Rocket

Rocket Model 204 Sir2000 User Language Tools

Reference Manual

July 2013
S2K-0704-RM-01

Notices

Edition

Publication date: July 2013

Book number: S2K-0704-RM-01

Product version: Rocket Model 204 Sir2000 User Language Tools

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Note

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Contact information

Website: www.rocketsoftware.com

Rocket Software, Inc. Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451–1468
USA
Tel: +1 781 577 4321
Fax: +1 617 630 7100

Contacting Global Technical Support

If you have current support and maintenance agreements with Rocket Software and CCA, contact Global Technical Support by email or by telephone:

Email: m204support@rocketsoftware.com

Telephone:

North America	+1 800 755 4222
United Kingdom/Europe	+44 (0) 20 8867 6153

Alternatively, you can access the Rocket Customer Portal and report a problem, download an update, or read answers to FAQs. You will be prompted to log in with the credentials supplied as part of your product maintenance agreement.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

Contents

Figures

Summary of Changes

This section describes significant changes to the documentation. In most cases these changes correspond to enhancements made to the underlying product.

Sirius Mods Version 5.4

- APPDATE RESRICT command.
- Customization zaps to disable or restrict APPDATE.

Sirius Mods Version 5.2

- Datetime processing general doc improvement and examples.

Sirius Mods Version 5.0

- SPANSIZE parameter introduced.
- CENTSPAN default changed from -75 to -50.
- Appropriate APPDATE clock passed to Fast/Unload.

Sirius Mods Version 4.6

First release of this product.

CHAPTER 1 Background

1.1 The Year 2000

Computer applications that use two-digit years will encounter problems beginning with dates starting on January 1, 2000. This is because, the two-digit year value for the year 2000 is “00” which is less than the previous year “99”. This violates a seemingly reasonable assumption that time constantly moves forward and, more specifically, that year numbers are constantly increasing. There are millions of lines of codes that depend on this assumption many of them using two-digit years. When these millions of lines of code are faced with dates on or after January 1, 2000 they will cease to work correctly. This is known as the “Year 2000 Bug”,

Strictly speaking, the 21st century does not begin until January 1, 2001. The day of reckoning for computer software, however, is January 1, 2000. Moreover, the popular perception is that a new century begins on that date. Given these two considerations, this document freely uses the vernacular meaning of the start of the next century (that is, January 1, 2000) rather than the pedantically correct "January 1, 2001".

Since the term "Year 2000" appears frequently in any discussion of computer software problems with years spanning the 20th and 21 centuries, it is often abbreviated "Y2K". One might, for example, see references to an application being "Y2K compliant" to mean that an application will work correctly with a mix of dates from the 20th and 21st centuries. Obviously, the "K" in "Y2K" refers to the vernacular meaning of "K", namely 1000, rather than the computer industry meaning of "K", namely 1024. As far as is known, there is no "Year 2048" problem with most software.

1.2 Sir2000

Sir2000 is a suite of add-on products for *Model 204* that addresses all aspects of preparing for the millennium rollover. *Sir2000* operates in conjunction with enhancements to existing Model 204 related products to dramatically reduce the resources required to achieve year 2000 compliance for *Model 204* applications. *Sir2000* supports a low-risk incremental approach, allowing the coexistence of current applications and databases with the parallel development of enhanced versions.

Sir2000 comprises an improved method for converting two-digit year values to four-digit year values, new and improved formats for date values, new facilities for manipulating date values, and three separately-packaged product sets:

- *Sir2000 User Language Tools*
- *Sir2000 Database Analysis Tools*

- *Sir2000 Field Migration Facility*

Several other Sirius products have been enhanced to address specific aspects of the year 2000 problem. In addition, all Sirius products are fully year-2000 compliant. *Sir2000* supports all current releases of *Model 204*, since and including Version 2.2.

1.3 Versions, Compatibility and Installation

The *Sir2000 User Language Tools* is delivered as object code enhancements to *Model 204* as part the *Sirius Mods*, as well as *Model 204* User Language subsystems packaged as part of *UL/SPF*. The *Sirius Mods* also includes products such as *Fast/Backup*, *Fast/Reload*, and the *Fast/Unload User Language Interface* that have nothing to do with the *Sir2000 User Language Tools*. None of these additional products are required to run the *Sir2000 User Language Tools*.

Therefore, to install *Sir2000 User Language Tools*, the *Sirius Mods* must be installed. When the *Sirius Mods* are installed, all other products owned by the installing site that are part of the *Sirius Mods* will also be (re)installed.

Since the *Sir2000 User Language Tools* is part of the *Sirius Mods* the version number of the *Sir2000 User Language Tools* is considered to be the version of the *Sirius Mods* in which it is contained. The *Sir2000 User Language Tools* was first available in version 4.6 of the *Sirius Mods*, so the first version of *Sir2000 User Language Tools* was called version 4.6. This document assumes that a site is running *Sirius Mods* version 4.6 or later. Any documented feature or facility that requires a later version of the *Sirius Mods* will be clearly marked to indicate this. For example, a parameter that is only available in versions 5.0 and later of the *Sirius Mods* will have a sentence such as "This parameter is only available in version 5.0 or later of the *Sirius Mods*" in its documentation. If a feature or parameter is not indicated as requiring any specific version of the *Sirius Mods*, it can be assumed that it is available in all versions of the *Sir2000 User Language Tools*; that is, all versions since version 4.6 of the *Sirius Mods*.

Any application that uses the *Sirius Mods* will run correctly on subsequent versions of the *Sirius Mods*. It is, therefore, always possible to upgrade the *Sirius Mods* without having to worry about significant changes to applications that use it.

Minor compatibility issues that have been introduced are explicitly listed for each release in “Summary of Changes” on page ix and in the Release Notes published for the *Sirius Mods*.

1.4 Related products

Another useful product for helping with year 2000 conversion is the *Sir2000 Field Migration Facility*. The *Sir2000 Field Migration Facility* dramatically decreases the costs, complexities, and risks associated with making large *Model 204* applications Year 2000 compliant.

Fast/Unload can be used to generate test data for year 2000 testing, since it is generally not sufficient for year 2000 testing to simply set the apparent clock forward to the 21st century. *Model 204* files must also be populated with large volumes of data that span the century boundary for thorough testing.

1.5 Related manuals

Since the *Sir2000 User Language Tools* requires the installation of the *Sirius Mods*, the person responsible for the installation of *Sir2000 User Language Tools* should refer to the *Sirius Mods Installation Guide*. The documentation of the *Sirius Mods* error messages (**http://m204wiki.rocketsoftware.com/index.php/Category:Sirius_Mods_messages**) might be useful to application programmers as well as installers. See **http://m204wiki.rocketsoftware.com/index.php/UL/SPF_installation_guide** for instructions for installing the User Language components of the *Sir2000 User Language Tools*.

The *Sir2000 Field Migration Facility* is documented in the *Sir2000 Field Migration Facility Reference Manual*.

Fast/Unload is documented in the *Fast/Unload Reference Manual*.

There are a number of Sirius date $functions that you are authorized to use in addition to the date $functions in the *Sir2000 User Language Tools*. These are documented in M204wiki at **http://m204wiki.rocketsoftware.com/index.php/List_of_$functions**.

1.6 System requirements

Sir2000 User Language Tools requires the following components to run:

- Mainframe operating systems:

  z/OS

  CMS (releases currently supported by IBM) running under:

  z/VM

- *Model 204* Version 6.1 or later

CHAPTER 2 Overview

The *Sir2000 User Language Tools* is a package of products designed to help you change and test User Language applications for correct operation across the millennium rollover. The package consists of several components that are described below.

2.1 APPDATE Command

The APPDATE command allows you to control the operation of date and time oriented User Language $functions, in the following two ways:

1. You can specify a system-wide or user-level clock that is used to obtain date and time values for User Language $functions. For application testing, this is preferred to the *Model 204* SYSDATE parameter, which is much less flexible and which greatly complicates the ability to do things such as share procedure files or read-only data files in the testing environment.

   The APPDATE command affects only date and time oriented User Language $functions; it does not affect any other date or time in the *Model 204* environment.

2. You can use the DATE_ERR clause to set switches that control the default system behaviour when errors are encountered in date and time oriented User Language $functions. The choices are to produce an error message along with request cancellation, produce a warning message, or silently continue with the request. This control is available at the system and user level, and can also be set for the duration of a User Language request, via the $SIR_DATE_ERR function. At the system and user level, you can also control whether procedure names and line numbers are available for error messages.

2.2 Date $Functions

Sir2000 User Language Tools includes $functions that manipulate datetime values, support improved datetime formats, while providing enhanced 2-digit year handling (CENTSPAN/SPANSIZE) and improved error handling. These $functions either replace and/or augment some standard Model 204 date-related functions.

2.2.1 Datetime Formats

The $functions included with the *Sir2000 User Language Tools* allow you to specify formats for datetime values that include all of the tokens and separator characters for any common application, including mixed case, variable length, and day of week. Date and time are handled uniformly, and there are no special $function names, special arguments, nor customization parameters to control the date formats.

To get the current date in a different format with the standard Model 204 date $functions, you must either specify a special switch and/or use combinations of different $functions, frequently involving string concatenation or substring to deal with composite date and time formats. With the Sirius date $functions, each function performs some datetime-related calculation, and you can specify a combination of tokens to determine the format, which may include time of day. For example, the following User Language code fragment would be required to print the current day of the week using the CCA $functions provided with Version 4.1 of *Model 204*:

```
PRINT $DAY($DAYI($DATECNV('CYY-MM-DD', -
           'CYYDDD', $DATE(2))))
```

With the *Sir2000 User Language Tools*, this set of nested $function calls would become:

```
PRINT $SIR_DATE('Wkday')
```

You can also get the current date and time conveniently for a report:

```
%D STRING LEN 40
%D = $SIR_DATE('Wkday, DAY Month YYYY @ HH:MI:SS AM')
SET HEADER 1 %D
```

This would set a line such as the following at the top of each page:

```
Monday, 1 January 2001 @ 01:11:10 PM
```

2.2.2 Processing two digit years (CENTSPAN/SPANSIZE)

The $functions included with the *Sir2000 User Language Tools* provide a robust technique for handling 2-digit years. An optional CENTSPAN argument supplies the beginning of year of the up to 100-year range of dates covered by the two-digit years. CENTSPAN provides both an absolute specification, in which the full 4 digit start year is specified, and a relative specification, in which the start year is specified relative to the current time, for example, 50 years ago. The default CENTSPAN is -50.

To reduce the ambiguities caused by mapping two-digit years into four digit years, *SPANSIZE* is used to limit the number of years mapped. SPANSIZE and its relationship to CENTSPAN is explained in “CENTSPAN” on page 46. The default SPANSIZE is 90.

2.2.3 Error handling

The Sirius datetime $functions that process user-provided datetime format strings or datetime values accept an optional last argument that is a character string specifying the action to be taken if the function detects an error. When this argument is omitted, the action taken upon an error is determined by the current state of the USER and SYSTEM DATE_ERR switches. The DATE_ERR switches can be set by an APPDATE command with a DATE_ERR clause or the $SIR_DATE_ERR function. DATE_ERR allows the interception of date processing errors at their source, making it much easier to perform ad-hoc testing and validation of date processing. The following error actions can be specified:

CANCEL If a datetime $function (including the non-Sirius functions) detects an error, an error message is produced describing the nature of the error and the current request is cancelled. Sirius datetime $functions produce detailed error messages describing the precise nature of the error.

REPORT If a datetime $function detects an error, a warning message is produced describing the nature of the error and the appropriate *standard error value* is returned by the $function.

IGNORE If a datetime $function detects an error, the appropriate *standard error value* is returned by the $function.

The errors that are detected include:

- Invalid datetime format specification
- Datetime string not matching format
- Datetime out of range for the format
- Invalid CENTSPAN value
- Datetime out of range for CENTSPAN/SPANSIZE combination.

In addition, an APPDATE command with a DATE_ERR clause can specify that warning and error messages should include the procedure name and line number of the failing $function.

2.3 Subroutines for FUNU

Included with the *Sir2000 User Language Tools* is a set of subroutines that can be invoked by user-provided assembly language (FUNU) $functions to obtain the current date and time using the appropriate APPDATE clock, if any. They are compatible with the similar set of subroutines CCA provides for current date only, with an additional entry point which allows you to obtain the current date and time using any of the Sirius datetime formats.

2.4 SirPro

SirPro provides a highly productive integrated environment for the development and maintenance of *Model 204* applications. SirPro efficiently manages very large procedure files while providing high performance search and global replace features. SirPro also provides a full screen interface for commonly used *Model 204* commands. SirPro is documented in the *SirPro User's Guide*.

2.5 SirLib

SirLib is a comprehensive change management system for *Model 204* User Language applications. SirLib enables multiple programmers to simultaneously work on the same procedures, with tools for resolving update conflicts and backing out individual changes or projects. SirLib is modelled on the change control system used by IBM and virtually all large scale mainframe product developers. SirLib support is integrated within SirPro to minimize its intrusion into the development process. SirLib is documented in the *SirLib User's Guide*.

CHAPTER 3 The APPDATE Command

The APPDATE command provides control over the operation of date and time oriented User Language $functions. It supports a system-wide or user-level clock that is used to obtain date and time values. It can be used to control the default handling of errors detected by datetime $functions, allowing the logging of error messages and request cancellation. The APPDATE command allows you to control these defaults at a user and system level, significantly reducing your effort to detect some date errors.

For application testing, the APPDATE clocks are preferred to the *Model 204* SYSDATE parameter, which is much less flexible and which greatly complicates the ability to do things such as share procedure files or read-only data files in the testing environment.

See “Values affected by the APPDATE clocks” on page 20 for a list of the values that are affected by the APPDATE command. See “APPDATE Command Usage Notes” on page 20 for descriptions of several common testing scenarios and how to use APPDATE to handle them, including testing the effect of changing the date into the future for any of the following:

- a single user's session
- all users' sessions
- all users' sessions except selected ones
- all users of a given APSY
- all users of all APSYs, except selected APSYs

In most contexts when describing APPDATE, this document speaks of a "date and time" or simply of a "time" (which is the same thing). Generally, for Y2K testing, you are only concerned about setting a "date," that is, setting the clock to a particular day.

APPDATE addresses many of your needs to test a User Language application at some date in the future, notably beyond the end of the 1900s. APPDATE allows you to make these tests with your current version of *Model 204*, and without modifying the *Model 204* internal date-related data structures, such as the File Parameter List, page trailers, and the Procedure Dictionary. APPDATE will not affect any file level operations such as recovery or DUMP, or the ability to open a file with different application date settings.

The system-wide and user-level clocks specified by the APPDATE command are called APPDATE clocks. In order to use these clocks, you must set the date (and optionally the time) of either the system level or user level clock. The syntax associated with setting these APPDATE clocks is described in “Setting the Clocks With the APPDATE Command” on page 10.

Another important component in a testing environment, or indeed in a production environment, is the ability to detect unanticipated errors. The APPDATE command with

a DATE_ERR clause and the $SIR_DATE_ERR function allow you to change the default error handling of arguments to date and time oriented $functions. The syntax associated with DATE_ERR is described in “Error Handling Control with DATE_ERR” on page 13.

A single invocation of the APPDATE command can either set APPDATE values or display APPDATE values. If you wish to examine the APPDATE values in effect, use the syntax described in “Examining the APPDATE Information” on page 16. The detailed explanation of setting APPDATE values is described separately for clock values and datetime argument error handling, but for a single level you can specify both clock values and error handling, in that order, in a single command. The complete syntax of the APPDATE command is shown in “Complete Syntax of the APPDATE Command” on page 19.

3.1 Setting the Clocks With the APPDATE Command

The two APPDATE clocks (**user** and **system**) effectively run in parallel with the **internal** clock, that is, the time that *Model 204* obtains from the CPU (note that the SYSDATE parameter cannot be specified if the APPDATE command is used). When you set either of the clocks, you are actually setting the amount that the clock runs ahead of the internal clock. The way you do that is to simply specify either the current time of the clock being set, or to specify the time which would be current when the internal clock is at some specific reference time.

In addition to the time setting of the clocks, you control which clock an application uses. This is determined primarily by the "state" of the USER clock: INTERNAL means applications use the INTERNAL clock, ON means applications use the USER clock, and OFF (which is the default) means that applications use the INTERNAL clock unless the SYSTEM clock is ON (the default SYSTEM clock is OFF).

The clock used to satisfy an application request for date or time is determined as shown in the following figure:

```
*----------------------------------------*
*     Application code requests date     *
*         (datetime $functions)          *
*----------------------------------------*
|
|-> Gets date/time from the USER clock
|   if it is ON
|     (e.g. APPDATE USER ON 2001/01/01)
|
|-> Else from SYSTEM clock if it is ON
|   and USER clock is OFF
|      e.g., APPDATE USER OFF (default)
|        and APPDATE SYSTEM ON 1999/12/31
|
|-> Else from INTERNAL clock:
       e.g., APPDATE USER INTERNAL
                    or
             APPDATE USER OFF
           and APPDATE SYSTEM OFF
                    or
             APPDATE USER OFF
           and APPDATE SYSTEM INTERNAL
```

Source of date returned to $function

The time of the USER or SYSTEM clock may be set by the APPDATE command:

```
APPDATE [USER|SYSTEM] [ON|OFF|INTERNAL] -
        targdate [targtime]  -
        [AT refdate [reftime]]
```

APPDATE syntax to set USER or SYSTEM clock

Whether or not a particular clock is used can be controlled by the APPDATE command, without changing the time of the USER or SYSTEM clock:

```
APPDATE [USER|SYSTEM] {ON|OFF|INTERNAL}
```

APPDATE syntax to change state of USER or SYSTEM clock

where

USER | SYSTEM

Optional specification of which clock is being modified. Defaults to USER. The USER clock specifies the running user's APPDATE clock; the SYSTEM clock specifies the APPDATE clock used for any user with an APPDATE USER clock which is OFF.

ON | OFF | INTERNAL

Optional. APPDATE USER ON indicates that the user clock is used by the current user; APPDATE SYSTEM ON indicates that the system clock is used by all users whose user clock is OFF. OFF and INTERNAL both indicate that the specified clock is not visible. If the user-level clock is set to ON, User Language $function values are obtained from the user-level clock. If the user-level clock is set to INTERNAL, User Language $function values are obtained from the internal clock. If the user-level clock is set to OFF, User Language $function values depend on the setting of the system-wide clock: if it is ON, they are obtained from the system-wide clock; otherwise (it is OFF or INTERNAL) they are obtained from the internal clock.

If this phrase is omitted in an APPDATE command that sets the time of a clock, the state of that clock is set to ON.

You can specify ON, OFF, or INTERNAL without a date, in which case the date/time in a clock is not changed, but the visibility of the corresponding clock is affected.

targdate [targtime] [AT refdate [reftime]]

targdate is required, the rest are optional. These phrases allow you to specify the date and time of the application clock. The dates are specified in YYYY/MM/DD format, and the times are specified in HH:MI:SS format. The application clocks continue to run so that they are a constant difference from the internal date and time. This difference is the difference between the target datetime, as specified by *targdate* and the optional *targtime*, and the reference datetime. The reference datetime can be omitted, in which case the reference datetime is the datetime when the APPDATE command is issued.

If *reftime* is specified, then *targtime* must be. If *targtime* is specified, then either *refdate* must be omitted or *reftime* must be specified. If *targtime* is omitted, the clock is shifted the number of days between *targdate* and *refdate*.

A successful APPDATE SYSTEM command has an immediate effect upon the date/time returned by User Language $functions for all threads which have user-level clocks set or defaulted to the OFF state.

The default value of the system-wide clock and all user-level clocks is OFF, with the same datetime as the internal clock. Hence, if the only command you issue is APPDATE SYSTEM yyyy/mm/dd, you will immediately shift the value returned as the current date by User Language $functions. The APPDATE USER clock is reset to OFF, with the same time as the INTERNAL clock, when a user logs off.

3.2 Error Handling Control with DATE_ERR

The APPDATE command with the DATE_ERR keyword and the $SIR_DATE_ERR function let you decide what default action the system should take when an error is detected by a datetime $function. This allows you to control validation of dates in your applications, significantly reducing your effort to detect and correct date errors. APPDATE DATE_ERR and $SIR_DATE_ERR detect the following errors:

- Invalid datetime format specification
- Datetime string not matching format
- Datetime out of range for the format
- Invalid CENTSPAN value
- Datetime out of range for CENTSPAN/SPANSIZE combination

You can set the default error handling for all $functions in the system with APPDATE SYSTEM DATE_ERR command; you can override this setting for an individual user with the APPDATE USER DATE_ERR command; you can override both settings for the duration of a User Language request with the $SIR_DATE_ERR function. If you are using the Sirius datetime $functions, you can also supply a parameter on the individual function call to control error handling, overriding the default set by $SIR_DATE_ERR or APPDATE.

The processing performed when a datetime function encounters an error is determined using the precedence indicated in the following figure:

```
*----------------------------------------*
*   Datetime $function argument error    *
*----------------------------------------*
|
|-> IGNORE is error handling handling arg:
|     return error value
|
|-> REPORT is error handling arg:
|     issue message, return error value
|
|-> Error handling supplied, and not OFF:
|     issue message, cancel request
|
++ Else OFF is error handling arg, or none:
|
|-> $SIR_DATE_ERR('REPORT') last invoked:
|     issue message, return error value
|
|-> $SIR_DATE_ERR('IGNORE') last invoked:
|     return error value
|
|-> $SIR_DATE_ERR('CANCEL') last invoked:
|     issue message, cancel request
|
++ Else error handling controlled by APPDATE:
|
|-> USER DATE_ERR REPORT:
|     issue message, return error value
|
|-> USER DATE_ERR IGNORE:
|     return error value
|
|-> USER DATE_ERR CANCEL:
|      issue message, cancel request
|
|-> USER DATE_ERR OFF (default) and
|   SYSTEM DATE_ERR REPORT
|     issue message, return error value
|
|-> USER DATE_ERR OFF (default) and
|   SYSTEM DATE_ERR IGNORE
|     return error value
|
|-> USER DATE_ERR OFF (default) and
|   SYSTEM DATE_ERR CANCEL
|     issue message, cancel request
|
++ Else error handling not affected
|  by Sir2000 User Language Tools:
|
+---> return error value
```

Datetime $function argument error handling

3.2.1 Setting Default Error Handling with the APPDATE Command

The APPDATE command may be used to set and change the default for processing performed when a datetime $function detects an error.

```
APPDATE [USER|SYSTEM] DATE_ERR -
        [OFF | CANCEL | REPORT | IGNORE] -
        [DEBUG | NODEBUG]
```

APPDATE syntax to set or change default error handling

where

USER | SYSTEM

optional specification of the scope for the current command, defaults to USER. The USER error handling overrides the SYSTEM error handling. Both USER and SYSTEM error handling are initially OFF.

OFF | CANCEL | REPORT | IGNORE

This is optional. OFF indicates that there is no default error handling at the given level. CANCEL indicates that the default error handling at the given level is issuing an error message describing the error and cancelling the User Language request. REPORT indicates that the default error handling at the given level is issuing an error message describing the error and returning an error indicator. IGNORE indicates that the default error handling at the given level is returning an error indicator.

DEBUG | NODEBUG

If DEBUG is specified, then any datetime error message produced at the given level (user or system) will include the name of the procedure file, the procedure name, and the line number within the procedure for the affected $function invocation. If neither DEBUG nor NODEBUG is specified, then either OFF, CANCEL, REPORT, or IGNORE must be, in which case the default DEBUG | NODEBUG setting is determined from the following table:

CANCEL	DEBUG
REPORT	DEBUG
IGNORE	NODEBUG
OFF	There is no DEBUG | NODEBUG setting at the level.

Since the initial DATE_ERR setting is OFF, there is no DEBUG | NODEBUG setting, at both SYSTEM and USER levels.

Both the USER and SYSTEM level defaults can be overridden by the setting of the $SIR_DATE_ERR function, which in turn is overridden by supplying a non-default value for the error handling argument, if a Sirius date $function is in use. The $SIR_DATE_ERR error handling is initially OFF. If $SIR_DATE_ERR('OFF') is in effect, the USER level handling controls default error handling, unless it is OFF, in which case the SYSTEM level controls the default.

Note that the APPDATE command and $SIR_DATE_ERR only set the *default* datetime argument error handling. It will affect the CCA date $functions, and it will affect the Sirius date $functions if they are coded without the error handling argument.

3.2.2 Setting Default Error Handling with the $SIR_DATE_ERR Function

The $SIR_DATE_ERR function provides the ability to set or override default datetime $function error handling at the request level:

```
%S STRING LEN 8
%S = $SIR_DATE_ERR('REPORT')
```

The effect of a $SIR_DATE_ERR invocation persists until the end of the User Language request, at which time its setting is reset to OFF. Note that $SIR_DATE_ERR only sets the *default* datetime argument error handling. It will affect the CCA date $functions, and it will affect the Sirius date $functions if they are coded without the error handling argument.

For a complete description of $SIR_DATE_ERR, see "$SIR_DATE_ERR: Set and query default error handling at request level" on page 27.

3.3 Examining the APPDATE Information

The *Sir2000 User Language Tools* provide mechanisms for displaying and saving the state of the APPDATE clocks and various DATE_ERR switches. Thus, it is possible for applications to retrieve the current states, temporarily change the states and then restore the previous states.

3.3.1 APPDATE DISPLAY Command

The following form of the APPDATE command allows you to view the setting of either the USER or SYSTEM clock, and to view the error handling setting:

```
APPDATE DISPLAY [ALL|USER|SYSTEM]
```

APPDATE syntax to display settings

where

ALL | USER | SYSTEM This is optional. It specifies to either display the current time and error handling setting for all levels, or to display either the USER or SYSTEM level in a form that is equivalent to the APPDATE command which set it.

ALL is the default and prints the current internal time, the clock ON|OFF|INTERNAL settings for both USER and SYSTEM, the DATE_ERR settings of both USER and SYSTEM, what the current times would be for the clocks if they were in effect, and an indication of which clock and which error handling setting is active for the current user. Following this is the display of the USER and SYSTEM levels in the "command setting" form, that is, the command that was issued to set the level.

APPDATE DISPLAY ALL is not suitable for placing in a global and re-invoking; you should use APPDATE DISPLAY {USER | SYSTEM} for that purpose. The display format of APPDATE DISPLAY ALL is shown in the following example:

```
LEVEL    CLOCK STATE/DATE_ERR    CURRENT CLOCK
-------- ---------------------- ------------------------
INTERNAL                        1997/05/14 11:27:51
SYSTEM   ON/OFF                 2000/01/02 11:27:51 (@)
USER     OFF/CANCEL DEBUG       1997/05/14 11:27:51 (*)

SYSTEM CLOCK ACTIVE (@); USER LEVEL IS ERROR HANDLING DEFAULT (*)
COMMAND SETTINGS:
APPDATE SYSTEM ON 2000/01/01 AT 1997/05/13 DATE_ERR OFF
APPDATE USER OFF 1997/05/13 AT 1997/05/13 DATE_ERR CANCEL DEBUG
```

The APPDATE DISPLAY {USER | SYSTEM} forms of the command produce results in the form of a legal APPDATE command which would set the levels to their current values. For example, the following lines would result from invocations of the APPDATE DISPLAY USER and APPDATE DISPLAY SYSTEM commands in the same context as above:

```
APPDATE SYSTEM ON 2000/01/01 AT 1997/05/13 DATE_ERR OFF
APPDATE USER OFF 1997/05/13 AT 1997/05/13 DATE_ERR CANCEL DEBUG
```

3.3.2 Retrieving Last Non-null $SIR_DATE_ERR Setting

If you call $SIR_DATE_ERR with no argument or the null string, $SIR_DATE_ERR will return a character string indicating the most recent valid and non-null value specified in a call to $SIR_DATE_ERR during the current request; it will return one of the following strings:

- 'OFF'
- 'CANCEL'

- 'REPORT'
- 'IGNORE'

```
%S STRING LEN 8
%S = $SIR_DATE_ERR( <no argument> | '' )
```

For a complete description of $SIR_DATE_ERR, see "$SIR_DATE_ERR: Set and query default error handling at request level" on page 27.

3.4 Privileges, Disabling, and Pre-User 0 APPDATE Command

The APPDATE SYSTEM form of the command can be issued only by user 0 (user zero, that is, in the CCAIN input stream) or by a user with system manager privileges; APPDATE USER and APPDATE DISPLAY can be issued by any user. APPDATE SYSTEM can also be used in the CCAIN input stream prior to the user 0 parameter line, so that the system-wide APPDATE clock and error handling can be set prior to any calculations performed by an IODEV 3 thread. Neither APPDATE USER nor APPDATE DISPLAY may be used prior to the user 0 parameter line.

The APPDATE RESTRICT command can be used to prohibit any issuance of the APPDATE command except prior to the user 0 parameter line. This could be used if you do not want anyone to change the APPDATE settings after initialization of the *Model 204* run. The APPDATE RESTRICT command itself can only be issued prior to the user 0 parameter line.

You can also disable or restrict the APPDATE command by applying a customization zap, based on the examples in the "Customization Zaps" chapter in the *Sirius Mods Installation Guide*. You might use a customization zap if, for example, you wanted to disallow the use of the APPDATE command in a production load module, or if you wanted to only allow one system-wide APPDATE setting for the entire *Model 204* run.

Note:

- Lines are not sent to the *Model 204* audit trail for commands, and subsequent error messages, issued prior to the user 0 parameter line. Output is, however, sent to CCAPRINT prior to the user 0 parameter line.

3.5 Complete Syntax of the APPDATE Command

The detailed explanation of setting APPDATE values is described separately for clock values (see “Setting the Clocks With the APPDATE Command” on page 10) and datetime argument error handling (see “Error Handling Control with DATE_ERR” on page 13) but for either the USER or SYSTEM level you can specify both clock values and error handling, in that order, in a single command. The complete syntax of the APPDATE command is shown below.

```
Appdate_cmd =  APPDATE { Level Clock_set [Err_set]  |
                         Level Err_set  |
                         Display |
                         RESTRICT }
Clock_set   =  Clock_state  |  [Clock_state] Date_times
Level       =  [USER  |  SYSTEM]
Clock_state =  ON  |  OFF  |  INTERNAL
Date_times  =  {targdate [AT refdate]  |
                targdate targtime [AT refdate reftime]}
Err_set     =  DATE_ERR [OFF | CANCEL | REPORT | IGNORE]
               [DEBUG | NODEBUG]
Display     =  DISPLAY [ALL | USER | SYSTEM]
```

Note: Either {OFF | CANCEL | REPORT | IGNORE} or {DEBUG | NODEBUG} must be specified with DATE_ERR.

Complete syntax of APPDATE command

As discussed in “Privileges, Disabling, and Pre-User 0 APPDATE Command” on page 18:

- The APPDATE SYSTEM command may only be issued in the user 0 input stream or by a system manager; an error message is produced if APPDATE SYSTEM is processed in other contexts.
- The APPDATE RESTRICT command may only be issued in the user 0 input stream; an error message is produced if APPDATE RESTRICT is processed in other contexts.
- The APPDATE command can also be completely disabled, or restricted to the user 0 input stream prior to the user 0 parameter line, by using a customization zap.
- Issuing an APPDATE command in a disabled or restricted context produces an error message but does no checking or processing of the command.

3.6 Values affected by the APPDATE clocks

The active APPDATE clock affects the value returned by the following User Language $functions:

- $DATE, $DATEJ, $DATEP
- $DATEDIF, $DATECNV, and $DAYI (when obtaining the year for handling the CENTSPLT argument)
- $TIME
- $SIR_DATE, $SIR_DATEN, $SIR_DATEND, $SIR_DATENM, $SIR_DATENS, $SIRTIME and $WEB_DATE
- $WEB_LAST_MODIFIED (when ensuring last_modified date isn't future)
- $SIR_DATECHG, $SIR_DATECHK, $SIR_DATECNV, $SIR_DATEDIF, $SIR_DATE2N, $SIR_DATE2ND, $SIR_DATE2NM, $SIR_DATE2NS, $DB_RPCPARAM, $SRV_BIND, and $SRV_PARMSET (when obtaining the year for relative CENTSPAN handling)

The only *Model 204* command affected by the APPDATE clock is the RESET command for changing the CENTSPLT user parameter. In this case, the BASECENT parameter, which is calculated using the current time, obtains the current time from the active APPDATE clock. The resulting value of BASECENT can be used to interpret 2-digit years passed to the following User Language $functions:

- $DATEDIF, $DATECNV, and $DAYI, if neither the CENTSPLT nor DEFCENT arguments are provided
- $DATECHG and $DATECHK

Finally, the default value of DEFCENT is taken from the active APPDATE clock if the APPDATE command is issued prior to the user 0 parameter line.

3.7 APPDATE Command Usage Notes

1. One kind of application testing is simply to see what happens when dates into the 21st century are returned by User Language $functions. This can be done in isolation by issuing a command such as the following:

   ```
   APPDATE USER 2000/01/01
   ```

   This can be simply issued by the user before entering the application, or it can be issued in the startup or logon proc of the application.

2. If your application stores results that are retrieved later by the application, you should probably put a command such as the following into the startup proc:

   ```
   APPDATE USER 2000/01/01 AT 1997/05/01
   ```

This indicates that the application clock will run ahead of the internal date and time so that on the 1st of May, 1997, the application clock will show the 1st of January, 2000. Your application can thus run as if it were "shifted" ahead in time, and the shift amount will be consistent, even if you bring down the online and bring it up again. We recommend placing it in the startup proc in case any dates are referenced in it.

3. If you want all applications (except those "shielded" as in 5. below) to use the same clock, you need not change any applications but simply code before the USER 0 parameter line:

```
APPDATE SYSTEM 2000/01/01 AT 1997/05/01
```

Note that this also changes the default DEFCENT; see (6.) below.

4. If you have an application that must use the internal date and time, even if the system-wide application clock is ON, you should enter a command such as the following in the startup proc:

```
APPDATE USER INTERNAL
```

This could be needed by an application that examines internal datetime values, such as those found in the audit trail. A robust application will preserve the APPDATE value and can do this using the $COMMNDL function (**http://M204WIKI.ROCKETSOFTWARE.COM/index.php/$CommndL**), since an SDAEMON thread initiated with $COMMxxx copies the application clock of its invoker.

5. A good date for APPDATE testing might be 2000/02/29, since that is a valid date, but 1900/02/29 is not.

6. The use of the APPDATE command is to establish a clock for all application-level operations. It does not affect static values that had been established prior to the APPDATE command. This is relevant to the value of the BASECENT parameter and the default value of the DEFCENT user parameter, as follows:

   - If the DEFCENT parameter has not been set, its value is the first two digits of the year during M204 initialization. If you issue an APPDATE SYSTEM ON ... command before the user 0 parameter line, this will affect the default value of DEFCENT. All other invocations of the APPDATE command will leave DEFCENT unaffected.

   - The BASECENT parameter is calculated based on the current time whenever the CENTSPLT parameer is set. This current time will be taken from the active APPDATE clock, but subsequent invocations of the APPDATE command will not affect the BASECENT parameter.

7. The APPDATE command does not affect the date and time printed on header 0 of each output page.

8. The APPDATE command and the use of the *Model 204* SYSDATE parameter are mutually exclusive.

9. $COMMAND, $COMMBG, and $COMMNDL (see **http://m204wiki.rocketsoftware.com/index.php/List_of_$functions**) provide a mechanism for passing a unit of work to an *SDAEMON*, or background thread that is different from the thread that issues the request. When the requested unit of work receives control, the APPDATE USER settings from the thread that invoked the $function will be in effect. If an APPDATE USER command is issued in the sdaemon thread, its effect will not be reflected back to the invoking thread.

10. The current application date used by #DATExxx functions in FUEL is set to the active APPDATE clock when the $FUNLOAD function is used.

CHAPTER 4 $Functions

This chapter lists the $functions provided with the *Sir2000 User Language Tools*. Each $function is presented here with a brief phrase denoting its use, a short explanation, the form and types of its arguments and the result (or "output value"), one or more examples, any error conditions, and special notes.

Unless otherwise indicated, all arguments shown are required and are input arguments.

In some instances the behavior of a $function will be compared to the behavior of a corresponding standard User Language $function described in M204wiki (see **http://m204wiki.rocketsoftware.com/index.php/Category:SOUL_$functions**).

In addition to the $functions that are only available with the *Sir2000 User Language Tools*, this chapter includes documentation of some additional $functions described in M204wiki at **http://m204wiki.rocketsoftware.com/index.php/List_of_$functions**.

In addition to the reference material in this chapter, the index contains a major heading labelled **$Function prototypes**. Under this heading are minor headings containing the form of the $functions, for your convenient reference.

4.1 Errors in Datetime $Functions

The *Sir2000 User Language Tools* provides you with several ways to detect datetime related errors and control the action taken by the system when these errors occur. You can control this on a system-wide or thread-only basis by using the APPDATE DATE_ERR command, on a request level by using the $SIR_DATE_ERR function, and in Sirius date $function calls by coding a special argument. You can control error handling in some standard User Language date $functions also, by using either APPDATE DATE_ERR or $SIR_DATE_ERR.

The Sirius date $functions provide you with simple and flexible calls to manipulate date/time data, and to detect and report error conditions that may be the result of data or coding errors related to year 2000 processing.

If you intend to use the $function error handling features of *Sir2000 User Language Tools*, you should also be sure to use the DEBUG feature of APPDATE. The DEBUG feature saves line number, procedure, and file information for date-related $functions. This information, along with argument values, is then displayed and reported to greatly simplify debugging.

The Sirius date $functions provide an optional error control argument for some datetime $functions. If you omit the error control argument and an error is detected in the $function, the action taken depends upon the latest invocation of $SIR_DATE_ERR, if any, or the switches established by the relevant APPDATE command with DATE_ERR clause.

See "Error Handling Control with DATE_ERR" on page 13 for a complete description of the type of errors encountered by date $functions, and for the control and detection approaches available to you with the *Sir2000 User Language Tools*.

For example, since the input date doesn't match the provided format, the following fragment:

```
%NEW_DT = $SIR_DATECHG('MM/DD/YY', '96/Ø1/Ø1',, -
                       7, 'IGNORE')
IF %NEW_DT EQ '' OR $SUBSTR(%NEW_DT, 1, 1) = '*' THEN
   PRINT 'Error incrementing date'
END IF
```

will print `Error incrementing date`, while the following fragment:

```
%NEW_DT = $SIR_DATECHG('MM/DD/YY', '96/Ø1/Ø1',, -
                       7, 'CANCEL')
```

will cause the request to be cancelled with an error message:

```
I DATERR
***  1  MSIR.Ø323: Error in $SIR_DATECHG call, no Sirius debug info
- Invalid date
***  MSIR.Ø326: $SIR_DATECHG - argument 1 = 'MM/DD/YY'
***  MSIR.Ø326: $SIR_DATECHG - argument 2 = '96/Ø1/Ø1'
***  MSIR.Ø326: $SIR_DATECHG - argument 3 = '7'
***  MSIR.Ø326: $SIR_DATECHG - argument 5 = 'CANCEL'
***  2  CANCELLING REQUEST: MSIR.Ø324: Cancelling request because of
$SIR_DATECHG error
```

If the APPDATE DATE_ERR DEBUG option is in effect, the User Language procedure name and line number will be reported, along with more detail about the error:

```
APPDATE USER DATE_ERR DEBUG
I DATERR
***  1  MSIR.Ø321: Error in $SIR_DATECHG call in line 523, procedure
DATERR, file PROCFILE - Invalid date
***  MSIR.Ø326: $SIR_DATECHG - argument 1 = 'MM/DD/YY'
***  MSIR.Ø326: $SIR_DATECHG - argument 2 = '96/Ø1/Ø1'
***  MSIR.Ø326: $SIR_DATECHG - argument 3 = '7'
***  MSIR.Ø326: $SIR_DATECHG - argument 5 = 'CANCEL'
***  2  CANCELLING REQUEST: MSIR.Ø324: Cancelling request because of
$SIR_DATECHG error
```

Each $function description in this chapter has a figure showing the error conditions detected by the $function and the value returned by the $function in case of an error with 'IGNORE' or 'REPORT' specified or defaulted.

4.2 $SIR_DATE: Get current datetime

For documentation of this $function, see
http://M204wiki.rocketsoftware.com/index.php/$Sir_Date.

4.3 $SIR_DATE_ERR: Set and query default error handling at request level

The $SIR_DATE_ERR function may be used to set or query the default request-level datetime error handling. For more information on datetime error control, refer to "Error Handling Control with DATE_ERR" on page 13.

```
%oldvalue = $SIR_DATE_ERR( newvalue or '' )
```

where

newvalue an optional error control string, which must contain one of the following values:

OFF indicates that default datetime $function error handling is determined by the USER level setting or, if that is OFF, by the SYSTEM level setting.

CANCEL indicates that when a datetime $function detects an error, the default behaviour will be to issue an error message and cancel the User Language request.

REPORT indicates that when a datetime $function detects an error, the default behaviour will be to issue a warning message and return the appropriate error value to the function caller.

IGNORE indicates that when a datetime $function detects an error, the default behaviour will be to silently return the appropriate error value to the function caller.

'' or missing argument
does not affect the current request-level error processing defaults. Provided as a method to obtain the current $SIR_DATE_ERR setting.

%oldvalue set to the most recent valid non-null setting of $SIR_DATE_ERR.

The request-level datetime error handling defaults set by a $SIR_DATE_ERR invocation persist until the end of the User Language request, at which time its setting is reset to OFF. Note that $SIR_DATE_ERR only sets the *default* datetime argument error handling. It will affect the standard User Language date $functions, and it will affect the Sirius date $functions only if they are coded without an error control argument.

4.4 $SIR_DATECHG: Add some days to datetime

The $SIR_DATECHG function expects a datetime format string, a datetime value string, and a number of days. It accepts an optional CENTSPAN value and an optional error control string. It returns the datetime string obtained by adding the indicated number of days to the input datetime; the return datetime string is in the same format as the input datetime string.

The $SIR_DATECHG function is similar to the $DATECHG function provided by CCA. However, the $DATECHG function does not support the specification of a CENTSPLT (**http://m204wiki.rocketsoftware.com/index.php/CENTSPLT_parameter**) or DEFCENT (**http://m204wiki.rocketsoftware.com/index.php/DEFCENT_parameter**) argument.

```
%odat = $SIR_DATECHG(fmt, dat, days, span, errctl)
```

where

fmt datetime format string, describes *dat* and *%odat*. Refer to "Datetime Formats" on page 41 for an explanation of valid datetime formats and valid datetime values. Strict matching is used for *fmt*.

dat datetime value string.

days number of days to add to *dat*.

span optional CENTSPAN value, default is -50. Refer to "CENTSPAN" on page 46.

errctl optional error control string, refer to "Error Handling Control with DATE_ERR" on page 13.

%odat set to *dat* plus *days* 24-hour periods, unless an error is detected.

For example, the following fragment prints the date one week after the run date:

```
%X = $SIR_DATE('DAY Month, YYYY')
PRINT $SIR_DATECHG('DAY Month, YYYY', %X, 7)
```

Error conditions are shown in the following figure (see the discussion in "Errors in Datetime $Functions" on page 23).

$SIR_DATECHG returns a string composed of all asterisks ("*"), whose length is the shorter of the length of the input date format string or 32, in the following error cases:

- *fmt* is not a valid datetime format.
- *date* does not match *fmt* or result date out of range.
- *days* is omitted.
- *span* contains an invalid CENTSPAN specification.

4.5 $SIR_DATECHK: Check if datetime matches format

The $SIR_DATECHK function expects a datetime format string and a datetime value string. It accepts an optional CENTSPAN value, and an optional error control string. $SIR_DATECHK returns the value 1 if the datetime value is valid and matches the provided format, otherwise the value 0 is returned.

The $SIR_DATECHK function is similar to the $DATECHK standard User language function. However, the $DATECHK function does not support the specification of a CENTSPLT (**http://m204wiki.rocketsoftware.com/index.php/CENTSPLT_parameter**) or DEFCENT (**http://m204wiki.rocketsoftware.com/index.php/DEFCENT_parameter**) argument.

```
%tst = $SIR_DATECHK(fmt, dat, span, errctl)
```

where

fmt datetime format string. Refer to "Datetime Formats" on page 41 for an explanation of valid datetime formats and valid dates. Strict matching is used for *fmt*.

dat datetime string.

span optional CENTSPAN value, default is -50. Refer to "CENTSPAN" on page 46.

errctl optional error control string, refer to "Error Handling Control with DATE_ERR" on page 13.

%tst set to 1 if *dat* matches *fmt*, otherwise set to 0.

The following fragment prints the string **Bad**, since February cannot have thirty days. Note that the request is not cancelled, even though an optional error control string was provided:

```
%X = $SIR_DATECHK('DAY Month, YYYY', -
                  '30 February, 1997',,'CANCEL')
IF %X = 1 THEN
   PRINT 'Good'
ELSE
   PRINT 'Bad'
END IF
```

Error conditions are shown in the following figure (see the discussion in "Errors in Datetime $Functions" on page 23).

$SIR_DATECHK returns a value of 0 if any of the following errors are detected:

- *fmt* is not a valid datetime format.
- *dat* is not a valid date, or does not match *fmt*, or is outside of range permitted for *fmt*. Note that these cases are always treated as if an error control string of **IGNORE** had been specified.
- *span* is not a valid CENTSPAN value.

4.6 $SIR_DATECNV: Convert datetime to different format

The $SIR_DATECNV function expects a datetime format string for the input string, a datetime format for the output string, and a datetime value to be converted. It also accepts an optional **CENTSPAN** value and an optional error control string. $SIR_DATECNV returns the input datetime converted to the output format.

```
%odat = $SIR_DATECNV(infmt, outfmt, dat, span, errctl)
```

where

infmt datetime format string for *dat*. Refer to "Datetime Formats" on page 41 for an explanation of valid datetime formats and valid dates. Strict matching is used for *infmt*.

outfmt datetime format string for function output (*%odat*).

dat input datetime string.

span optional CENTSPAN value, default is -50. Refer to "CENTSPAN" on page 46.

errctl optional error control string, refer to "Error Handling Control with DATE_ERR" on page 13.

%odat Set to the value of *dat*, converted from the format in *infmt* to the format in *outfmt*, unless an error is detected.

For example, the following fragment prints the string **19970101**:

```
PRINT $SIR_DATECNV('YYMMDD', 'YYYYMMDD', '970101', 1950)
```

Error conditions are shown in the following figure (see the discussion in "Errors in Datetime $Functions" on page 23).

$SIR_DATECNV returns a string composed of all asterisks (*), whose length is the shorter of the length of the output date format string or 32, in the following error cases:

- *infmt* or *outfmt* is not a valid datetime format string.
- *dat* does not match *infmt*.
- *dat* is outside of range permitted for *infmt* or that permitted for *outfmt*.
- *span* is invalid.

4.7 $SIR_DATEDIF: Difference between two dates

The $SIR_DATEDIF function expects a datetime format string and two datetime value strings. It accepts a second optional datetime format string, an optional **CENTSPAN** value, and an optional error control string. $SIR_DATEDIF subtracts the second datetime value from the first datetime value, returning the number of complete days of difference. If an error is detected, the value 99999999 is returned.

```
%dif = $SIR_DATEDIF(fmta, data, fmtb, datb, span, errctl)
```

where

fmta first datetime format string. Refer to "Datetime Formats" on page 41 for an explanation of valid datetime formats and valid datetime values. Strict matching is used for *fmta*.

data first datetime value string.

fmtb optional second datetime format string, default is to use *fmta*. Strict matching is used for *fmtb*.

datb second datetime value string.

span optional CENTSPAN value, default is -50. Refer to "CENTSPAN" on page 46.

errctl optional error control string, refer to "Error Handling Control with DATE_ERR" on page 13.

%dif set to the number of days obtained from subtracting *datb* from *data*. If an error is detected, the value returned is 99999999.

For example, the following fragment prints the string **-7 days**:

```
PRINT $SIR_DATEDIF('YYMMDD', '970301', , -
                   '970308') AND 'days'
```

Error conditions are shown in the following figure (see the discussion in "Errors in Datetime $Functions" on page 23).

$SIR_DATEDIF returns the value 99999999 in the following error cases:

- *fmta* or *fmtb* is not a valid datetime format.
- *data* or *datb* does not match the respective datetime format.
- *data* or *datb* is outside of range permitted for the respective datetime format..
- *span* is invalid.

Notes:

Time is ignored in the subtraction.

In the example above, even though the input dates have 2-digit years, the correct answer would be given for any valid *span* argument, since the dates do not span the end of February.

As in the standard User Language $DATEDIF function, the single *span* argument is used for both dates; if it is necessary to get the difference between two dates which both have 2-digit years and are in different 100-year windows, you must first use $SIR_DATECNV to convert one of them to some 4-digit year format. Alternatively, you could use $SIR_DATE2ND to convert both dates to number of days values, and subtract those values.

4.8 $SIR_DATEFMT: Validate datetime format

For documentation of this $function, see
http://M204wiki.rocketsoftware.com/index.php/$Sir_DateFmt.

4.9 $SIR_DATEN/ND/NM/NS: Current date and time as integer

For documentation of these $functions, see for example, **http://M204wiki.rocketsoftware.com/index.php/$Sir_DateN**.

4.10 $SIR_DATE2N/ND/NM/NS: Convert datetime string to integer

For documentation of these $functions, see for example,
http://M204wiki.rocketsoftware.com/index.php/$Sir_Date2N.

4.11 $SIR_ND2DATE/NM2DATE/NS2DATE/N2DATE: Convert datetime integer to string

For documentation of these $functions, see for example, **http://M204wiki.rocketsoftware.com/index.php/$Sir_ND2Date**.

CHAPTER 5 Datetime Processing Considerations

This chapter presents date processing issues, including usage of the *Sir2000 User Language Tools* past the year 1999, an explanation of its processing of dates, and any rules and restrictions you must follow to achieve correct results using date values with the *Sir2000 User Language Tools*.

The *Sir2000 User Language Tools* uses dates in the following ways:

- To examine the CPU clock (as returned by the STCK hardware instruction) to determine the current date, in case the *Sir2000 User Language Tools* is under a rental or trial agreement
- As arguments to various $functions, and returned values from them
- To set a date for testing with the APPDATE command
- See the *SirLib User's Guide* and the *SirPro User's Guide* for date processing considerations for those products as part of the *Sir2000 User Language Tools*.

To correctly use the *Sir2000 User Language Tools* past the year 1999, version 5.0 of the *Sirius Mods* and version 5.0 of *UL/SPF*, or later, are required. For headers on pages or rows that occur on printed pages or displayed screens, Sirius Software products generally use a full four digit year format, although they may display dates with two digit years in circumstances where the proper century can be inferred from the context.

Above and beyond the post-1999 requirements specific to the *Sir2000 User Language Tools*, you must examine all uses of date values in your applications to ensure that each of your applications produces correct results. Furthermore, both the operating system and *Model 204* must correctly process and transmit dates beyond 1999 in order for the *Sir2000 User Language Tools* to operate properly.

Most Sirius date processing involves the use of datetime $functions. This chapter refers to datetime $functions in two product groups:

1. The *Sirius Functions*, which are documented in the M204wiki at **http://m204wiki.rocketsoftware.com/index.php/List_of_$functions**. All of these $functions that concern dates are available to users of the *Sir2000 User Language Tools*.
2. The *Sir2000 User Language Tools Functions*, which are documented in the *Sir2000 User Language Tools Reference Manual*. These $functions are only available to users of the *Sir2000 User Language Tools*.

The occasional references to "all Sirius datetime $functions" stand for all date processing $functions formerly delivered by Sirius Software, in any product.

In operational terms, there are two classes of datetime $functions:

1. $Functions using a numeric value to represent a datetime, where 0 represents 12:00 AM, 1 January 1900; for example, $SIR_DATE2NM and $SIR_NM2DATE (number of milliseconds since the start of 1900).

   These $functions, and $SIR_DATE, have the following error return values:

   - **-9.E12** for numeric result $functions
   - **null string** for string result $functions

   They also perform **non-strict** matching of date strings to date formats; for example, a leading blank is allowed for the HH token.

   All numeric datetime $functions, and $SIR_DATE, are part of the *Sirius Functions*.

2. Other $functions that only manipulate strings and associated datetime formats ($SIR_DATE not included in this class); for example, $SIR_DATECHG (add number of days to given date).

   These $functions have error return values of a variable number of asterisks (or, in the case of $SIR_DATEDIF, the value 99,999,999). They also perform **strict** matching of date strings to date formats; for example, a leading blank is **not** allowed for the HH token. These $functions produce the same results as CCA $DATExxx functions, with additional enhancements.

   These string format datetime $functions are available only with the *Sir2000 User Language Tools*.

See “Strict and non-strict format matching” on page 48 for a discussion of strict and non-strict format matching, including a technique for accomplishing strict date checking using the non-strict $functions.

The rest of this chapter contains a discussion of datetime formats, valid datetime strings, and processing of two-digit year values. It also contains example datetime formats and corresponding example datetime strings. Finally, there is a list of benefits of Sirius datetime processing.

5.1 Datetime Formats

The representation of a date is determined by a *datetime format*. This value is a character string, composed of the concatenation of tokens (for example, "YYYY" for a four-digit year, and "MI" for minutes) and separator characters (for example, "/" in "MM/DD/YY" for two-digit month, day, and year separated by slashes).

These *datetime format* strings are used in many products in addition to the *Sir2000 User Language Tools*. The products using datetime format strings are:

- *Fast/Unload*
- *Janus Open Client*
- *Janus Open Server*
- *Janus Specialty Data Store*
- *Janus Web Server*
- *SirDBA*
- *Sirius Functions*
- *Sir2000 Field Migration Facility*
- *Sir2000 User Language Tools*

The rules for these *datetime format* strings are consistent throughout all these products, though certain uses of these strings might impose extra restrictions. For example, a leading blank is allowed for the HH, DD, and MM parts of a date argument using a non-strict date $function, such as $SIR_DATE2NS, but is not allowed for the strict date $functions (i.e., the *Sir2000 User Language Tools Functions*).

There are certain rules applied to determine if a format is valid. The basic rules are:

1. If a format string contains a numeric datetime token (i.e. "ND", "NM", or "NS"), then the format string must consist of only one token. Numeric datetime tokens are only supported in format strings for the *Sir2000 Field Migration Facility*.

2. You must specify at least one time, weekday, or date token.

3. Except for "weekday", you can't specify redundant information. More specifically this means

    - Except for "I", no token can be specified twice.

    - At most one year format (contains Y) can be specified.

    - At most one month format (contains MON, Mon, or MM) can be specified.

    - At most one day format (DD or Day) can be specified.

    - At most one weekday format (WKD, Wkd, WKDAY, or Wkday) can be specified.

    - If AM is specified, then PM can not be specified.

- At most one fractions-of-a-second format (contains X) can be specified.
- If DDD is specified, then neither a day nor month format can be.

4. If ZYY is specified in a format string, no other token that denotes a variable-length value may be used.

5. If a format string contains other tokens that denote variable length values, then an * token may only appear as the last character of the format string.

6. The DAY token may not be immediately followed by another token whose value may be numeric, regardless of whether the following token repsents a variable length value. Thus, DAY may not be followed by *, I, YY, YYYY, CYY, MM, HH, MI, SS, X, XX, or XXX; DAY may not be followed by a decimal digit separator, and DAY may not be followed by a quote followed by a decimal digit.

7. When a pair of format strings are used for transforming date values, for example for $SIR_DATECNV or processing of updates to SIRFIELD RELATEd fields, additional rules apply to the pattern matching tokens:

   - If one of the format strings includes one or more "I" tokens, then the other format string must contain the same number of "I" tokens. Note that the placement of "I" tokens within the format strings is not restricted. The "I" tokens are processed left to right, with each character from the input string that corresponds to the nth "I" token in the input format being copied unchanged to the character position in the output string that corresponds to the nth "I" token in the output format.
   - If one of the format strings contains an asterisk (*) token, then the other format string must also contain an asterisk token. All of the characters from the input string that correspond to the asterisk token in the input format, if any, are copied unaltered to the output string, begining in the position that corresponds to the asterisk token in the output format.

   SIRFIELD is part of the *Sir2000 Field Migration Facility*.

8. The maximum length of a format string is 100 characters.

Note: A common mistake is to use "MM" for minutes; it should be "MI".

The valid tokens in a date format are shown in the following list. In general, the output format rule for a token is shown. For some of the $functions, the input format rule for a token is the same as the output format rule; this is the definition of "strict date format matching". However, non-strict $functions sometimes allow a string to match a token on input that would not be produced by that token on output.

All of the tokens that match alpabetic strings (for example, "MON") match any case for non-strict matching. All other tokens that have differing strict and non-strict matching

rules are listed under "Special date format rules" in the index at the back of the manual, and usage notes for them are contained in “Datetime and format examples” on page 48. Each input datetime format argument in the description of a $function specificies whether the use of the format observes strict or non-strict format matching. See “Strict and non-strict format matching” on page 48.

NM numeric datetime value containing the number of milliseconds (1/1000 of a second) since January 1, 1900 at 12:00 AM. (This token is allowed only in the *Sir2000 Field Migration Facility*.)

NS numeric datetime value containing the number seconds since January 1, 1900 at 12:00 AM. (This token is allowed only in the *Sir2000 Field Migration Facility*.)

ND numeric date value containing the number of days since January 1, 1900. (This token is allowed only in the *Sir2000 Field Migration Facility*.)

***** Ignore entire variable-length substring matching pattern, if any, when only retrieving a date value. Substitute with null string when only creating a date value. When copying date values, copy entire variable-length substring matching pattern, if any, from input value to location identified by * token in output string. See “Datetime and format examples” on page 48.

I Ignore corresponding input character when only retrieving a date value. Store a blank in corresponding output character when only creating a date value. When copying date values, copy each character matching an I token from from the input value to location in the output string identified by the corresping I token in the output format. See “Datetime and format examples” on page 48.

" Following character is "quoted", that is, it acts as a separator character. See “Datetime and format examples” on page 48.

YYYY Four-digit year

YY Two-digit year

CYY Year minus 1900 (three digits, including any leading zero). See “Datetime and format examples” on page 48.

ZYY Year minus 1900, two-digit or three-digit year number, excluding any leading zero (variable length data). Non-strict $functions allow a three-digit number with leading zero on input, but any number less than 100 always produces a two-digit number on output. See “Datetime and format examples” on page 48.

MONTH Full-month name (uppercase variable length). Non-strict $functions allow any mixture of uppercase and lowercase on input, but all uppercase is always produced on output.

Month Full-month name (mixed-case variable length). Non-strict $functions allow any mixture of upper and lowercase on input, but initial uppercase letter followed by all lowercase is always produced on output.

MON Three-character month abbreviation (uppercase). Non-strict $functions allow any mixture of upper and lowercase on input, but all uppercase is always produced on output.

Mon Three-character month abbreviation (mixed case). Non-strict $functions allow any mixture of upper and lowercase on input, but initial uppercase letter followed by all lowercase is always produced on output.

MM	Two-digit month number. Non-strict $functions allow a two-character number with leading blank on input, but two decimal digits are always produced on output. See "Datetime and format examples" on page 48.
BM	Two-character month number; if less than 10, first character is blank. Non-strict $functions allow a two-digit number with leading zero on input, but any number less than 10 always produces a blank followed by a decimal digit on output. See "Datetime and format examples" on page 48.
DDD	Three-digit Julian day number
DD	Two-digit day number. Non-strict $functions allow a two-character number with leading blank on input, but two decimal digits are always produced on output. See "Datetime and format examples" on page 48.
BD	Two-character day number; if less than 10, first character is blank. Non-strict $functions allow a two-digit number with leading zero on input, but any number less than 10 always produces a blank followed by a decimal digit on output. See "Datetime and format examples" on page 48.
DAY	One-digit or two-digit day number (variable length data). Non-strict $functions allow a two-digit number with leading zero on input, but any number less than 10 always produces a one-digit number on output. See "Datetime and format examples" on page 48.
WKDAY	Full day of week name (uppercase variable length). Non-strict $functions allow any mixture of upper and lowercase on input, but all uppercase is always produced on output.
Wkday	Full day of week name (mixed case variable length). Non-strict $functions allow any mixture of upper and lowercase on input, but initial uppercase letter followed by all lowercase is always produced on output.
WKD	Three-character day of week abbreviation (uppercase). Non-strict $functions allow any mixture of upper and lowercase on input, but all uppercase is always produced on output.
Wkd	Three-character day of week abbreviation (mixed case). Non-strict $functions allow any mixture of upper and lowercase on input, but initial uppercase letter followed by all lowercase is always produced on output.
HH	Two-digit hour number. Non-strict $functions allow a two-character number with leading blank on input, but two decimal digits are always produced on output. See "Datetime and format examples" on page 48.
BH	Two-character hour number; if less than 10, first character is blank. Non-strict $functions allow a two-digit number with leading zero on input, but any number less than 10 always produces a blank followed by a decimal digit on output. See "Datetime and format examples" on page 48.
MI	Two-digit minute number
SS	Two-digit second number
X	Tenths of a second
XX	Hundredths of a second
XXX	Thousandths of a second (milliseconds)
AM	AM/PM indicator
PM	AM/PM indicator

The valid separators in a date format are:

blank (" ")
apostrophe ("'")
slash ("/")
colon (":")
hyphen ("-")
back slash ("\")
period (".")
comma (",")
underscore ("_")
left parenthesis ("(")
right parenthesis (")")
plus ("+")
vertical bar ("|")
equals ("=")
ampersand ("&")
at sign ("@")
sharp ("#")
the decimal digits ("0" - "9").

In addition, any character may be a separator character if preceeded by the quoting character (").

See "Datetime and format examples" on page 48 for examples which include use of various separator characters.

5.2 Valid Datetimes

For a datetime string to be valid it must meet the following criteria:

- Its length must be less than 128 characters.
- It must be compatible with its corresponding format string.
- It must represent a valid date and/or time. For example, at most 23:59:59.999 for a time, 01-12 for a month, 01-31 or less (depending on the month) for a day, February 29 is only valid in leap years (only centuries divisible by 4 are leap years: 2000 is but neither 1800, 1900, nor 2100 are). Note: weekdays are not checked for consistency against the date; for example, both Saturday, 02/15/97 and Friday, 02/15/97 are valid.
- It must be within the date range allowed for the corresponding format. A datetime string used with a CYY or ZYY format can only represent dates from 1900 to 2899, inclusive. A datetime string used with a YY format can only represent dates in a range of 100 or less years, as determined by CENTSPAN and SPANSIZE. The valid range of dates for all other formats is from 1 January 1753 thru 31 December 9999.

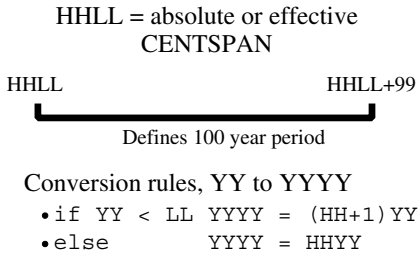
5.3 Processing Dates With Two-Digit Year Values

A date field with only two digits for the year value is capable of representing a range of up to one hundred years. When we compare a pair of two-digit year values we are accustomed to thinking of the century as fixed, so that all dates are either “19xx” or "20xx". However, a date field with two-digit year values can actually represent dates from two different centuries, provided that the *range* of dates does not exceed 100 years.

5.3.1 CENTSPAN

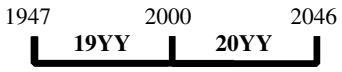
CENTSPAN provides a mechanism for unambiguously converting dates with two-digit year values into dates with four-digit year values. The CENTSPAN mechanism allows two-digit year values to span two centuries without confusion. CENTSPAN identifies the four-digit year value that is the *start* of a range of years represented by the two-digit year values.

CENTSPAN may be specified as an *absolute* unsigned four digit value between 1753 and 9999, or it may be specified as a *relative* signed value between -99 and +99, inclusive. A relative CENTSPAN value is dynamically converted to an *effective* absolute value before it is used to perform a YY to YYYY conversion. The effective CENTSPAN value is formed by adding the relative CENTSPAN to the current four-digit year value at the time the relative value is converted.



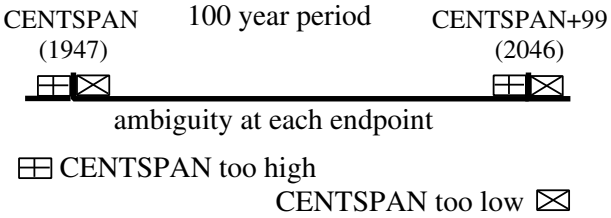
Example:

```
CENTSPAN = -50
current date = 1997
effective CENTSPAN = 1947
```



A simple algorithm is used to convert a two-digit year value (YY) to a four-digit year value, using a four-digit absolute or effective CENTSPAN value (HHLL). If the two-digit year value is less than the low-order two digits of the CENTSPAN value, then the resulting century is one greater than the high-order two digits of the CENTSPAN value. Otherwise the resulting century is the same as the high-order two digits of the CENTSPAN value.

Using all one hundred available years for mapping two-digit year values can cause significant confusion and result in data integrity errors. This is because dates just above and just below the 100-year window are mapped to the other end of the window. From our previous example, the date "47" will be intepreted as 1947, when it could have conceivably been 2047. Simlarly, the date "46" will be intepreted as 2046, when it might have been 1946.



If CENTSPAN is set to a value that is too high, dates that are just prior to CENTSPAN will appear to occur 100 years hence. If CENTSPAN is set to a value that is too low, dates that fall just after CENTSPAN+99 will appear to have occured 100 years earlier. A full one-hundred year window also can not detect attempts to represent more than one hundred years of values with a two-digit year.

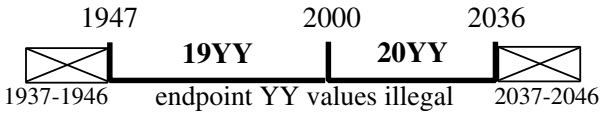
5.3.2 SPANSIZE

There is a method to protect from the ambiguities that can occur at each end of the 100-year window defined by CENTSPAN. SPANSIZE is used to restrict the size of the window used for mapping two-digit year values. The effect is to create two *guard bands*, one just below the date window and one just above. An attempt to represent a date value that lands in a guard band produces an error.

Each guard band contains CENTSPAN-SPANSIZE years, hence a SPANSIZE of 100 removes the protection. The default SPANSIZE is 90, which provides protection for two ten year windows: one below the CENTSPAN setting and one starting at CENTSPAN+90. From our previous example:

Example:

```
CENTSPAN = -50
SPANSIZE = 90
current date = 1997
```



An attempt to represent the values "37" through "46" will be rejected. This protects the range 1937 through 1946 as well as the range 2037 through 2046. Note that an intended value of 2047, expressed as "47" will be accepted and interpreted as 1947. In general a smaller SPANSIZE provides the highest assurance of correct mappings. However, any setting of SPANSIZE less than 100 will probably detect the case where a range greater than one hundred years is being used.

5.4 Strict and non-strict format matching

As mentioned in “Datetime Formats” on page 41, for some of the $functions, the input format rule for a token is the same as the output format rule; this is the definition of "strict date format matching". However, non-strict $functions sometimes allow a string to match a token on input that would not be produced by that token on output. The types of strict matching are as follows:

Alpha tokens For alphabetic tokens (for example, **Month**), a strict match requires the input value to be the correct case. For example, the "MON" token is strictly matched by "JAN" but not by "Jan", and the reverse is true for the "Mon" token. For non-strict matching, the alpabetic tokens are matched by any combination of upper and lowercase input.

HH, MM, DD For these tokens, a strict match requires a leading zero for values less than 10. For non-strict matching, a value less than 10 can also be represented by a leading blank followed by a single numeric digit.

BH, BM, BD For these tokens, a strict match requires a leading blank for values less than 10. For non-strict matching, a value less than 10 can also be represented by a leading zero followed by a numeric digit.

DAY For this token, a strict match requires a single digit for values less than 10. For non-strict matching, a value less than 10 can also be represented by a leading zero followed by a numeric digit.

ZYY For this token, a strict match requires two digits for values less than 100. For non-strict matching, a value less than 100 can also be represented by a leading zero followed by a two numeric digits.

If you want to check a datetime string using strict rules, you can use the following technique with the non-strict date $functions:

```
IF <date> EQ '' OR <date> NE $SIR_NM2DATE(-
   $SIR_DATE2NM(<date>, <fmt>), -
   <fmt>) THEN
   <error handling>
END IF
```

5.5 Datetime and format examples

There is an extensive set of format tokens, as shown in “Datetime Formats” on page 41. These tokens and the various separator characters can be combined in almost limitless possibility, giving rise to an extremely large set of datetime formats. This section provides examples of some common datetime formats, and also tries to explain the use of some of the format tokens which might not be obvious. It also has examples for

formats which have usage with the *Sir2000 User Language Tools* which differs from their usage with other Sirius products. These are noted in the examples and are indexed at the back of this manual under the heading “Special date format rules”. Each example format is explained and also presented with some matching datetimes; again, bear in mind that these tokens can be combined in very many ways and only a very few are shown here. It is assumed that these examples are invoked sometime between the years 1998-2040, as the basis for relative CENTSPAN calculations.

YYMMDD This is the common 6-digit date format which supports sort order if all dates are within a single century. The following User Language fragment

```
PRINT $SIR_DATECHK('YYMMDD', '960229')
```

prints the value 1.

YYYYMMDD

This is the common 8-digit date format which supports sort order with dates in two centuries. The following User Language fragment

```
PRINT $SIR_DATECNV('YYMMDD', -
   'YYYYMMDD', '921212')
```

prints the value 19921212.

MM/DD/YY

This is the U.S. 6-digit date format for display. The following User Language fragment

```
PRINT $SIR_DATECHK('MM/DD/YY', -
   '12/14/94')
```

prints the value 1.

Note:
0

- With non-strict format matching, such as $SIR_DATE2ND, the leading zero corresponding to an MM token may be given as a blank, thus allowing " 7/15/98". With strict matching, however, such a leading blank is not allowed for MM; a leading blank month value with a strict $function (that is, one of the *Sir2000 User Language Tools Functions*) requires the BM token. If the data contains leading zeroes in some month instances and leading blanks in others, you must use a non-strict $function.

DD.MM.YY

This is a European 6-digit date format for display. The following User Language fragment

```
PRINT $SIR_DATECHK('DD.MM.YY', -
   '14.12.94')
```

prints the value 1.

Note:
0

- With non-strict format matching, such as $SIR_DATE2ND, the leading zero corresponding to a DD token may be given as a blank, thus allowing " 1.01.00". With strict matching, however, such a leading blank is not allowed for DD; a leading blank day value with a strict $function (that is, one of the *Sir2000 User Language Tools Functions*) requires the BD token. If the data contains leading zero days in some instances and leading blanks in others, you must use a non-strict $function.

Wkday, DAY Month YYYY "A"T HH:MI

This is a format which could be used for report headers. The following User Language fragment

```
PRINT $SIR_DATE( -
  'Wkday, DAY Month YYYY "A"T HH:MI')
```

prints a value like "Friday, 7 February 1998 AT 21:33".

Notes:

- If an input format contains AM or PM, then the time (HH:MI) must be between 00:01 and 12:00 and must be accompanied by either AM or PM.

- If an input format contains DAY (for example, "DAY MON YY") with non-strict format matching, such as $SIR_DATE2ND, the string matching it may have a leading zero, thus allowing "06 MAY 98". With strict matching $functions (that is, one of the *Sir2000 User Language Tools Functions*) however, such leading zero is not allowed for DAY; a single digit must be supplied for days 1 through 9.

- If an input format contains HH with non-strict format matching, such as $SIR_DATE2ND, the string matching it may have a leading blank, thus allowing " 8:30". With strict matching, however, such a leading blank is not allowed for HH; a leading blank hour value with a strict $function (that is, one of the *Sir2000 User Language Tools Functions*) requires the BH token. If the data contains leading zero hours in some instances and leading blanks in others, you must use a non-strict $function.

YYIIII This is a format that could be used for data that contains a 2-digit year prefixing other information, such as a sequence number. The following User Language fragment

```
PRINT $SIR_DATECNV('YYIIII', -
   'YYYYIIII', '92ABCD')
```

prints the value “1992ABCD”.

Note:

- When a pair of format strings are used for transforming date values, for example for $SIR_DATECNV or processing of updates to SIRFIELD RELATEd fields, both formats must have the same number of I tokens.

  SIRFIELD is part of the *Sir2000 Field Migration Facility*.

YY*

This is a format which could be used for data which contains a 2-digit year prefixing other information, such as a sequence number, when the other information is variable length. The following User Language fragment

```
PRINT $SIR_DATECNV('YY*', -
   'YYYY*', '92')
PRINT $SIR_DATECNV('YY*', -
   'YYYY*', '92XYZ')
```

prints the values "1992" and "1992XYZ".

Notes:

- At most one occurrence of the * token may appear in a datetime format.
- When a pair of format strings are used for transforming date values, for example for $SIR_DATECNV or processing of updates to SIRFIELD RELATEd fields, then if a * token appears in one of the formats, a * must also appear in the other format.

  SIRFIELD is part of the *Sir2000 Field Migration Facility*.

CYYDDD

This is a compact 6-digit date format with explicit century information, from 1900 through and including 2899. The following User Language fragment

```
PRINT $SIR_DATECHK('CYYDDD', '097031')
```

prints the value 1.

ZYYMMDD

This is a compact 6- or 7-digit date format with explicit century information, from 1900 through and including 2899, that can often be used with "old" YYMMDD date values in the 1900s. The following User Language fragment

```
* Check 1 Dec, 1997:
PRINT $SIR_DATECHK('ZYYMMDD', -
   '971201')
* Check 1 Dec, 2000:
PRINT $SIR_DATECHK('ZYYMMDD', -
   '1001201')
```

prints the values 1 and 1.

Note:
0

- With non-strict format matching (such as $SIR_DATE2ND), a three digit number with a leading zero may correspond to a ZYY token, thus allowing "0971201". With strict matching, however, a 3-digit value with leading zero is not allowed for ZYY; a 3-digit value less than 100 with a strict $function (that is, one of the *Sir2000 User Language Tools Functions*) requires the CYY token. If the data contains values less than 100 as three digits in some instances and as two digits in others, you must use a non-strict $function.

YY0000 Decimal digits can be used as separator characters. The following User Language fragment

```
PRINT $SIR_DATECNV('YY0000', -
  'YYYY"N"A', '920000')
```

prints the value "1992NA".

Note:

- Numeric separators, unlike alphabetic separators, do not need to be preceeded by a quote character (").

5.6 $SIR_DATExxx Functions CENTSPAN Argument

Many of the $SIR_DATExxx functions accept an optional argument containing a CENTSPAN value to be used for the call. The default value of any CENTSPAN argument is -50, excepting the $WEB_DATE2xx functions without a format argument, in which case the CENTSPAN argument is ignored and a CENTSPAN of 1990 is used. The default value should be adequate in most cases; if you have carefully determined it should be different in some application, code the value on the relevant $function invocations.

For a different approach, see the description of the CENTSPLT and DEFCENT parameters (for example, **http://m204wiki.rocketsoftware.com/index.php/CENTSPLT_parameter**) and $function arguments.

5.7 Benefits of Sirius datetime processing

Following is a list of benefits offered by Sirius datetime processing. To provide concrete comparisons, there are some references to the standard User Language date $functions.

SPANSIZE

The SPANSIZE processing creates a very strong barrier to detecting otherwise un-noticed 2-digit year processing errors. This is unique to Sirius datetime processing.

Relative CENTSPAN

The relative CENTSPAN specification (for example, "-50") allows you to maintain a flexible "rolling" window for 2-digit year processing.

Default CENTSPAN

One significant advantage of a relative CENTSPAN is that it allows the default (**1990** for $WEB_DATE2xx functions without a format, and **-50** otherwise) of a reasonable value without parameter changes in all batch and online jobs.

Format tokens

There is a very large set of tokens in the Sirius datetime formats. For example, there are 4 different tokens representing the day of the week, and time of day can be represented. Standard User Language date formats do not have any day of week nor time of day tokens, and other standard User Language token variations, for example, CYY vs. ZYY, is done by a complex argument setting.

Pattern match tokens

The Sirius datetime formats can contain single-character ("I") or variable length character ("*") match-any tokens in datetime formats. For example, you can specify that a string has an imbedded year, and process that year as a date.

Format-free representations

Non-string datetime values allow you to pass around dates simply as numbers, without the complexities of carrying the corresponding string format (you only need to establish the scale to operate on a value).

Operating on numeric representations

Numeric date values can be operated on directly with User Language, especially allowing you to add datetime differences (for example, "+"), rather than calling a DATECHG $function and providing a format.

Time

All Sirius datetime $functions allow any reference to a "date" to include time of day. The only standard User Language datetime $function which provides a time of day is $TIME, the current time of day, in one fixed format.

$SIR_DATE formats

$SIR_DATE allows you to specify any format to return the current date and time; $DATE has only a few numeric codes for a few formats.

Error control args

The *Sir2000 User Language Tools* provides error handling control that applies to all datetime $functions &mdash. Sirius and standard User Language. Additionally, all Sirius datetime $functions (except $SIR_DATEFMT, of course) allow you to specify it for a single $function invocation.

Error values of numeric date $functions

The $functions that use non-string datetime values provide very uniform error return values: -9.E12 or a null string for numeric or string result $functions, respectively.

CHAPTER 6 Assembler Language Subroutines for FUNU

If a site has coded its own dollar functions that obtain the system date or time, you will need to modify these functions to work correctly with the APPDATE clocks. The *Sir2000 User Language Tools* provides four entry points to assist you in this conversion effort.

Three of these entry points (SDATE4, SDATE3, and SDATE) are compatible with entry points provided by CCA in *Model 204* V4R1 for FUNU (DATE4, DATE3, and DATE). In addition, a fourth entry point, SDATEF, allows you to obtain the current date and time using any of the Sirius datetime formats (described in "Datetime Formats" on page 41.)

If a datetime argument error occurs during execution of these subroutines, the error handling performed will be controlled by T3 in the case of SDATEF, or by the $SIR_DATE_ERR or APPDATE DATE_ERR settings. The datetime argument errors which can occur are:

- Invalid format, for SDATEF
- Date out of range, for SDATE3 (if you have used APPDATE to set your application clock outside the range 1900-2899!)

The subroutines are summarized below:

```
SDATE4 — Datetime routine for FUNU, with 4-digit year.
SDATE3 — Datetime routine for FUNU, with 3-digit year.
SDATE  — Datetime routine for FUNU, with 2-digit year.
SDATEF — Datetime routine for FUNU, format specified by
         argument.

  All of the SDATE/3/4/F routines obtain the time based
  on the current active APPDATE clock.

    Input    T1 — Address of area for result
             T2 — Address of counted format (SDATEF)
             T3 — Address of error handling string (SDATEF);
                  values may be:
                  CANCEL — Cancel request
                  IGNORE — Return T4=1
                  REPORT — Issue error message, return T4=1
                  OFF — $SIR_DATE_ERR and APPDATE DATE_ERR
                        determine error handling action

    Returns  T4 — Month number (1-12) (SDATE/3/4)
             T4 — Ø: format OK, 1 format bad (SDATEF)
             Output area set to current date & time:
             SDATE4: YYYY.DDD  MON DD  HH:MI:SS  (26 bytes)
             SDATE3:  CYY.DDD  MON DD  HH:MI:SS  (25 bytes)
             SDATE:    YY.DDD  MON DD  HH:MI:SS  (24 bytes)
             SDATEF: Counted string datetime,
                     in format requested.

 T1-T3, T5, R6-R1 preserved.
```

APPENDIX A Messages

Please refer to the M204wiki (see **http://m204wiki.rocketsoftware.com/index.php/Category:Sirius_Mods_messages**) for messages related to the *Sir2000 User Language Tools*.

Index

Figures

Figures