

---

# Janus Specialty Data Store Reference Manual

## Model 204 Interoperability Products

---



Sirius Software, Inc.  
875 Massachusetts Avenue, Suite 21  
Cambridge, MA 02139

Telephone: (617) 876-6677  
FAX: (617) 234-1200  
E-mail: [support@sirius-software.com](mailto:support@sirius-software.com)  
World Wide Web: <http://sirius-software.com>

August 3, 2010

© 2010 Sirius Software, Inc.



---

## *Proprietary Notices*

The following products:

- *Fast/Backup*
- *Fast/Reload*
- *Fast/Unload*
- *Fast/Unload User Language Interface*
- *Janus Network Security*
- *Janus Open Client*
- *Janus Open Server*
- *Janus Sockets*
- *Janus Specialty Data Store*
- *Janus TCP/IP Base*
- *Janus Web Server*
- *SirDBA*
- *Sirius Mods*
- *SirMon*
- *SirPro*
- *SirScan*
- *Sir2000 Field Migration Facility*
- *Sir2000 User Language Tools*
- *UL/SPF*

are proprietary products of Sirius Software, Inc.:

**Sirius Software, Inc.**  
**875 Massachusetts Avenue, Suite 21**  
**Cambridge, Massachusetts 02139**  
**USA**

**Model 204®** is a proprietary product of Computer Corporation of America, a wholly-owned subsidiary of Rocket Software, Inc., which owns the trademark:

**Rocket Software Corporate Office**  
**M204 Division**  
**275 Grove Street**  
**Suite 3-410**  
**Newton, Massachusetts 02466-2272**  
**USA**

Sybase, DB-Library, Omni SQL Server and Adaptive Server are registered trademarks of Sybase Inc.:

**Sybase Inc.**  
**6475 Christie Ave.**  
**Emeryville, California 94608**  
**USA**

---

## Contents

<b>Proprietary Notices</b> . . . . .	<b>iii</b>
<b>Contents</b> . . . . .	<b>v</b>
<b>Summary of Changes</b> . . . . .	<b>ix</b>
Sirius Mods Version 6.0 . . . . .	ix
Sirius Mods Version 5.5 . . . . .	ix
Sirius Mods Version 5.4 . . . . .	ix
Sirius Mods Version 4.6 . . . . .	x
 <b>Chapter 1: Overview</b> . . . . .	 <b>1</b>
Janus, the Sirius Mods, and UL/SPF . . . . .	3
Versions and compatibility . . . . .	4
Related manuals . . . . .	5
Related products . . . . .	6
System requirements . . . . .	6
 <b>Chapter 2: Janus / Connectivity Concepts</b> . . . . .	 <b>7</b>
Server Ports . . . . .	7
Specialty Data Store or Omni Access Module . . . . .	8
 <b>Chapter 3: Getting Started</b> . . . . .	 <b>11</b>
 <b>Chapter 4: Janus Commands</b> . . . . .	 <b>13</b>
JANUS command overview . . . . .	14
JANUS DEFINE . . . . .	16
ALLOCC . . . . .	18
AUDTERM . . . . .	18
AUTOLOAD . . . . .	18
BINDADDR xxx . . . . .	19
BSIZE xxx . . . . .	19
CMD 'xxx' . . . . .	19
FDWOL . . . . .	20
IBSIZE xxx . . . . .	20
JANCAT xxx . . . . .	21
MASTER . . . . .	21
MAXCURS xxx . . . . .	21
MAXSAVE xxx . . . . .	22
NOAUDTERM . . . . .	22

NOUPCASE . . . . .	22
OBSIZE xxx . . . . .	23
OMNIACCT xxx . . . . .	23
OMNIUSER xxx . . . . .	23
OPEN list . . . . .	23
PRELOGINUSER userid . . . . .	24
RAWINPUT . . . . .	24
RAWINPUTONLY . . . . .	25
RBSIZE xxxx . . . . .	26
SDSACCT xxx . . . . .	26
SDSUSER xxx . . . . .	26
SSL . . . . .	26
SSLBSIZE xxxx . . . . .	27
SSLCACHE xxxx . . . . .	28
SSLCIPH xxx . . . . .	29
SSLCLCERT and SSLCLCERTR . . . . .	29
SSLIBSIZE xxxx . . . . .	30
SSLMAXAGE xxx . . . . .	31
SSLMAXCERTL xxx . . . . .	32
SSLOBSIZE xxxx . . . . .	32
SSLPROT xxx . . . . .	33
SSLUNENC . . . . .	33
TCPKEEPALIVE . . . . .	34
TIMEOUT xxxx . . . . .	35
TRACE xxx . . . . .	35
UPCASE . . . . .	36
XTAB table . . . . .	36
<b>Chapter 5: Subsystem JANCAT . . . . .</b>	<b>37</b>
Open File screen . . . . .	37
Tables screen . . . . .	39
Autobuild screen . . . . .	42
Columns display . . . . .	46
Column Update screen . . . . .	51
JANCAT security system . . . . .	54
Security Groups screen . . . . .	54
Modify Security Group screen . . . . .	55
Security Entries screen . . . . .	56
Add Users to Table screen . . . . .	57
<b>Appendix A: Specialty Data Store Catalog File Structure . . . . .</b>	<b>59</b>
<b>Appendix B: Datetime Processing Considerations . . . . .</b>	<b>69</b>
Datetime Formats . . . . .	70
Valid Datetimes . . . . .	73
Processing Dates With Two-Digit Year Values . . . . .	74

CENTSPAN . . . . .	74
SPAN SIZE . . . . .	75
Datetime and format examples . . . . .	76
<b>Index . . . . .</b>	<b>79</b>

## **Figures**

<b>Figure 1: Model 204 TCP/IP Connectivity using Janus . . . . .</b>	<b>2</b>
<b>Figure 2: JANUS DEFINE command syntax . . . . .</b>	<b>16</b>
<b>Figure 3: JANCAT Open File screen . . . . .</b>	<b>37</b>
<b>Figure 4: JANCAT Tables screen . . . . .</b>	<b>39</b>
<b>Figure 5: JANCAT Autobuild . . . . .</b>	<b>42</b>
<b>Figure 6: JANCAT Columns display . . . . .</b>	<b>46</b>
<b>Figure 7: JANCAT Column Update screen . . . . .</b>	<b>51</b>
<b>Figure 8: JANCAT Security Groups screen . . . . .</b>	<b>54</b>
<b>Figure 9: JANCAT Modify Security Group screen . . . . .</b>	<b>55</b>
<b>Figure 10: JANCAT Security Entries screen . . . . .</b>	<b>56</b>
<b>Figure 11: JANCAT Add Users to Table screen . . . . .</b>	<b>57</b>



---

## *Summary of Changes*

This section describes significant changes to the documentation. Usually, these changes correspond to enhancements made to the underlying product, although they might be simple documentation improvements.

### **Sirius Mods Version 6.0**

The following changes correspond to changes in *Janus Specialty Data Store* in version 6.0 of the *Sirius Mods*.

- DEBUG keyword replaced by TRACE
- AUDTERM replaced by NOAUDTERM as port default (JANUS DEFINE), and now applicable to SDS ports

### **Sirius Mods Version 5.5**

The following changes correspond to changes in *Janus Specialty Data Store* in version 5.5 of the *Sirius Mods*.

- New parameters SDSUSER and SDSACCT for JANUS DEFINE documented.

### **Sirius Mods Version 5.4**

The following changes correspond to changes in *Janus Specialty Data Store* in version 5.4 of the *Sirius Mods*.

- Manual name changed from ***Janus Omni Access Module Reference Manual*** to ***Janus Specialty Data Store Reference Manual***. This reflects the change in name of the product from *Janus OmniSQL Access Module* to *Janus Specialty Data Store*.
- New port types SDS and OAS documented.
- New parameter MAXCURS for JANUS DEFINE documented.

## **Sirius Mods Version 4.6**

The following changes correspond to changes in *Janus Specialty Data Store* in version 4.6 of the *Sirius Mods*.

- Manual converted for layout changes

---

**CHAPTER 1** *Overview*

Janus is a family of products that provides direct bi-directional access, via a TCP/IP network, between *Model 204* and programs running on other platforms. One of the Janus products, *Janus TCP/IP Base*, can be used by itself, and is also required by the other Janus products, each of which is optional. The Janus product family consists of the following:

*Janus TCP/IP Base*

*Janus TCP/IP Base* provides TCP/IP connectivity to *Model 204*.

It also includes a *Janus IFDIAL Library* for access to *Model 204* similar to IFDIAL, enabling TCP/IP access to *Model 204* from Unix workstations, DOS and Windows-based PCs, and other machines that support the C language and the TCP/IP protocol layers. The library has C routines for communication with a *Model 204* Online and C programs that can be used to communicate with a *Model 204* Online without additional programming, similar to the BATCH2 utility.

*Janus Open Server*

*Janus Open Server* provides a set of \$functions that allow *Model 204* to be a server in response to Sybase CT and DB-Library Open Client calls and SQL EXECUTE statements.

A server application consists of a set of User Language procedures which are invoked by a client application's request. Client applications request the execution of a procedure via Sybase remote procedure calls (RPCs), which are implemented as part of Sybase's DB-Library Open Client code. The Sybase client sends the name of a stored procedure and an arbitrary number of parameters, and the Janus server executes the corresponding User Language procedure and returns the requested data in *Model 204* images (which appear as "rows" to the client) or as RPC return parameters.

*Janus Open Client*

*Janus Open Client* provides a set of \$functions that allow *Model 204* applications to be client applications to Sybase SQL Servers or Open Servers. A *Model 204* client application sends RPCs or language requests (for example, SQL) to a Sybase open server or Sybase SQL server, and retrieves the results for further processing. It is possible for a User Language application to act as a client to several different servers simultaneously, and it is possible for a server application to simultaneously act as a client application.

### *Janus Specialty Data Store*

*Janus Specialty Data Store* provides access to *Model 204* from Sybase Adaptive Server OmniConnect or from the older Sybase Omni SQL Server or Gateway. This includes optimized translation of SQL into User Language and a cataloging facility for mapping *Model 204* files to SQL structures. A user application issues SQL requests, which are routed to Sybase Omni and then routed to *Janus Specialty Data Store*, which provides the SQL response using data from one or more *Model 204* files.

The cataloging facility, JANCAT, maps *Model 204* files and fields onto a table/column structure, which Sybase Omni can then re-map onto its own table/column definitions. *Janus Specialty Data Store* does not require UNIQUE attributes nor any other alteration of your *Model 204* files.

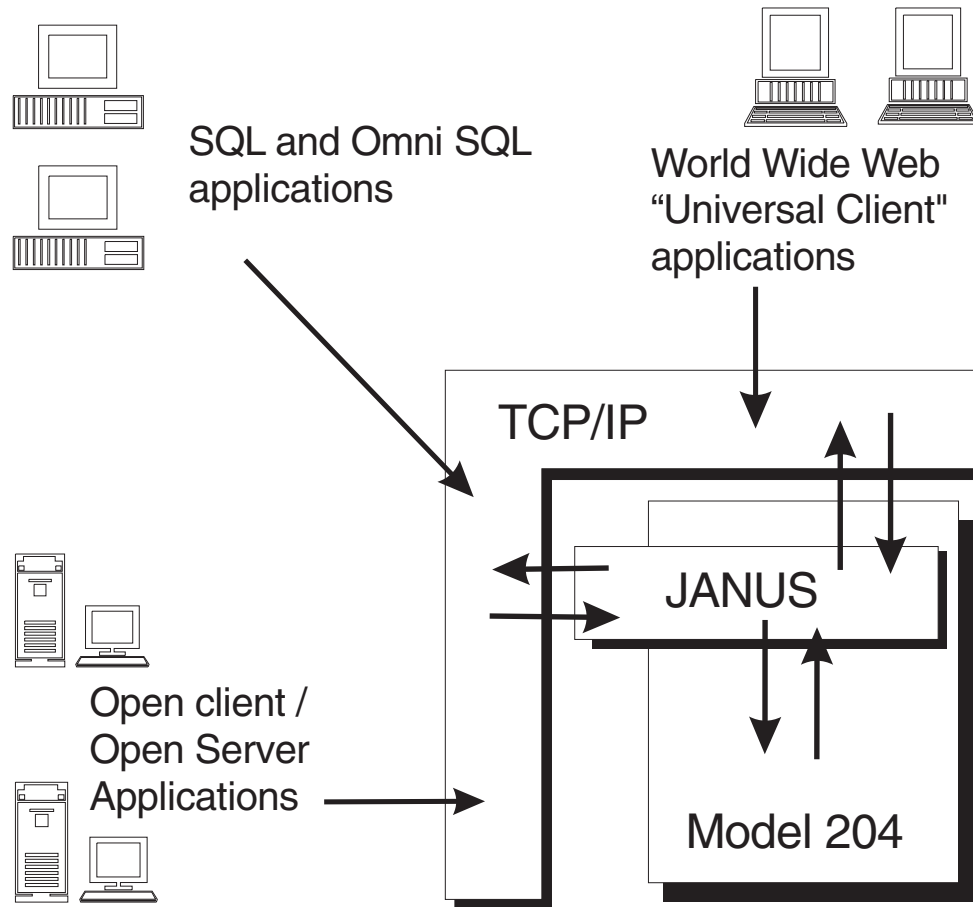
### *Janus Web Server*

*Janus Web Server* is a full-featured Hyper-Text Transmission Protocol (http) server for the World Wide Web (WWW). It provides an interface to all WWW data types stored in *Model 204* procedures or database files. It can pass plain text, Hyper-Text Markup Language (HTML), and binary data types to web browser applications, and it can control access to secured, high-performance applications. *Janus Web Server* also provides a full API for building web applications in User Language.

### *Janus Network Security*

*Janus Network Security* supports the SSL (Secure Sockets Layer) protocol, which provides secure communications for users of Janus products. It can be used with any Janus product, but it is most commonly used to secure communications of Janus Web Server applications. Users of web browsers communicating with such applications can be confident that their communications will be encrypted, and the identity of the server with which they are communicating is authenticated.

The use of all Janus products involves the **JANUS *Model 204*** command, which is documented in the ***Janus TCP/IP Base Reference Manual***. This command provides Janus port definition, starting, stopping, and monitoring of ports; for Web implementations it also defines the Web Server rules.



**Model 204 TCP/IP Connectivity using Janus**

The above figure shows that with TCP/IP as the communications protocol, Janus provides high-speed bi-directional access to *Model 204* from any client.

## 1.1 Janus, the Sirius Mods, and UL/SPF

*Janus Specialty Data Store* is part of the Janus family of products that provides connectivity to the *Model 204* database. A site that has *Janus Specialty Data Store* must have the *Janus TCP/IP Base* because without it, it is impossible to use *Janus Specialty Data Store*. A *Janus Specialty Data Store* site might also have one or more of the other products in the Janus family, though no others are required. Note that if *Limited Janus Web Server* is available, then *Janus TCP/IP Base* is automatically authorized. *Limited Janus Web Server* is a free, restricted version of *Janus Web Server*; they are both documented in the ***Janus Web Server Reference Manual***.

The Janus family of products is itself made up of two distinct components:

- A collection of object code enhancements to the *Model 204* database-engine nucleus.

These enhancements are distributed as components of the **Sirius Mods** and make up a collection of products including those in the Janus family. The *Sirius Mods* include many non-connectivity related products (such as *Fast/Backup*, *Fast/Reload*, and the *Fast/Unload User Language Interface*) that are not part of the Janus family. No *Sirius Mods* products are required to run *Janus Specialty Data Store* other than itself and *Janus TCP/IP Base*.

- A collection of *Model 204* procedures that contain User Language, documentation, and assorted other data.

These *Model 204* procedures install and implement the components of the User Language Structured Programming Facility, also known as **UL/SPF**. All the *UL/SPF* files reside in the SIRIUS procedure file (as of *Sirius Mods* version 6.8). which also contains code and data useful to Janus users including *Janus Specialty Data Store* users.

*UL/SPF* also includes files that are components of non-connectivity related products such as *SirPro*, *SirScan*, and *SirMon*. No other *UL/SPF* products are required to run *Janus Specialty Data Store*, or any other Janus product, for that matter.

Thus, to install *Janus Specialty Data Store*, both the *Sirius Mods* and *UL/SPF* must be installed, following the instructions in the ***Sirius Mods Installation Guide*** and the ***UL/SPF Installation and Maintenance Guide***, respectively. When the *Sirius Mods* are installed, all other products owned by the installing site that are part of the *Sirius Mods* will also be installed. Similarly, when *UL/SPF* is installed, all other products owned by the installing site that are part of *UL/SPF* will be installed.

## 1.2 Versions and compatibility

Because the *Sirius Mods* and *UL/SPF* have somewhat different release cycles, the version numbers for these two components will often differ in a distribution. For example, version 7.6 of the *Sirius Mods* might be shipped with version 7.3 of *UL/SPF*. All the products in *UL/SPF* depend on certain features being present in the version of the *Sirius Mods* that is installed in the *Model 204* load module under which *UL/SPF* is running. This implies, obviously, that the *Sirius Mods* must be installed for any *UL/SPF* component to operate correctly. And, as of version 6.8, the *Sirius Mods* version must match or be higher than the *UL/SPF* version number.

The *Sirius Mods* however, do not depend on any particular features of the *UL/SPF* product, merely the presence of the *UL/SPF* SIRIUS file. The SIRIUS file contains the code for the sample Janus Web Server, and Janus port definitions have default rules that call to this file.

Sirius Software has a strong commitment to backward compatibility with the *Sirius Mods*. This means that any User Language application (including *UL/SPF*) that uses the *Sirius Mods* will run correctly on subsequent versions of the *Sirius Mods*. It is, thus, always possible to upgrade the *Sirius Mods* without having to worry about upgrading *UL/SPF*. This is not to say that this is always a good idea, only that it is possible and that the installed version of a *UL/SPF* product will continue to run as it had before the *Sirius Mods* upgrade.

While the Janus family of products has a *UL/SPF* component, most of the critical code is actually in the *Sirius Mods* — object code enhancements to the *Model 204* nucleus. The *UL/SPF* component of the Janus family consists mostly of utilities, examples, and documentation. Because of this, the version number of a Janus product is generally considered to be the version of the *Sirius Mods* in which it is contained.

This document, the ***Janus Specialty Data Store Reference Manual***, assumes that a site is running *Sirius Mods* version 6.7 or later and has installed *UL/SPF* version 6.2 or later. Any documentation that requires a later version of the *Sirius Mods* or *UL/SPF* will be clearly marked to indicate this. For example, a JANUS DEFINE parameter that is only available in versions 7.7 and later of the *Sirius Mods* will have a sentence such as “This parameter is only available in version 7.7 and later of *Sirius Mods*” in its documentation. If a feature, \$function, command, or parameter is not indicated as requiring any specific version of the *Sirius Mods*, it can be assumed that it is available, as documented, in all versions of *Janus Specialty Data Store*; that is, all versions since version 6.7 of the *Sirius Mods* and version 6.2 of *UL/SPF*.

### **1.3 Related manuals**

As mentioned in “[Janus, the Sirius Mods, and UL/SPF](#)” on page 3, *Janus Specialty Data Store* requires the installation of both the *Sirius Mods* and *UL/SPF*. As such, the person responsible for the installation of *Janus Specialty Data Store* should refer to the ***Sirius Mods Installation Guide*** and the ***UL/SPF Installation and Maintenance Guide***. Also, the ***Sirius Messages Manual*** contains documentation on *Sirius Mods* error messages, so it might be useful to application programmers.

Also, as stated in [on page 3](#), *Janus Specialty Data Store* depends on the *Janus TCP/IP Base* product. Hence, there is much useful information in the ***Janus TCP/IP Base Reference Manual***.

## 1.4 Related products

The *Janus TCP/IP Base* must be installed to use *Janus Specialty Data Store*. This is the only other *Sirius Mods* product that must be installed in a *Model 204* region to use *Janus Specialty Data Store*.

If security is a concern, whether it be internet or intranet security, SSL (Secure Socket Layer) is the de-facto standard for providing encryption and validation security for web-based applications. The *Janus Network Security* product provides SSL support for *Janus Specialty Data Store* (as well as other products in the Janus family).

One of the convenient debugging features available with *Janus Specialty Data Store* is a TRACE facility which logs Janus request/response information to the *Model 204* journal. If you don't have good tools to view the journal, using it for debugging is a tedious process. AUDIT204 and ISPF provide some capabilities for viewing the journal, but they have many inherent shortcomings and inefficiencies. Because of this, it is **strongly** recommended that any site that installs *Janus Specialty Data Store* also install *SirScan*.

*SirScan* is a product in the *UL/SPF* family that facilitates the interactive extraction of journal information within the *Model 204* region. It does so via a user-friendly web browser or full-screen 3270 interface and low-level routines to provide efficient access to in-memory and on-disk journal buffers. *SirScan* can provide an order of magnitude improvement in debugging efficiency for non-terminal-related *Model 204* processes such as *Janus Specialty Data Store*, Horizon, BATCH2 and other Janus server applications.

## 1.5 System requirements

The current release of *Janus Specialty Data Store* requires the following components to run:

- Mainframe operating systems:
  - Any supported version of z/OS
  - z/VSE Version 4 or later or
  - CMS (releases currently supported by IBM) running under any supported version of z/VM
- *Model 204* Version 6 Release 1 or later
- One of the following mainframe TCP/IP implementations:
  - IBM TCP/IP for z/VM or z/OS
  - InterLink TCP/IP for MVS - Version 1.1 or later
  - TCP/IP for VSE (Connectivity Systems, Inc., Columbus, OH) - Version 1 Release 4.0 or later

## **2.1 Server Ports**

In order for a client application to communicate with a server application it must have a way to identify the server application on the network. Under the TCP/IP protocols the identity of a server has two parts. The first part identifies the machine on which the server runs. This part is called the machine's (or host's) IP address. The second part distinguishes the server application from other applications on the host. This part is called the port number.

A host's IP address is a 32 bit unsigned binary number that is displayed in "dotted" format, i.e. 198.242.244.33. To avoid having to refer to these types of addresses most networks have nameservers or names files that map names to IP addresses. That way a client application can connect to a host by a name (such as IBM3090) rather than an address.

A port number is a number from 1 to 65535 that is assigned to every server application that is available on a host. In the case of a Janus IFDIAL Server or a Janus Open Server, this port number is specified by the second parameter on the JANUS DEFINE command. Since this port number must be unique for the host, it is impossible to start (JANUS START) a port with a port number that matches a port number for any other application running on the same host. This includes any Janus port on the same or different Online. This also includes any other non-Janus server application. For example, port number 23 is almost always used by the telnet server. An attempt to start a Janus server for port number 23 will undoubtedly encounter a port in use situation and be unable to start. On a system with several local server applications, or more than one Online (maybe test and production) with several Janus ports, a simple strategy to keep port numbers from conflicting is to simply assign a range of ports to each Online. For example, port numbers 300-399 might be reserved for the test Online and port numbers 400-499 might be reserved for the production Online.

Sybase provides a way of mapping an application name to a host name (or address) and port number. This makes it possible to access a specific application by specifying only a single application name. This mapping is done through a mapping file called the interfaces file on Unix workstations, and through entries in WIN.INI under Microsoft Windows. For more information on this mapping refer to the Sybase DB/Library manuals.

## 2.2 Specialty Data Store or Omni Access Module

It can be extremely useful to be able to access any database using what is the lingua franca of data manipulation languages, namely SQL. This capability provides the following advantages:

- Off-the-shelf data query tools can be used against the database.
- It provides a simple way of loading data into workstation applications such as spreadsheets and word processors.
- Applications can be written with no (or at least relatively little) knowledge of the underlying database.

All of these advantages apply to the *Model 204* database indicating that there is a clear benefit to providing SQL access to *Model 204*.

But providing SQL access to *Model 204* is more than just a matter of adding the ability to convert SELECT statements to FIND statements. There are quite a number of other issues that need to be addressed :

- A SQL catalog must be provided that most workstation applications “understand”.
- A protocol must be provided for transporting structured request, table and row data over TCP/IP or other underlying transport protocols.
- Libraries must be provided on a wide variety of workstation platforms for accessing *Model 204* via SQL.
- These libraries must be integrated with off-the-shelf tools and development environments.
- Under Windows, at least, an ODBC interface must be provided.
- A full dialect of SQL must be supported and documented.

These issues present a monumental undertaking that makes the work of supporting some basic SQL statements pale in comparison. Fortunately, Sybase Inc., one of the leading vendors of SQL databases, provides a way of taking advantage of the equivalent work done for their SQL database to provide SQL access to other databases. Using either a Sybase Adaptive Server's OmniConnect facility or the older Omni SQL Server (or Gateway) as “middleware” a client can communicate with a *Model 204* database (and most other databases as well) exactly the way it communicates with a Sybase Adaptive Server (or Sybase SQL Server). The Sybase Adaptive Server OmniConnect facility provides a catalog and supports the standard Sybase protocols on behalf of *Model 204*. It translates SQL requests into a reasonable subset of SQL that *Model 204* can support efficiently, passes this simpler SQL on to *Model 204*, receives the results from *Model 204* and then uses these results to process more complex requests if necessary.

This facility can even make multiple heterogeneous back-end databases look like a single Sybase Adaptive Server to a client.

Because this facility makes *Model 204* and any other back-end database appear to be a Sybase Adaptive Server to a client, all tools and products that work with Sybase Adaptive Server will also work with *Model 204*. Because Sybase is one of the industry leading SQL database vendors this means that almost all workstation based tools will work with Sybase Adaptive Server and so with *Model 204* accessed via Sybase Adaptive Server's OmniConnect facility.

A Janus port defined as an SDS port acts as a *Specialty Data Store* to a Sybase Adaptive Server. It can also act as a *Generic Access Module* though *Generic Access Modules* are missing many transaction integrity and performance improvements that are available with *Specialty Data Stores*. *Generic Access Modules* were also sometimes called *Omni Access Modules* and *Specialty Data Stores* were also called *Open Access Servers* at some point. Because of these historical name issues, a Janus SDS port can also be defined as an OMNI port or an OAS port so that

```
JANUS DEFINE M204SQL 1047 SDS 10
JANUS DEFINE M204SQL 1047 OAS 10
JANUS DEFINE M204SQL 1047 OMNI 10
```

are functionally equivalent. Whether the port interacts with the Sybase Adaptive Server (or Sybase Omni SQL Server) as a *Specialty Data Store* or a *Generic Access Module* depends on how it is defined to the Sybase middleware. The *Janus Specialty Data Store* port must be defined to the Sybase server with the `sp_addserver` RPC, as in

```
sp_addserver M204SQL,sds
```

*Janus Specialty Data Store's* support for `sds` server types and the “SDS” and “OAS” keywords are only available on *Sirius Mods* version 5.4 and later. Before this release, *Janus Specialty Data Store* could only act as a *Generic Access Module* and was, in fact, called *Janus OmniSQL Access Module*.

A Sybase Adaptive Server accepts Transact-SQL statements and converts them to a subset of SQL that can be more easily processed by non-Transact-SQL based systems. This SQL subset is forwarded to the appropriate (based on Adaptive Server table definitions) *Specialty Data Store* or *DirectConnect*. If the required data is in a *Model 204* database, the request can be handled by a *Janus Specialty Data Store*.

Because *Model 204* data structures do not map naturally to SQL data structures, the mapping must be specified on a file/group, field, table and column specific basis. This means that the first step in providing Adaptive Server SQL access to *Model 204* data is to define *Model 204* to SQL mappings. This is done via the JANCAT subsystem. Once these definitions are made, the data can be made accessible to a Sybase Adaptive Server with the Adaptive Server's “create existing table” command. Appropriate “create existing table” commands can be generated automatically from the *Model 204* to SQL mappings in JANCAT with the “defgen” utility which comes with the Sybase Adaptive Server.

*Model 204* to SQL mappings are kept in a *Model 204* file normally called JANCAT. When an SDS port is started an *SDS catalog SDAEMON* is logged on. This SDAEMON copies the JANCAT data into virtual storage and CCATEMP. The SDS catalog SDAEMON remains logged on until the SDS port is stopped. This SDAEMON holds enqueues on all files and groups associated with tables defined as enqueued in JANCAT. The SDS catalog SDAEMON is also responsible for reloading JANCAT into virtual storage and CCATEMP when the JANUS RELOAD command is issued, or at subsequent user connections after tables are re-mapped for ports that have the AUTOLOAD switch turned on.

Using *Janus Specialty Data Store* involves several steps.

- Install the *Sirius Mods* that contain *Janus Specialty Data Store*. For instructions on doing this see the *Sirius Mods Installation Guide*.
- Map the appropriate *Model 204* files and groups into SQL tables using JANCAT.
- Install and configure a Sybase Adaptive Server (if this is not already done). Most of the documentation for Adaptive Server can be found on the web at <http://sybooks.sybase.com/as.html>. Adaptive Server or Adaptive Server Enterprise must be purchased from Sybase.
- Define a *Janus Specialty Data Store* port to *Model 204* using the JANUS DEFINE command. For more information see “[Janus Commands](#)” on page 13.
- Define the *Janus Specialty Data Store* port to Sybase Adaptive Server. This is done by adding the *Janus Specialty Data Store* name, IP address and port number to the *interfaces* file or *WIN.INI* depending on the operating system and then issuing an `sp_addserver` for the appropriate name on the Sybase Adaptive Server. The `sp_addserver` command must be followed by the name of the *Janus Specialty Data Store* port as entered in the *interfaces* or *WIN.INI* file. It is recommended but not necessary that this also match the name of the port on the “JANUS DEFINE” command. After the port name the `sp_addserver` command should have the port type which can either be “generic” or “sds”, the latter being strongly recommended (unless running *Sirius Mods* 5.3 or earlier in which case only “generic” is supported). The Sybase Adaptive Server command to define a *Janus Specialty Data Store* port called “M204SDS” would be, then

```
sp_addserver M204SDS,sds
```

- Run the “defgen” utility to download the *Janus Specialty Data Store* SQL table definitions to the workstation.
- Run the output of the “defgen” utility (a bunch of “create existing table” commands) back into the Sybase Adaptive Server.

After all these steps are completed, the *Model 204* data being handled by *Janus Specialty Data Store* can be accessed by any client exactly as if it were a Sybase Adaptive Server database.



The Janus command set (simply referred to as "Janus commands") consists of commands and subcommands that begin with the string **JAN**. The two Janus commands currently supported as of *Sirius Mods* version 6.3 are **JANUS** and **JANUSDEBUG**. For a full description of the Janus commands, see the *Janus TCP/IP Base Reference Manual*. The manual you are reading describes the commands that are specific to *Janus Specialty Data Store*.

You use Janus commands to:

- Define *Model 204* as a server on the TCP/IP network. Janus commands set port numbers for your Janus server applications and start, stop, and monitor Janus activity in the *Model 204* address space.
- Define remote servers to the *Model 204* client for access by *Janus Open Client* applications and *Janus Sockets* client applications, and define which remote hosts can establish connections with *Janus Specialty Data Store*, *Janus Open Server*, and *Janus Sockets*.
- Add security to Janus ports using Secure Sockets Layer (SSL) or Transport Layer Security (TLS) to provide encrypted communications.

Janus commands require the executing user to have System Manager privileges. Exceptions are the **JANUSDEBUG** command, which can be issued by any logged-in user, and commands that are executed through the **JANMAN** subsystem. **JANMAN** is an optional application subsystem that ships with Janus and is described in the *Janus TCP/IP Base Reference Manual*. As of *Sirius Mods* version 6.2, Janus commands can also be issued as operator commands (on the Online virtual console under VM) or as replies to the HALT message under OS/390.

Janus commands make use of the following wildcard characters:

- \* An asterisk represents any string of characters.
- ? A question mark represents any character.
- " A double quote escapes wildcard translation of the special character that follows it.

For example, the following command starts all Janus ports whose names begin with the string **BA** (like **BART**, **BARNEY**, **BALES**, **BAY**):

```
JANUS START BA*
```

The following command drains all Janus ports whose names are three characters long, beginning with BA (BAY, BAD, BAX, etc.):

```
JANUS DRAIN BA?
```

The following command starts all Janus ports whose names end in ? (WHODONEIT?, WHERESTHEBEEF?, WHAT?\_ME\_WORRY?, WHO\_YA\_GONNA\_CALL?, etc.):

```
JANUS START *"? 
```

### 4.1 JANUS command overview

The principal command of the Janus command set is the JANUS command, which consists of a set of mutually exclusive subcommands. To execute a subcommand, you specify it with the prefix **JANUS:** for example, JANUS DEFINE ..., JANUS STATUS ..., etc.

The following list shows the JANUS subcommands with a brief description of what they do. For more information about the subcommands not described in this manual, see the *Janus TCP/IP Base Reference Manual*.

Subcommand execution requires System Manager privileges, unless the command is executed through the JANMAN subsystem.

<b>ADDCA</b>	Adds a trusted certifying authority's certificate to a port.
<b>CHARSET</b>	Specifies the default character set.
<b>CLSOCK</b>	Specifies rules to allow a User Language program to access a CLSOCK port.
<b>CONFIGURATION</b>	Displays global configuration values.
<b>DEFINE</b>	Defines a Janus port.
<b>DEFINEIPGROUP</b>	Defines a grouping of IP addresses for web access control.
<b>DEFINEREMOTE</b>	Defines a remote server for <i>Janus Open Client</i> , and associates it with a Janus OPENSERV or SDS port.
<b>DEFINEUSGROUP</b>	Defines a grouping of user IDs for web access control.
<b>DELCA</b>	Deletes a trusted certifying authority's certificate from a port.
<b>DELETE</b>	Deletes a port definition.
<b>DELETEIPGROUP</b>	Deletes a grouping of IP addresses.

<b>DELETEREMOTE</b>	Deletes an association between a remote server and a Janus OPENSERV or SDS port.
<b>DELETEUSGROUP</b>	Deletes a grouping of user IDs.
<b>DISPLAY</b>	Displays Janus port definitions.
<b>DISPLAYCA</b>	Displays the contents of a trusted certifying authority's certificate.
<b>DISPLAYREMOTE</b>	Displays remoter server definitions.
<b>DISPLAYSOCK</b>	Displays CLSOCK and SRVSOCK port rules.
<b>DISPLAYWEB</b>	Displays WEBSERV port rules.
<b>DISPLAYXT</b>	Displays translate table definitions.
<b>DOMAIN</b>	Specifies the domain; used with IBM TCP/IP to resolve unqualified host names.
<b>DRAIN</b>	Prevents new connections to port and stops port when last connection is closed.
<b>FORCE</b>	Breaks all connections to port and stops port when last connection is closed.
<b>FTP</b>	Specifies Janus FTP Server processing rules.
<b>LANGUAGE</b>	Specifies default <i>Janus Open Server</i> language.
<b>LIMITS</b>	Displays the Janus connection limits for an Online.
<b>LOADXT</b>	Loads or reloads a translate table and, optionally, an entity translate table.
<b>NAMESERVER</b>	Specifies IP address and port number of the domain name server used with <i>Janus Sockets</i> CLSOCK applications and <i>Janus Open Client</i> applications; only used with the IBM TCP/IP interfaces.
<b>RELOAD</b>	Reloads the <i>Model 204</i> -to-SQL mappings from the JANCAT file for a <i>Janus Specialty Data Store</i> port.
<b>SRVSOCK</b>	Specifies rules that determine which SRVSOCK connections to allow.
<b>SSLSTATUS</b>	Displays SSL (Secure Sockets Layer) statistics for SSL ports.
<b>START</b>	Makes a port available for connections.

<b>STATUS</b>	Displays port status.
<b>STATUSCA</b>	Displays the status of a trusted certifying authority's certificate.
<b>STATUSREMOTE</b>	Displays status of remote servers.
<b>TCPLOG</b>	Stores all input and output streams to and from a port.
<b>TRACE</b>	Changes trace settings for a port or for specific IP addresses connected to a port.
<b>TSTATUS</b>	Displays thread utilization statistics.
<b>WEB</b>	Specifies <i>Janus Web Server</i> processing rules.

## 4.2 JANUS DEFINE

The JANUS DEFINE command is used to specify the characteristics of a *Janus Specialty Data Store* port as well as any other Janus port. It defines the usage of the named port as one of the following:

- Access by IFDIAL clients
- Open Server or Open Client connections
- Specialty Data Store access
- Web access
- FTP server connections
- Telnet server connections
- Generic Sockets usage — with the *Model 204* online either requesting (CLSOCK) or accepting (SRVSOCK) the connection
- Connection between the *Janus Debugger* or *Sirius Debugger* workstation GUI and programs being debugged in *Model 204*

For any except a CLSOCK or DEBUGGERCLIENT port, this subcommand associates a service with a TCP/IP port number.

Among the characteristics specified by JANUS DEFINE is whether the port will use Secure Sockets Layer (SSL) for encrypted communications.

`JANUS DEFINE portname portnum type maxcon other_parms...`

### JANUS DEFINE command syntax

Where each of the first four parameters is positional and required:

- portname** A 1- to 30-character name by which the port is identified. It is used on other JANUS subcommands, such as JANUS START and JANUS DISPLAY.
- portnum** The TCP/IP port number at which the service is available. *portnum* is the server port number, and it must be between 1 and 65535, inclusive. This number is used by client applications on the network when they require access to the *Model 204* server. The server port number must be unique on the host. Several “well-known” port numbers for various TCP/IP services (for example, 53 for nameserver) should be avoided. There is a discussion of server ports in the *Janus TCP/IP Base Reference Manual*.
- type** Port type. For *Janus Specialty Data Store* ports, specify SDS.
- maxcon** Maximum number of simultaneous active connections to be allowed on the port. This number must be less than or equal to the number of TCP/IP connections for which the site is licensed.

Before *Sirius Mods* version 6.8, the *maxcon* value had to be less than or equal to the number of sdaemons defined to the online. If you are defining multiple ports for your site, however, the sum of the *maxcon* connections you define is allowed to be greater than the number for which the site is licensed. In this case, *Janus Web Server* will automatically prevent any connection that would exceed the site license limit.

For *Janus Specialty Data Store*, note that a server-to-server connection requires an extra connection for the **site handler**. Thus, a single connection to a remote server would use two connections, while 10 connections to a remote server would use 11.

Under *Sirius Mods* version and later, restrictions on the allowed values for *maxcon* are no longer present, but licensed thread limits are still enforced at the time a connection is made.

You can use the JANUS TSTATUS command to view the thread usage and connection limits for your port, and you can use the JANUS LIMITS command to view similar information for your Online.

- other\_parms** A set of blank-delimited parameters that describe the characteristics of and processing to be performed on the port. These parameters are keywords, sometimes followed by values. They are all optional.

Once again, all the parameters allowed on the JANUS DEFINE command are documented in the *Janus TCP/IP Base Reference Manual*. For your convenience, only those applicable to port definitions used with *Janus Specialty Data Store* are described in the following subsections.

### 4.2.1 ALLOCC

This parameter indicates that input, output and request buffers are to be allocated when a connection is established and are to be freed when the connection is closed. If ALLOCC is not specified, all necessary buffers are allocated when the JANUS START command is executed and are kept until the port is stopped, after a JANUS DRAIN or JANUS FORCE command. All buffers are allocated above the line using space reserved by SPCORE.

### 4.2.2 AUDTERM

This parameter is used to control whether the server thread sends “non-compiler terminal output” to the audit trail. Compiler terminal output is always sent to the audit trail. Any terminal output sent to the audit trail is sent as RK lines.

AUDTERM specifies that terminal output **is sent** to the audit trail; NOAUDTERM (“NOAUDTERM” on page 22), which is the default port setting, specifies that (non-compiler) terminal output **is not sent** to the audit trail.

Note that some “print output” can be “captured” on a Sirius \$list, a Janus Socket, or a USE output stream, and thus it would not be sent as terminal output -- to the audit trail or anywhere else. For further description of terminal output, Starting with version 6.0, this parameter applies to all Janus “server” port types, and the default setting is NOAUDTERM. Prior to this, the parameter only applied to WEB ports, and the default setting was AUDTERM.

This introduces a small incompatibility. Starting with version 6.0, compared to earlier versions, any WEB port connection without an explicit AUDTERM or NOAUDTERM will probably generate fewer audit trail lines, as will any SDS or OPENSERV port. This should be a benefit, since most of this output is either uninteresting or already logged to the audit trail as ER, AD or MS lines. Logging these messages as RK lines as well is just a waste of journal space and I/O and makes application diagnosis and debugging from the audit trail more difficult because of the extra noise data. For WEB, OPENSERV, or SRVSOCK applications that wish to explicitly audit information, the User Language AUDIT statement should be used, not the PRINT statement.

### 4.2.3 AUTOLOAD

This parameter, indicates that the incore copy of Janus tables are to be reloaded at the first request subsequent to a table redefinition. This eliminates the need to issue JANUS RELOAD commands to reload table definitions after changes have been made. AUTOLOAD does have a few disadvantages, however. First, if several table definitions are being changed while SQL requests are being processed on a port, Janus might end up performing the reload process several times where issuing a JANUS RELOAD command at the end of all the changes would result in a single reload. Doing multiple reloads could be a significant expense if the number of JANCAT table definitions is

large. The other disadvantage of AUTOLOAD is that the reload doesn't happen until a request arrives on the port. This means that the user issuing the request could encounter a significant delay as the Janus port is reloaded. A JANUS RELOAD command before the request comes in would ensure that the table definitions would already be loaded so no extra delay would be encountered. Note that it is perfectly valid to issue a JANUS RELOAD command on an AUTOLOAD port.

#### 4.2.4 **BINDADDR xxx**

This parameter specifies the IP address to which the port will be bound, if the host (machine) on which *Model 204* is running supports multiple IP addresses. The IP address must be an IP address of the host. If BINDADDR is not specified, the port binds the port number for all IP addresses associated with the host; that is, it can be accessed via any IP address associated with the host.

This parameter only really makes sense on a host with more than one IP address. For example, if a host on which an Online is running has IP addresses 198.242.244.47 and 198.242.244.130, a **BINDADDR 198.242.244.47** specification indicates that the port can only be reached through IP address 198.242.244.47.

This parameter is especially useful for allowing a single mainframe host or even an Online to act as more than one web server without the inconvenience of having port numbers on URLs. This can be done because there can be multiple port 80's (the default web port number) on the host, each accessed by its indicated BINDADDR. The separate IP addresses could, in turn, be associated with different DNS host names even though these separate names refer to the same underlying machine.

Note that there is not likely to be much, if any, performance benefit to having multiple Janus ports with the same port number but different BINDADDRs in the same Online. There might certainly be, however, some organizational advantages to running such a configuration.

#### 4.2.5 **BFSIZE xxx**

This parameter specifies the size of the TCP/IP input and output buffers. The default is 4096 for IBSIZE and 8192 for OBSIZE. BFSIZE is a shorthand way of specifying both IBSIZE and OBSIZE when their sizes are the same.

#### 4.2.6 **CMD 'xxx'**

This parameter specifies the *Model 204* commands to be executed after the files and groups specified in the OPEN parameter ([“OPEN list” on page 23](#)) are opened. Multiple commands must be separated by the word “AND”, and any command that contains blanks must be enclosed in quotes. Multiple commands in the CMD clause are only supported in version 6.0 and later of *Sirius Mods*.

CMD may span more than one line — continued with a hyphen (-) — but the total length of commands plus one overhead byte per command cannot exceed 255 bytes.

For an OPENSERV, SRVSOCK, or TNSERV port, the commands specified by CMD specify the processing performed for each connection to the port. If the user logs off and logs on again during the same connection, the CMD command(s) are not executed. Because of this, using AUTOSYS is probably preferable to using CMD parameters for most TNSERV applications.

For SDS ports, the commands are executed before the port begins acting as a Specialty Data Store. It is strongly recommended that this command be used mainly to set user table sizes and user parameters for SDS ports. This might be necessary because *Janus Specialty Data Store* might have very different table size requirements than other applications running on an sdaemon.

For WEB ports the commands specified by CMD are executed after all rules are executed except the ON rules. The specified commands can be used to invoke an APSY subsystem when using the Janus Web UL API or to reset UTABLEs and other parameters.

Examples of some valid CMD clauses:

```
JANUS DEFINE MYWEB 80 WEBSERV 10 CMD WEBAPSY
JANUS DEFINE SDS204 1777 SDS 20 -
      CMD 'R MCPU 5000' AND 'UTABLE LQTBL 1000' -
      AND 'R PROMPT 16'
JANUS DEFINE OPENXXX 1234 OPENSERV 15 -
      OPEN FILE OPENPROC CMD 'R PROMPT 16' AND -
      OPENAPSY
```

### 4.2.7 FDWOL

This parameter specifies that all FINDs done by the *Janus Specialty Data Store* are to be done as FIND WITHOUT LOCKS. This can minimize the impact of *Janus Specialty Data Store* applications on 3270 applications and reduce record locking conflicts caused by large *Janus Specialty Data Store* request. On the other hand, FDWOL could cause severe *Model 204* errors especially in reuse record number files. The default for SDS ports is no FDWOL.

### 4.2.8 IBSIZE xxx

This parameter specifies the size of the TCP/IP input buffer. The default is 4096, the minimum is 512, and the maximum is 65534 (prior to version 5.2 the maximum was 32767). There is one input buffer used for each connection.

A larger input buffer size provides better CPU performance in both *Model 204* and the TCP/IP address space at the expense of more virtual (and real) storage. Generally, the size of the input buffer has an impact only on a port being used for *Janus Open Client* or *Janus Sockets* connections or on a *Janus Web Server* port used for file uploads.

#### 4.2.9 JANCAT xxx

This parameter specifies the name of the file that contains the *Model 204* to SQL mappings. These mappings are generated using the JANCAT subsystem. The default JANCAT file is JANCAT.

#### 4.2.10 MASTER

This parameter specifies that this is the default port for *outgoing* connections to remote servers. A single MASTER port will serve as the access route to multiple external server address spaces. The servers may be other *Model 204* servers or may be Sybase/Microsoft servers that are to receive *Janus Open Client* function calls.

Users accessing the *Model 204* address space over an OPENSERVICES port will use the same port they came in on for any outgoing *Janus Open Client* connections. Users accessing the *Model 204* address space with other threads (for example, 3270 or web based applications) must have a MASTER port defined in order for the *Janus Open Client* functions to establish connections to other address spaces.

Note that the port number is irrelevant for outgoing purposes, though it must still be specified.

Multiple ports may be DEFINEd and STARTed with the MASTER parameter specified, but the one used in any particular instance will not be predictable.

#### 4.2.11 MAXCURS xxx

This parameter specifies the maximum number of cursors that can be opened for a single *Janus Specialty Data Store* connection. The default for MAXCURS is 5.

A Sybase Adaptive Server or Omni SQL Server will open multiple cursors on a connection to an SDS thread. Typically, even for fairly complex requests the number of simultaneously open cursors will not exceed 3 so that the default of 5 should be fine. If a Sybase Adaptive Server attempts to open more cursors than is allowed by MAXCURS, the request will fail.

The cost of having a high MAXCURS is the storage for each possible cursor is allocated either at port start time for each thread or at connection time for ALLOCC ports. The storage required for each cursor is about 64 bytes so that on a 40 thread port with MAXCURS set to 5, the virtual storage used for cursor blocks would be 64\*5\*40 or 12,800 bytes.

### 4.2.12 **MAXSAVE xxx**

This parameter specifies the maximum number of compiled SQL requests to save in CCATEMP. When the Specialty Data Store receives an SQL request from the Sybase Adaptive Server it parses the request, translates it into User Language, compiles the User Language into quads and finally evaluates the compiled User Language. When MAXSAVE is greater than 0, compiled requests are saved and future identical requests (except for constants) can simply load the compiled requests from CCATEMP (much like an APSY load), bypassing the translation and compilation steps. The default for MAXSAVE of 16 should be adequate for most situations, ensuring that frequently run requests will perform well without wasting CCATEMP and I/O bandwidth for infrequently run requests.

### 4.2.13 **NOAUDTERM**

This parameter is used to control whether the server thread sends “non-compiler terminal output” to the audit trail. Compiler terminal output is always sent to the audit trail. Any terminal output sent to the audit trail is sent as RK lines.

NOAUDTERM, which is the default port setting, specifies that (non-compiler) terminal output **is not sent** to the audit trail; AUDTERM specifies that terminal output **is sent** to the audit trail.

Note that some “print output” can be “captured” on a Sirius \$list, a Janus Socket, or a USE output stream, and thus it would not be sent as terminal output -- to the audit trail or anywhere else. For further description of terminal output, Starting with version 6.0, this parameter applies to all Janus “server” port types, and the default setting is NOAUDTERM. Prior to this, the parameter only applied to WEB ports, and the default setting was AUDTERM.

This introduces a small incompatibility. Starting with version 6.0, compared to earlier versions, any WEB port that does not specify either AUDTERM or NOAUDTERM will probably generate fewer audit trail lines, as will any SDS or OPENSERV port. This should be a benefit, since most of this terminal output is either uninteresting or already logged to the audit trail as ER, AD or MS lines. Logging these messages as RK lines as well is just a waste of journal space and I/O and makes application diagnosis and debugging from the audit trail more difficult because of the extra noise data. For WEB, OPENSERV, or SRVSOCK applications that wish to explicitly audit information, the User Language AUDIT statement should be used, not the PRINT statement.

### 4.2.14 **NOUPCASE**

This parameter indicates that no client data is to be converted to upper case. By setting NOUPCASE the userid and password must be specified by the client in the correct case (probably upper case). Note that it is possible to have lower case userids and passwords in *Model 204*. For example, the userids HOMER, homer and Homer would

be treated as three separate userids by *Model 204*. The NOUPCASE parameter simplifies the interaction between clients where names tend to be in lower case or case-insensitive and *Model 204* where they tend to be in upper case.

The NOUPCASE parameter is the opposite of UPCASE. The default is for all ports to have UPCASE set.

#### 4.2.15 **OBSIZE xxx**

This parameter specifies the size of the TCP/IP output buffer. The default is 8192, the minimum is 512, and the maximum is 65534 (prior to version 5.2 the maximum was 32767). There is one output buffer used for each connection.

A larger output buffer size provides better CPU performance in both *Model 204* and the TCP/IP address space at the expense of more virtual (and real) storage.

#### 4.2.16 **OMNIACCT xxx**

This parameter specifies the *Model 204* ACCOUNT for the SDS catalog sdaemon for the port. If no OMNIACCT is specified the ACCOUNT for the user executing the JANUS DEFINE command is used. This is a synonym for SDSACCT.

#### 4.2.17 **OMNIUSER xxx**

This parameter specifies the *Model 204* user ID for the SDS catalog sdaemon for the port. If no OMNIUSER is specified the ID of the user executing the JANUS DEFINE command is used. This is a synonym for SDSUSER.

#### 4.2.18 **OPEN list**

This parameter specifies the name of one or more *Model 204* files or groups to be opened when a server session is initiated.

If you specify multiple files or groups in an OPEN clause, they must be separated by an AND keyword. You can also specify individual file open privileges; if not, they default to X'0221'. Multiple files or groups and explicitly specified privileges in the OPEN clause are supported only in version 6.0 and later of *Sirius Mods*.

The syntax of each file or group specification is:

```
[FILE | GROUP] name [ [WITH] privs]
```

The first file or group listed in an OPEN clause is set as the default file or group context for the thread. If neither the keyword FILE nor the keyword GROUP is specified, OPEN

looks first for a permanent group, then for a file, to open. It does not look for a temporary group, since one cannot yet exist at open time.

You can use the CMD parameter to specify a command to execute just after a file or group opens. If the CMD parameter specifies an INCLUDE command, the included procedure is assumed to come from the first file or group specified in the OPEN clause.

Examples of valid OPEN clauses follow:

```
JANUS DEFINE WEBXXX 80 WEBSERV 20 -
        OPEN WEBPROC AND -
        FILE DATAPROC WITH X'0761'
JANUS DEFINE OPENDOOR 1234 OPENSERV 40 -
        OPEN GROUP DOORPROC AND -
        FILE DOORDATA X'BFFF' -
        CMD 'I DRIVER'
```

### 4.2.19 PRELOGINUSER userid

This parameter indicates the userid under which pre-login processing runs. Pre-login processing is that which occurs before a user login.

This parameter is only available in version 6.0 and later of the *Sirius Mods*. Before this version, these were all true about processing that occurred before user login:

- It ran under “NO USERID”.
- It was not visible with *SirMon*, the MONITOR command, or LOGWHO.
- It was not BUMP'able.

After version 6.0 of the *Sirius Mods*, pre-login processing runs under the default userid of “NO USERID” or under the userid specified by the PRELOGINUSER parameter; it is visible to *SirMon*, the MONITOR command, and the LOGWHO command; and it is BUMP'able.

On many port types, much processing can take place before a thread is actually logged on to a user. The PRELOGINUSER parameter can be useful in helping distinguish users in pre-login processing on different ports.

### 4.2.20 RAWINPUT

This parameter tells *Janus Web Server* to save the raw input stream for an HTTP POST, regardless of the mime type set by the client in the `content-type` header. This has two basic advantages:

1. The raw input content for an HTTP POST is always available to *Janus Web Server* applications (via `$web_input_content`) regardless of the content-type. This could be useful for debugging, or perhaps for logging, input content.

2. It is possible for *Janus Web Server* to interact correctly with clients that don't set the mime type, regardless of what content they send. Prior to the availability of RAWINPUT, if a client sent, say, XML data, but it did not set the content-type, *Janus Web Server* would assume that the content was `application/x-www-form-urlencoded` (form POST) encoded. If after it read some of the content, *Janus Web Server* discovered that it was not HTML form data, it was too late: the request had to be rejected for having an invalid format.

With the RAWINPUT parameter set, however, *Janus Web Server* proceeds as follows:

- a. It loads the input content into CCATEMP.
- b. If the mime type is set to `application/x-www-form-urlencoded`, or if it is not set at all, *Janus Web Server* determines if the input has the `application/x-www-form-urlencoded` format.
- c. If the format is *not* `application/x-www-form-urlencoded`, the request is *not* rejected, and the *Janus Web Server* application can still access the data.

This parameter is only available in version 6.7 and later of *Sirius Mods*.

#### 4.2.21 RAWINPUTONLY

RAWINPUTONLY indicates that, regardless of the POST data content-type set by the client, *Janus Web Server* should do both of the following:

- Save the raw input stream of an HTTP POST.
- Refrain from parsing the input content into form fields.

RAWINPUTONLY is very similar to the RAWINPUT port definition parameter (“RAWINPUT” on page 24), except that:

- RAWINPUTONLY can be an ON rule parameter, so it can be set for specific URLs.
- RAWINPUT does not prevent *Janus Web Server* from trying to parse the form parameters, if the `content-type` for the POST is set to `application/x-www-form-urlencoded` or `multipart/form-data`. RAWINPUTONLY prevents this parsing, so it protects *Janus Web Server* applications from errors in this parsing. These errors include invalid-form-data errors and request-buffer-full errors.

For more information about RAWINPUTONLY processing, see the ***Janus Web Server Reference Manual***.

This parameter is only available in version 6.8 and later of *Sirius Mods*.

### **4.2.22 RBSIZE xxxx**

This parameter specifies the Janus RPC or Request buffer size. On a Web port, the Janus request buffer holds browser request information such as header data, cookies and form data. On other port types, it holds RPC input and output parameters.

If a Janus server thread is started, and the data sent by the client program requires more space for its parameters than is allocated by RBSIZE, the connection to the client is broken and message MSIR.0154 is issued. If a *Janus Open Client* program sends or receives parameters too long for RBSIZE, the connection to the client is broken and message MSIR.0186 is issued.

The Janus RPC buffer is also used by Open Server programs to contain column descriptions set by the \$SRV\_BIND function; if the buffer is too small, \$SRV\_BIND returns an error code indicating so.

The default for RBSIZE is 4096. The maximum is 65534.

### **4.2.23 SDSACCT xxx**

This parameter specifies the *Model 204* ACCOUNT for the SDS catalog sdaemon for the port. If no SDSACCT is specified the ACCOUNT for the user executing the JANUS DEFINE command is used. This is a synonym for OMNIACCT.

### **4.2.24 SDSUSER xxx**

This parameter specifies the *Model 204* user ID for the SDS catalog sdaemon for the port. If no SDSUSER is specified the ID of the user executing the JANUS DEFINE command is used. This is a synonym for OMNIUSER.

### **4.2.25 SSL**

The SSL parameter indicates that communications on this port should be encrypted using *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) support. The parameter has the following mutually exclusive options:

#### **SSL procfile procname**

Identifies the file (typically JANSSL) and procedure that contain the certificate to be presented to clients on server ports and to the server on CLSOCK ports.

**SSL \*** Presents to the client or server the “self-signed certificate” provided for your site by Sirius Software.

**SSL 0** Indicates for CLSOCK ports that, although the connection is encrypted, the client is not to provide a certificate to the server if requested.

Server certificates are required to establish an encrypted connection, but client certificates are optional and are not used at all by many secured servers.

Certificates and authentication are described further in the *Janus Network Security Reference Manual*.

Other optional DEFINE command parameters used in conjunction with the SSL parameter include:

- For server sockets:  
SSLBSIZE, SSLCIPH, SSLCLCERT/SSLCLCERTR,  
SSLIBSIZE, SSLOBSIZE, SSLPROT, SSLSES
- For client sockets:  
SSLOPT
- For both types of sockets:  
SSLCACHE, SSLMAXAGE, SSLMAXCERTL, SSLUNENC

Other JANUS commands useful for SSL ports and described in the *Janus TCP/IP Base Reference Manual* include:

- For ports that authenticate incoming certificates:  
ADDCA, DELCA, DISPLAYCA, STATCA
- For monitoring a port's SSL activity:  
SSLSTAT

*Janus Web Server* \$functions useful for SSL applications and described in the *Janus Web Server Reference Manual* include:

\$WEB\_CERT\_INFO, \$WEB\_CERT\_LEVELS, \$WEB\_CIPHER,  
\$WEB\_PROTOCOL, \$WEB\_SECURE

#### 4.2.26 **SSLBSIZE xxxx**

This tuning parameter specifies the size of the input buffer used for reading encrypted data for an SSL port. An SSL port is a Janus port whose definition includes an SSL parameter (“SSL” on page 26) setting, which indicates that communications on this port may be encrypted using *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) support.

Under version 6.0 and later of the *Sirius Mods*, the `SSLBSIZE` parameter also specifies the size of the SSL output buffer. To set the input and output buffer sizes independently, you use the `SSLIBSIZE` and `SSLOBSIZE` parameters.

The default for `SSLBSIZE` is 4096 bytes; the minimum and maximum values are 1024 and 32767, respectively.

If you set `SSLBSIZE` greater than the SSL specification maximum buffer size of 16000, the port's input buffer size is set to the `SSLBSIZE` value, but the output buffer size is set to 16000 bytes. Setting the input buffer greater than 16000 bytes might be necessary if the port will have connections with SSL implementations that don't fully conform to the SSL specification. For more information about buffer sizing and about Janus handling of oversized packets, see ([“SSLIBSIZE xxxx” on page 30](#)) and ([“SSLOBSIZE xxxx” on page 32](#)).

#### **4.2.27 SSLCACHE xxxx**

This parameter specifies the number of entries in virtual storage to be allocated for caching information related to this port's SSL sessions. A Janus port whose definition includes an SSL parameter ([“SSL” on page 26](#)) setting supports *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) encrypted sessions.

The SSL cache helps limit the CPU overhead of establishing an SSL session. It does not reduce the effectiveness of security, but it does reduce the overhead at the cost of a relatively small amount of virtual storage.

SSL sessions can persist for a length of time determined by either the client or server. *Janus Network Security* limits the life-span of SSL V2 connection sessions to the lesser of 2 minutes or the value of `SSLMAXAGE` ([“SSLMAXAGE xxx” on page 31](#)), and it limits SSL V3 and TLS connections to 1440 minutes (24 hours). For most sites, the default `SSLCACHE` should be sufficient.

Each session requires approximately 512 bytes per entry to cache session related information. A further `SSLMAXCERTL` ([“SSLMAXCERTL xxx” on page 32](#)) bytes are required to hold server certificates for `CLSOCK` ports, or to hold client certificates for Janus server ports that request them by including `SSLCLCERT` or `SSLCLCERTR` ([“SSLCLCERT and SSLCLCERTR” on page 29](#)).

If the `SSLCACHE` value is too small, and a larger than anticipated number of users attempt to access an SSL-secured port, entries in the cache are removed on a least-recently-used basis. This may lead to greater overhead for re-execution of the CPU intensive initial public-key/private-key encryption/decryption operations. The indicator that the `SSLCACHE` value is not large enough to hold all the contemporaneous SSL sessions is a non-zero value in the “SesNF” column of the `JANUS SSLSTAT` command result. This is not necessarily problematic as long as the `SesNF` value is relatively small, because it is not unreasonable to suffer an occasional lost session in order to reduce virtual storage.

**Note:** SSLCACHE is specified in *entries*, and the default SSLCACHE allocation is the number of storage entries required for 16 times the number of threads defined on the port. So by default, 10 threads would result in 160 entries; at 512 bytes per entry, this would require 81,920 bytes of virtual storage. 100 threads would require 819,200 bytes.

The default SSLCACHE value is likely to be excessively large for CLSOCK ports that only connect to a single or to a few servers. All CLSOCK connections to a particular server use the same SSL session regardless of how many different threads initiate connections.

#### 4.2.28 SSLCIPH xxx

This parameter lets you limit the stream ciphers (encryption algorithms) that this port offers for SSL connections. A Janus port whose definition includes an SSL parameter ([“SSL” on page 26](#)) setting supports *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) encrypted connections.

Typically, SSLCIPH is allowed to default to 0: all the Janus-supported ciphers are available, and the cipher that is ultimately used depends on the outcome of the handshake negotiation with the client that seeks the service at this port. The negotiation selects the strongest available cipher that the client can support.

However, to make only a subset of the server ciphers available, you can specify SSLCIPH followed by the (bitmask) value that selects the subset. For example, SSLCIPH 2 indicates that only strong RC4 encryption is available.

Currently, these ciphers are supported:

- 1 RC4 bulk cipher with MD5 digest algorithm with 40 bits of the 128 bit RC4 key transmitted encrypted, the rest transmitted "in the clear" (unencrypted). This is considered a moderately strong encryption algorithm and is available on virtually every client implementation of SSL.
- 2 RC4 bulk cipher with MD5 digest algorithm with all 128 bits of the RC4 key transmitted encrypted. This is considered a very strong encryption algorithm but is only available on clients that have been specially configured to support this cipher. This encryption level is not available for export from the United States.

#### 4.2.29 SSLCLCERT and SSLCLCERTR

These parameters specify that an SSL server port will request an SSL certificate from the client. An SSL port is a Janus port whose definition includes an SSL parameter ([“SSL” on page 26](#)) setting, which indicates that communications on this port may be encrypted using *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) support.

If the client does *not* present a certificate when requested:

- SSLCLCERT specifies that normal processing should continue.
- SSLCLCERTR specifies either of the following:
  - The connection should be closed with no further processing (and “MSIR.0646: Error requesting client certificate - client did not have required certificate” is journaled).
  - Processing continues to run the SSLNOCERTERR exception handler, if this is a WEBSERV port and an ON SSLNOCERTERR clause is part of the port definition. For information about this exception handler, see

To verify a certificate that is passed by a client, you must first have added to the port one or more CA-signed certificates by using the JANUS ADDCA command, which is described in the *Janus TCP/IP Base Reference Manual*.

When a client presents a certificate, that certificate is available to User Language code via \$WEB\_CERT\_LEVELS and \$WEB\_CERT\_INFO on WEBSERV ports, and it is available via \$SOCK\_CERT\_LEVELS and \$SOCK\_CERT\_INFO on SRVSOCK ports.

The following example shows a web server SSL port definition that specifies the SSLCLCERTR parameter, JANUS ADDCA commands that are needed to store CA-signed certificates to authenticate the client certificate, and a rule that specifies the ONSSLCERTERR exception handler for cases where the client does not present a certificate:

```
JANUS DEFINE CLCERTWEB 9733 WEBSERV 10 HTTPVERSION 1.1 -  
      SSL JANSSL TM2008.PKEY SSLCLCERTR  
  
JANUS ADDCA CLCERTWEB MYPROC SECURESE.CERT  
JANUS ADDCA CLCERTWEB MYPROC THAWTE.CERT  
JANUS ADDCA CLCERTWEB MYPROC VERIJUNK.CERT  
  
JANUS WEB CLCERTWEB ON SSLNOCERTERR OPEN FILE MYPROC -  
      CMD 'INCLUDE MISSING_CERTIFICATE_ERROR'
```

The SSLCLCERT and SSLCLCERTR parameters are only available in version 6.0 and later of *Sirius Mods*.

#### **4.2.30 SSLIBSIZE xxxx**

This parameter specifies the size of the SSL input buffer to be used on SSL ports. An SSL port is a Janus port whose definition includes an SSL parameter (“SSL” on page 26) setting, which indicates that communications on this port may be encrypted using *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) support.

Before version 6.0 of the *Sirius Mods*, the size of the SSL input buffer was specified with the SSLBSIZE parameter (“SSLBSIZE xxxx” on page 27) because the SSLIBSIZE parameter was not available.

Technically, the maximum “legal” SSL buffer size is 16000, but it may be necessary to use a larger input buffer if there will be connections with SSL implementations that don't fully conform to the SSL specification. If an application tries to send an SSL packet larger than SSLIBSIZE to a Janus SSL port, the connection will be broken and an error written to the audit trail (MSIR.0386 SSL INPUT MESSAGE TOO LONG - INCREASE SSLBSIZE). The other side of the SSL connection will not receive this error message or any other indication of why the connection was broken. There will be no effect on other users on the same port.

The default for SSLIBSIZE is 4096, and the minimum and maximum allowable values are 1024 and 32767, respectively.

For WEBSERV ports that are used for file uploads (HTTP PUT or form-based uploads), it will probably be necessary to set SSLIBSIZE to at least 16000, because most browsers will send SSL packets that are as large as possible. For most other applications, the SSLIBSIZE default is probably sufficient, though web applications that POST very large forms might require a slight increase of SSLIBSIZE.

#### 4.2.31 SSLMAXAGE xxx

This parameter specifies the maximum number of minutes that an SSL session is to be maintained. A Janus port whose definition includes an SSL parameter (“SSL” on page 26) setting supports SSL (Secure Sockets Layer) or TLS (Transport Layer Security) encrypted sessions. The discussion of this SSLMAXAGE parameter uses “SSL” to refer to SSL or TLS.

An SSL session is a series of SSL connections that are made using a single “master secret” shared by the SSL client and server. To set up an SSL session, the master secret must be exchanged using computationally expensive public-key/private-key encryption/decryption. SSL sessions are a way of reducing the overhead of SSL by reducing the number of public-key/private-key encryption/decryption operations.

The SSLMAXAGE default is 1440 (24 hours), which is the specified maximum life-span of an SSL V3 or a TLS session. The maximum life-span of an SSL V2 session is 2 minutes, so larger values of SSLMAXAGE are ignored for SSL V2 sessions. Before version 6.0 of the *Sirius Mods*, only SSL V2 was supported and the SSLMAXAGE parameter was not available, so an implicit SSLMAXAGE value of 2 was always used.

The 24-hour life-span of SSL V3 and TLS sessions is generally considered “safe”, but if even greater security is required, a smaller SSLMAXAGE can be specified. Setting SSLMAXAGE to 0 forces a new session for every request, which forces a public-key/private-key encryption/decryption operation for every connection. This might be useful for benchmarking the overhead associated with the public-key/private-key

operations. The JANUS SSLSTAT command can provide useful information in monitoring the efficacy of SSL session caching.

#### **4.2.32 SSLMAXCERTL xxx**

For a Janus port defined (by the `SSL` parameter) to support encrypted connections, this parameter indicates the number of bytes of virtual storage to be allocated to hold incoming certificates presented for authentication. Authentication verifies (or not) the certifying authority signature on the incoming certificate. Such a certificate may be:

- A server certificate sent in reply to a CLSOCK port.
- A client certificate sent in reply to a WEBSERV, SRVSOCK, OPENSERV, or SDS port that has the SSLCLCERT or SSLCLCERTR parameter in its definition.

Since incoming certificates are cached, SSLMAXCERTL bytes are allocated for each SSL session in the cache, the size of which is determined by the explicit or implicit setting of the SSLCACHE parameter (“[SSLCACHE xxxx](#)” on [page 28](#)).

The default SSLMAXCERTL size is 1024, which should be large enough to hold most certificates received from clients or servers. The minimum and maximum SSLMAXCERTL values are 256 and 32767, respectively. It is unlikely that any incoming certificate will be smaller than 512 bytes, and it is extremely unlikely that an incoming certificate will be larger than 2048 bytes. If an incoming certificate is larger than SSLMAXCERTL, an error message is logged to the audit trail and the connection is closed.

The SSLMAXCERTL parameter is only available in version 6.0 and later of the *Sirius Mods*.

#### **4.2.33 SSLOBSIZE xxxx**

This parameter specifies the size of the SSL output buffer to be used on SSL ports. An SSL port is a Janus port whose definition includes an SSL parameter (“[SSL](#)” on [page 26](#)) setting, which indicates that communications on this port may be encrypted using *Janus Network Security* SSL (Secure Sockets Layer) or TLS (Transport Layer Security) support.

Before version 6.0 of the *Sirius Mods*, the size of the SSL output buffer was always 256 bytes and the SSLOBSIZE parameter was not available.

There is little or no performance benefit to using large SSL output buffers, because the amount of work associated with creating an SSL output packet is almost directly proportional to the size of the packet. Typically, it is sensible to use the default SSLOBSIZE of 4096, or even to make it smaller to save on memory.

The default for SSLOBSIZE is 4096, and the minimum and maximum allowable values are 1024 and 16000, respectively.

For *Model 204* to *Model 204* applications, the SSLOBSIZE on each side must be less than or equal to the SSLIBSIZE (“SSLIBSIZE xxxx” on page 30) on the other side.

#### 4.2.34 SSLPROT xxx

This parameter lets you specify the degree of SSL-like encryption available at this port. *Janus Network Security* currently supports two Secure Socket Layer (SSL) protocols (SSL V2 and SSL V3) and the Transport Layer Security (TLS) protocol, an extension to SSL V3 but developed by the IETF Internet standards group.

During the negotiation for a connection to or from this port, Janus will offer the most secure protocol available, then, if necessary, will fall back to the next lower one available, and so on. The SSLPROT parameter lets you explicitly disallow one or more protocols from the negotiation.

SSLPROT is a bitmask parameter whose main values are:

- X'01'** SSL, V2 support. This is less secure than SSL V3 or TLS.
- X'02'** SSL, V3 support. This is less secure than TLS.
- X'04'** TLS, V1 support.
- X'07'** The default. SSL V2, SSL V3, and TLS are available. Janus will try for them in the order: TLS, SSL V3, SSL V2.

A typical reason for explicitly specifying an SSLPROT value is to require a more secure connection for a port. If a client attempts to connect to a Janus server port using a protocol explicitly disallowed by SSLPROT, the connection is immediately broken, except for WEBSERV ports where the SSLPROTOCOLERR exception handler will be run if available.

Janus CLSOCK ports will attempt to connect under the most secure protocol available, and will fall back to the next-most secure protocol available; if less-secure protocols are disallowed by SSLPROT, the connection attempt will fail.

Before version 6.0 of the *Sirius Mods*, only SSL V2 was supported and the SSLPROT parameter was not available.

#### 4.2.35 SSLUNENC

This parameter indicates that an unencrypted private key is being used in the certificate specified by the SSL parameter (“SSL” on page 26) on this Janus server port definition.

As of *Sirius Mods* version 6.2, this parameter is obsolete — as of this version, *Janus Network Security* automatically determines whether or not the private key is encrypted, and if not, prompts for a password. A corrupted private key procedure could lead *Janus Network Security* to believe that the private key must be encrypted, and so result in a password prompt.

Regardless of the *Sirius Mods* version, the use of unencrypted private keys is discouraged.

Before *Sirius Mods* version 6.2, SSLUNENC must have been specified on a port definition if an unencrypted private key was used. Otherwise, the JANUS START command for an SSL-secured port would prompt for a password (technically, a seed for the encryption algorithm) to use to decrypt the private key. Any data, or even a null value, entered for the password will incorrectly be used in an attempt to decrypt the private key (rendering the key unusable), and the START will fail.

Similarly, if an encrypted private key **is** used in the certificate specified on the SSL parameter, the SSLUNENC parameter **must not** be specified. Specifying SSLUNENC will prevent password prompting for that key, thus bypassing decryption of the private key (rendering it unusable), and causing the START to fail.

The certificate and private key generation process is described further in the *Janus Network Security Reference Manual*.

### 4.2.36 TCPKEEPALIVE

This parameter specifies that connections on the port should use TCP keepalives. TCP keepalives request that the TCP stack send periodic “keepalive” packets to the communications partner to see if it is still there. The time interval between these packets, which cannot be set by Janus, is set in the TCP/IP stack configuration. For example, with the IBM stacks, the keepalive interval is set in the TCPCONFIG INTERVAL parameter for BPX (IBM Communications Server) and in the KEEPALIVEOPTIONS INTERVAL parameter for VM TCP/IP.

In some sense, the term “keepalive” is a misnomer — keepalive packets that are not responded to cause a connection to be closed, so keepalives actually cause connections to be closed faster than they might be otherwise.

TCPKEEPALIVE probably only makes sense for ports where connections are held open for long periods of time. TNSERV ports are the most likely candidate. For these ports, TCPKEEPALIVE might be useful for two reasons:

1. It can detect connections lost due to a client failure (say, a turned-off workstation), reducing threads wasted for connections to lost clients.
2. It can reassure certain routers, especially those doing network address translation (NAT) that the connection is still active. Some routers will stop routing packets for

connections on which no activity is seen for some period of time. Keepalives ensure that there is periodic activity on a connection, even if there is no user interaction. Of course, for this to be successful, the TCP/IP stack's keepalive interval must be less than any applicable router's inactivity timeout. For this particular application, keepalives live up to their name.

Since the TCP/IP stack does the keepalives, the overhead in *Model 204* for setting this parameter is virtually zero.

This parameter is only available in *Sirius Mods* version 6.9 and later.

#### 4.2.37 **TIMEOUT xxxx**

This parameter specifies the number of seconds of inactivity after which clients connected to this port will be disconnected. The default for TIMEOUT is 0, which means that connections never time out.

**For WEBSERV ports** Browser requests never involve waits on user input so the TIMEOUT parameter for WEBSERV ports involves terminating connections when network response is **extremely** slow or cases where the client workstation has been turned off before a response is received from *Janus Web Server*. Because of this, TIMEOUT can be set fairly aggressively for WEBSERV ports. A value of 60 (seconds) would be reasonable.

**For all other port types** The TIMEOUT value should reflect the fact that a connection might require user input waits.

#### 4.2.38 **TRACE xxx**

This parameter specifies the initial TRACE setting for the port. The TRACE setting controls what Janus-related trace information is logged to the audit trail. The port TRACE setting can be overridden by the JANUS TRACE command.

Like the JANUS TRACE command, the TRACE parameter value is a bit mask integer that sums the values of the options that will be logged. The default value is 3 for SDS and OPENSERV ports, and it is 0 for WEBSERV and all other ports. For a description of the individual bit options and for more information about the TRACE setting, see the JANUS TRACE command in the *Janus TCP/IP Base Reference Manual*.

**Note:** The TRACE keyword was introduced in version 6.0 of the *Sirius Mods*. Before that, trace operations were controlled by the DEBUG keyword, which is no longer available as of version 6.5.

### 4.2.39 **UPCASE**

This parameter indicates that all client “names” are to be converted to upper case. “Names” includes userids and passwords, variable names for OPENSERV ports, column names for SDS ports and header parameters, header values, cookie names, and form field names for WEBSERV ports. By setting UPCASE as a port parameter, the userid and password can be specified by the client in case insensitive form, that is, it can be specified in lower case.

Note that it is possible to have lower case userids and passwords in *Model 204*. For example, the userids HOMER, homer, and Homer would be treated as three separate userids by *Model 204*. The UPCASE parameter simplifies the interaction between clients (where names tend to be in lower case) and *Model 204* (where they tend to be in upper case).

**Note:** The UPCASE parameter never results in data being converted to upper case. That is, if a client sends variable “@customer” with a value of “Dolly Dinkle”, and UPCASE is active for the connection, the User Language application would see a variable called “@CUSTOMER” with a value of “Dolly Dinkle”.

For SDS ports, the UPCASE parameter means that all table and column names passed from the Adaptive Server will be converted to upper case. This means that when defining the columns and tables (using JANCAT), the names must all be upper case. It also means that if an SDS port has the UPCASE parameter set but has mixed case table and column names, those tables and columns will be inaccessible.

The UPCASE parameter is the opposite of NOUPCASE. The default is for all ports to have UPCASE set.

### 4.2.40 **XTAB table**

This parameter indicates the EBCDIC-to-ASCII, ASCII-to-EBCDIC, and character entity translation tables to be used for the port.

You can specify a translation table that has not yet been loaded with the JANUS LOADXT command, but the table must be loaded before the port can be started.

The default translation table is **STANDARD**, which is a fairly generic pair of EBCDIC-to-ASCII and ASCII-to-EBCDIC translate tables that was the only available option before *Sirius Mods* version 6.0.

You can replace a translate table with the JANUS LOADXT command at any time, even if the port has active connections.

Valid for all port types, the XTAB parameter is available in *Sirius Mods* version 6.0 and later.

The *Janus Specialty Data Store* requires *Model 204* data to be “mapped” to SQL tables before it can be accessed by Sybase Adaptive Server or an SQL-based client application. This mapping is called cataloging, and is performed within the *Model 204* region by subsystem JANCAT.

To install JANCAT, follow the instructions in the ***UL/SPF Installation and Maintenance Guide***.

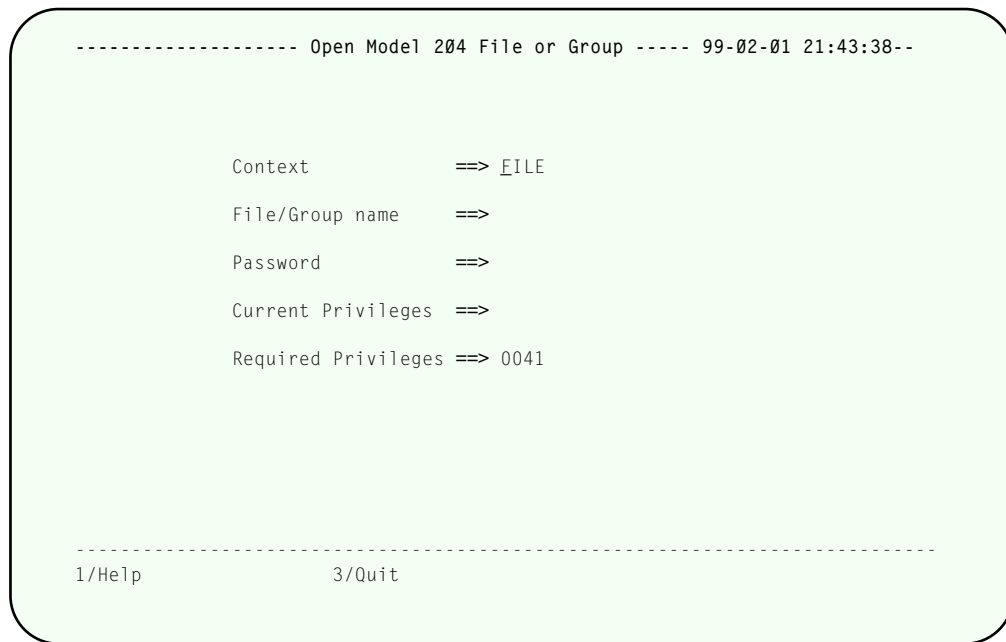
Cataloging *Model 204* data as Specialty Data Store tables stores *Model 204* records in the catalog file, which defaults to file JANCAT. *Janus Specialty Data Store* supports the use of multiple catalogs. If you wish to use a catalog file other than JANCAT you must duplicate the file and field definitions of JANCAT in the alternate file. Catalog data can be stored manually, but it is much easier and more accurate to let subsystem JANCAT perform the cataloging. The JANCAT file format is described in an appendix to this manual.

JANCAT is invoked as an APSY from *Model 204* command level by simply typing the subsystem name: **JANCAT**. The following sections list the screens that are used in JANCAT.

## **5.1 Open File screen**

The first time JANCAT is accessed at a site, the user is presented with the Open File screen, allowing him or her to enter the name of the file or group to catalog, then is presented with the Autobuild screen to automatically generate tables for that file or group. Subsequently, each time a user enters the system the Tables screen is presented.

The Open File screen is also presented when a user presses **PF6** in the Tables screen, or enters ADD <file or group> in the command area of the Tables screen with default privileges for the file or group insufficient to allow JANCAT to read the necessary record and field descriptor information.



### JANCAT Open File screen

Use the above screen to open a *Model 204* FILE or GROUP so the field characteristics can be determined and cataloging (or other JANCAT activity) may commence.

JANCAT only invokes this screen when it cannot open the file or group using default privileges, meaning either the file or group requires a password or the file is not allocated to the online (or the group is not defined). If JANCAT did open the file but privileges were not adequate for the desired action, this screen is also displayed, allowing password input.

Input/Display fields on this screen are:

- |                            |  |
|----------------------------|--|
| <b>Context</b>             | Context of the FILE or GROUP targeted for Specialty Data Store cataloging. Group context always means permanent group-- JANCAT cannot map a TEMP or ADHOC group. |
| <b>File/Group name</b>     | <i>Model 204</i> file or group name for cataloging.  |
| <b>Password</b>            | <i>Model 204</i> password that will open the file or group with the necessary privileges.  |
| <b>Current privileges</b>  | Privileges currently available in the file or group.   |
| <b>Required Privileges</b> | Privileges required for JANCAT to perform the desired action on the file or group.   |

PF1 Help  
PF3 Quit

## 5.2 Tables screen

The Tables screen provides a view of all existing Specialty Data Store tables defined in the specified catalog file:

```

----- Janus Specialty Data Store Tables -----99-02-01 21:43:30--
==> _
          Catalog File: JANCAT                Table 1/73
      TableName      File/   Default Priv N
      CCASYS_SCLS    CCASYS  PUBLIC  YYYY Y APSURTYP  SCLS
      CCASYS_SCLS_APSFG CCASYS  PUBLIC  YYYY Y APSURTYP  SCLS      APSFGN
      CCASYS_SCLS_APSF L CCASYS  PUBLIC  YYYY Y APSURTYP  SCLS      APSFLGSD
      CCASYS_SDEF    CCASYS  PUBLIC  YYYY Y APSURTYP  SDEF
      CCASYS_UDEF    CCASYS  PUBLIC  YYYY Y APSURTYP  UDEF
      DATALINK       DATALINK PUBLIC  YYYY Y
      JANCAT_COLUMNS JANCAT  PUBLIC  YNNN N RECTYPE   C
      JANCAT_SECURITY JANCAT  PUBLIC  YNNN N RECTYPE   S      SEC_UGNAME
      JANCAT_TABLES  JANCAT  PUBLIC  YNNN N RECTYPE   T
      JANCAT_USER_GROU JANCAT  PUBLIC  YNNN N RECTYPE   G      GRP_USER
      PUBS2_AUTHORS  PUBS2   PUBLIC  YYYY Y RECTYPE   AUTHORS
      PUBS2_PUBLISHERS PUBS2   PUBLIC  YYYY Y RECTYPE   PUBLISHERS
      PUBS2_ROYSCHED  PUBS2   PUBLIC  YYYY Y RECTYPE   ROYSCHED
      PUBS2_SALES     PUBS2   PUBLIC  YYYY Y RECTYPE   SALES
      PUBS2_SALESDETA I PUBS2   PUBLIC  YYYY Y RECTYPE   SALESDETAI
      PUBS2_TITLEAUTHO PUBS2   PUBLIC  YYYY Y RECTYPE   TITLEAUTHO
-----
1/Help      2/Sort-Name 3/Quit      4/Sort-File 5/Groups    6/Autobuild
7/Up        8/Down      9/Repeat    10/Refresh   11/Security

```

### JANCAT Tables screen

Use the above screen to select an action to execute against an existing Specialty Data Store table, or to add a new table to the Specialty Data Store catalog.

Input/Display options on this screen are:

**Catalog File** The Specialty Data Store interface allows multiple catalog files (currently the catalog must be a file, not a group). JANCAT is the default, and the JANUS installation process will have added the necessary fields to this file. If you wish to use another file as your catalog, or if you wish to use multiple catalogs (say, one for development and a different one for production), you must customize and execute the file creation and field definitions in the procedure CREATE.JANCAT in file SIRULSPF against the alternate catalog file. To do this make a copy of the procedure and replace the string "JANCAT" with the name of your alternate catalog file.

**Number of Tables** The total number of tables that exist in the designated catalog.

Following this, a block of lines describes each Table in the designated catalog file. The left hand side of the block looks like this:

```
      Sel  Tablename  File/
      Sel  Tablename  Group      ...
--  -----  -----
_  AUTHORS  PUBS2      ...
_  AGENTS   NAMES
_  HARDCOVERS  INVENTORY
```

and the right hand side of the block looks like this:

```
      File/  Default Priv N
      ... Group  Access  SUID Q  Subset Field/Value  NestField
-----  -----
      ... PUBS2  PRIVATE YYY Y RECTYPE  AUTHORS
      NAMES  PUBLIC  YNNN Y JOB DESC  AGENT  STATE
      INVENTORY PUBLIC  YNNN N BOOK TYPE  HARDCOVER
```

where:

**Sel** Enter a selection criterion in this field, as follows:

**E or S** Select table for update.  
**B** Browse table information.  
**R** Rename table  
**C or N** Create a new table based on this one.  
**D** Delete this table.

**Tablename** Table names may be up to 30 characters long and must consist of only the alphanumeric character set and the characters @, #, \$, % and \_. The first character must be alpha or “\_”. Under Specialty Data Store, table names must be unique across *Model 204* files.

**File/Group** The name of the *Model 204* file or group from which the table data is derived.

**Default Access** PUBLIC, PRIVATE or SEMIPUB. PUBLIC tables are accessible to all users only with the default privileges (shown in the next column). SEMIPUB tables are accessible with the default privileges by users that do not have table-specific security entries. Users with security entries for the table may override the default privileges. Default privileges for PRIVATE tables are bypassed for all users. Only users with specific security entries may access PRIVATE tables.

**Priv SUID** A positional Y or N switch indicates the default privilege on this table for SELECT, UPDATE, INSERT and DELETE commands. These access privileges are bypassed for all users of PRIVATE tables, and for users with specific security entries for SEMIPUB tables.

**NQ (Enqueue)** Y or N (Yes or No) indicator of whether JANUS should lock the file or group from which the data is drawn to prevent field redefinitions while any SDS port is running.

**Subset Field/Value** The Subset Field is a *Model 204* field that defines a set of records on which tables should be based. In *Model 204* terms this is usually a Record Type field. This display area shows the Subset fieldname and value that determines the records JANUS considers part of this table. For instance, if RECTYPE HST is shown, then the *Model 204* field RECTYPE is the subset field, and the table contains all *Model 204* records that contain the fieldname/value pair RECTYPE=HST.

**NestField** *Model 204* fieldname that determines the number of nested table rows in a *Model 204* record. A nested table contains a repeating group of fields. The nesting field is the field whose occurrences count as the row indicator, and it may or may not belong to the nested table itself.

See the description of **Nest Field** in “Columns display” on page 46.

The following commands may be entered on the Command line:

- ADD <context> <file/group> or NEW <context> <file/group>:

Either ADD or NEW is used to invoke the AUTOBUILD feature to add Specialty Data Store SQL tables for a designated file or group. Groups must be permanent groups. Example formats for the command are:

```
ADD CUSTHIST
NEW FILE CUSTHIST
ADD GROUP CUSTHIST
NEW FILE CUSTHIST
```

NEW and ADD perform the same function as **PF6** except you can specify the file or group name on this screen.

- QUIT, STOP, END or LOG:

Perform the same function as **PF3**, which exits JANCAT.

<b>PF1</b>	Help
<b>PF2</b>	Sort the displayed list by Tablename.
<b>PF3</b>	Quit
<b>PF4</b>	Sort the displayed list by File/Groupname, then Tablename.
<b>PF5</b>	Transfer to the Security Groups screen.
<b>PF6</b>	Build tables for a file or group to be designated on the Autobuild screen.

- PF7** Scroll the display up.
- PF8** Scroll the display down.
- PF9** Repeat the last command.
- PF10** Refresh the list.
- PF11** Transfer to the Security Entries screen.

### 5.3 Autobuild screen

The JANCAT Autobuild routine analyzes the record structure of a *Model 204* file or group, and populates the catalog file with a mapping of the records and fields onto SQL tables and columns. Autobuild presents the following options screen before doing the mapping.

```
----- Janus Specialty Data Store Auto Build -----99-02-01 21:44:02

Map tables for FILE PUBS2          Table name prefix ==> PUBS2

Record type field ==> 1
                    1. RECTYPE
                    2. TITLE_ID
                    3. STOR_ID

Max record types ==> 20          Records to scan ==> 200
Char fudge factor ==> 2          Delete/Update/Leave ==> L
Assume At Most One ==> N

Security:  Access: ==> PUBLIC    Default SELECT (Y/N) ==> Y
                                         Default UPDATE (Y/N) ==> Y
                                         Default INSERT (Y/N) ==> Y
                                         Default DELETE (Y/N) ==> Y

Date/Time formats used in file: ==>
                                         ==>

-----
1/Help          3/Quit
```

#### JANCAT Autobuild

Use the above screen to set parameters or to accept JANCAT's calculations for mapping the file or group onto a table structure.

Pressing **ENTER** will initiate the JANCAT Autobuild routine which performs the 204-to-SQL mapping. When Autobuild is complete, the Tables screen will be presented and the Specialty Data Store tables built during this session will be displayed along with those that already existed. This process is quite involved and for a large file may take several minutes.

Input/Display fields on this screen are:

**FILE/GROUP**

Context of the FILE or GROUP targeted for Specialty Data Store cataloging. Group context always means permanent group. This is a display-only field.

**File/Group name**

*Model 204* file or group name for cataloging. This is a display-only field.

**Table name prefix**

JANCAT will either build a single table from a complete file, or it will build a table for each value of a record type field and for each nested table (group of repeating fields). JANCAT's default naming scheme is to prefix all tables from the same file with the file name. If a valid prefix is entered in this field, it is used instead of the file name. For instance, if you want to map CCASYS but don't want table names like CCASYS\_SCLS and CCASYS\_SDEF you could put "APSY" in the prefix field and you would get table names APSY\_SCLS and APSY\_SDEF, etc.

**Record type field**

*Model 204* fieldname whose values will determine the table structure of the cataloged Specialty Data Store tables. JANCAT will have attempted to determine the likely record type fields, and will display, at most, 3 candidates. These fields may be selected by number, or any other field in the target file or group may be selected by name.

If no record type field is selected the entire *Model 204* file or group is mapped to a single table or set of tables. If a record type field is selected, a table or set of tables is built for each value of the record type field.

**Max record types**

Limit the number of values of the record type field for which Specialty Data Store tables will be built.

**Records to scan**

Number of records to scan--within each value of the record type field if one is selected--to determine the columns (fields) and contents of any repeating group (multiply occurring fields). This defaults to 200 records. JANCAT performs a record-skipping algorithm based on the number of records in the file, so records are not all sampled from the beginning of the file.

**CHAR fudge factor**

This is an integer which is the number of standard deviation units to add to the average of the sampled field occurrence lengths to calculate the default column length.

JANCAT calculates the average and longest lengths for every observed occurrence of each field. It also calculates the standard deviation for the length of each field. Each field's default column length is set to the observed mean plus as many standard deviation units as specified in *fudge factor*.

The *fudge factor* default is two standard deviation units, which should result in an adequate length for 99% to 100% of occurrences of all fields. Note that the algorithm using +2 standard deviations works even when a field is fixed length, as there will be no variance from the mean, and the standard deviation will be 0. The *fudge factor* of 2 standard deviations is almost always the preferred value.

### **Delete/Update/Leave**

If you are cataloging tables for a file that has already been cataloged, JANCAT can take one of three actions on the tables already existing for the file.

D (Delete) -- Delete all tables for the file before building the new tables

U (Update) -- Update any tables named the same as JANCAT's standard naming convention, but leave any other tables for this file as they are.

L (Leave) -- Leave alone any existing tables, but add new ones.

### **Assume At Most One**

*Janus Specialty Data Store* can generate much more efficient User Language from incoming SQL if the catalog tells which fields are guaranteed not to be multiply occurring. If this field is set to "Y", the autobuild routine will turn on the "at most one" switch for all fields which do not occur more than one time within records sampled for the file (and within each record type value, if one is specified). This is the preferred setting and should be overridden only if it is known that certain fields may contain multiple occurrences even though they currently do not. Note that Janus generates the most efficient User Language anyway if the AT MOST ONE *Model 204* file attribute is set. The At Most One setting in JANCAT is only needed for fields that don't explicitly have the *Model 204* AT MOST ONE setting but which are known to occur no more than once per 204 record.

### **Access**

Access defines the default level of security to apply to tables built from this file. Default access may be one of the following:

PUBLIC -- Grant only default privileges to all users.

PRIVATE -- Grant no default privileges. Require security.

SEMIPUB -- Grant default privileges to users without security entries, and specific privileges to those with security entries.

### **Default SELECT (Y/N)**

A "Y" means to grant SQL SELECT privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Default UPDATE (Y/N)**

A “Y” means to grant SQL UPDATE privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Default INSERT (Y/N)**

A “Y” means to grant SQL INSERT privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Default DELETE (Y/N)**

A “Y” means to grant SQL DELETE privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Date/Time formats used in file**

JANCAT automatically scans for 10 different date formats to determine if a field can be mapped to a datetime column. If any field contains date/time data not in one of the formats, JANCAT will map it to CHAR unless the format is specified in this input area. Two custom date/time formats may be input. The characters that make up valid date/time format specifications are described in “[Datetime Formats](#)” on page 70.

The ten datetime formats JANCAT automatically scans for are:

```

YYYYMMDD
YYYYDDD
YYMMDD
YYDDD
YYYY MONTH DAY
YY MONTH DAY
DD MON YYYY
MON DD YYYY
YYIMMIDDIHHIMMISS
YYYYMMDDHHMMSS

```

JANCAT's method for naming tables is:

```
<filename>_<rectype.value>_<nestfield>
```

so the table built for APSURTYP=SCLS in FILE CCASYS would be

```
CCASYS_SCLS
```

If no repeating group existed, and

```
CCASYS_SCLS_APSFGN
```

if a repeating group existed and the field APSFGN defined a repeating group (a nested table) on the record type.

Specialty Data Store Table names have a limit of 30 characters, so names are truncated and a uniqueness check is performed.

If you wish to rebuild tables for a file but don't want any of the existing tables overwritten, they can be renamed in the Tables screen.

**PF1** Help  
**PF3** Quit

### 5.4 Columns display

When an SQL table is selected from the Tables screen (using “S” in the selection column), the following screen is presented:

```
--- FILE PUBS2 --- Janus Specialty Data Store Columns -- 99/02/02 08:57:34 --
==> _                                     Tot Columns: 9

Table ==> PUBS2_AUTHORS                    Enqueue ==> Y
Access ==> PUBLIC      Select ==> Y Update ==> Y Insert ==> Y Delete ==> Y
Nest Field ==>
Subset Field ==> RECTYPE
Subset Value ==> AUTHORS

Sel Column Name      Type      Null 1 Occ  Field Name
AU_ID                CHAR(11)  N N 1  AU_ID
AU_LNAME             CHAR(14)  N N 1  AU_LNAME
AU_FNAME             CHAR(11)  N N 1  AU_FNAME
PHONE                CHAR(12)  N N 1  PHONE
ADDRESS              CHAR(21)  N N 1  ADDRESS
CITY                 CHAR(14)  N N 1  CITY
STATE                CHAR(2)   N N 1  STATE
COUNTRY              CHAR(3)   N N 1  COUNTRY
POSTALCODE           DATETIME  N N 1  POSTALCODE

-----
1/Help              3/Quit
7/Up                8/Down          9/Repeat
12/End-Save
```

**JANCAT Columns display**

Values displayed on this screen are the results of JANCAT's analysis of the file selected for SQL table mapping.

Any portion of a *Model 204* record structure can be mapped to an SQL table, but the most likely way to perform mapping is to either map every field in the file onto a column in a single table, or to create a table for value of a “record type” field if the file has one, and to map into each table only those fields that apply.

If no Record type field was entered on the Autobuild screen, and if JANCAT did not detect a likely candidate for record type mapping, then this screen shows the entire file being mapped to a single SQL table. If a record type field exists this JANCAT screen allows an SQL table to be stored for each value of that field and JANCAT will have derived a table name from a combination of the file name and the record type value.

While it's likely that you'll want to alter the SQL table name automatically generated by JANCAT, some effort should be made to maintain consistent column names across tables mapped from the same file. The column names automatically generated by JANCAT are the field names converted to conform to the restrictions imposed on Specialty Data Store column names.

Input/Display fields on this screen are:

**Sel** Enter a selection criterion in this field, as follows:

- S, E, or U** (Select, Edit or Update) Invokes the Column Update screen for the selected column.
- M** (Move) Moves the column to a specified location.
- P or B** (Prior, Before) Specifies the location for a Move.
- A or F** (After, Following) Specifies the location for a Move.
- R** (Rename) Renames the column.
- D** (Delete) Removes the column from the table definition.
- I** (Insert) Inserts a new column after the specified column and invokes the Column Update screen for the new column.

**Table** JANCAT generates a table name based on the file name and the record type if there is one. Remember that Specialty Data Store Server requires table names to be unique, even across *Model 204* files.

JANCAT's full method for naming tables is:

<filename>\_<rectype.value>\_<nestfield>

so the table built for APSURTYP=SCLS in FILE CCASYS would be

CCASYS\_SCLS

if no repeating group existed, and

CCASYS\_SCLS\_APSFGN

if a repeating group existed and the field APSFGN defined a repeating group (a nested table) on the record type.

**Enqueue** “Y” tells JANCAT to place an update lock on the file or group once the Specialty Data Store port is started, preventing field names from being changed in the source *Model 204* file or group. Enter “N” to not lock the file or group.

**Access** Access defines the default level of security to apply to tables built from this file. Default access may be one of the following:

PUBLIC -- Grant only default privileges to all users.

PRIVATE -- Grant no default privileges. Require security.

SEMIPUB -- Grant default privileges to users without security entries, and specific privileges to those with security entries.

**Select** A “Y” means to grant SQL SELECT privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Update** A “Y” means to grant SQL UPDATE privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Insert** A “Y” means to grant SQL INSERT privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Delete** A “Y” means to grant SQL DELETE privileges to all users of PUBLIC tables and to default users of SEMIPUB tables. This setting is ignored for PRIVATE tables.

**Nest Field** *Model 204* fieldname that determines the number of nested table rows in a *Model 204* record. A nested table consists of a repeating group of 1 or more fields (that are meaningfully related and occur in sync on the source records) and an optional set of non-repeating fields that are “constant” for all nested rows in a record. The occurrences of the field designated as the Nest Field define the repeating group, and it may or may not belong to the nested table itself.

For example, if a single record is used to hold a customer call history, including CALL\_DATE\_TIME, ORDER\_NUMBER and ORDER\_TOTAL, any of these three fields could be designated the Nest Field. ORDER\_NUMBER(1) would correspond to ORDER\_TOTAL(1) and CALL\_DATE\_TIME(1), etc., and this set would make up the 1st row of the nested table, the second set of occurrences would make up the second row, etc.

There are two special indicators that may be used as the Nest Field:

- Asterisk ('\*') indicates Janus should use as the Nest Field that repeating field with the highest number of occurrences on a given record. This guarantees that no data will be lost when some occurrences of multiply occurring fields are missing. Each record produces as many rows in the table as the highest number of occurrences among the multiply occurring fields; thus a record with no occurrences of any repeating field will not produce any rows.
- Pound sign ('#') also uses as the Nest Field the field with the highest number of occurrences, except that a record with no occurrences of the multiply occurring fields produces one row in the table (with null values for all the repeating fields).

See the description of **Occurrence Number** in this section, which shows how to designate non-repeating fields that are “constant” for all nested rows in a record.

**Subset Field** The “record type” field whose values determine which *Model 204* records qualify as part of this SQL table.

**Subset Value** The subset field value that defines the *Model 204* data for this table.

For each column in the table the following display values are shown:

**Column Name**

The column name is derived by JANCAT by converting spaces, periods and other characters not allowed by Specialty Data Store Server to underscores (“\_”), and truncating the name to 30 characters (the maximum length allowed by Sybase Adaptive Server). Column names can be changed by the user.

**Type of column and (length)**

*Janus Specialty Data Store* supports most Sybase Adaptive Server data types for its columns. The following table shows valid data types for a column, and lengths that can be assigned to each type.

Column type	Description	Length
-----	-----	-----
INT	Integer	N/A
SMALLINT	Integer	N/A
TINYINT	Integer	N/A
FLOAT	Floating point	N/A
REAL	Floating point	N/A
CHAR	Character string	1 - 255
VARCHAR	Character string	1 - 255
DATETIME	Date/Time data	1 - 255
-----	-----	-----

Specialty Data Store Server does not distinguish between CHAR and VARCHAR, so neither does *Janus Specialty Data Store* support. Users may map any type *Model 204* field onto any column, because *Model 204* allows data not of the field's nominal type to be entered into the file.

Specify the length on CHAR, VARCHAR, and DATETIME columns by appending an integer value to the type designation, e.g.:

```
CHAR(40)
VARCHAR(231)
DATETIME(10)
```

**Null - allow or not**

“Y” allows null values to be held in the returned SQL columns. “N” prevents nulls being saved.

**1 (At Most One)**

“Y” indicates the field occurs at most one time in a record. “N” indicates the field may occur more than once.

**Occurrence number**

Field occurrence number to map onto this column. For record structures with known, fixed numbers of occurrences, users can map specific occurrence numbers onto different columns. JANCAT initially shows each field only once, but any field/column line can be duplicated via [PF6](#), and the columns mapped to unique occurrences. For instance, if a *Model 204* record is known to almost always contain three occurrences of the field FOOD, the SQL mapping could be done like this:

Column name	Type	Nul	1	Occ	Field Name
PROTEIN	CHAR(25)	N	N	1	FOOD
VEGETABLE	CHAR(25)	N	N	2	FOOD
STARCH	CHAR(25)	N	N	3	FOOD

The other use of **Occurrence number** is for repeating groups. To implement a different application, and using a database with different meanings of the field, FOOD could be mapped onto a repeating group, by designating FOOD as the *Subset Field* and specifying null or 0 in the **Occurrence number**. Any column mapped to a null or 0 occurrence number is part of the repeating group. So, for instance:

Subset Field: FOOD

Column name	Type	Nul	1	Occ	Field Name
FOOD_ITEM	CHAR(25)	N	Y	Ø	FOOD
MEALS_PER_WK	INT	N	Y	Ø	WEEKLY.IN
AVG_MEAL_SIZ	INT	N	Y	Ø	WEEKLY.OZ

Note that the Subset Field does not necessarily have to be part of the repeating group, it just has to be in the file, and has to have occurrence numbers that match those of the field-components of the repeating group.

When mapping data from a *Model 204* record structure onto an SQL row structure, it is often necessary to add non-repeating “constant” fields to the repeating group columns of the nested table. This is done by specifying a positive occurrence number for the non-repeating column fields (NAME and ID in the example below) in the table definition. This causes *Janus Specialty Data Store* to return the fixed occurrences of those fields for each row of the nested table.

Column name	Type	Nul	1	Occ	Field Name
-----	-----	---	-	---	-----
NAME	CHAR(40)	N	N	1	CUST.NAME
ID	CHAR(25)	N	N	1	CUST.ID
PURCHASE_LOC	CHAR(25)	N	N	0	STORE
PURCHASE_DT	CHAR(10)	N	N	0	DATE
PURCHASE_AMT	INT	N	N	0	COST

### Field Name

*Model 204* field from which the Specialty Data Store column gets its data.

PF1	Help
PF3	Quit
PF7	Scroll the display up.
PF8	Scroll the display down.
PF9	Repeat the last command.
PF12	Save this table and exit. If you exit this routine NOT via PF12, no changes made to this table, including anything done on the Column Update screen, will be saved.

## 5.5 Column Update screen

When a specific column is selected from the Columns display, the following screen is displayed:

```

--- FILE PUBS2 --- Janus Specialty Data Store Columns -- 99/02/02 11:26:24 --

Column Name    ==> AU_ID
Type/Length    ==> CHAR(11)
Allow Null     ==> N
At most one    ==> N
DateTime Format ==>
Field Occurs   ==> 1

Fieldname
==> AU_ID

-----
1/Help          3/Quit
12/End-Save
    
```

**JANCAT Column Update screen**

Use the above screen to update the mapping of a selected column.

If this screen was presented as part of a DELETE function, all fields are protected and it is used only for verification purposes. If the column is being RENAMED or COPIED, all screen fields are protected except the column name.

Values on the screen are:

**Column name**

The column name is derived by JANCAT by converting spaces, periods and other characters not allowed by Sybase Adaptive Server to underscores (“\_”), and truncating the name to 30 characters, the maximum length allowed by Sybase Adaptive Server. Column names can be changed by the user.

**Type of column and (length)**

*Janus Specialty Data Store* supports most Sybase Adaptive Server data types for its columns. The following table shows valid data types for a column, and lengths that can be assigned to each type.

Column type	Description	Length
INT	Integer	N/A
SMALLINT	Integer	N/A
TINYINT	Integer	N/A
FLOAT	Floating point	N/A
REAL	Floating point	N/A
CHAR	Character string	1 - 255
VARCHAR	Character string	1 - 255
DATETIME	Date/Time data	1 - 255

Specialty Data Store Server does not distinguish between CHAR and VARCHAR, so neither does *Janus Specialty Data Store* support. Users may map any type *Model 204* field onto any column, because *Model 204* allows data not of the field's nominal type to be entered into the file.

Specify the length on CHAR, VARCHAR, and DATETIME columns by appending an integer value to the type designation, e.g.:

```
CHAR(40)
VARCHAR(231)
DATETIME(10)
```

### Column number

The relative position of the column in the row returned to Sybase Adaptive Server. This number is display only on this screen, but corresponds to the column order which can be manipulated on the Columns display screen.

### Allow nulls

“Y” to allow null values to be held in the returned SQL column. “N” prevents nulls being preserved.

### At Most One

“Y” indicates the field occurs at most one time in a record. “N” indicates the field may occur more than once.

If *Janus Specialty Data Store* knows that a field will occur no more than once per record, it can generate much more efficient User Language from the incoming SQL. To determine this, Janus checks for the AT MOST ONE field attribute; if it is not present, the **At Most One** JANCAT column-level indicator is used.

### DateTime format

If the Column Type (above) is DATETIME, a format must be specified. Date/time formats are described in [“Datetime Formats” on page 70](#).

### Field occurs

Field occurrence number to map onto this column. For record structures with known, fixed numbers of occurrences, users can map specific occurrence numbers onto different columns. See the description of the **Occurrence number** values in [“Columns display” on page 46](#).

### Field name

*Model 204* field from which the Specialty Data Store column gets its data.

PF1	Help
PF3	Quit
PF12	Save Column and exit.

## 5.6 JANCAT security system

The JANCAT security system is governed by mapping user IDs or groups of user IDs onto specific privilege sets for each table. For instance, if you wanted all your data entry clerks to have SELECT, UPDATE, INSERT and DELETE privileges to a PERSONEL file, you could enter them one at a time, assigning the four privileges to each, or you could build a security Group, calling it DATAENTRY, add each data entry clerk's ID to the group and then give the group the necessary privileges.

Using Groups greatly simplifies management of table security. Instead of tracking every accountant's access privileges, you can define an ACCOUNTANTS group, designate privileges for members of the group, then add or remove IDs from the group as the accounting team changes.

### 5.6.1 Security Groups screen

**PF5** from the Tables or Security Entries screen invokes the Security Groups screen.

Security Groups are displayed alphabetically by group name. Each line of users may contain up to 255 characters, and if the size of the user list exceeds the screen width, an ellipsis is shown (“...”) at the right. The group can be selected to display all names assigned to the group. If a group contains a very large number of users, multiple lines of users will be shown, and selecting the group will result in a display of all users.

```

- JANCAT - Janus Specialty Data Store Security Groups -- 99/02/01 21:44:24 --
==> _                                     Groups: 1/25

*- Group -* *----- User Ids -----*
- ACCOUNTNTS BOBBREED QUICKEN VINNY
- ADULTCONTE BILLYJOEL KENNYG WHITNYHSTN YANNY ZANFIR
- BOSSES     BILLGATES DAVE GARY JGOTTI SREDSTONE
- DEVELOPERS ALAN ALEX DME TOM
- DWARFS7    BASHFUL DOC DOPEY GRUMPY HAPPY SLEEPY SNEEZY
- HACKS      GHENGISKHN IVANTERRIB JULIACHILD OJ PHILLIPEKN SPECKINPAW
- HASBEENS   HRMNSHRMTS MONKEES SIRMIXALOT TERRYJAX
- ICONS      BILLGATES BUDDHA CINDY ELVIS JAMESDEAN MARILYN
- KINGS      CRIMSON EDWARD1 EDWARD2 GEORGE1 GEORGE2 GEORGE3 GYPSY HENRY1...
- MARKETING  DAVE JEFF MSAATCHI
- NEWGRP     NEWUSER0 NEWUSER1 NEWUSER11 NEWUSER2 NEWUSERS5 NEWUSER6 OLDUS...
- PUNKS      RATSORIZZO SIDVICIOUS
- QUEENS     ANTOINETTE BEATRICE ELIZABETH1 ELIZABETH2 HENRIETTA ISAAC PE...
- ROCKNROLL  ACDCASN AEROSMITH ALLMANBROS AMADEUS ANNIELENOX ARETHAFRKL B...
              KINKS KISS KTICKELL KURTCOBAIN LINDARNDST LITTLEFEAT LITTLER...
              RICKOCASIC RNEWMAN SALTNPPEPPA SEBASTION SHERYLC SLYSTONE SNO...

-----
1/Help          3/Quit
7/Up            8/Down        9/Repeat      11/Security

```

**JANCAT Security Groups screen**

Use the above screen to build or update a security group. ADD <group ID> or NEW <group ID> may be entered in the command line at the top of the screen to build a new security group.

The selection field to the left of the group name allows the following entries:

**S** Select the group to view the complete list of users and/or to make changes to the list of users.

**D** Delete the group.

- PF1** Help
- PF3** Quit
- PF7** Scroll the display up.
- PF8** Scroll the display down.
- PF9** Repeat the last command.
- PF11** Transfer to the Security Entries screen.

### 5.6.2 Modify Security Group screen

If the ADD or NEW command is entered from the Security Groups screen, or if a group is selected via the “S” prefix command on the Security Groups screen, the Modify Security Group screen is displayed. This screen shows the complete list of users assigned to the group, and allows the names to be changed or deleted. JANCAT will automatically sort the list alphabetically, so there's no point trying to put the user IDs into any specific order.

```

- From: JANCAT ----- Modify Security Group - Add Users - 99/02/02 12:36:30 --
==> _                                     Users: 1/117

Group Name ==> ROCKNROLL

*----- User Id's -----*
ACDCASN   AEROSMITH ALLMANBROS AMADEUS   ANNIELENOX ARETHAFRKL
BASH      BEACHBOYS BECK       BEEFYBOYS BILLHALLEY BILLYJOEL
BOBBYCFRN BONO      BUDDYHOLLY BUSTERBILL BUSTERKEAT B52S
CARS      CHRISSEYHNS CINDYLAUPR CLASH     COCOTAYLOR COURTNEYLV
CREAM     DAVIDBYRNE DAVIDGAT   DME       DURANDURAN DWEEZIL
DWIGHTYOKM ECOSTELLO ELO        ELP       ELVIS     EUROPE
EURYTHMICS FLEETWOODM FRANKZAPPA GARTH     GHOSTBUSTR GRATEFLDED
HAMMER    IGGYPOP   ITZAK      JADDICTION JEFFBECK   JGEILS
JHAMMOND  JIMMYHNDRX JOHANN     JOHNNYCASH JOHNNYWINT JSBACH
JULIE     KINGCRIM  KINKS      KISS       KTICKELL   KURTCOBAIN
LINDARNDST LITTLEFEAT LITTLERICH LSPHOONFUL MADONNA    MAMASPAPAS
METALLICA MOONUNIT  NEWCRYSTYM NIRVANA    PAUPER     PEARLJAM
PETERTOSH PETERWOLF PGABRIAL   PINKFLOYD POGUES     POLICE
POPPAJOHN PRINCE    RAYCHARLES RICHIEHAVN RICKOCASIC RNEWMAN
-----

1/Help      3/Quit
7/Up        8/Down    9/Repeat   12/Save

```

**JANCAT Modify Security Group screen**

Use the above screen to add user IDs to a security group. Each field allows entry of a user ID. User IDs are scanned for invalid characters and duplicates.

Either Groups or individual user IDs may be mapped onto particular table/privilege combinations. Groups provide far more flexibility for anything other than trivial Janus implementations.

- PF1** Help
- PF3** Quit
- PF7** Scroll the display up.
- PF8** Scroll the display down.
- PF12** Save this security group and exit.

### 5.6.3 Security Entries screen

**PF11** from the Tables or Security Groups screen invokes the Security Entries screen.

By default, tables are displayed alphabetically by name, and then by group/user ID. For each Table/ID combination, you simply tab over to alter the privileges associated with that entry. A “Y” indicates the user or group is permitted to either SELECT, UPDATE, INSERT or DELETE in the named table. Any other character is converted to “N” to revoke permission.

Table entries may contain the “\*” wildcard character as the last character of the identifier. Thus, entries for “BILL\*” would grant user and group permission for table names “BILLING”, “BILLBO”, “BILLYBONG”, etc.

```
- JANCAT -- Janus Specialty Data Store Security Entries -- 99/02/01 21:44:36 -
==> _                               Entries: 1/55

  Table                               Type  User/Group  Sel  Upt  Ins  Del
-  AUTHORS                             USER  ALAN        Y   Y   Y   N
-  AUTHORS                             USER  SA          Y   Y   Y   Y
-  DATALINK                             GROUP  SCURRILOUS  Y   N   N   N
-  DATALINK                             GROUP  SIRIUS      Y   Y   Y   Y
-  DATALINK                             USER  ALAN        Y   Y   Y   Y
-  DATALINK                             USER  BETTYBOOP   Y   N   N   N
-  DATALINK                             USER  FOOLHARDY   Y   N   N   N
-  DATALINK                             USER  FOOLONTHHL Y   N   N   N
-  DATALINK                             USER  NOBYSFOOL   Y   N   N   N
-  DATALINK                             USER  SIRIUS      Y   Y   N   N
-  DATALINK                             USER  SNOOPY      Y   Y   Y   Y
-  DATALINK                             USER  SOMEBDYFOO Y   N   N   N
-  DISCOUNTS                          GROUP  ROCKNROLL   Y   Y   Y   Y
-  DISCOUNTS                          GROUP  SIRIUS      Y   Y   Y   Y
-  DISCOUNTS                          USER  ALAN        Y   N   N   N
-  DISCOUNTS                          USER  ANNIEPROLU Y   N   N   N
-----
1/Help  2/Sort-Table  3/Quit  4/Sort-User  5/Groups
7/Up    8/Down       9/Repeat 10/Refresh
```

#### JANCAT Security Entries screen

Use the above screen to add, delete, or modify access privileges to tables, for user or group IDs. ADD <table\_name> or NEW <table\_name> may be entered in the command line at the top of the screen to add a list of user and/or group IDs.



- Update** “Y” indicates that SQL UPDATE privileges will be granted to all users and groups designated on this screen.
- Insert** “Y” indicates that SQL INSERT privileges will be granted to all users and groups designated on this screen.
- Delete** “Y” indicates that SQL DELETE privileges will be granted to all users and groups designated on this screen.
- User/Group identifier** 10 character identifiers of users or groups.
- Group indicator** If the entry is a group ID it must be flagged as such with a “Y” in this field.

- PF1** Help
- PF3** Quit
- PF9** Repeat the last command.
- PF12** Save this security group and exit.

*Specialty Data Store Catalog File Structure*

Users of *Janus Specialty Data Store* may want to customize their SQL catalog in a manner not directly supported by Sirius, or to write custom reports or screen queries against the catalog. JANCAT, the default catalog file, is a normalized file that contains the table structures in *Model 204* record format. If you want to use a file other than JANCAT as your catalog, you must apply the JANCAT field definitions in CREATE.JANCAT in file SIRULSPF. CREATE.JANCAT also contains the record mappings that allow SQL access to the tables that make up JANCAT itself. These STORE RECORD statements are a good model to work from if you want to build record mappings manually.

The JANCAT field definitions are:

```

IN JANCAT DEFINE FIELD RECTYPE           WITH ORDERED
IN JANCAT DEFINE FIELD COL_NAME          WITH ORDERED
IN JANCAT DEFINE FIELD COL_TAB_NAME     WITH ORDERED
IN JANCAT DEFINE FIELD COL_FIELD_NAME
IN JANCAT DEFINE FIELD COL_FIELD_OCC
IN JANCAT DEFINE FIELD COL_NULL
IN JANCAT DEFINE FIELD COL_NUM
IN JANCAT DEFINE FIELD COL_TYPE
IN JANCAT DEFINE FIELD COL_LEN
IN JANCAT DEFINE FIELD COL_DATEFORMAT
IN JANCAT DEFINE FIELD COL_AT_MOST_ONE
IN JANCAT DEFINE FIELD GRP_NAME         WITH ORDERED
IN JANCAT DEFINE FIELD GRP_USER        WITH ORDERED
IN JANCAT DEFINE FIELD SEC_TAB_NAME     WITH ORDERED
IN JANCAT DEFINE FIELD SEC_UGNAME      WITH ORDERED
IN JANCAT DEFINE FIELD SEC_UGTYPE      WITH ORDERED
IN JANCAT DEFINE FIELD SEC_SELECT
IN JANCAT DEFINE FIELD SEC_UPDATE
IN JANCAT DEFINE FIELD SEC_INSERT
IN JANCAT DEFINE FIELD SEC_DELETE
IN JANCAT DEFINE FIELD TAB_NAME         WITH ORDERED UNIQUE
IN JANCAT DEFINE FIELD TAB_FGNAME      WITH ORDERED
IN JANCAT DEFINE FIELD TAB_FGTYPE     WITH ORDERED
IN JANCAT DEFINE FIELD TAB_ENQ
IN JANCAT DEFINE FIELD TAB_ACCESS
IN JANCAT DEFINE FIELD TAB_SELECT
IN JANCAT DEFINE FIELD TAB_UPDATE
IN JANCAT DEFINE FIELD TAB_INSERT
IN JANCAT DEFINE FIELD TAB_DELETE
IN JANCAT DEFINE FIELD TAB_SUBSET_FIELD
IN JANCAT DEFINE FIELD TAB_SUBSET_VALUE
IN JANCAT DEFINE FIELD TAB_NEST_FIELD

```

The values for RECTYPE are 'T' for a Table record, 'C' for a Column record, 'G' for a Group record and 'S' for a Security record.

A Specialty Data Store table consists of a single 'T' record with associated 'C' records. TAB\_NAME and COL\_TAB\_NAME are the primary key that holds together 'T' and 'C' records and secured access to them on the 'S' record. Thus a simple table structure might look like this:

```
RECTYPE = T                (This is a Table record)
TAB_NAME = CARTOONS        (of table "CARTOONS")
TAB_FGNAME = CARTOONS     (from "CARTOONS")
TAB_FGTYPE = FILE         (which is a FILE)
TAB_ENQ = Y               (Enqueue on port startup)
TAB_ACCESS = PUBLIC       (allow everybody access)
TAB_SELECT = Y            (but only for SELECT,
                           not for UPDATE
                           or INSERT
                           or DELETE privileges)
TAB_UPDATE = N
TAB_INSERT = N
TAB_DELETE = N
RECTYPE = C                (Here's a column record)
COL_NUM = 1                (for column 1)
COL_TAB_NAME = CARTOONS   (of table "CARTOONS")
COL_NAME = CHARACTER_NAME (first col is CHARACTER NAME)
COL_FIELD_NAME = CHARACTER (from field CHARACTER)
COL_FIELD_OCC = 1         (first occurrence)
COL_NULL = N              (Field can't be set to null)
COL_TYPE = CHAR           (column type is character)
COL_LEN = 50              (up to 50 characters long)
COL_AT_MOST_ONE = Y       (there's only one per record)
RECTYPE = C                (Here's another col record)
COL_NUM = 2                (for column 2)
COL_TAB_NAME = CARTOONS   (of table "CARTOONS")
COL_NAME = BROADCAST_TIME (called BROADCAST_TIME)
COL_FIELD_NAME = T.O.D.   (from field T.O.D.)
COL_FIELD_OCC = 1         (first occurrence)
COL_NULL = N              (Field can't be set to null)
COL_TYPE = DATETIME       (column holds DATETIME data)
COL_LEN = 20              (up to 20 characters long)
COL_DATETIME = HH:MM      (even though format len is 5)
COL_AT_MOST_ONE = Y       (and it too only occurs once)
RECTYPE = C                (Here's another col record)
COL_NUM = 1                (for column 3)
COL_TAB_NAME = CARTOON    (of table "CARTOONS")
COL_NAME = CHANNEL        (called CHANNEL)
COL_FIELD_NAME = NETWORK  (from field NETWORK)
COL_FIELD_OCC = 1         (first occurrence)
COL_NULL = N              (Field can't be set to null)
COL_TYPE = CHAR           (it's character data)
COL_LEN = 10              (up to 10 characters long)
COL_AT_MOST_ONE = Y       (and it too only occurs once)
```

JANCAT itself is built of tables, allowing remote queries to analyze the structure of JANCAT itself. The following STORE RECORD stream builds the table/column and security entries that describes JANCAT:

```
IN JANCAT STORE RECORD
RECTYPE           = 'T'
TAB_NAME          = 'JANCAT_TABLES'
TAB_FGNAME        = 'JANCAT'
TAB_FGTYPE        = 'FILE'
TAB_ENQ           = 'N'
TAB_ACCESS        = 'PUBLIC'
TAB_SELECT        = 'Y'
TAB_UPDATE        = 'N'
TAB_INSERT        = 'N'
TAB_DELETE        = 'N'
TAB_SUBSET_FIELD  = 'RECTYPE'
TAB_SUBSET_VALUE  = 'T'
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'T'
TAB_NAME          = 'JANCAT_COLUMNS'
TAB_FGNAME        = 'JANCAT'
TAB_FGTYPE        = 'FILE'
TAB_ENQ           = 'N'
TAB_ACCESS        = 'PUBLIC'
TAB_SELECT        = 'Y'
TAB_UPDATE        = 'N'
TAB_INSERT        = 'N'
TAB_DELETE        = 'N'
TAB_SUBSET_FIELD  = 'RECTYPE'
TAB_SUBSET_VALUE  = 'C'
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'T'
TAB_NAME          = 'JANCAT_USER_GROUP'
TAB_FGNAME        = 'JANCAT'
TAB_FGTYPE        = 'FILE'
TAB_ENQ           = 'N'
TAB_ACCESS        = 'PUBLIC'
TAB_SELECT        = 'Y'
TAB_UPDATE        = 'N'
TAB_INSERT        = 'N'
TAB_DELETE        = 'N'
TAB_NEST_FIELD    = 'GRP_USER'
TAB_SUBSET_FIELD  = 'RECTYPE'
TAB_SUBSET_VALUE  = 'G'
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'T'
TAB_NAME          = 'JANCAT_SECURITY'
TAB_FGNAME        = 'JANCAT'
TAB_FGTYPE        = 'FILE'
TAB_ENQ           = 'N'
TAB_ACCESS        = 'PUBLIC'
TAB_SELECT        = 'Y'
TAB_UPDATE        = 'N'
TAB_INSERT        = 'N'
TAB_DELETE        = 'N'
TAB_NEST_FIELD    = 'SEC_UGNAME'
TAB_SUBSET_FIELD  = 'RECTYPE'
TAB_SUBSET_VALUE  = 'S'
```

```
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'TABLE_NAME'
COL_FIELD_NAME   = 'TAB_NAME'
COL_FIELD_OCC    = 1
COL_NUM          = 1
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN         = 30
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'FILE_OR_GROUP'
COL_FIELD_NAME   = 'TAB_FGTYPE'
COL_FIELD_OCC    = 1
COL_NUM          = 2
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN         = 5
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'FILE_OR_GROUP_NAME'
COL_FIELD_NAME   = 'TAB_FGNAME'
COL_FIELD_OCC    = 1
COL_NUM          = 3
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN         = 8
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'ENQUEUE_ON_FILE'
COL_FIELD_NAME   = 'TAB_ENQ'
COL_FIELD_OCC    = 1
COL_NUM          = 4
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN         = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'TABLE_ACCESS_MODE'
COL_FIELD_NAME   = 'TAB_ACCESS'
COL_FIELD_OCC    = 1
COL_NUM          = 5
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN         = 7
END STORE
IN JANCAT STORE RECORD
```

```

RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'TABLE_SELECT_PRIV'
COL_FIELD_NAME   = 'TAB_SELECT'
COL_FIELD_OCC    = 1
COL_NUM          = 6
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'TABLE_UPDATE_PRIV'
COL_FIELD_NAME   = 'TAB_UPDATE'
COL_FIELD_OCC    = 1
COL_NUM          = 7
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'TABLE_INSERT_PRIV'
COL_FIELD_NAME   = 'TAB_INSERT'
COL_FIELD_OCC    = 1
COL_NUM          = 8
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'TABLE_DELETE_PRIV'
COL_FIELD_NAME   = 'TAB_DELETE'
COL_FIELD_OCC    = 1
COL_NUM          = 9
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'
COL_NAME         = 'SUBSET_FIELD_NAME'
COL_FIELD_NAME   = 'TAB_SUBSET_FIELD'
COL_FIELD_OCC    = 1
COL_NUM          = 10
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 30
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_TABLES'

```

```
COL_NAME          = 'SUBSET_FIELD_VALUE'  
COL_FIELD_NAME    = 'TAB_SUBSET_VALUE'  
COL_FIELD_OCC     = 1  
COL_NUM           = 11  
COL_NULL          = 'N'  
COL_TYPE          = 'CHAR'  
COL_LEN          = 8  
END STORE  
IN JANCAT STORE RECORD  
RECTYPE          = 'C'  
COL_TAB_NAME     = 'JANCAT_TABLES'  
COL_NAME         = 'NESTING_FIELD_NAME'  
COL_FIELD_NAME   = 'TAB_NEST_FIELD'  
COL_FIELD_OCC    = 1  
COL_NUM         = 12  
COL_NULL        = 'N'  
COL_TYPE        = 'CHAR'  
COL_LEN        = 3Ø  
END STORE  
IN JANCAT STORE RECORD  
RECTYPE          = 'C'  
COL_TAB_NAME     = 'JANCAT_COLUMNS'  
COL_NAME         = 'TABLE_NAME'  
COL_FIELD_NAME   = 'COL_TAB_NAME'  
COL_FIELD_OCC    = 1  
COL_NUM         = 1  
COL_NULL        = 'N'  
COL_TYPE        = 'CHAR'  
COL_LEN        = 3Ø  
END STORE  
IN JANCAT STORE RECORD  
RECTYPE          = 'C'  
COL_TAB_NAME     = 'JANCAT_COLUMNS'  
COL_NAME         = 'COLUMN_NAME'  
COL_FIELD_NAME   = 'COL_NAME'  
COL_FIELD_OCC    = 1  
COL_NUM         = 2  
COL_NULL        = 'N'  
COL_TYPE        = 'CHAR'  
COL_LEN        = 3Ø  
END STORE  
IN JANCAT STORE RECORD  
RECTYPE          = 'C'  
COL_TAB_NAME     = 'JANCAT_COLUMNS'  
COL_NAME         = 'FIELD_NAME'  
COL_FIELD_NAME   = 'COL_FIELD_NAME'  
COL_FIELD_OCC    = 1  
COL_NUM         = 3  
COL_NULL        = 'N'  
COL_TYPE        = 'CHAR'  
COL_LEN        = 5Ø  
END STORE  
IN JANCAT STORE RECORD  
RECTYPE          = 'C'  
COL_TAB_NAME     = 'JANCAT_COLUMNS'  
COL_NAME         = 'FIELD_OCCURRENCE'  
COL_FIELD_NAME   = 'COL_FIELD_OCC'
```

```

COL_FIELD_OCC      = 1
COL_NUM            = 4
COL_NULL          = 'N'
COL_TYPE          = 'INT'
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'C'
COL_TAB_NAME      = 'JANCAT_COLUMNS'
COL_NAME          = 'COLUMN_NUMBER'
COL_FIELD_NAME    = 'COL_NUM'
COL_FIELD_OCC     = 1
COL_NUM           = 5
COL_NULL         = 'N'
COL_TYPE         = 'INT'
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'C'
COL_TAB_NAME      = 'JANCAT_COLUMNS'
COL_NAME          = 'COLUMN_TYPE'
COL_FIELD_NAME    = 'COL_TYPE'
COL_FIELD_OCC     = 1
COL_NUM           = 6
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 8
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'C'
COL_TAB_NAME      = 'JANCAT_COLUMNS'
COL_NAME          = 'COLUMN_LENGTH'
COL_FIELD_NAME    = 'COL_LEN'
COL_FIELD_OCC     = 1
COL_NUM           = 7
COL_NULL         = 'N'
COL_TYPE         = 'INT'
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'C'
COL_TAB_NAME      = 'JANCAT_COLUMNS'
COL_NAME          = 'NULLS_ALLOWED'
COL_FIELD_NAME    = 'COL_NULL'
COL_FIELD_OCC     = 1
COL_NUM           = 8
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE           = 'C'
COL_TAB_NAME      = 'JANCAT_COLUMNS'
COL_NAME          = 'DATE_TIME_FORMAT'
COL_FIELD_NAME    = 'COL_DATEFORMAT'
COL_FIELD_OCC     = 1
COL_NUM           = 9
COL_NULL         = 'Y'
COL_TYPE         = 'CHAR'
COL_LEN          = 3Ø

```

```
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_COLUMNS'
COL_NAME         = 'AT_MOST_ONE'
COL_FIELD_NAME   = 'COL_AT_MOST_ONE'
COL_FIELD_OCC    = 1
COL_NUM          = 10
COL_NULL         = 'Y'
COL_TYPE         = 'CHAR'
COL_LEN          = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_USER_GROUP'
COL_NAME         = 'GROUP_IDENTIFIER'
COL_FIELD_NAME   = 'GRP_GROUP'
COL_FIELD_OCC    = 1
COL_NUM          = 1
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 10
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_USER_GROUP'
COL_NAME         = 'USERID'
COL_FIELD_NAME   = 'GRP_USER'
COL_FIELD_OCC    = 0
COL_NUM          = 2
COL_NULL         = 'N'
COL_TYPE         = 'CHAR'
COL_LEN          = 10
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME         = 'TABLE_NAME'
COL_FIELD_NAME   = 'SEC_TAB_NAME'
COL_FIELD_OCC    = 1
COL_NUM          = 1
COL_NULL         = 'Y'
COL_TYPE         = 'CHAR'
COL_LEN          = 30
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME         = 'USER_GROUP_NAME'
COL_FIELD_NAME   = 'SEC_UGNAME'
COL_FIELD_OCC    = 1
COL_NUM          = 2
COL_NULL         = 'Y'
COL_TYPE         = 'CHAR'
COL_LEN          = 10
END STORE
IN JANCAT STORE RECORD
```

```
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME        = 'USER_GROUP_TYPE'
COL_FIELD_NAME   = 'SEC_UGTYPE'
COL_FIELD_OCC    = 1
COL_NUM         = 3
COL_NULL        = 'Y'
COL_TYPE        = 'CHAR'
COL_LEN         = 5
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME        = 'SELECT_PRIVILEGE'
COL_FIELD_NAME   = 'SEC_SELECT'
COL_FIELD_OCC    = 1
COL_NUM         = 4
COL_NULL        = 'Y'
COL_TYPE        = 'CHAR'
COL_LEN         = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME        = 'UPDATE_PRIVILEGE'
COL_FIELD_NAME   = 'SEC_UPDATE'
COL_FIELD_OCC    = 1
COL_NUM         = 5
COL_NULL        = 'Y'
COL_TYPE        = 'CHAR'
COL_LEN         = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME        = 'INSERT_PRIVILEGE'
COL_FIELD_NAME   = 'SEC_INSERT'
COL_FIELD_OCC    = 1
COL_NUM         = 6
COL_NULL        = 'Y'
COL_TYPE        = 'CHAR'
COL_LEN         = 1
END STORE
IN JANCAT STORE RECORD
RECTYPE          = 'C'
COL_TAB_NAME     = 'JANCAT_SECURITY'
COL_NAME        = 'DELETE_PRIVILEGE'
COL_FIELD_NAME   = 'SEC_DELETE'
COL_FIELD_OCC    = 1
COL_NUM         = 7
COL_NULL        = 'Y'
COL_TYPE        = 'CHAR'
COL_LEN         = 1
END STORE
```



---

**APPENDIX B** *Datetime Processing Considerations*

This chapter presents date processing issues, including usage of *Janus Specialty Data Store* 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 *Janus Specialty Data Store*.

*Janus Specialty Data Store* 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 *Janus Specialty Data Store* is under a rental or trial agreement
- as values of DATETIME type fields

Please note that in addition to the above date processing performed by *Janus Specialty Data Store*, you also can use it to update, retrieve, and perform other SQL operations on *Model 204* fields which might contain two digit year date values and which you have not declared in JANCAT as DATETIME fields. 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 *Janus Specialty Data Store* past the year 1999, version 4.6 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 *Janus Specialty Data Store*, 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 *Janus Specialty Data Store* to operate properly.

If a column is declared in JANCAT as type DATETIME, *Janus Specialty Data Store* will convert all values that are retrieved from or operate on this field to the date format declared for the field. If a DATETIME column has a two digit year format, all values will be interpreted using a CENTSPAN of -50.

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.

## B.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 (e.g., “YYYY” for a 4 digit year, and “MI” for minutes) and separator characters (e.g., “/” in “MM/DD/YY” for two-digit month, day, and year separated by slashes).

These *datetime format* strings are used in many Sirius Software products in addition to *Janus Specialty Data Store*. 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 Sirius products, though certain uses of these strings might impose extra restrictions. For example, a field value with a leading blank may be a valid date matching an HH, DD, or MM token in *Janus Specialty Data Store*, but it may not in some cases in other Sirius products.

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

1. If a format string contains a numeric datetime token (i.e. “ND”, “NM”, or “NS”), then the format string must consist of only one token. Numeric datetime tokens are only supported in format strings for the *Sir2000 Field Migration Facility*.
2. You must specify at least one time, weekday, or date token.
3. Except for “weekday”, you can't specify redundant information. More specifically this means
  - Except for “l”, 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 represents 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. The maximum length of a format string in *Janus Specialty Data Store* is 31 characters; in most other Sirius products it 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, that is, the result when a value from an SQL client is converted to operate on a DATETIME field. The **input format rules** which convert a value from a DATETIME field are less strict; for example, all of the tokens which convert **from** an alphabetic string (e.g., “MON”) will properly convert a value of the field which contains any case string (e.g., “jan” or “JAN” or “Jan”).

<b>NM</b>	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 <i>Sir2000 Field Migration Facility</i> .)
<b>NS</b>	numeric datetime value containing the number seconds since January 1, 1900 at 12:00 AM. (This token is allowed only in the <i>Sir2000 Field Migration Facility</i> .)
<b>ND</b>	numeric date value containing the number of days since January 1, 1900. (This token is allowed only in the <i>Sir2000 Field Migration Facility</i> .)
*	Ignore entire variable-length substring matching pattern, if any. See <a href="#">“Datetime and format examples” on page 76</a> .
I	Ignore corresponding input character. See <a href="#">“Datetime and format examples” on page 76</a> .
"	Following character is “quoted”, that is, it acts as a separator character. See <a href="#">“Datetime and format examples” on page 76</a> . This token is available starting with version 5.2 of the <i>Sirius Mods</i> .
<b>YYYY</b>	4 digit year
<b>YY</b>	2 digit year
<b>CYY</b>	Year minus 1900 (3 digits, including any leading zero). See <a href="#">“Datetime and format examples” on page 76</a> .
<b>ZYY</b>	Year minus 1900, two-digit or three-digit year number (variable length data). See <a href="#">“Datetime and format examples” on page 76</a> .

<b>MONTH</b>	Full month name (upper case variable length). When used to convert <b>from</b> a DATETIME field value, this is the same as Month.
<b>Month</b>	Full month name (mixed case variable length). When used to convert <b>from</b> a DATETIME field value, this is the same as MONTH.
<b>MON</b>	Three character month abbreviation (upper case). When used to convert <b>from</b> a DATETIME field value, this is the same as Mon.
<b>Mon</b>	Three character month abbreviation (mixed case). When used to convert <b>from</b> a DATETIME field value, this is the same as MON.
<b>MM</b>	Two-digit month number. When used to convert <b>from</b> a DATETIME field value, this is the same as BM (leading blank is allowed). See <a href="#">“Datetime and format examples” on page 76</a> .
<b>BM</b>	Two-character month number; when used to convert <b>from</b> a DATETIME field value, <b>from</b> a string, this is the same as MM. See <a href="#">“Datetime and format examples” on page 76</a> . This token is available starting with version 5.2 of the <i>Sirius Mods</i> .
<b>DDD</b>	Three digit Julian day number
<b>DD</b>	Two-digit day number. When used to convert <b>from</b> a DATETIME field value, this is the same as BD (leading blank is allowed). See <a href="#">“Datetime and format examples” on page 76</a> .
<b>BD</b>	Two-character day number; when used to convert <b>from</b> a DATETIME field value, this is the same as DD. See <a href="#">“Datetime and format examples” on page 76</a> . This token is available starting with version 5.2 of the <i>Sirius Mods</i> .
<b>DAY</b>	One-digit or two-digit day number (variable length data). See <a href="#">“Datetime and format examples” on page 76</a> .
<b>WKDAY</b>	Full day of week name (upper case variable length). when used to convert <b>from</b> a DATETIME field value, this is the same as Wkday.
<b>Wkday</b>	Full day of week name (mixed case variable length). when used to convert <b>from</b> a DATETIME field value, this is the same as WKDAY.
<b>WKD</b>	Three character day of week abbreviation (upper case). When used to convert <b>from</b> a DATETIME field value, this is the same as Wkd.
<b>Wkd</b>	Three character day of week abbreviation (mixed case). When used to convert <b>from</b> a DATETIME field value, this is the same as WKD.
<b>HH</b>	Two-digit hour number. When used to convert <b>from</b> a DATETIME field value, this is the same as BH (leading blank is allowed). See <a href="#">“Datetime and format examples” on page 76</a> .
<b>BH</b>	Two-character hour number; When used to convert <b>from</b> a DATETIME field value, this is the same as HH. See <a href="#">“Datetime and format examples” on page 76</a> . This token is available starting with version 5.2 of the <i>Sirius Mods</i> .
<b>MI</b>	Two-digit minute number
<b>SS</b>	Two-digit second number
<b>X</b>	Tenths of a second
<b>XX</b>	Hundredths of a second
<b>XXX</b>	Thousandths of a second (milliseconds)
<b>AM</b>	AM/PM indicator
<b>PM</b>	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 preceded by the quoting character (“”).

See [“Datetime and format examples” on page 76](#) for examples which include use of various separator characters.

Decimal digits as separator characters, and the quoting character are available starting with version 5.2 of the *Sirius Mods*.

## **B.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; e.g., 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.





## **B.4 Datetime and format examples**

There is an extensive set of format tokens, as shown in “[Datetime Formats](#)” on page 70. 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. Each example format is explained and also presented with some matching datetimes; again, bear in mind that these tokens can be combined in very many ways and only a very few are shown here. It is assumed that these examples are invoked sometime between the years 1998-2040, as the basis for relative CENTSPAN calculations.

**YYMMDD** This is the common 6-digit date format which supports sort order if all dates are within a single century. The value 960229 can be stored in a field with this format.

### **YYYYMMDD**

This is the common 8-digit date format which supports sort order with dates in 2 centuries. The value 19921212 can be stored in a field with this format.

### **MM/DD/YY**

This is the U.S. 6-digit date format for display. The value 12/14/94 can be stored in a field with this format.

Notes:

- A field with this format can have a value with a leading blank corresponding to MM, thus allowing “ 7/15/98”, but any value stored in the field by *Janus Specialty Data Store* will have a leading zero.
- To define a field into which *Janus Specialty Data Store* will store a leading blank for the month, use the BM token instead of MM.

### **DD.MM.YY**

This is a European 6-digit date format for display. The value 14.12.94 can be stored in a field with this format.

Notes:

- A field with this format can have a value with a leading blank corresponding to DD, thus allowing “ 7/04/89”, but any value stored in the field by *Janus Specialty Data Store* will have a leading zero.
- To define a field into which *Janus Specialty Data Store* will store a leading blank for the day, use the BD token instead of DD.

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

This is a format which would be quite unlikely in a file, but just to round out the discussion of some date format tokens, the value “Friday, 7 February 1998 AT 21:33” can be stored in a field with this format.

Notes:

- If a format contains AM or PM, then the time (HH:MI) must be between 00:01 and 12:00 and must be accompanied by either AM or PM.
- The day number (string matching DAY) may have a leading zero.
- A field with this format can have a value with a leading zero corresponding to DAY, thus allowing “Friday, 07 February 1998 AT 21:33”, but any day value less than 10 stored in the field by *Janus Specialty Data Store* will have a one-digit value for DAY.
- A field with this format can have a value with a leading blank corresponding to HH, thus allowing

Friday, 31 February 1998 AT 8:33

but any hour value less than 10 stored in the field by *Janus Specialty Data Store* will have only one blank after the string “AT”.

- To define a field into which *Janus Specialty Data Store* will store a leading blank for the hour, use the BH token instead of HH.

**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. The value 92A123 can be stored in a field with this format.

**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. The values 92 and 92ABC123 can be stored in a field with this format.

Note:

- At most one occurrence of the \* token may appear in a datetime format.

**CYYDDD**

This is a compact 6-digit date format with explicit century information, from 1900 through and including 2899. The value 097031 (representing 31 January 1997) can be stored in a field with this format.

**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. The values 970501 (representing 1 May 1997) and 1000501 (representing 1 May 2000) can be stored in a field with this format.

Notes:

- A field with this format can have a value with a leading zero corresponding to ZYY, thus allowing 097051, but any year value less than 2000 stored in the field by *Janus Specialty Data Store* will result in a 6 digit value stored in the field.

**YY0000** Decimal digits can be used as separator characters. The value 980000 can be stored in a field with this format.

Notes:

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

---

## *Index*

**A**

ADDCA, JANUS subcommand ... 14  
 ALLOCC parameter, JANUS DEFINE ... 18  
 AUDTERM parameter  
   JANUS DEFINE command ... 18  
 AUTOLOAD parameter, JANUS DEFINE ... 18

**B**

BINDADDR parameter, JANUS DEFINE ... 19  
 BSIZE parameter, JANUS DEFINE ... 19

**C**

Cataloging *Model 204* data as SQL tables ... 37  
   SDS Table catalog facility ... 37  
 Cataloging *Model 204* data for SDS ... 37  
 CENTSPAN ... 74  
 Certificate, SSL ... 26  
   in SSL cache ... 28, 32  
   requested by server ... 29  
 CHARSET, JANUS subcommand ... 14  
 Client certificate ... 27, 29  
 CLSOCK, JANUS subcommand ... 14  
 CMD parameter  
   JANUS DEFINE command ... 19  
 CONFIGURATION, JANUS subcommand ... 14  
 Connection limit ... 17  
 Content type  
   client Post data ... 24-25

**D**

Date processing ... 69  
 DEFINE, JANUS subcommand ... 14  
 DEFINEIPGROUP, JANUS subcommand ... 14  
 DEFINEREMOTE, JANUS subcommand ... 14  
 DEFINEUSGROUP, JANUS  
   subcommand ... 14  
 DELCA, JANUS subcommand ... 14  
 DELETE, JANUS subcommand ... 14  
 DELETEIPGROUP, JANUS  
   subcommand ... 14  
 DELETEREMOTE, JANUS subcommand ... 14

DELETEUSGROUP, JANUS  
   subcommand ... 15  
 DISPLAY, JANUS subcommand ... 15  
 DISPLAYCA, JANUS subcommand ... 15  
 DISPLAYREMOTE, JANUS  
   subcommand ... 15  
 DISPLAYSOCK, JANUS subcommand ... 15  
 DISPLAYWEB, JANUS subcommand ... 15  
 DISPLAYXT, JANUS subcommand ... 15  
 DOMAIN, JANUS subcommand ... 15  
 DRAIN, JANUS subcommand ... 15

**E**

Encoding  
   HTML form ... 24-25

**F**

FDWOL parameter, JANUS DEFINE ... 20  
 FORCE, JANUS subcommand ... 15  
 FTP, JANUS subcommand ... 15

**I**

IBSIZE parameter, JANUS DEFINE ... 20  
 Installation ... 4  
 Introduction to JANUS ... 1

**J**

JANCAT ... 2, 37-39, 42, 46, 52, 54-57  
   Add Users to Table screen ... 57  
   Autobuild screen ... 42  
   Column Update screen ... 52  
   Columns display ... 46  
   Modify Security Group update screen ... 55  
   Open file screen ... 38  
   SDS Table catalog facility ... 37  
   Security Entries screen ... 56  
   Security Groups screen ... 54  
   Tables screen ... 39  
 JANCAT parameter, JANUS DEFINE ... 21  
 Janus commands  
   introduction to ... 13  
   wildcards used with ... 13

JANUS concepts ... 7

JANUS, introduction to

    Janus IFDIAL Library ... 1

*Janus TCP/IP Base* ... 1

## **K**

Keepalive connection, TCP ... 34

## **L**

LANGUAGE, JANUS subcommand ... 15

LIMITS, JANUS subcommand ... 15

LOADXR, JANUS subcommand ... 15

## **M**

MASTER parameter, JANUS DEFINE ... 21

    defining OPEN CLIENT ports ... 21

MAXCURS parameter, JANUS DEFINE ... 21

MAXSAVE parameter, JANUS DEFINE ... 22

*Model 204* resource requirements ... 18

    buffer space requirements ... 18

## **N**

NAMESERVER, JANUS subcommand ... 15

Nested Tables ... 41, 48

    defined ... 41, 48

    special characters '\*' and '#' ... 48

NOAUDTERM parameter

    JANUS DEFINE command ... 22

NOUPCASE parameter ... 22

    Converting client data to upper case ... 22

## **O**

OBSIZE parameter, JANUS DEFINE ... 23

Omni Access Module ... 8

OMNIACCT parameter, JANUS DEFINE ... 23

OMNIUSER parameter, JANUS DEFINE ... 23

OPEN parameter

    JANUS DEFINE command ... 23

OPENSERV port type ... 17

## **P**

Performance ... 20, 23

Port, Janus

    definition ... 16

Ports ... 7

PRELOGINUSER parameter, JANUS

    DEFINE ... 24

## **R**

RAWINPUT parameter, JANUS DEFINE ... 24

RAWINPUTONLY parameter, JANUS

    DEFINE ... 25

RBSIZE parameter, JANUS DEFINE ... 26

RELOAD, JANUS subcommand ... 15

## **S**

Sdaemon threads

    for SDS catalog ... 23, 26

    limit per Online ... 17

SDS catalog SDAEMON ... 10, 23, 26

    specifying account and user ID ... 10, 23, 26

SDS port type ... 17

SDSACCT parameter ... 26

SDSUSER parameter ... 26

Server ports ... 7

SIRIUS file ... 4

*Sirius Mods* ... 4

Specialty Data Store ... 8, 37

    JANCAT ... 37

Specialty Data Store catalog file structure ... 59

SRVSOCK, JANUS subcommand ... 15

SSL certificate ... 29

*See also* Certificate, SSL

SSL parameter, JANUS DEFINE ... 26

SSLBSIZE parameter, JANUS DEFINE ... 27

SSLCACHE parameter, JANUS DEFINE ... 28

SSLCIPH parameter, JANUS DEFINE ... 29

SSLCLCERT parameter, JANUS DEFINE ... 29

SSLCLCERTR parameter, JANUS

    DEFINE ... 29

SSLIBSIZE parameter, JANUS DEFINE ... 30

SSLMAXAGE parameter, JANUS

    DEFINE ... 31

SSLMAXCERTL parameter, JANUS

    DEFINE ... 32

SSLOBSIZE parameter, JANUS DEFINE ... 32

SSLPROT parameter, JANUS DEFINE ... 33

SSLSTATUS, JANUS subcommand ... 15

SSLUNENC parameter, JANUS DEFINE ... 33

START, JANUS subcommand ... 15

STATUS, JANUS subcommand ... 15

STATUSCA, JANUS subcommand ... 16

STATUSREMOTE, JANUS subcommand ... 16

**T**

TCP keepalives ... 34  
TCPKEEPALIVE parameter, JANUS  
    DEFINE ... 34  
TCPLOG, JANUS subcommand ... 16  
Terminal output ... 18, 22  
TIMEOUT parameter  
    JANUS DEFINE ... 35  
Timeouts, session ... 35  
TNSERV port type  
    TCPKEEPALIVE processing ... 34  
TRACE parameter ... 35  
TRACE, JANUS subcommand ... 16  
Translation, character set ... 36  
TSTATUS, JANUS subcommand ... 16

**U**

UL/SPF ... 4  
UPCASE parameter ... 36  
    Converting client data to upper case ... 36  
Userid and password ... 22, 36  
    Converting to upper case ... 22, 36

**W**

WEB, JANUS subcommand ... 16  
WEBSERV port type ... 17

**X**

XTAB parameter  
    JANUS DEFINE command ... 36



**Figures**

**Figures**

