreted as 1900 or 2000, depending on whether it is greater than or equal to 38, or less than 38. Examples below with the date mask DD-MM-YYYY and the parameter Default year = Y in the application data:

<table>
<thead>
<tr>
<th>Year YY</th>
<th>interpreted as YYYY</th>
<th>Examples of input</th>
<th>interpreted as</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>1900</td>
<td>101097</td>
<td>10-10-1997</td>
</tr>
<tr>
<td>= 38</td>
<td>1900</td>
<td>151238</td>
<td>15-12-1938</td>
</tr>
<tr>
<td>38</td>
<td>2000</td>
<td>310537</td>
<td>31-05-2037</td>
</tr>
<tr>
<td>-”-</td>
<td>2000</td>
<td>310100</td>
<td>31-01-2000</td>
</tr>
<tr>
<td>- ”-</td>
<td>2000</td>
<td>311202</td>
<td>31-12-2002</td>
</tr>
</tbody>
</table>

With the parameter Default year = N in the application data, the value of Starting year in the application data determines the interpretation:
With the year YY is interpreted as YYYY

<table>
<thead>
<tr>
<th>Year</th>
<th>Interval as input</th>
<th>Interpreted as</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>00 - 99</td>
<td>1900 - 1999</td>
</tr>
<tr>
<td></td>
<td>310100</td>
<td>31-01-1900</td>
</tr>
<tr>
<td>1900</td>
<td></td>
<td>31012000</td>
</tr>
<tr>
<td></td>
<td>3100</td>
<td>31-01-2000</td>
</tr>
<tr>
<td>2000</td>
<td>00 - 99</td>
<td>2000 - 2099</td>
</tr>
<tr>
<td></td>
<td>101097</td>
<td>10-10-2097</td>
</tr>
<tr>
<td></td>
<td>101097</td>
<td>10-10-1997</td>
</tr>
<tr>
<td></td>
<td>310100</td>
<td>31-01-1997</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>31011997</td>
</tr>
<tr>
<td></td>
<td>101097</td>
<td>10-10-1997</td>
</tr>
<tr>
<td></td>
<td>310100</td>
<td>31-01-2000</td>
</tr>
</tbody>
</table>

The solutions above apply to all date data-types and to both date and combined date and time fields in Nectar. The exception is the limitations that the maximum values of the data types have. Date integers (Unix date) are stored in seconds relative to 01-01-1970 00:00:00, which is why the interval for this data type (4 byte integer) is from 14-12-1901 to 19-01-2038.

**Interpretation as current month and year**

The input of dates is simplified considerably by setting the parameter `Default year = Y` in the application data in Nectar 6.1. If the current year is being referred to, the year can be omitted completely during input in a date field. For instance, entering 3112 is interpreted as 31-12-1997. If the current year and month are being referred to, both of these can be omitted. For instance, 31 is interpreted as 31-10-1997 when this is entered in October 1997.

This method of writing rationalizes and reduces the frequency of error in date-intensive dialogs.
**Recommended actions**

**Upgrading to Nectar 6.1**

Like many other software vendors, we recommend upgrading to year 2000 certified versions. We have had Nectar 6.1 year 2000 certified, which entails that the product correctly handles dates before, during and after the turn of the millennium, in addition to correctly handling leap years too. However, since Nectar is a development tool, a system developer can define and handle dates incorrectly by mistake, for instance with two year-digits, in developed applications.

Upgrading to Nectar 6.1 entails the application version being converted to Nectar 6.1. This is implemented using conversion programs followed by a certain amount of hands-on in order to make the old application work with new concepts such as:

- User groups (added in Nectar 5.1): Structured authorization systems where users with equal authorization are placed in the same group.
- Terminal groups (Nectar 5.0): Directs printouts to the correct printer, depending on from where/whom the printout was started.
- Logical printers (Nectar 5.0): Enables selecting, during report printouts, different printers (prompt).
- Nectar Basic (Nectar 6.0): Processing rules and Triggers are implemented with access to complete BASIC.

Known changes which impact on compatibility are described in the documents Revision information - Incompatibility for each respective edition.
To bear in mind when deciding not to upgrade

If, in spite of everything, you choose to keep an application under an older version of Nectar over the turn of the millennium, you will have to take the following into account:

- Do you fully realize the consequences for the interpretation of the year “00”?
- Will you have to upgrade your operating system and/or database handler prior to the year 2000? In which case you might have to upgrade Nectar anyway.
- The aids which considerably facilitate the surveying of date usage in Nectar applications are only available in later versions of Nectar.

Recommended approach

In order to isolate year 2000 issues in a Nectar application, you do a survey in which you focus on where and how the date concept is used. Broadly speaking, dates occur in calculations and/or as information.

Dates, as information, are supplemented by expanding the field length so that four year-digits can be displayed/entered. Dates included in calculations are, by nature, of critical importance to operations. Program code that carries out date calculations is analyzed, tested and revised to handle the year 2000 correctly.

Aids for year 2000 certification

Instances of date and date-and-time items are appropriately surveyed with regard to the data types used.

The easiest way to search Nectar’s data dictionary is by using the Documentation pack (from Nectar 5.1 and upwards).
For Nectar Basic, ZNBLIST is used (from Nectar 5.1 and upwards) which creates searchable text files.

C programs are analysed using fundamental search and editing functions or special programs.

Nectar Systems AB has unique expertise in the programs included in system environments that contain Nectar. We thus offer our services in implementing the year 2000 certification of Nectar applications. Please contact us for further information.

This document is a guide in the analysis and revision of Nectar applications with regard to year 2000 issues.