Rocket

Rocket M204 Sir2000 Field Migration Facility

Reference Manual

Version 7.4

September 2013
S2K-0704-RM-01

Notices

Edition

Publication date: September 2013

Book number: S2K-0704-RM-01

Product version: Rocket M204 Sir2000 Field Migration Facility Version 7.4

Copyright

Trademarks

Examples

License agreement

Contact information

Website: www.rocketsoftware.com

Rocket Software, Inc. Headquarters
77 Fourth Avenue
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

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 5.1

The following changes, documented elsewhere in this manual, are new in this version.

- Null exception value
- Date format ALTERNATES
- ERROR [WARN|CANCEL|NOWARN] format
- Strict date matching

  Please see the *Sirius Mods* Release Notes for any compatibility issues. Also, if you need to validate any fields to ensure they follow strict date matching rules, you can use the following $functions contained in the *Sir2000 User Language Tools*:

  $SIR_DATECHG

  $SIR_DATECHK

  $SIR_DATECNV

  $SIR_DATEDIF

- DELETE EACH and FILE RECORDS UNDER of related fields

  Related fields are now encompassed for all updates. Please see the *Sirius Mods* Release Notes for any compatibility issues.

- LAI of INVISIBLE related fields

  Related fields are now encompassed for all UAI/LAI reorgs. Please see the *Sirius Mods* Release Notes for any compatibility issues.

- Alpha EXCEPTIONS values must be quoted

  This may be a compatibility issue.

Sirius Mods Version 5.0

First release of this product.

CHAPTER 1 *Background*

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 will freely use 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 reference 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.

1.1 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 other *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, and three separately-packaged product sets:

- *Sir2000 User Language Tools*
- *Sir2000 Database Analysis Tools*
- *Sir2000 Field Migration Facility*

Several other Sirius products were enhanced to help you address specific aspects of the year 2000 problem. *Sir2000* supports all releases of *Model 204* since and including Version 2.2. Thus, you can begin your millennium preparations before you change your *Model 204* version. You do not need to upgrade your *Model 204* base in the crucial months before starting extensive User Language testing and conversion to ensure year 2000 compliance of your applications.

Also note that you can use the *Sir2000 User Language Tools* to perform that part of your testing which requires an altered “current date” facility, without depending on installation of a *Model 204* release containing the SYSDATE parameter, and also without encountering the complexities in testing with SYSDATE.

1.2 Versions, compatibility, and installation

The *Sir2000 Field Migration Facility* is delivered as object code enhancements to *Model 204* as part the *Sirius Mods*, which also includes products such as *Fast/Backup*, *Fast/Reload*, and the *Fast/Unload User Language Interface* that have nothing to do with the *Sir2000 Field Migration Facility*. No *Sirius Mods* products are required to run *Sir2000 Field Migration Facility* other than itself. To install *Sir2000 Field Migration Facility*, 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* are also (re)installed.

Since the *Sir2000 Field Migration Facility* is part of the *Sirius Mods*, the version number of the *Sir2000 Field Migration Facility* is considered to be the version of the *Sirius Mods* in which it is contained. The *Sir2000 Field Migration Facility* was first available in version 5.0 of the *Sirius Mods* so the first version of *Sir2000 Field Migration Facility* was called version 5.0. This document, the *Sir2000 Field Migration Facility Reference Manual*, assumes that a site is running *Sirius Mods* version 5.0 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 5.1 and later of the *Sirius Mods* will have a sentence such as "This parameter is only available in version 5.1 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 Field Migration Facility*; that is, all versions since version 5.0 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*.

1.3 Related products

Another useful product for helping with year 2000 conversion is the *Sir2000 User Language Tools*. The *Sir2000 User Language Tools* provide excellent facilities for doing application level testing of Y2K related changes.

Since *Sir2000 Field Migration Facility* will probably require *Model 204* file reorgs to be used to full advantage, the *Fast/Unload* and *Fast/Reload* products would prove extremely useful to sites using *Sir2000 Field Migration Facility*. *Fast/Unload* can also 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.

1.4 Related manuals

Since the *Sir2000 Field Migration Facility* requires the installation of the *Sirius Mods*, the person responsible for the installation of *Sir2000 Field Migration Facility* 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.

The *Sir2000 User Language Tools* is documented in the *Sir2000 User Language Tools Reference Manual*

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

There are a number of Sirius $functions that you are authorized to use along with the *Sir2000 Field Migration Facility*, including date processing $functions and *$list processing $functions*. These are documented at **http://m204wiki.rocketsoftware.com/index.php/List_of_$functions**.

1.5 System requirements

Sir2000 Field Migration Facility 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 Field Migration Facility* is an enhancement to *Model 204* that simplifies conversion of an application from using one field format to another. The initial motivation and most likely use for *Sir2000 Field Migration Facility* is migrating a field from using two-digit years to using four-digit years.

Two-digit year fields present a wide variety of problems when the dates in the fields cross a century boundary. Date conversion functions might fail, date comparisons might produce incorrect results and ordering of date values might be incorrect. Many of the problems with two-digit year fields can be solved purely on an application level. That is, the data can be kept in the database with two-digit years and the application code can be corrected to handle years in different centuries. While this approach can make application code a little messier — a simple comparison statement might become several statements — it has many benefits.

- Since there are no changes to the data in the database, there is no issue with synchronizing application changes to database changes.
- Application changes can be limited to problematic uses of two-digit year fields. Benign uses of the two-digit year field can be left untouched.
- Applications can be changed piecemeal instead of all at once. This is a huge benefit in

  Development
  The developer can work in manageable chunks.

  Testing Testing could be spread out over time as changes to the application are made instead of being done all at once when a mass of application changes are completed.

  Cutover The application changes could be phased into production as they are completed and tested rather than all at once. This reduces the likelihood of the disasters that inevitably happen when massive changes are rolled into production. In addition, it makes it easier to back-out changes should they prove to be problematic.

Unfortunately, many references to two-digit years cannot be easily corrected. SORT, FOR EACH RECORD IN ORDER, range FIND, and FOR EACH VALUE IN ORDER statements would require difficult and possibly high overhead changes to the code to make them work. If a field is stored with a two-digit year and any applications use these statements with that field, the only simple way to make the application continue to work correctly in the year 2000 is to convert the field to a four-digit year format.

Such a migration can present a daunting synchronization task. First, the field format must be converted. This will immediately “break” all applications that reference the field. Now all references to the field must be corrected to account for the new format. All the application changes **must** be made and tested before rolling them into production since any applications that aren't converted will not work correctly when the production data format is changed.

Finally, when the new format is rolled into production, a file reorg is likely to be required followed by the roll-in of the application. With the large number of changes likely to be rolled in, odds for application problems are high. If these occur and they prove serious, the need to reorg back to the old format and the fact that the original reorg is likely to have eaten a large chunk of the batch window means that either serious problems will have to be tolerated until they are fixed or there will be a long outage while the changes are backed out.

Sir2000 Field Migration Facility solves these problems by allowing you to make the minimum of User Language and IFAM changes, to make those changes smoothly as part of a conversion effort, and to assure that all needed changes are made.

2.1 SIRFIELD related fields

Sir2000 Field Migration Facility provides a facility that combines all the benefits of an application change only approach to migration with a solution to the more intractable problems in two-digit year handling. How does *Sir2000 Field Migration Facility* do this? By making it possible to store a date field as both a two-digit year and a four-digit year without **any** application changes.

It accomplishes this by “intercepting” all updates to a two-digit field (STORE, ADD, INSERT, CHANGE, DELETE, DELETE EACH, and FILE RECORDS UNDER) and reproducing those updates in a corresponding four-digit field. These fields are called “related” fields and are set up with the SIRFIELD RELATE command. For example, when a date such as “12/11/98” is stored into a two-digit year field, the corresponding date, “12/11/1998”, is *automatically* stored into a four-digit year field. A few SIRFIELD RELATE commands and a file reorganization are all that are required to populate the *Model 204* file with the four-digit year fields that correspond to the two-digit year fields.

What does this accomplish? First, it immediately populates a *Model 204* file with four-digit year versions of all two-digit year fields. More importantly, it ensures that the two-digit year and four-digit year fields are kept in synch, that is, all updates to two-digit year fields will be automatically propagated to the four-digit year field and vice-versa. This means that with no programming effort and a single batch cycle, a file can be fully “Y2K compliant”.

Of course, providing four-digit year date fields in *Model 204* files is a small part of the battle: the bigger challenge is bringing applications into compliance. SIRFIELD makes this process considerably easier.

By keeping both two-digit year and four-digit year copies of a field in a file, programmers can gradually convert references to the two-digit year variant into four-digit year references. In fact, these changes can be limited to places where the four-digit year is required. For example, suppose all references to a particular two-digit year field are "clean" (that is, will work correctly even when the years are in different centuries) except for a single range FIND on the date field. That single range find could be changed to use a *related* four-digit year field and immediately **all** application uses of the field are brought into Y2K compliance.

SIRFIELD related fields are a powerful facility that provide three different options for doing a Y2K migration of applications.

1. "Problematic" references to a two-digit year field can be converted to four-digit year field references. All other references can be left as is, even past the year 2000. This is the lowest cost alternative from a programming perspective but does have a few costs.

    - Since the same date is stored twice, the cost of updates to the date field will generally be double what it would be if the date were only stored singly. Unless the field is heavily updated, this overhead is likely to be barely measurable.

    - Table B and index table usage will also increase because of the duplication of data. Unless a file is dominated by related date fields, this effect is likely to be relatively small.

    - Buffer pool utilization might also increase. Once again, except for certain pathological cases, this effect is likely to be small.

    - Retrieval operations might be more expensive due to the need to scan over two copies of what is essentially the same data. This overhead will only be noticeable in files with large records with large numbers of related fields in each record.

    - Application code might be slightly confusing in that references to what is essentially the same data will use two different field names. This can be mitigated by using similar names for two-digit year fields and their four-digit year variants. For example, calling two related fields "EXPDATE" and "EXPDATE4" is a good idea; calling them "EXPDATE" and "TERMINATION" is a bad idea.

2. "Problematic" references to a two-digit year field can be converted to four-digit year field references as part of bringing an application into Y2K compliance. Work can then continue to gradually convert **all** two-digit year references into four-digit year references. As long as all the problematic references are taken care of, this work can continue past the year 2000 without affecting application functionality.

3. **All** references to a two-digit year field can be converted to four-digit year field references as part of bringing an application into Y2K compliance. Even in this case, SIRFIELD can help significantly.

- First, programmers can gradually bring applications into Y2K compliance in a development region instead of being faced with a situation where all applications are immediately “broken” by a change in the field format.

- Second, it simplifies the process of rolling changes into production. Specifically, the data-structure changes can be separated from the application changes. A file reorganization can be done one weekend to populate a file with four-digit year fields, leaving, of course, the two-digit year fields in place. Application changes could then be rolled-in on a subsequent weekend with no associated file reorganization. If the updated application has severe problems it can easily be backed-out since the two-digit years will still be in place. Finally, when the updated applications have performed to everyone's satisfaction for a reasonable period of time, another file reorganization can be done to remove the two-digit year fields.

The *Sir2000 Field Migration Facility*'s relate facility provides a powerful tool for migrating data and applications into Y2K compliance. It unlinks the process of migrating data-structures and applications and provides a great deal of flexibility in application conversion.

2.2 SIRFIELD aliases

In a year 2000 conversion effort, it might be deemed desirable to leave many references to two-digit year fields intact. There is no reason, for example, to convert an application that simply takes a two-digit year field and displays it on a screen field if there is no ambiguity about what the year means. In many applications, an end-user will readily understand that the date “01/13/01” means “January 13, 2001” and not “January 13, 1901”.

From a conversion management perspective, however, it is important to be certain that someone has actually scanned the code and made sure that the use of a two-digit year is acceptable in a particular case. One way to accomplish this is with the use of SIRFIELD aliases. A SIRFIELD alias is simply another name by which a field might be referenced.

A two-digit year field called EXPDATE, might be given an alias of EXPDATE2. In a conversion effort, whenever a programmer verifies that code that refers to EXPDATE can continue to use a two-digit year that programmer can convert the reference to EXPDATE2, marking the code as having been validated as “two-digit year tolerant”. If a programmer determines that a use of EXPDATE must be converted to use a four-digit year, the programmer can change the reference to a related four-digit year field, say EXPDATE4.

In this way, through the use of an alias and a related field, programmers could gradually remove all references to the original two-digit date field. When all such references have been removed, a manager can be certain that someone has scanned all references to the original two-digit date field and so made an honest effort to bring the code into Y2K compliance.

Management tracking could be taken to a higher level with the use of multiple aliases where each programmer has his or her own alias for a particular two-digit year field and its four-digit year related field. For example, suppose there is a two-digit year field called ARREST_DATE and suppose there are three programmers that will be involved in a Y2K conversion project — Felix Katz, Wanda Fish and Peter Frumble. A related four-digit year field called ARREST_DATE4 could be created. Aliases could be created for ARREST_DATE called ARREST_DATE2FK, ARREST_DATE2WF and ARREST_DATE2PF for use by Felix, Wanda and Peter, respectively. Similarly, aliases can be created for ARREST_DATE4 called ARREST_DATE4FK, ARREST_DATE4WF and ARREST_DATE4PF.

By using these aliases the programmers can not only mark code as having been “visited” and checked for Y2K compliance but also indicate the individual responsible for checking a particular reference to a field.

The use of SIRFIELD aliases can be extended beyond Y2K conversion efforts to any field content conversion effort. For example, if an id field that used to contain 6-digit ids is soon to start holding 7-digit ids, it is imperative that all code that refers to this field be checked for its ability to handle a 7-digit field. SIRFIELD aliases can be used in this case to mark a reference to this field as having been checked.

Once all code has been “visited” *Sir2000 Field Migration Facility* provides the ability to request a warning if any code should be run that references the original field name. These warnings would indicate that some code has been missed in the conversion effort and should be checked. As the certainty grows that all code has been checked, SIRFIELD can be told to cancel any request that refers to the original field name. Eventually, the original field name could be removed from the file's data dictionary and an alias could become the primary field name.

2.3 Name families

With related fields and aliases *Sir2000 Field Migration Facility* provides the capability to associate dozens of different names with what is essentially the same underlying data — a single date field. All these names are called a “family” of names. This family can include either a single fieldname and all its aliases or, in the case of a related field, the names of the related fields and all their aliases.

The concept of a name family is especially important with the PAI and PAI INTO statements. Because of these statements' use in file reorganizations and record copying operations, it would be a major problem if more than one of the names in a family were to be output in a PAI statement. For example, consider the case of two related fields EXPDATE and EXPDATE4. If a PAI output both of these as part of a reorganization process:

```
*
RECTYPE = E
ID = 186475
EXPDATE = 001113
EXPDATE4 = 20001113
```

the reload would result in EXPDATE and EXPDATE4 being loaded twice. This is because when EXPDATE is loaded, EXPDATE4 will automatically be loaded with the corresponding four-digit year since EXPDATE4 is *related* to EXPDATE. Then when EXPDATE4 is loaded, EXPDATE will be loaded with the corresponding two-digit year. As a result, EXPDATE and EXPDATE4 will be loaded twice with the same data. Identical issues arise for aliases.

Because of this and similar problems with PAI INTO, there is a simple rule for SIRFIELD name families and that is: one and only one name from a family of related fields and aliases will ever be output by a PAI or PAI INTO statement. The name to be used can be left to *Sir2000 Field Migration Facility* or it can be controlled explicitly with the SIRFIELD SET PAI command.

If you wish to display all stored values in a record, showing both of each pair of related fields, use the $SIRFIELD_PAF function, which is described in "The $SIRFIELD_PAF function" on page 50.

2.4 Variant date formats

Using any approach to achieving Y2K compliance, many customers will find that their date fields do not conform to strict date formats as they either convert (with a file reorganization) or map (with runtime date conversion $functions or other code) values in date fields.

With the *Sir2000 Field Migration Facility*, you simply specify the date formats of your existing date fields, and RELATE those fields to 4 digit year date fields. A file reorganization then makes both 2 digit and 4 digit occurrences of your date fields available for use in your applications.

Specifically:

- The SIRFIELD FORMAT command establishes a date format for a field which is checked before the field may be updated.
- This uses the corresponding format for a related field to produce the corresponding related value.
- As an additional benefit, this lets you control the quality of your data.

If you have non-date values in fields which contain date information, the *Sir2000 Field Migration Facility* gives you a range of choices for proceeding:

1. Only use strict date formats, and clean up all existing non-strict date values and correct all applications which introduce them.
2. Provide specific exception values which can occur.
3. Provide additional formats for “patterns” of non-date values that can occur.
4. Specify that all existing non-date values should be accepted, but not allow applications to introduce new non-date values.
5. Specify that any non-date value can be introduced, and clean up your applications later, and clean up your data after applications have been cleaned up.
6. Never modify the quality of the data.

This section discusses several situations for dealing with data that cannot be constrained by strict date formats. Most of the examples in this section are motivated by actual usage uncovered by users and some are contrived to illustrate the mechanisms available in the *Sir2000 Field Migration Facility* We offer here no judgement about the applications and data that are assumed to exist before use of the SIRFIELD command.

2.4.1 I/* formats - Fields with embedded year

Frequently, a system is designed with a field containing a year concatenated with some other information. For example, events that are represented in the database could be assigned “control IDs”, which contain the (2 digit) year number followed by a numeric sequence number that is incremented throughout the year.

If you need to relate these fields, for example to sort by control ID so that the year 2000 (00) follows 1999 (99), you need a date format which specifies the year portion of the field and to “ignore and copy” the rest of the value into the related field. The following commands would serve this purpose if the sequence number is 5 digits:

```
SIRFIELD ID    FORMAT DATE YYIIIII
SIRFIELD ID_2K FORMAT DATE YYYYIIIII
SIRFIELD ID    RELATE ID_2K
```

The following commands would work if the sequence number is of different lengths in different records (they would also handle the fixed length case):

```
SIRFIELD ID    FORMAT DATE YY*
SIRFIELD ID_2K FORMAT DATE YYYY*
SIRFIELD ID    RELATE ID_2K
```

2.4.2 Exception values

Frequently, a few special values are chosen to represent “missing” or other “exception” values. This can be handled by the EXCEPTIONS keyword, which can be abbreviated EXC.

For example, you might use the string “999999” in the EXPIRES field to represent “never expires”, and the string “UNK” to indicate a missing expiration. This can be handled by the following commands:

```
SIRFIELD EXPIRES    FORMAT DATE YYMMDD
SIRFIELD EXPIRES_2K FORMAT DATE YYYYMMDD
SIRFIELD EXPIRES    RELATE EXPIRES_2K EXC -
   (999999 999999) (UNK UNK)
```

2.4.3 Fields with alternate formats

Sometimes a system is designed with a field containing different formats on different records. This can be handled with the ALTERNATES keyword, which can be abbreviated ALT; this section gives some examples of its use.

1. If a value in field DTFLD ends in the letter “U”, the field contains a date in the form MMDDYY followed by the letter U. Otherwise, the field contains a date in the form YYMMDD. (This is a fairly contrived example, used later to illustrate the errors which would occur if you tried to “normalize” alternate date formats.) This can be handled by the following commands:

   ```
   SIRFIELD DTFLD    FORMAT DATE YYMMDD
   SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
   SIRFIELD DTFLD    RELATE DTFLD_2K ALT -
      (MMDDYY"U MMDDYYYY"U)
   ```

2. If the value in DTFLD is 4 digits, it starts with a 2 digit year; if it is 6 digits, the second and third digits are a 2 digit year. This can be handled by the following commands:

   ```
   SIRFIELD DTFLD    FORMAT DATE YYII
   SIRFIELD DTFLD_2K FORMAT DATE YYYYII
   SIRFIELD DTFLD    RELATE DTFLD_2K ALT -
      (IYYIII IYYYYIII)
   ```

3. If ID usually contains “year and sequence number” and sometimes does not have the year, but instead has as a prefix the letters “AA”. this can be handled by these commands:

   ```
   SIRFIELD ID    FORMAT DATE YYIIIII
   SIRFIELD ID_2K FORMAT DATE YYYYIIIII
   SIRFIELD ID    RELATE ID_2K ALT -
      ("A"AIIIII "A"A"A"AIIIII)
   ```

Note that you can choose which and how many ***separator characters*** (here, the “A”s) to define in the related field; you might want to use as many as the length of the year, because you might have code that extracts the sequence number from the 4 digit year field, whether or not the first 4 characters are an actual year value.

4. If DTFLD sometimes has a leading blank for the month, you can use the following commands:

```
SIRFIELD DTFLD    FORMAT DATE MMDDYY
SIRFIELD DTFLDY2K FORMAT DATE MMDDYYYY
SIRFIELD DTFLD    RELATE DTFLD2K ALT -
   (BMDDYY BMDDYYYY)
```

Note: In these examples you cannot use RELATE with the ALTERNATES clause to “normalize” all values. For example, in the first example you may not specify “ALT (MMDDYY"U YYYYMMDD)” (see “Date variants update error conditions” on page 16).

2.4.4 Zero day or month numbers

An application may allow part of the date to be entered as zero, indicating an unknown specific date within a year. This also can be handled with the ALTERNATES keyword:

```
SIRFIELD DTFLD    FORMAT DATE YYMMDD
SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
SIRFIELD DTFLD    RELATE DTFLD_2K ALT -
   (YY0000 YYYY0000)
```

Again, you cannot use RELATE to “correct” the date value; for example, you may not specify “ALT (YY0000 YYYY0101)” (see “Date variants update error conditions” on page 16).

2.4.5 Error data

If a you need to relate a field to a field with four digit years, and it contains some values which do not match any specified set of formats, and you cannot clean up the data, or your applications continue to store error values, you must use the ERROR clause. For example:

```
SIRFIELD DTFLD    FORMAT DATE YYMMDD
SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
SIRFIELD DTFLD    RELATE DTFLD_2K ERROR (* "Z*)
```

This will cause any value stored in DTFLD that is not a valid YYMMDD date to also store a value in DTFLD_2K with a leading character “Z” prefixed to the value.

Note that the quotation mark (") before the Z is used to insert a letter Z as a separator character in the date format (see “Datetime Formats” on page 59).

Do not use a quotation mark before the asterisk (*), because it is the wild-card or “catch-all” format: ERROR ("* "Z"*) would mean the asterisk is used as a separator character. (The SIRFIELD command would reject this anyway, because the ERROR format must have at least one “catch-all” format.)

2.4.6 ERROR (* *)

As mentioned, the ERROR clause on the RELATE subcommand must have at least one “catch-all” format. Usually, the applications which are storing error values are storing them into the 2 digit year field. If this is true, you can, and probably should, avoid (* *) as your ERROR clause, and instead concatenate some character to your 4 digit year fields, for example:

```
SIRFIELD DTFLD    FORMAT DATE YYMMDD
SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
SIRFIELD DTFLD    RELATE DTFLD_2K ERROR (* "Z*)
```

You should note the following considerations:

Auditing This approach allows you to write a quick User Language program which uses the ordered index to find (using the LIKE condition) all records with error values.

Reduce edge case rejections
This approach prevents a certain class of updates that are rejected even though your SIRFIELD definitions contain an ERROR clause (see “Date variants update error conditions” on page 16). For example, if instead of the ERROR clause above you had “ERROR (* *).”, then you would not be able to store the value 19980401 in DTFLD.

Continued use of 2 digit year field
This approach may require you to retain some references to the 2 digit year field, in cases where the “error” data is processed for special meaning. Note this can also be true for alternate date formats.

Trailing character
If your applications use the leading character of non-date values for special processing, you may be able to minimize any changes to this code by using a “trailing” marker character in the ERROR format, for example, “ERROR (* *"Z)”. However, this diminishes the benefit of using the marker for auditing purposes, since a FIND statement with the LIKE clause using a trailing character can be quite time-consuming.

(* *) required
If your applications store error data into the 4 digit year field, you must provide the “catch-all” format for both fields. You can avoid this either by leaving all update instances which introduce errors as references to the 2

digit year field; in time you may be able to fix updates that introduce errors and thus remove all refernces to the 2 digit year field.

2.4.7 Controlling error data

With the CANCEL, WARN, or NOWARN keyword, you can, at any time, modify the use of the ERROR format for a field.

For example, you may have historical data for a field which does not match any exception value, the primary date format, or an alternate format; therefore you must use the ERROR clause to continue to maintain these field values when you reorganize your file. If your application only stores values matching the exceptions or primary/alternate formats, you can allow the errors during reorganization but prevent them during application operation with the following scheme in your reorganize job:

```
//RELOAD EXEC PGM=...
Other file load JCL ...
//CCAIN DD *
User zero parameter, etc. ...
OPEN MYFILE
INITIALIZE
Field definitions, if not using LAI ...
SIRFIELD commands, if not using LAI ...
SIRFIELD DTFLD    RELATE DTFLD_2K ERROR NOWARN
FILELOAD (or FLOD) ...
END
SIRFIELD DTFLD    RELATE DTFLD_2K ERROR CANCEL
```

As a result of this, error data which has been on the file will be allowed during the file load, but after the file is reorganized, any application which attempts to store an error value will be prevented from doing so.

As another example, even if your applications continue to introduce errors into the 2 digit year field, you can replace the last SIRFIELD command above with the following:

```
SIRFIELD DTFLD    RELATE DTFLD_2K ERROR -
   (NOWARN CANCEL)
```

This ensures that applications are not introducing new error values via the 4 digit year field (of course, if your ERROR format is “(* "Z*)” or something similar, chances of this should be quite low).

2.4.8 Null exception values

Storing a null, or zero-length, string in a *Model 204* field will either cause the field occurrence to be deleted, or a field occurrence with the null string to be stored, depending on several factors. For this reason, the null string is not allowed to match any format (even the ERROR format).

If your applications store the null string into fields controlled by *Sir2000 Field Migration Facility*, you need to specify the null string as an exception value; for example:

```
SIRFIELD DTFLD    FORMAT DATE YYMMDD
SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
SIRFIELD DTFLD  RELATE DTFLD_2K EXC ('' '')
```

The only exception value that can be paired with the null string is the null string. The null string may not be specified as an exception value for related fields if one has the OCCURS attribute and the other does not.

2.4.9 Date variants update error conditions

When you issue the SIRFIELD FORMAT command for a field, the *Sir2000 Field Migration Facility* guarantees that updates to the field are validated by the criteria you have specified in the exception values, date format (and CENTSPAN and SPANSIZE), alternate formats, and error format.

When you issue the SIRFIELD RELATE command for a pair of fields, the *Sir2000 Field Migration Facility* ensures that each update to one of these fields is reflected to a ***corresponding*** update of the other one. An update to a field is allowed because the value matches one of the items in the exception values, the primary format, and alternate format, or the error format. The corresponding update is defined to be the mapping of that value and the item it matched, using the corresponding item defined on a SIRFIELD RELATE command.

For example, with the following commands:

```
SIRFIELD DTFLD    FORMAT DATE YYMMDD
SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
SIRFIELD DTFLD    RELATE DTFLD_2K EXC -
   (XXX XXXXX)
```

If the following updates are performed:

```
ADD DTFLD = 980101
ADD DTFLD = XXX
```

- The first would cause 19980101 to be added to DTFLD_2K (since 980101 matches YYMMDD, and this corresponds to YYYYMMDD)
- The second would cause XXXXX to be added to DTFLD_2K (since XXX matches XXX, and this corresponds to XXXXX).

When you reorganize your files, each addition of a field occurrence is done by taking a value from one of the related fields (the one with the PAI attribute) and performing a normal *Sir2000 Field Migration Facility* update, which will update the other related field as well. Therefore, if the correspondence between related items changes after a value is stored in the file, a reorganization of the file would cause the related field occurrence to be different than when the original update took place.

The *Sir2000 Field Migration Facility* has a number of restrictions to ensure this will not happen. Some of them concern which SIRFIELD commands may be entered, such as are described in "EXC/ALT/ERROR after data added to file" on page 18 and other sections. Other restrictions cause the *Sir2000 Field Migration Facility* to reject an update to a field, even though that update matches, for example, the primary date format. Those other restrictions are described in this section.

For the following examples, assume these rather contrived commands:

```
    SIRFIELD DTFLD    FORMAT DATE YYMMDD
    SIRFIELD DTFLD_2K FORMAT DATE YYYYMMDD
    SIRFIELD DTFLD    RELATE DTFLD_2K EXC -
                      (991231 99999999)
    SIRFIELD DTFLD    RELATE DTFLD_2K ALT -
                      ("A* "A*) -
                      ("C"D* "A"B*) -
                      (YY0000 YYYY0101)
    SIRFIELD DTFLD    RELATE DTFLD_2K ERROR (* *)
```

Note that the second alternates pair, (`"C"D* "A"B*`), will produce failures that could be avoided if it were switched with the first alternates pair.

Note that the last alternates pair, (`YY0000 YYYY0101`), is a misguided attempt to "normalize" dates with zeroes into legal dates. This is an ALTERNATES clause which is allowed in the SIRFIELD command but which will never be matched and produce the intended "normalization".

The possible update types and "mismatched items" errors are shown in the following list:

Exception value There are no restrictions against adding an exception value; the related field is updated with the corresponding exception value.

Primary format If a value is not an exception value, and it matches the primary format, the corresponding value must not be an exception value.

For example:

```
    ADD DTFLD_2K = 19991231
```

This will fail because the corresponding value is 991231, which is an exception value.

Alternate format If a value is not an exception value and doesn't match the primary format, and it matches an alternate format, the corresponding value must not be an exception value and must not match the primary corresponding format nor any alternate corresponding format which occurs earlier in the alternates list.

For example:

```
ADD DTFLD = 990000
```

This will fail because the corresponding value is 19990101, which matches the primary corresponding format.

Another example is:

```
ADD DTFLD = CDAB
```

This will fail because the corresponding value is ABAB, which matches "A*, which is an earlier alternate corresponding format.

Error format

If a value is not an exception value and doesn't match the primary format nor any alternate format, and it matches the error format, the corresponding value must not be an exception value and must not match the primary corresponding format nor any alternate format.

For example:

```
ADD DTFLD = 19990101
```

This will fail because the corresponding value is 19990101, which matches the primary corresponding format.

2.4.10 EXC/ALT/ERROR after data added to file

In general, you can add exeptions, alternates, and an error format to a field after data has already been stored in the file. This allows you to compensate for applications which might introduce unforseen data. However, in order to ensure the integrity enforced by the rules of the previous section, some kinds of SIRFIELD commands are not allowed **after data has been added** to the file. These are:

- You cannot specify an EXCEPTIONS value which matches either the primary or one of the alternate formats for the field.
- You cannot specify any EXCEPTIONS value if the error format exists for the field.
- You cannot specify any ALTERNATES format if the error format exists for the field.
- You may not change the SPANSIZE (or CENTSPAN) if any alternate formats or the error format exists for the field.

2.4.11 Specify EXC/ALT/ERROR on RELATE

As mentioned in the syntax of the RELATE and FORMAT subcommands, EXCEPTIONS, ALTERNATES, and ERROR can be specified on both of these commands. However, if a field is being related, these clauses are specified on the RELATE subcommand, not on the FORMAT subcommand. This is because each of these clauses defines a pair that is used to correspond between the two related fields.

2.4.12 Other ALT syntax restrictions

In addition to the restrictions already mentioned, there are other restrictions for ALTERNATES.

Two-digit years You may not enter a two-digit year date format for a field in the ALTERNATES clause if the primary format for the field is not also a two-digit year date format.

Respecifying When you enter multiple SIRFIELD commands with the ALTERNATES clause for a field or a related pair of fields, each command must have one of the following forms:

1. It can be an initial sublist of the alternates previously defined (of course, this doesn't add new alternates).

2. It can be the complete list of the alternates previously defined, followed by additional new alternates.

3. It can be entirely new alternates.

In the second and third cases, each new alternate for each field must not already be an alternate for that field, and it must not be the same as the primary format for the field.

2.5 IFAM and batch

Sir2000 Field Migration Facility is integrated into *Model 204* so that it is **impossible** for an application program to bypass its controls. By relating fields, and setting either warning or cancel controls on a name, a system administrator can be certain that these controls will be in effect for all programs. This is even true for IFAM and BATCH2 or BATCH204 programs, including file load, that might be difficult to track any other way.

IFAM programs are usually kept separate from User Language programs because they are written in Cobol, Fortran, Assembler or some other language. It is even possible that the source code for some IFAM applications might be lost. Regardless of the status of IFAM program source, *Sir2000 Field Migration Facility* can be used to ensure that any two-digit year update by an IFAM program will be automatically reflected in the *related* four-digit year field. Similarly, if there is any doubt as to whether any IFAM programs refer to a two-digit date field, SIRFIELD warnings can be used as an alternative to searching for IFAM programs that might do so.

BATCH2 and BATCH204 jobs can also present a management problem in that these might contain in-line User Language or FLOD code that is outside of the standard User Language management control. *Sir2000 Field Migration Facility*'s tight integration with *Model 204* ensures that BATCH2 and BATCH204 jobs cannot bypass the *Sir2000 Field Migration Facility* controls placed on a file.

BATCH204 jobs are among the most likely jobstreams to reference *Model 204* load modules which may be different from the latest installed and maintained versions in use at a shop. Therefore they will benefit from the protection against access using a load module without *Sir2000 Field Migration Facility*, which produces the M204.1521 error message. This is described in "The SIRFIELD Command" on page 25.

CHAPTER 3 *Operational Considerations*

This chapter documents various operation considerations associated with using the *Sir2000 Field Migration Facility*. Perhaps the most obvious consideration is that once a **SIRFIELD** command has been used to modify a file, for example to register formatting information for a field or to create a relation between two fields, only load modules with support for the *Sir2000 Field Migration Facility* can be used to access the file. Attempts to access such a file using a load module without *Sir2000 Field Migration Facility* produce an M204.1521 error message.

3.1 DISPLAY FIELD command

The DISPLAY FIELD command has been extended to provide better integration with the *Sir2000 Field Migration Facility*:

- If you enter `D FIELD` *`name`*, and *name* is an alias, you will receive the error message **M204.1265 NO SUCH FIELD NAME**, followed by the output for a SIRFIELD *name* DISPLAY command.

- If you enter `D FIELD [(options)] ALL` for a file that contains any SIRFIELD information, then the normal output for the D FIELD command is followed by the output from a SIRFIELD ALL DISPLAY command.

- If you enter `D FIELD (NAMES` *`other_options`*`)` *`name1`* `...`, then aliases will not be looked up (assuming *other_options* does not override NAMES; for example, `NAMES DDL` is processed as if you never specified `NAMES`).

3.2 Non-TBO processing

If the update of a field succeeds, and the update of its related field fails, under normal conditions with the *Model 204* Transaction Backout (TBO) feature active, the transaction is backed out, removing the update to the first field and leaving the file consistent (neither of the two related fields are updated). Such an update failure can happen, for example, if the related field is a fixed OCCURS field, and an attempt is made to store too many occurrences.

However, if you are processing with TBO turned off (FRCVOPT parameter has 8 bit on) and the first update succeeds and the related update fails, the first update cannot be backed out, therefore the file is physically inconsistent. As of version 5.1, in this circumstance, the *Sir2000 Field Migration Facility* will mark the file as physically broken (FISTAT parameter 2 bit is turned on) and the user is restarted.

3.3 Error conditions for IFAM updating

SIRFIELD processing introduces various new updating error conditions (for example, a value supplied for an update may not match a date format specified on a SIRFIELD command). In order to ensure that your IFAM application does not proceed without corrective action, these errors will cause a soft restart and an immediate termination of the IFAM connection.

3.4 PAI statement or IFAM GET DATA specification

As discussed in "Selecting the PAI field with RELATE" on page 35, PAI or an IFGET call with a "DATA;" specification will only return one value from a pair of related fields. This is to facilitate the use of PAI for file reorganizations. As described in "The $SIRFIELD_PAF function" on page 50, you may use the $SIRFIELD_PAF $function to retrieve or display all of the physical fields contained in a record.

3.5 Restrictions to IFAM field specifications

The "searched" option of the G format with an EDIT specification is currently not allowed with the *Sir2000 Field Migration Facility*.

3.6 Server Table settings (UTABLE)

An application that contains updating statements to fields that have a format established by the SIRFIELD command may need to increase some of the server table sizes. The following sections list the maximum additional server table sizes used.

3.6.1 LPDLST

Your application may use additional LPDLST size, up to 300 (decimal) bytes.

3.6.2 STBL

Your application may use additional LSTBL size, up to 1 plus the length of the longest string you provide for update, or for the value converted according to the related field format. Actual usage will be reflected in the STBL statistic.

3.6.3 VTBL (User Language)

Your User Language application may use additional LVTBL size, but only up to 16 bytes. VTBL usage is specified in units of 32 bytes. Actual usage will be reflected in the VTBL statistic (also shown in units of 32 bytes).

CHAPTER 4 The SIRFIELD Command

The SIRFIELD command is used to assign special attributes to a field name; the purpose of these attributes is to provide additional checking of uses of the field, and to ease a migration period for applications using the field. The general form of the SIRFIELD command is:

```
[IN file] SIRFIELD name subcommand operands
```

SIRFIELD command syntax

where

IN file	specifies the file to which the SIRFIELD command applies. This is only necessary if *file* is not the default file. *IN file* can be specified with any SIRFIELD subcommand but, for readability, is not presented in subsequent syntax diagrams.
name	is alias name or fieldname in the current file (it must already be defined). If it contains blanks or special characters, the entire name must be enclosed in apostrophes, for example, SIRFIELD 'DATE OF BIRTH' ... Note that certain SIRFIELD subcommands accept either a fieldname or an alias, while other subcommands accept only a fieldname or only an alias.
subcommand	indicates the operation being performed, it is one of the words *ALIAS*, *DELETE*, *FORMAT*, *DISPLAY*, *RELATE*, or *SET*. The meanings of these are described below.
operands	The operands specific to the operation. These are described below.

The SIRFIELD command allows you to control access to and update of fields in a file. This control is provided in any *Model 204* load module (BATCH204, ONLINE, IFAM4, IFAM1, etc.) with the *Sir2000 Field Migration Facility*. In order to prevent a load module without this product from updating or accessing controlled fields and circumventing this control, the SIRFIELD command adds a special entry to the dataset list for all files with SIRFIELD controlled fields. This special dataset list entry has the name `*SIRIUS*`.

All load modules with the *Sir2000 Field Migration Facility* understand that this special dataset list entry does not actually correspond to a real dataset but is simply a marker indicating that the file has at least one SIRFIELD controlled field. If a load module without *Sir2000 Field Migration Facility* attempts to open a logical file with *Sir2000 Field Migration Facility* controlled fields, it will attempt to open a physical file with a ddname of `*SIRIUS*`. This will result in a message:

```
    M2Ø4.1521: *SIRIUS* DOES NOT EXIST OR REQUESTED -
               ACCESS NOT AUTHORIZED
```

since *SIRIUS* is not even a valid DDNAME, and the attempt to open the file will fail.

This mechanism ensures that once one or more fields in a file have *Sir2000 Field Migration Facility* access controls placed on them, it will be impossible to bypass those controls, even with a load module that doesn't have *Sir2000 Field Migration Facility*.

The SIRFIELD command may not be issued against a file that has `CCA` as the first three characters of the file name.

Note that the FORMAT and RELATE subcommands are done as part of the initial file definition process anyway.

4.1 SIRFIELD ALIAS command

The purpose of this subcommand is to provide an additional name that programs can use to refer to a field. Generally this is useful if you want to keep track of which applications have been modified to observe some new field semantics, and you can enforce such modification by use of the SET subcommand. The REFERENCE WARN or the REFERENCE CANCEL parameters of the SIRFIELD SET command allow warnings or request cancellations to be issued on references to a "controlled" name.

```
SIRFIELD fieldname ALIAS aliasname
```

SIRFIELD ALIAS command syntax

where

fieldname	is the name of a field in the current file.
aliasname	is a name that applications can use to refer to the field. It must be "new", that is, it must not already be an alias for another field in the file and it must be different from any field name in the file. If it contains blanks or special characters, the entire name must be enclosed in apostrophes.

The SIRFILE ALIAS command adds *aliasname* to a family of names that includes *fieldname*, any other aliases of *fieldname*, the field that is *RELATED* to *fieldname* (if one exists), and all aliases of the *RELATED* field. Since all the names in this effectively refer to the same underlying data, one and only one name from this family will ever be output by a PAI or PAI INTO along with its value. The default setting for an alias is to not be output in a PAI or PAI INTO statement for the file. This can be overridden with a SIRFIELD SET command as in

```
    SIRFIELD aliasname SET PAI
```

The SIRFIELD ALIAS command can be issued whether or not the file against which it is being issued is empty.

Multiple alises are allowed for a field.

Aliases can be deleted with the SIRFIELD DELETE command.

If you issue an ALIAS subcommand which defines the same alias name for the same field name, it is accepted without any error or warning message.

Ad hoc update privileges on the file are required for this subcommand.

After a successful ALIAS subcommand, the *Model 204* REDEFINE command will reject an attempt to change the field level security level of the underlying field.

4.2 SIRFIELD DELETE command

The purpose of this subcommand is to remove an alias name. Future references to the name will be handled as unknown field names.

```
SIRFIELD aliasname DELETE
```

SIRFIELD DELETE command syntax

where

aliasname	is the name of an alias as defined by the SIRFIELD ALIAS subcommand.

It is not valid to delete an alias that has the PAI attribute set. To delete an alias with the PAI attribute, set the PAI attribute for another field in the alias's family of names.

The SIRFIELD DELETE command can be issued whether or not the file against which it is being issued is empty.

Ad hoc update privileges on the file are required for this subcommand.

4.3 SIRFIELD DISPLAY command

The purpose of this subcommand is to show the attributes which SIRFIELD has associated with a field. The output from this command is a set of SIRFIELD commands that can be used to re-create the SIRFIELD attributes for the file. The output from this command can be captured and used to define SIRFIELD attributes during a file reorganization.

```
SIRFIELD [ALL | name] DISPLAY
```

SIRFIELD DISPLAY command syntax

where

ALL	Specifies that all names with SIRFIELD attributes be displayed, with their attributes.
name	The name of a field or alias in the current file.

If you use the DISPLAY subcommand to create a file reorganization input stream, be sure to use SIRFIELD ALL DISPLAY, rather than doing it a field at a time.

A SIRFIELD DISPLAY for a fieldname or an alias will display SIRFIELD attributes that apply to **all** names in that fieldname's or alias's family.

No privileges on the file are required for this subcommand.

4.4 SIRFIELD FORMAT command

The purpose of this subcommand is to ensure that only values of the specified format(s) are stored in the field, and also to associate the format, CENTSPAN, and SPANSIZE with the field for use in applications or in the SIRFIELD RELATE subcommand. When an update occurs to the field, it is checked against its format and the field update is issued. If the format check fails, the request is cancelled and the field is not updated.

In most cases, the SIRFIELD FORMAT command is used to define a field so that it can be used in the SIRFIELD RELATE command. This section will describe SIRFIELD FORMAT for a field which is used in RELATE. See "SIRFIELD FORMAT for unrelated field" on page 30 for a description of the SIRFIELD FORMAT command that can be used to restrict updating of a field but without relating the field.

```
SIRFIELD fieldname FORMAT -
         [DATE format [CENTSPAN cspan]] -
         [SPANSIZE ssize]
```

SIRFIELD FORMAT command syntax

where

fieldname	is the name of a field in the current file.
format	must be a valid datetime format (see "Datetime Formats" on page 59) and must include a year specification. The phrase starting with DATE is optional if you are using FORMAT to change CENTSPAN or SPANSIZE.

The format specified after the DATE keyword is called the ***primary*** format.

If you specify the FORMAT subcommand for a field which already has a primary format specified, you can specify the same format and the command will be accepted without any warning or error message. If you specify a different primary format, the command is rejected with an error message.

cspan is the CENTSPAN specification for the field. *Cspan* can be either a 4-digit absolute year or a relative year number as indicated by a leading plus or minus sign. This parameter is meaningful only if *format* describes a date value with 2-digit year values, and is not allowed otherwise. See "CENTSPAN" on page 64 for a discussion of CENTSPAN. The default CENTSPAN value for a 2-digit year format is -50.

When you specify a relative CENTSPAN value, it is relative to the date when the first FORMAT subcommand is specified for that field. The effective absolute CENTSPAN value is stored with the field and is used for future DISPLAY sub-commands.

If you specify the FORMAT subcommand for a field which already has a CENTSPAN value specified (or defaulted), you can omit CENTSPAN, specify the same relative CENTSPAN as originally specified, if any, or specify the absolute CENTSPAN value currently associated with the field (and displayed by the DISPLAY subcommand) and the command will be accepted without any warning or error message. If you specify a larger (effective) CENTSPAN, the FORMAT sub-command will be rejected with an error message. If you specify a smaller (effective) CENTSPAN, you must also increase SPANSIZE by the difference between the old and new CENTSPANs, or the FORMAT sub-command will be rejected with an error message.

You may not change the SPANSIZE (or CENTSPAN) if alternate formats or the error format have been defined for the field (either on the FORMAT or RELATE subcommands) if records have been added to the file.

ssize is the SPANSIZE specification for the field. SPANSIZE sets the number of years in the valid date range for the field. This parameter is meaningful only if *format* describes a date value with 2-digit year values, and is not allowed otherwise; hence *ssize* must be in the range of 1 to 100. The default SPANSIZE value for a 2-digit year format is 90. See "SPANSIZE" on page 65 for a discussion of SPANSIZE.

SPANSIZE may be increased at any time, up until its maximum of 100, but may not be decreased.

For example

```
    SIRFIELD HIREDATE FORMAT YYYYMMDD
```

specifies that field HIREDATE has format “YYYYMMDD”.

The FORMAT subcommand must be specified if you are going to subsequently RELATE the field.

The only way to change or remove the date format for a field is to re-initialize the file.

You must specify some form of year in the format (YY, ZYY, CYY, or YYYY). The YY format and its corresponding absolute CENTSPAN and current SPANSIZE define a valid range of dates (the “SPANSIZE” years starting with the absolute CENTSPAN). CYY or ZYY format defines a valid range of dates from 1900 through 2899, inclusive. An attempt to set a field to a value outside of the range of dates implied by these formats will result in an error message and request cancellation, and the field is not updated.

If the field is FLOAT, then any attempt to store a value which has a leading plus sign, zero, or blank results in an error message and request cancellation, and the field is not updated.

A date format may not be specified for a field which is PURE DBCS or MIXED DBCS.

A date format may not be specified for the record security field.

Ad hoc update privileges on the file are required for this subcommand.

After a successful FORMAT subcommand, the *Model 204* REDEFINE command will reject an attempt to change any of the following characteristics of the underlying field:

```
    DEFERRABLE         FOR-EACH-VALUE      KEY
    FEW VALUED         CODED               NUMERIC RANGE
    INVISIBLE          LEVEL               UPDATE-IN-PLACE
    FLOAT              ORDERED NUMERIC     ORDERED CHARACTER
    UNIQUE             AT-MOST-ONE
```

4.4.1 SIRFIELD FORMAT for unrelated field

As mentioned in the previous section, the common use of the SIRFIELD FORMAT command is to define the datetime format so that the field can be used in relate processing. However, you can also use FORMAT to restrict updating of a field that is not related to another field. In this case, and this case only, you can specify exception values, alternate formats, and the ERROR format for the field on the SIRFIELD FORMAT command.

```
SIRFIELD fieldname FORMAT -
         [DATE format [CENTSPAN cspan]] -
         [SPANSIZE ssize] -
         [EXCEPTIONS values] -
         [ALTERNATES altformats] -
         [ERROR [NOWARN | CANCEL | WARN] [*]]
```

SIRFIELD FORMAT command syntax (unrelated field)

where

fieldname	is the name of a field in the current file.
format	See description in preceding section. The phrase starting with DATE is optional if you are using FORMAT to change CENTSPAN or SPANSIZE or to add an exception or an alternate format or to use the ERROR clause.
cspan	See description in preceding section.
ssize	See description in preceding section.
values	is a list of values to accept for the field, even if they do not match the primary or an alternate format. If any value consists only of uppercase letters, it must be enclosed in apostrophes ('). **EXCEPTIONS** may also be written **EXC**.
altformats	is a list of alternate formats to accept for the field. **ALTERNATES** may also be written **ALT**.
ERROR ... *	is a specification that any value should be accepted for updating this field. NOWARN, which is the default if ERROR is specified, indicates that updates which don't match an exception value, the primary format, any alternate format should be accepted without any warning message. CANCEL specifies that such updates should not be allowed (thereby “blocking” the ERROR format). WARN specifies that the update be accepted, but that a warning message be issued.

For example

```
SIRFIELD HIREDATE FORMAT YYYYMMDD -
                  EXC 'NONE' 99999999 -
                  ALT YYYY0000 -
                  ERROR WARN *
```

specifies that field HIREDATE has format “YYYYMMDD” but might also contain values “NONE” or “99999999” or a 4 digit year followed by 4 zeroes; if an update is provided which matches none of these, it should be processed and a warning messsage issued.

You may use the FORMAT subcommand to add new exceptions values. If you specify a value which has already been specified, the command is accepted and no warning or error message is issued.

You may use the FORMAT subcommand to add new alternate formats to accept for the field. Such a list of formats must have one of the following forms:

1. It can be an initial sublist of the alternates previously defined (of course, this doesn't add new alternates).

2. It can be the complete list of the alternates previously defined, followed by additional new alternates.

3. It can be entirely new alternates.

In the second and third cases, each new alternate for each field must not already be an alternate for that field, and it must not be the same as the primary format for the field.

A null string ('') may be specified as one of the EXCEPTIONS values.

It would be unusual, but you may specify an exception value which matches either the primary or any alternate format. However, any exception value which matches a format must be added before any records have been stored in the file.

If records have been stored in the file, you may not specify new EXCEPTIONS nor format ALTERNATES if ERROR has been specified for the field.

The only way to remove an exception value, alternate format, or the ERROR forat, is to re-initialize the file.

All of the rules specified in the preceding section for the SIRFIELD FORMAT command apply to this form as well.

See “Variant date formats” on page 10 for additional information about formats, exceptions, and handling non-date values.

You may not specify EXCEPTIONS, ALTERNATES, nor ERROR on the FORMAT subcommand if the field is going to be referenced on a RELATE subcommand — RELATE has its own provision for pairs of these items.

4.5 SIRFIELD RELATE command

The purpose of this subcommand is to cause any update of a field to be reflected by the equivalent update to another field. These fields are said to be **RELATED**. You may only RELATE two fields which have FORMAT attributes. When an update occurs to one of the fields, it is checked against its format and the field update is issued; then the value is converted to the format of the other field and if that conversion is successful the update to the other field is issued. If either format check fails, the request is cancelled and neither field is updated.

```
SIRFIELD fieldname1 RELATE fieldname2 -
                    [EXCEPTIONS values] -
         [ALTERNATES altformats] -
         [ERROR -
            warn_can *
          | [warn_can] [(err1 err2)]
          | (warn_can err1 warn_can err2)
          | (warn_can warn_can) ]
```

SIRFIELD RELATE command syntax

where

fieldname1 is the name of a field in the current file.

fieldname2 is the name of a field in the current file. Must be different from *fieldname1*.

values is a list of special values for the related fields. Each item in *values* is either a single value or a pair of values enclosed in parentheses; the single value case is identical to having the same value repeated inside parentheses, except that if a single value contains only uppercase alphabetic characters, it must be contained in apostrophes. The first and second value corresponds to values for *fieldname1* and *fieldname2* respectively.

An exception value for one field may not occur in two pairings with different values for the other field. The list of values can contain any values, including those which match the primary format or any of the alternate formats.

When an attempt is made to store a value in the exception list for a field, no attempt is made to convert the value based on the primary or an alternate format. Instead, the value is stored and the corresponding value for the related field is also stored.

A null string ('') may be specified as both of the EXCEPTIONS values, if both fields are preallocated (have the OCCURS attribute) or are both nonpreallocated; otherwise the null string is not allowed. The null string may not be paired with a non-null string. Null strings are allowed.

EXCEPTIONS may also be written **EXC**.

altformats is a list of alternate formats to accept for the fields. Each item in *values* is either a single format or a pair of formats enclosed in parentheses; the single format case is identical to having the same format repeated inside parentheses. The first and second format corresponds to formats for *fieldname1* and *fieldname2* respectively.

ALTERNATES may also be written **ALT**.

ERROR ... is a specification that any value should be accepted for updating at least one of these fields. It specifies a format for each field (***err1*** and ***err2***) and/or an action (***warn_can***) to be taken if the error format is matched.

At least one of the formats must be a single asterisk ("*", the "catch-all" format), which matches any value. Both formats can be "*" (which is also specified by a single "*" without parentheses), but this is not recommended; see "ERROR (* *)" on page 14.

The use of ***err1*** and ***err2*** can each be controlled by ***warn_can***, which is:

```
    NOWARN | CANCEL | WARN
```

NOWARN, which is the default when ERROR is first specified for a field, indicates that updates which don't match an exception value, the primary format or any alternate format, but which match the specified error format, should be accepted without any warning message. CANCEL specifies that such updates should not be allowed (thereby "blocking" the ERROR format). WARN specifies that the update be accepted, but that a warning message be issued.

If ***warn_can*** is specified outside parentheses, it applies to both error formats; otherwise the first it applies to the first format and the second to the second format.

The only way to change the field to which a given field is related or to remove a relationship is to re-initialize the file. The only way to remove an exception value, alternate format, or the ERROR forat, is to re-initialize the file.

You may issue the RELATE subcommand multiple times for a given pair of fields, in order to add to the exception list, the alternates list, to add the error formats, or to change the ***warn_can*** attribute of the error formats. RELATE subcommands after the first for a pair of fields will **not** affect the PAI attributes of any names in the family. The order of field names in each RELATE subcommand for a given pair of fields must be the same as the command which originally related them.

See “Variant date formats” on page 10 for additional information and examples about exceptions, alternate formats, and handling non-date values.

Neither field may be the HASH key field, nor may it be the SORT key field, since it would then have various implicit AT-MOST-ONE and “not-null” attributes. The INVISIBLE, UPDATE-IN-END, and LEVEL attributes of both fields must be identical.

Both fields must have had FORMAT attributes specified with a SIRFIELD FORMAT command before the SIRFIELD RELATE command is issued for the field. Exception values, alternate formats, and the ERROR format must **not** have been specified on the FORMAT subcommands for either of the fields. The precision of both date formats must be the same (for example, if one specifies time in HH:MI:SS, the other must also).

A field can be related to at most one other field.

Ad hoc update privileges on the file are required for this subcommand.

4.5.1 Selecting the PAI field with RELATE

The SIRFIELD RELATE command joins the family of names (fieldname and its aliases) associated with *fieldname1* with the family of names associated with *fieldname2* into a single family. Before the SIRFIELD RELATE command, one name from each family would be output in a PAI statement. Since the RELATE combines the two families into a single family and since only one name in the family can be output in a PAI, the RELATE subcommand ends up removing the PAI attribute from one of the names in one of the RELATE'ed families.

For example, in

```
SIRFIELD STARTDATE  FORMAT 'YYMMDD'
SIRFIELD STARTDATE4 FORMAT 'YYYYMMDD'
SIRFIELD STARTDATE  ALIAS  STARTDATE2
SIRFIELD STARTDATE2 SET PAI
SIRFIELD STARTDATE4 RELATE STARTDATE
```

the SIRFIELD RELATE command will result in STARTDATE2 **not** being output in a PAI because STARTDATE4 has “more information”, that is, a four-digit year instead of a two-digit year. STARTDATE4 is the name used to PAI this family.

Note that if the SET PAI subcommand were to be moved after the RELATE, as in

```
SIRFIELD STARTDATE  FORMAT 'YYMMDD'
SIRFIELD STARTDATE4 FORMAT 'YYYYMMDD'
SIRFIELD STARTDATE  ALIAS  STARTDATE2
SIRFIELD STARTDATE4 RELATE STARTDATE
SIRFIELD STARTDATE2 SET PAI
```

the name STARTDATE2 would be output in a PAI and not STARTDATE4.

The rules for which name in the family of names created by a SIRFIELD RELATE command is to be output in a PAI are :

- The PAI name will **always** have been a PAI name before the RELATE subcommand. That is, if an alias for a field is set to PAI, the relate command will either cause that alias to be the PAI field for the combined family if is is determined that the fieldname for which it is an alias is the PAI field.
- The field with the most year digits will be the PAI field. If that field has an alias with the PAI attribute set, that alias will be the PAI name.
- If the RELATE'd fields have an equal number of year digits, the first field listed on the SIRFIELD RELATE command will be the PAI field. If the PAI field has an alias with the PAI attribute set, that alias will be the PAI name.

After the RELATE subcommand, an informational message will be issued indicating the PAI name from the two family of names being joined by the RELATE that will no longer be output in a PAI.

If you wish to display all stored values in a record, showing both of each pair of related fields, use the $SIRFIELD_PAF function, which is described in "The $SIRFIELD_PAF function" on page 50.

Note that the set of names and values produced by the PAI statement is the same as the set of names and values returned by the IFGET function with the "DATA;" (no fieldnames) specification in IFAM.

4.6 SIRFIELD SET command

The purpose of this subcommand is to specify special processing options for a field or alias name.

```
SIRFIELD name SET [PAI] -
        [REFERENCE {NOWARN | WARN | CANCEL}]
```

SIRFIELD SET command syntax

where

name	is the name of a field or alias in the current file.
REFERENCE ...	specifies the action taken when the field or alias is referenced in your User Language/IFAM programs. *NOWARN* indicates that references should be allowed, *WARN* indicates that they should be allowed but a warning should be sent to the journal every time a reference to the field is compiled or occurs as the result of a

fieldname variable interpretation and *CANCEL* indicates that a reference to the field should result in a compilation error or request cancellation in the fieldname variable case.

The default value for *REFERENCE* is *NOWARN*. “REFERENCE” can also be written “REF”.

PAI indicates that *name* is the name to be output for the name family in PAI's and PAI INTO's. A name family consists of a field, any aliases for that field, the field, if any, *RELATED* to that field, and all aliases of the *RELATED* field. Since all the names in this effectively refer to the same underlying data, one and only one name from this family will ever be output by a PAI or PAI INTO along with its value. A SET PAI subcommand for a field or alias not only results in the indicated name from being output on a PAI but also prevents the name that would have been previously PAI'ed from being output.

For example

```
SIRFIELD OLDDATE FORMAT 'MM/DD/YYYY'
SIRFIELD NEWDATE FORMAT 'YYYYMMDD'
SIRFIELD OLDDATE RELATE NEWDATE
SIRFIELD NEWDATE SET PAI
SIRFIELD OLDDATE SET REF WARN
```

indicates that NEWDATE should appear in PAI's and PAI INTO's and that OLDDATE should not appear in these. It also indicates that any references to OLDDATE should result in warnings going to the *Model 204* journal.

A SET REF WARN for a name that is the PAI name for its family will generate a warning since a PAI is an implicit reference to the name. A SET REF CANCEL for a name that is the PAI name for its family will not be allowed.

You may issue the SET subcommand multiple times for a given name. The SET subcommand can be issued at any time — even after data has been added to a file.

Ad hoc update privileges on the file are required for this subcommand.

After a successful SET subcommand, the *Model 204* REDEFINE command will reject an attempt to change the field level security level of the underlying field.

CHAPTER 5
File Reorganizations

Sir2000 Field Migration Facility defines rules for the contents of and relationships among fields (and aliases) in a *Model 204* file. Since a *Model 204* file reorganization involves the wholesale movement and possibly manipulation of all the fields in a file it is important to understand how *Sir2000 Field Migration Facility* rules affect the reorg process. For simplicity *Sir2000 Field Migration Facility* rules will be referred to here as SIRFIELD rules since it is the SIRFIELD command that specifies the field content and relationship rules for a file.

File reorganizations can be generally classified according to three basic types :

UAI/LAI	Reorgs done using the UAI command of *Fast/Unload* and the LAI command of *Fast/Reload*.
PAI	Reorgs done using the User Language or *Fast/Unload* PAI statement and a generic FLOD/FILELOAD program.
Structured	Reorgs done using the User Language PRINT or WRITE statements or *Fast/Unload* PUT statements to produce a structured flat file that is loaded with a FLOD/FILELOAD program specific to the file.

Each of these reorg types has specific issues in relation to *Sir2000 Field Migration Facility* so any issues specific to a particular reorg type are described where applicable.

One current limitation of *Sir2000 Field Migration Facility* is that unless the FLOD or FILELOAD is done using LAI, *Fast/Reload* will hand the FLOD program off the standard FLOD or FILELOAD and standard FLOD or FILELOAD will have its “go faster” feature disabled. This means that the table B load portion of a FLOD or FILELOAD will be considerably slower than when *Sir2000 Field Migration Facility* is not in the picture. For this reason, it is **strongly** recommended that UAI and LAI be used for reorgs involving *Sir2000 Field Migration Facility* controlled files.

Since the SIRFIELD FORMAT and RELATE commands are only allowed in empty files a file reorg is required to take advantage of these features. Special attention is therefore payed to the initial file reorg to activate *Sir2000 Field Migration Facility* processing.

5.1 The Initial Sir2000 FMF Reorg

To take advantage of *Sir2000 Field Migration Facility* the appropriate SIRFIELD FORMAT and RELATE commands must be issued in an empty file. This involves several steps.

5.1.1 Unloading the original data

Unload the original dataset. This can be done with *Fast/Unload* or with User Language. The file can be unloaded in UAI format (*Fast/Unload* only), PAI format or a structured format. The simplest and fastest way to unload the file is using *Fast/Unload*'s UAI command. This not only unloads the field values but also the field definitions. It also allows sorting of the unloaded dataset by a primary key or keys which is often a good idea for producing an efficiently organized file.

5.1.2 Allocating and sizing files

Allocate and size the file to contain the reorged data. The file and table sizes might need to be increased from the unloaded file because of some of the extra data loaded by *Sir2000 Field Migration Facility*. Some file parameters (such as BRECPPG and maybe BRESERVE) might need to be adjusted also. *Sir2000 Field Migration Facility* data structures are not likely to be a concern here as they are unlikely to take up more than 2 table D pages. The more significant changes are likely to result from RELATE'd fields.

Specifically, each occurrence of a date field in the unloaded file will result in the addition of the RELATE'd date field in the loaded file which would require extra space in each table B record as well as in table C if the field is a KEY or NUMERIC RANGE field and in table D for ORDERED, KEY and NUMERIC RANGE fields.

Calculating the exact extra amount of space required in each table for RELATE'd fields can be difficult. Table C utilization is especially difficult to estimate as is table D. Fortunately, table D is easy to manage as there is no performance penalty in oversizing table D. The simplest file-sizing task, therefore, is to make sure that DSIZE in the file being loaded is considerably larger than DPGSUSED in the unloaded file if there are any ORDERED, KEY or NUMERIC RANGE RELATE'd field being created.

Sizing table B is a bit more challenging. First, the average number of occurrences per record of each field for which a RELATE'd field is to be generated should be estimated. If a field has the OCCURS attrbute this number is simply the value of the OCCURS attribute. This number should then be multiplied by the expected bytes used by the RELATE'd field to then adjust BRECPPG and BSIZE. The bytes used by a RELATE'd field can be determined by the field attributes.

OCCURS	All OCCURS fields use the number of bytes indicated by the LEN attribute. For BINARY and CODED fields, each occurrence uses 4 bytes.
BINARY	All non-OCCURS BINARY fields use 6 bytes per occurrence.

CODED	All non-OCCURS CODED fields use 6 bytes per occurrence.
FLOAT	All non-OCCURS FLOAT fields use 2 plus the LEN attributes value per occurrence. A “FLOAT LEN 8” field uses 2+8 or 10 bytes per occurrence.
STRING	All non-OCCURS STRING fields use 3 plus the length of the string data. If, for example, a RELATE'd field has a “YYYYMMDD” format and few occurrences that are exception values, one can expect each occurrence of this field to use 3+8 or 11 bytes.

Suppose you determine that a RELATE'd field is expected to occur, on average, 2 times per record and is expected to use 11 bytes per occurrence. Furthermore, suppose the current BRECPPG is set to 18 because the average record uses about 300 bytes. 18*300 is equal to 5400 which is less than the page size of 6144 but 18 is used for BRECPPG because variance in the lengths of the records would cause excessive extension records if the pages were allowed to fill more completely. To leave about the same amount of extra space per page after the reorg, BRECPPG would have to be adjusted to be 5400/(300+22) or 16.8. Since BRECPPG must be a whole number it should be set to either 16 or 17. The chosen value would depend on the variance in the number occurrences of the RELATE'd field. If it will occur exactly 2 times per record, the value of 17 should probably be chosen. If, on the other hand, the number of occurrences will vary widely between zero and several dozen a BRECPPG of 16 might be better because the increased variance of the record lengths will produce more extension records than is deemed desirable.

Sizing table C for a *Sir2000 Field Migration Facility* initial reorg is the most difficult challenge. A simple estimator that might be as good as anything would be to simply increase the size of table C by the percentage of added KEY and NUMERIC RANGE fields added. For example, if before the *Sir2000 Field Migration Facility* initial reorg there were 10 KEY and NUMERIC RANGE fields and there are 2 new RELATE'd fields that are KEY or NUMERIC RANGE, simply increase CSIZE by 20 percent. Obviously, this is a very rough estimator and for a very precise sizing of table C, iterative reorgs would probably be required. Table C sizing, in general, is a difficult thing to do correctly and is just one more reason why the ORDERED attribute might be preferrable to the KEY attribute in most cases.

5.1.3 Specifying SIRFIELD commands

After the target reorg file has been sized and allocated it must, of course, be FORMAT'ed and INITIALIZE'd. If the original file was unloaded with a UAI it would, at this point be possible to simply run the FLOD or FILELOAD LAI program which would automatically define the fields with the same attributes as the unloaded file. With a PAI or structured unload, all the fields have to be DEFINE'd before the FLOD or FILELOAD.

When the load is the initial *Sir2000 Field Migration Facility* load, the appropriate SIRFIELD commands must be issued to specify the format and relationship of the date fields. Since the SIRFIELD command can only be issued against DEFINE'd fields, this means that all fields on which SIRFIELD is to operate **must** be DEFINE'd before the FLOD or FILELOAD, even if the input dataset was created with UAI and so has the field definitions in it. This will not cause a problem during the load since *Fast/Reload* will simply not auto-define any fields that are already DEFINE'd at load time. In fact, there is a choice here in that one can DEFINE only the fields that are to be specified on a SIRFIELD command and let *Fast/Reload* auto-define the others or one can simply DEFINE all the fields before the load.

In any case, once all fields or at least the fields to be operated on by SIRFIELD commands are DEFINE'd, the appropriate SIRFIELD commands can be issued. The SIRFIELD command is documented in "The SIRFIELD Command" on page 25.

5.1.4 Loading the data

Once all or some (in the LAI case) of the fields have been DEFINE'd and the appropriate SIRFIELD commands have been issued, it is time to load the data. This is generally done with a FLOD or FILELOAD program. Whether the load is being done with a LAI, a generic PAI FLOD program or a structured FLOD program, *Sir2000 Field Migration Facility* will automatically verify the format of any field with a SIRFIELD format and automatically generate the RELATE'd field value for any loaded value. That is, in the most common case, *Sir2000 Field Migration Facility* will populate the file with 4-digit year equivalents of 2-digit year fields during the reorg process,

It is at this point that errors in the SIRFIELD command will often be caught. The most likely errors are mistakes in the specified date format and missed exception values. This latter type of error is the most likely error during an initial *Sir2000 Field Migration Facility* load and can happen quite easily since it is an error of omission and can be caused by not being aware of some special non-date format in date fields. This can be avoided by using the DATESTAT statement in the *Fast/Unload* FUEL program (if *Fast/Unload* was used to unload the data) and examining the FUNPRINT dataset. The date statistics could actually be collected while the unload is running though this would not be helpful if the unload and reload were part of a single job stream since there would be no opportunity to use the date statistics to correct the SIRFIELD commands.

Whether an LAI or FLOD program load, any SIRFIELD format errors during the FLOD or FILELOAD will leave the loaded file *physically inconsistent*. This means that either the original data must be cleansed and the whole reorg process re-done or the loaded file must be re-initialized, fields re-defined and the formats and/or exception values in the SIRFIELD commands must be corrected before loading the data again.

Even though a single SIRFIELD format error is sufficient to leave the loaded file *physically inconsistent*, *Sir2000 Field Migration Facility* will report up to the *Model 204* ERMX parameter number of errors before terminating regardless of whether the load is an LAI or FLOD program. Too small an ERMX and it might take many iterations of the reorg process to catch all the errors; too big an ERMX and the FLOD or FILELOAD might run an excessively long period of time after it has already been determined that the FLOD or FILELOAD should fail. In general, the default ERMX of 30 should be fine for most reorgs.

Once the file has been loaded with the appropriate SIRFIELD commands it can be considered to be under *Sir2000 Field Migration Facility*'s control. That is, *Sir2000 Field Migration Facility* will automatically ensure that values stored into fields with SIRFIELD specified formats will have the correct format and that updates to any field will be reflected in the field's RELATE'd field if there is one.

If after an initial *Sir2000 Field Migration Facility* reorg it becomes necessary to add new formats or related fields to a file, steps virtually identical to the original reorg must be followed. The only real differences are that a UAI/LAI reorg will preserve current SIRFIELD rules (unless explicitly prevented from doing so) so they would not need to be re-specified and the considerations specified in "Reorg considerations for Sir2000 FMF controlled files" should be kept in mind.

5.2 Reorg considerations for Sir2000 FMF controlled files

Reorganizing a *Sir2000 Field Migration Facility* controlled file using UAI/LAI can be quite simple. The *Fast/Unload* UAI statement automatically unloads all the SIRFIELD command definitions and the *Fast/Reload* LAI statement automatically applies these commands to the new file just as it automatically applies the unloaded field definitions. Unless overridden or prevented with the NOFDEF parameter, **all** SIRFIELD rules in the unloaded file will be inherited by the loaded file. These include SIRFIELD ALIAS, FORMAT, RELATE and SET rules.

Note that if a SIRFIELD command was specified with a format using a relative CENTSPAN (the default is a relative CENTSPAN of -50), the rule is automatically converted to an absolute CENTSPAN at the time the command is issued. For example, if a SIRFIELD FORMAT is specified with the default CENTSPAN of -50 in 1998, it is automatically converted to an absolute CENTSPAN of 1948. It is the absolute CENTSPAN that is preserved over a UAI/LAI reorg when LAI is allowed to auto-define the SIRFIELD rules.

If field definitions are explicitly specified during a reorg either because UAI/LAI is not being used or for some other reason, it makes sense to keep the field definitions in a file or *Model 204* procedure where they can be re-executed consistently. In this way, no matter the reorg strategy one can be assured that a reorg will not unintentionally lose or change field attributes or SIRFIELD rules.

When issuing a SIRFIELD FORMAT command with a relative CENTSPAN (the default is a relative CENTSPAN of -50) the relative CENTSPAN is converted into an absolute CENTSPAN when the command is issued. That is, if a CENTSPAN of -40 is specified for a field in 1999, it is converted to an absolute CENTSPAN of 1959. If this file is reorganized in 2001 and the same SIRFIELD FORMAT command is used for the same field, the CENTSPAN would then be converted to an absolute CENTSPAN of 1961. This is not likely to cause a problem unless the file contained dates in 1959 and 1960. If this is the case, however, there is no danger that these dates would be blindly converted to 2059 and 2060 as long as a reasonable SPANSIZE is used. Instead, the load of these dates would fail with a invalid date violation and the FLOD or FILELOAD would have to be re-run after, perhaps, the SIRFIELD FORMAT command has its CENTSPAN changed. It is highly recommended that reorgs not be done with SPANSIZE's greater than 90 since this reduces the protection against accidental century shifting of dates.

When unloading data for a PAI reorg only one field from a pair of related fields is unloaded. In fact the name used for the PAI of a field value can be an alias of the field name. In any case, only one value from a family of aliases and related fields will be unloaded with a PAI. For reorg purposes it shouldn't really matter which field is unloaded though it is generally considered “safer” to unload the field with the greater number of digits in the year. For this reason, the default *Sir2000 Field Migration Facility* PAI field in a pair of related fields is the field with the most digits in the year. If the default is not satisfactory it can be overridden with the SIRFIELD SET PAI commmand. The PAI name should not be a field or alias with the SIRFIELD SET REFERENCE WARN attribute since this will cause a warning at reload time. The SIRFIELD command will issue a warning if a SIRFIELD command make a WARN name a PAI name. SIRFIELD will not allow a PAI name to have the REFERENCE CANCEL attribute to be set.

When doing a structured unload of data (using PRINT or WRITE statements) for reorg purposes, it is essential that only one value from a pair of RELATE'd fields be unloaded. If both values were unloaded, both fields would be loaded twice with the same value, once each as the primary loaded value and once each as the automatically generated related value. While it shouldn't really matter which of a pair of RELATE'd fields is unloaded in a structured reorg, it is probably a good idea to unload the one with the most digits in the year.

5.3 Deleting a RELATE'd field with a reorg

Once a field in a file is under *Sir2000 Field Migration Facility*'s control the only way to eliminate that control is with a file reorg. A common example of when this would be done is after completion of a year 2000 conversion project. While *Sir2000 Field Migration Facility* gives one the ability to continue running with 2-digit years even after the year 2000, converting 2-digit year field references to 4-digit year field reference only where absolutely necessary, another approach is to simply convert **all** references from 2-digit year fields to 4-digit year fields. Once this conversion is completed, there is no longer a need to keep the 2-digit year field around any more.

5.3.1 Unloading the data

To eliminate a RELATE'd field a file reorganization must be done. The first step in this reorg process is unloading the data. With a *Fast/Unload* UAI unload there is no issue about which field of a RELATE'd pair is to get unloaded because UAI unloads both fields in the pair. Ordinarily, if the pair is still RELATE'd at LAI time *Fast/Reload* uses only the

"PAI" field at the time of the unload to reload the field, automatically generating the RELATE'd field value based on the SIRFIELD rules for both fields. However, if one of the fields no longer exists at the time of the reload, *Fast/Reload* will simply load the values associated with the remaining field. The bottom line is, once again, there need be no concern about which field of a RELATE'd pair is to be unloaded — they both are.

When unloading the data with either PAI or a structured unload with the intent of deleting one of a pair of RELATE'd fields over the reorg, it is essential that the correct field is unloaded. With a PAI unload, this means that the SIRFIELD PAI attribute should be set for the field being preserved. Typically, nothing special needs to be done to ensure this since the default PAI name is the field with the most digits in the year and that is almost certainly going to be the field that is to be preserved. With a structured unload, the User Language program that does the unload needs to be changed (if it hasn't been already) so that it outputs the field being preserved rather than the field being related. This is likely to require a change in the corresponding FLOD program used to load the data since the field name and format will have changed.

5.3.2 Allocating and sizing files

Allocate and size the file to contain the reorged data. The file and table sizes can be decreased from the unloaded file because the reorged file will no longer need space for the deleted fields. Some file parameters (such as BRECPPG and maybe BRESERVE) might need to be adjusted also. The sizing considerations for this step are basically the inverse of the sizing considerations for the initial *Sir2000 Field Migration Facility* reorg of the file as explained in "Allocating and sizing files" on page 40.

5.3.3 Specifying field definitions and SIRFIELD commands

When doing an LAI, *Fast/Reload* will automatically define any field that was defined in the unloaded file but is not defined in the target file. But to delete a field during a reorg, *Fast/Reload* must be prevented from doing this. This can be accomplished with the *NOFDEF* parameter on the LAI command. Specifying the NOFDEF parameter on the LAI statement prevents *Fast/Reload* from auto-defining any fields but does not prevent *Fast/Reload* from seeing values unloaded for the field. Once again, UAI unloads the values for both fields of a RELATE'd pair. Ordinarily, *Fast/Reload* would terminate with an error if it encounters data for an undefined field. To prevent it from doing so, the DELFIELD parameter must also be specified on the LAI command to indicate that values for undefined fields are to be deleted. An example of a FILELOAD program that can be used to delete one or more fields is

```
FILELOAD -1,-1,0,1000000,10000,100000,100000,50
LAI NOFDEF DELFIELD
END
```

For more information on the NOFDEF and DELFIELD parameters see the *Fast/Reload Reference Manual*. Unfortunately, the NOFDEF parameter prevents the definition of

any fields and their associated SIRFIELD rules. Because of this, any load (LAI or not) for a reorg being used to delete a field must be preceded by the field definitions and SIRFIELD rules for the remaining fields. This shouldn't be much of a problem since these rules will probably be saved in a operating system file or a *Model 204* procedure. Deleting a field simply entails editing this file or procedure and eliminating (or commenting out) any references to the field being deleted. Even if such a file or procedure doesn't exist, it can be created easily enough from the the output of a DISPLAY FIELD command with the DDL parameter as in

```
    DISPLAY FIELD (DDL) ALL
```

There are many ways the output from such a command can be moved into a file or procedure (USE statement, $COMMNDL, terminal emulator cut-and-paste, etc..) and it is assumed that the reader is familiar with one or more of them. If this is not the case contact Technical Support for assistance.

5.3.4 Loading the data

Once all the fields are defined and SIRFIELD commands are executed the data can be loaded using a FLOD or FILELOAD program. If doing an LAI program, the NOFDEF and DELFIELD parameters must have been specified on the LAI statement to delete a field. Be very, very careful when doing this as these two parameters together make it very easy to accidentally delete other fields. If a standard reorg job is modified for the purpose of doing a field-delete reorg, make sure that the NOFDEF and DELFIELD parameters are removed from the LAI statement as soon as the reorg is done so that subsequent reorgs will not be able to delete fields.

For a PAI reorg, the reload will fail if the field being deleted had been unloaded via PAI. Similarly, the FLOD compile will fail for a structured reorg if the FLOD program contains references to the deleted field.

5.3.5 UAI/LAI and Sir2000 Field Migration Facility rules

Much of the complexity of doing a file reorg is hidden from the user by the UAI/LAI facilities. This is generally a good thing in that basic reorg can be done with simple UAI and LAI statements and everything will automatically work as expected. If, however, a reorg is being done to affect some complex change in the structure of the data, especially as it pertains to *Sir2000 Field Migration Facility*, it becomes important to have a better understanding of some of the processing that goes on “under the covers” for UAI and LAI. Following is a list of some basic rules followed by UAI and LAI that might be useful in planning a complex reorg. If, even with all these rules, it is still unclear as to how certain situations would be handled by UAI or LAI, users should feel free to contact Technical Support for assistance and advice.

- Unless NOFDEF was specified on the LAI statement, *Fast/Reload* will automatically DEFINE any fields in the target file that were defined in the source file but are not already defined. These fields will be automatically DEFINE'd with the same attributes as they had in the source file. If a field is already defined in the target file at load time, its attributes are not changed. A way to change a field's attributes over a UAI/LAI reorg is to specify a DEFINE statement for the field with the new attributes before the LAI.

- Unless NOFDEF was specified on the LAI statement, *Fast/Reload* will automatically specify the SIRFIELD rules that were in effect in the unloaded file. This means that any SIRFIELD ALIAS, FORMAT, RELATE and SET commands would be effectively reissued in the target file with some restrictions. Among these restrictions, *Fast/Reload* will not override explicit SIRFIELD SET rules and will not override a SIRFIELD FORMAT for a field. If either one of a pair of fields that used to be RELATE'd is RELATE'd to another field, *Fast/Reload* will not automatically RELATE the original two fields (as this would be invalid, anyway). Finally, if an ALIAS for a field is already defined in the target file as a fieldname or an alias, the old alias definition would not be generated (as this would be invalid, anyway).

- UAI always unloads both fields of a pair of RELATE'd fields. If these fields are still RELATE'd at load time, *Fast/Reload* only loads the values for the field with the SIRFIELD PAI attribute set **in the source file at the time of the unload**. The RELATE'd field is then automatically generated based on the fields' formats.

  If these fields are no longer RELATE'd at load time, *Fast/Reload* will attempt to load each one individually, however, since this is breaking the logical link between the fields, it is considered a “family split” and will not be allowed unless the *FAMSPLIT* parameter is specified on the LAI statement, indicating that the such a break in logical links is intentional.

- If a name that was a fieldname in the unloaded data is an alias in the target file, the data associated with the original field is loaded into the field for which the field is an alias. If the target field also has data associated with its own name or another alias the FLOD or FILELOAD will be terminated because of an attempt to load data into a single field from multiple source fields. This is simply not allowed in an LAI.

- If a name that was an alias in the unloaded data is a field in the target file, any data associated with the field for which it used to be an alias is loaded into the new field. If the original field is also in the target file and is part of a different name family (see “Name families” on page 9) the data is also moved into that other family. This is not allowed unless the FAMSPLIT option is specified on the LAI command to indicate that this splitting is intentional (see the *Fast/Reload Reference Manual*).

CHAPTER 6 $Functions for SIRFIELD

This chapter describes the User Language $functions that are specifically intended for use with the SIRFIELD command. In addition to these $functions, see also the Sirius $functions documentation in M204wiki (**http://m204wiki.rocketsoftware.com/index.php/List_of_$functions**), and if you are licensed to use it, the $UL2KR, for $functions that can be used to process date values. In particular, the $SIR_DATE2ND function (and the $SIR_DATECHK function in the *Sir2000 User Language Tools*) can be used by your applications to accept only valid data.

6.1 The $SIRFIELD function

You can use this $function to retrieve the value of the datetime format or CENTSPAN of a field or alias.

```
%out = $SIRFIELD([filename], name, parameter)
```

Where:

%out	A User Language string %variable; for CENTSPAN its length should be at least 4, and for FORMAT it should be as long as the longest datetime format in your file (the longest valid datetime format is 100 characters).
filename	An optional string containing the name of the file of interest; if omitted, the default compile-time file is used.
name	A string containing the name of the field or alias of interest. This argument is required.
parameter	The parameter of interest for *name*:

FORMAT	Directs $SIRFIELD to return the date format of *name*, for example, **YYDDD** or **MM/DD/YYYY**.
CENTSPAN	Directs $SIRFIELD to return the CENTSPAN value for *name*.

If *name* is an alias, the FORMAT or CENTSPAN for the field associated with the alias is returned.

Any errors such as *name* not found, in group context, *filename* not open, or no attributes set for *name* will result in $SIRFIELD returning a null string.

No special file privileges are required to execute the $SIRFIELD function.

6.2 The $SIRFIELD_PAF function

The $SIRFIELD_PAF function may be used to print or retrieve the values and field names or aliases for *all* field occurrences in the current record, including values for fields managed by *Sir2000 Field Migration Facility*.

```
%num = $SIRFIELD_PAF([listID], [option], start_seq)
```

Where:

%num	A User Language numeric %variable; it will either be set to the number of field and/or alias names obtained by $SIRFIELD_PAF, or it will be set to a negative error return code.
listID	An optional identifier of a $list created by some other list $function, such as $ListNew (**http://m204wiki.rocketsoftware.com/index.php/$ListNew**). If this argument is supplied, the names and values are stored in the identified list; otherwise they will be processed as printed output.
option	An optional string which, if supplied, must contain one of the following values:

ALIAS	Output for a physical field occurence should include the field name as well as any alias names for the field.
FIELD	Output for a physical field occurence should only identify the associated field name.
PAI	Only physical field occurrences that have a PAI name will be processed, and the PAI name will be used to identify the occurrence.

PAI is the default if a $list is used, otherwise ALIAS is the default.

start_seq	The first sequence number to be used for $list output.

$SIRFIELD_PAF returns the following negative return codes if an error is detected:

```
-3   Not enough room to save in $list
-4   Invalid argument value
-6   Invalid QTBL offset for $list (argument 1)
```

The $SIRFIELD_PAF function must be called from within a FOR EACH RECORD loop.

6.2.1 $SIRFIELD_PAF output to a $list

If the first argument of $SIRFIELD_PAF is supplied, the names and field values are stored in a $list, which must have been previously created by another $function. The format of each $list item is as follows:

```
FLD_OR_ALIAS IS STRING LEN 1
*      'F' if the item is a fieldname, 'A' if an alias
WARN_CAN IS STRING LEN 1
*      'W' if the name has REFERENCE WARN
*      'C' if the name has REFERENCE CANCEL
*      ' ' otherwise
PAI IS STRING LEN 1
*      'P' if the name is the PAI name for the family
*      'N' otherwise
SKIP 5 POSITIONS
SEQ_NO IS BINARY LEN 4
*      Incremented for each $list item
NAME_LEN IS BINARY LEN 1
NAME IS STRING LEN 255
VALUE_LEN IS BINARY LEN 1
VALUE IS STRING LEN UNKNOWN
*      Maximum value length is 255
```

6.2.2 $SIRFIELD_PAF output to the terminal

If the first argument of $SIRFIELD_PAF is omitted, the names and field values are output to the user's current output stream. The format of the output depends upon the output option in effect, as follows:

The format of print lines when the ALIAS option is in effect:

```
    fieldname[(warn fld_pai)]: value
    **alias_name(ALIAS warn alias_pai)
```

Where:

warn	Optional indicator. If the SET REFERENCE WARN subcommand has been issued for the field or alias, the value will be **WARN**. If the SET REFERENCE CANCEL subcommand has been issued for the field or alias, the value will be **CANCEL**.
fld_pai	Optional indicator. If the field is the PAI name of a family with a related field, the value will be **PAI**. If an alias of the field is the PAI name of the family, the value will be **NOT PAI - SEE ALIAS**. If the field is related to a field which is the PAI name of the family or which has an alias which is the PAI name of the family, the value will be **NOT PAI - SEE RELATED**.
alias_pai	Optional indicator. If the alias is the PAI name of the family, the value will be **PAI**.

A comma is used to separate the items in the parenthesized description list.

The format of print lines when the FIELD option is in effect:

```
     fieldname[(warn fld_pai)]: value
```

Where:

warn	Optional indicator. If the SET REFERENCE WARN subcommand has been issued for the field, the value will be **WARN**. If the SET REFERENCE CANCEL subcommand has been issued for the field, the value will be **CANCEL**.
fld_pai	Optional indicator. If the field is the PAI name of a family with a related field, the value will be **PAI**. If an alias of the field is the PAI name of the family, the value will be **NOT PAI - SEE ALIAS**. If the field is related to a field which is the PAI name of the family or which has an alias which is the PAI name of the family, the value will be **NOT PAI - SEE RELATED**.

A comma is used to separate the items in the parenthesized description list.

The format of print lines when the PAI option is in effect:

```
     name = value
```

This is the same as the output of the PAI statement.

6.2.3 $SIRFIELD_PAF examples

The following subsections provide a variety of examples of output from $SIRFIELD_PAF. There examples are based upon the following hypothetical SIRFIELD attributes:

```
    SIRFIELD PS2Ø DISPLAY
    SIRFIELD 'PS2Ø' ALIAS 'STRING'
    SIRFIELD 'PS2Ø' SET REFERENCE WARN

    SIRFIELD BDY4 DISPLAY
    SIRFIELD 'BDY4' FORMAT DATE 'YYYY/MM/DD'
    SIRFIELD 'BDY4' ALIAS 'BIRTH_DTY4'
    SIRFIELD 'BDY2' FORMAT DATE 'YYDDD' CENTSPAN 1922 SPANSIZE 9Ø
    SIRFIELD 'BDY2' ALIAS 'BIRTH_DTY2'
    SIRFIELD 'BDY4' RELATE 'BDY2'
    SIRFIELD 'BIRTH_DTY4' SET PAI
    SIRFIELD 'BDY2' SET REFERENCE WARN
```

In addition, the field **SIMPLE** is defined, but has not been the subject of any SIRFIELD command.

```
B
%X FIXED
%L FIXED
%L = $LISTNEW
PRINT 'Begin $SIRFIELD_PAF...'
FR
%X=$LISTDEL(%L)
%X=$SIRFIELD_PAF(%L)
PRINT 'Returned: ' %X '.  $LISTCNT: ' $LISTCNT(%L) '.'
FOR %X FROM 1 TO $LISTCNT(%L)
PRINT 'Name: ' $DEBLANK($LISTINF(%L, %X, 14, 255))
PRINT 'Flags: ' $LISTINF(%L, %X, 1, 3)
PRINT 'Value: ' $LISTINF(%L, %X, 15+255, 255)
PRINT '............'
END FOR
PRINT '------------'
END
```

$SIRFIELD_PAF $list Example: ALIAS Option

The preceding User Language code produces the following output:

```
Begin $SIRFIELD_PAF...
Returned: 4.  $LISTCNT: 4.
Name: PS2Ø
Flags: FWP
Value: BASE
............
Name: BIRTH_DTY4
Flags: A P
Value: 1997/Ø1/Ø1
............
Name: BIRTH_DTY4
Flags: A P
Value: 1997/Ø1/Ø2
............
Name: SIMPLE
Flags: F P
Value: DREAMS
............
------------
```

```
B
%X IS FIXED
PRINT 'Begin FIELD...'
FR
* Note that 'FIELD' is default for second arg:
%X = $SIRFIELD_PAF
PRINT '------------'
END
```

$SIRFIELD_PAF Output Example: FIELD Option

The preceding User Language code produces the following output:

```
Begin FIELD...
PS2Ø(WARN): BASE
BDY4(NOT PAI - SEE ALIAS): 1997/Ø1/Ø1
BDY2(WARN, NOT PAI - SEE RELATED): 97ØØ1
BDY4(NOT PAI - SEE ALIAS): 1997/Ø1/Ø2
BDY2(WARN, NOT PAI - SEE RELATED): 97ØØ2
SIMPLE: DREAMS
------------
```

```
B
%X IS FIXED
PRINT 'Begin PAI...'
FR
%X = $SIRFIELD_PAF( , 'PAI')
PRINT '------------'
END
```

$SIRFIELD_PAF Output Example: PAI Option

The preceding User Language code produces the following output:

```
Begin PAI...
PS2Ø = BASE
BIRTH_DTY4 = 1997/Ø1/Ø1
BIRTH_DTY4 = 1997/Ø1/Ø2
SIMPLE = DREAMS
------------
```

```
B
%X IS FIXED
PRINT 'Begin ALIAS...'
FR
%X = $SIRFIELD_PAF(, 'ALIAS')
PRINT '------------'
END
```

$SIRFIELD_PAF Output Example: ALIAS Option

The preceding User Language code produces the following output:

```
Begin ALIAS...
PS2Ø(WARN): BASE
**STRING(ALIAS)
BDY4(NOT PAI - SEE ALIAS): 1997/Ø1/Ø1
**BIRTH_DTY4(ALIAS, PAI)
BDY2(WARN, NOT PAI - SEE RELATED): 97ØØ1
**BIRTH_DTY2(ALIAS)
BDY4(NOT PAI - SEE ALIAS): 1997/Ø1/Ø2
**BIRTH_DTY4(ALIAS, PAI)
BDY2(WARN, NOT PAI - SEE RELATED): 97ØØ2
**BIRTH_DTY2(ALIAS)
SIMPLE: DREAMS
------------
```

CHAPTER 7 Datetime Processing Considerations

This chapter presents date processing issues, including usage of the *Sir2000 Field Migration Facility* 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 Field Migration Facility*.

The *Sir2000 Field Migration Facility* 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 Field Migration Facility* is under a rental or trial agreement
- as values to be validated before storing into fields identified by the SIRFIELD FORMAT command
- as values to be converted when storing into fields identified by the SIRFIELD RELATE command

Please note that in addition to the above date processing performed by the *Sir2000 Field Migration Facility*, you also can retrieve fields with $SIRFIELD_PAF which might contain two digit year date values. The customer must ensure that any application using that data has an algorithm or rule for unambiguously determining the correct century for the values.

To correctly use the *Sir2000 Field Migration Facility* past the year 1999, version 5.0 of the *Sirius Mods*, or later, is 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 Field Migration Facility*, 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 Field Migration Facility* to operate properly.

Most Sirius date processing, with the notable exception of the *Sir2000 Field Migration Facility*, involves the use of datetime $functions. To better explain the full range of Sirius date processing, 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 Field Migration Facility*.

2. The *Sir2000 User Language Tools Functions*, which are documented in the *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*. Some references to these functions are made in this manual, nonetheless, to illustrate some datetime $function considerations.

See “Strict and non-strict format matching” on page 66 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.

7.1 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 Field Migration Facility*. 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, the SIRFIELD FORMAT command requires that a year token (for example, “YYYY”, “CYY” or “YY”) be present in the primary format. Also, for example, when a value is stored into a field with any SIRFIELD format, a leading blank is **not** allowed for the HH, DD, nor MM tokens, but 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.

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, except for *Sir2000 Field Migration Facility* alternate or error formats.

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 “*” token, then the other format string must also contain an “*” token. All of the characters from the input string that correspond to the “*” token in the input format, if any, are copied unaltered to the output string, begining in the position that corresponds to the “*” token in the output format.

   The $functions with both an input and output format, for example $SIR_DATECNV, are only available in the *Sir2000 User Language Tools*.

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 the *Sir2000 Field Migration Facility*, the input format rule for a token is the same as the output format rule; this is the definition of “strict date format matching”. This ensures that values accepted as the update of a field will be preserved across a reorg, for related fields. 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 “SIRFIELD datetime and format examples” on page 66. See “Strict and non-strict format matching” on page 66.

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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
"	Following character is “quoted”, that is, it acts as a separator character. See “SIRFIELD datetime and format examples” on page 66.
YYYY	4 digit year
YY	2 digit year
CYY	Year minus 1900 (3 digits, including any leading zero). See “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
MONTH	Full month name (upper case variable length). Non-strict $functions allow any mixture of upper and lower case on input, but all upper case is always produced on output.
Month	Full month name (mixed case variable length). Non-strict $functions allow any mixture of upper and lower case on input, but initial upper case letter followed by all lower case is always produced on output.
MON	Three character month abbreviation (upper case). Non-strict $functions allow any mixture of upper and lower case on input, but all upper case is always produced on output.

Mon	Three character month abbreviation (mixed case). Non-strict $functions allow any mixture of upper and lower case on input, but initial upper case letter followed by all lower case 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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
WKDAY	Full day of week name (upper case variable length). Non-strict $functions allow any mixture of upper and lower case on input, but all upper case is always produced on output.
Wkday	Full day of week name (mixed case variable length). Non-strict $functions allow any mixture of upper and lower case on input, but initial upper case letter followed by all lower case is always produced on output.
WKD	Three character day of week abbreviation (upper case). Non-strict $functions allow any mixture of upper and lower case on input, but all upper case is always produced on output.
Wkd	Three character day of week abbreviation (mixed case). Non-strict $functions allow any mixture of upper and lower case on input, but initial upper case letter followed by all lower case 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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66.
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 “SIRFIELD datetime and format examples” on page 66 for examples which include use of various separator characters.

7.2 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.

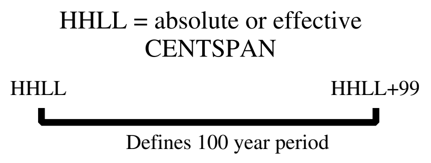
7.3 Processing Dates With Two-Digit Year Values

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.

7.3.1 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.

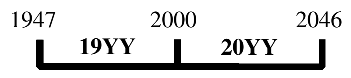


Conversion rules, YY to YYYY

```
•if YY < LL YYYY = (HH+1)YY
•else       YYYY = HHYY
```

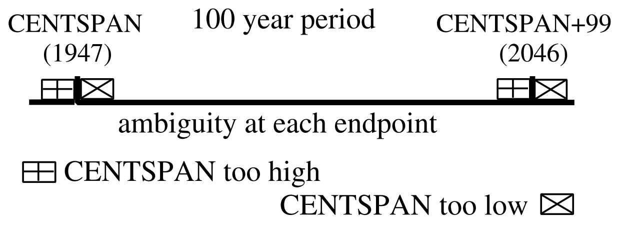
Example:

```
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.



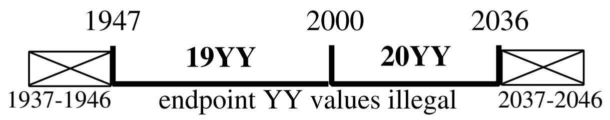
If CENTSPAN is set to a value that is too high, dates that are just prior to CENTSPAN will appear to occur 100 years hence. If CENTSPAN is set to a value that is too low, dates that fall just after CENTSPAN+99 will appear to have occured 100 years earlier. A full one-hundred year window also can not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

7.3.2 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.

7.4 Strict and non-strict format matching

As mentioned in "Datetime Formats" on page 59, for the *Sir2000 Field Migration Facility*, 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 lower case 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 wish 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
```

7.5 SIRFIELD datetime and format examples

There is an extensive set of format tokens, as shown in "Datetime Formats" on page 59. 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 Field Migration Facility* 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.

YYMMDD This is the common 6-digit date format which supports sort order if all dates are within a single century. With the following SIRFIELD command

```
SIRFIELD FOO FORMAT DATE YYMMDD
```

the value 960229 would be allowed to be stored in field FOO.

Notes:

- Since leading zeroes would be lost, a field with this format should not have the FLOAT or ORDERED NUMERIC attribute if it can hold YY values less than 10 (e.g., for 2001).

YYYYMMDD

This is the common 8-digit date format which supports sort order with dates in 2 centuries. With the following SIRFIELD commands

```
SIRFIELD FOO    FORMAT DATE YYMMDD
SIRFIELD FOOY2K FORMAT DATE YYYYMMDD
SIRFIELD FOOY2K RELATE FOO
```

storing the value 921212 in the field FOO would also cause the value 19921212 to be stored in field FOOY2K.

MM/DD/YY

This is the U.S. 6-digit date format for display. With the following SIRFIELD command

```
SIRFIELD FOO FORMAT DATE MM/DD/YY
```

the value 12/14/94 would be allowed to be stored in field FOO.

Notes:

- In the non-strict $functions., the leading zero corresponding to an MM token may be given as a blank, thus allowing “ 7/15/98”; however, in the *Sir2000 Field Migration Facility*, such leading blank is not allowed for MM. If a field contains leading blanks for some values of a month, use the BM token; if the field contains leading zeroes in some instances and leading blanks in others, use alternate date formats. Alternate date formats and the BM token are available starting with version 5.1 of the *Sir2000 Field Migration Facility*.

DD.MM.YY

This is a European 6-digit date format for display. With the following SIRFIELD command

```
SIRFIELD FOO FORMAT DATE DD.MM.YY
```

the value 14.12.94 would be allowed to be stored in field FOO.

Notes:

- In the non-strict $functions, the leading zero corresponding to a DD token may be given as a blank, thus allowing “ 7.04.89”; however, in the *Sir2000 Field Migration Facility*, such leading blank is not allowed for DD. If a field contains leading blanks for some values of a day, use the BD token; if the field contains leading zeroes in some instances and leading blanks in others, use alternate date formats. Alternate date formats and the BD token are available starting with version 5.1 of the *Sir2000 Field Migration Facility*.

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

This is a format which could be used for report headers. This would be very unlikely in a SIRFIELD command, but just to round out the discussion of some date format tokens, assuming the following SIRFIELD command

```
SIRFIELD FOO FORMAT DATE -
   'Wkday, DAY Month YYYY "A"T HH:MI'
```

the value “Friday, 7 February 1998 AT 21:33” would be allowed to be stored in field FOO.

Notes:

- The apostrophes must enclose the date format on the SIRFIELD command; any format containing a blank, comma, apostrophe, equal sign, or left or right parenthesis must be enclosed in apostrophes.

- If the format had contained 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.

- The day number (string matching DAY) must not have a leading zero. Note that in the non-strict $functions, a leading zero is allowed. If a field contains leading zeroes for some values of a day, use the DD token; if the field contains leading zeroes in some instances and a one-digit day in others, use alternate date formats. Alternate date formats are available starting with version 5.1 of the *Sir2000 Field Migration Facility*.

- In the non-strict $functions, the leading zero corresponding to an HH token may be given as a blank, thus allowing

  ```
  Friday, 31 February 1998 AT  8:33
  ```

  However, in the *Sir2000 Field Migration Facility*, such leading blank is not allowed for HH. If a field contains leading blanks for some values of an hour, use the BH token; if the field contains leading zeroes in some instances and leading blanks in others, use alternate date formats. Alternate date formats and the BH token are available starting with version 5.1 of the *Sir2000 Field Migration Facility*.

YYIIII This is a format which could be used for a field which contains a 2-digit year prefixing other information, such as a sequence number. With the following SIRFIELD commands

```
SIRFIELD FOO    FORMAT DATE YYIIII
SIRFIELD FOOY2K FORMAT DATE YYYYIIII
SIRFIELD FOOY2K RELATE FOO
```

storing the value 923142 in the field FOO would also cause the value 19923142 to be stored in field FOOY2K.

Notes:

- This format could also be used to handle a field which sometimes contains a valid date (for example, dates in the format YYMMDD) and sometimes not (in the same example, sometimes alphabetic characters for MMDD). The year will be validated (as numeric and in the CENTSPAN/SPANSIZE range) and the appropriate 2-digit to 4-digit year conversion will be performed, so you can use FOOY2K for range searches, etc.

  Note that the combintation of YYMMDD and YYIIII data can also be handled using alternate date formats, available in version 5.1 of the *Sir2000 Field Migration Facility*. This could allow storing a special marker for non-date values.

- Both formats of a pair of related fields must have the same number of I tokens. See "Variant date formats" on page 10.

YY* This is a format which could be used for a field which contains a 2-digit year prefixing other information, such as a sequence number, when the other information is variable length. With the following SIRFIELD commands

```
SIRFIELD FOO    FORMAT DATE YY*
SIRFIELD FOOY2K FORMAT DATE YYYY*
SIRFIELD FOOY2K RELATE FOO
```

storing the value 92736AB in the field FOO would also cause the value 1992736AB to be stored in field FOOY2K, and storing the value 92991 in the field FOO would also cause the value 1992991 to be stored in field FOOY2K,

Notes:

- As in the preceding example, this format can also be used to handle a field which sometimes contains a valid date and sometimes not.
- At most one occurrence of the * token may appear in a datetime format.
- If a * token appears in the format of a field and that field is related, a * must also appear in the format of the related field.

CYYDDD This is a compact 6-digit date format with explicit century information, from 1900 through and including 2899. With the following SIRFIELD command

```
SIRFIELD FOO FORMAT DATE CYYDDD
```

the value 097031 would be allowed to be stored in field FOO.

Notes:

- Since leading zeroes would be lost, a field with this format should not have the FLOAT attribute if it can hold values in the 1900's (i.e., 31 Jan 1997 is 097031).
- This format supports range, etc., processing.

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 1900's. With the following SIRFIELD command

```
SIRFIELD FOO FORMAT DATE ZYYDDD
```

the values 970501 (representing 1 May 1997) and 1000501 (representing 1 May 2000) would each be allowed to be stored in field FOO.

Notes:

- As long as dates are from 1910 or later, a field with this format **may** have the FLOAT attribute, because the years 1900 - 1999 can also be represented as 0 - 99, and after that the values will not have leading zeroes.

- Range processing is supported with a field of this format if it is ORDERED NUMERIC, and SORT must use the NUMERICAL option to produce the correct order for the dates before and after 2000. For dates before and after 2000, FOR EACH VALUE processing will present the values in order if, and only if, the field is ORDERED NUMERIC.

- The century and year number (string matching ZYY) must not have a leading zero. Note that in the non-strict $functions, a leading zero is allowed. If a field contains zeroes for some values of the century, use the CYY token; if the field contains leading zeroes in some instances and a two-digit ZYY in others, use alternate date formats. Alternate date formats are available starting with version 5.1 of the *Sir2000 Field Migration Facility*.

"Z* This format could be used to relate non-date values so they can be marked as such. With the following commands:

```
SIRFIELD DATE1 FORMAT DATE YYMMDD
SIRFIELD DATE1Y2K FORMAT DATE YYYYMMDD
SIRFIELD DATE1Y2K RELATE DATE1 -
   ERROR (*, "Z*)
```

If you attempt to store a non-date value in DATE1, the value will be stored with a leading Z in DATE1Y2K, and you can later find all records with errors in this field, with the following User Language request:

```
BEGIN
FOR EACH RECORD WHERE -
   DATE1Y2K IS LIKE 'Z*'
   PRINT '*'
   PAI
END FOR
END
```

Note:

- The ERROR clause of the SIRFIELD command is available starting with version 5.1 of the *Sir2000 Field Migration Facility*.

- The quoting token (“"”) is available starting with version 5.1 of the *Sirius Mods*.

YY0000 You can translate the separator characters between a pair of related fields. For example, with the following commands:

```
SIRFIELD DATE1 FORMAT DATE YYMMDD
SIRFIELD DATE1Y2K FORMAT DATE YYYYMMDD
SIRFIELD DATE1Y2K RELATE DATE1 ALT DATE (YY0000,
YYYY9999)
```

you can change specific non-date values (in this case, zeroes for month and day), here causing them to sort high rather than low.

Notes:

- Numeric separators, unlike alphabetic separators, do not need to be preceeded by a quote character (").
- In terms of 2- to 4-digit year conversions, usually an I token will suffice instead of a numeric separator.
- Numeric separators are available starting with version 5.1 of the *Sirius Mods*.

7.6 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 *Model 204* date $functions provided by CCA.

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. CCA date formats do not have any day of week nor time of day tokens, and other CCA 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.

CHAPTER 8

Messages

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

Index