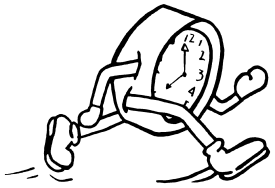
PJS

Personal Job Scheduler



Installation Guide

Release 3.1

Draft 2/9/05

Personal Job Scheduler (PJS) 3.1.0 was released for distribution in February, 2005.

Table of Contents

Figures

About This Manual

This manual describes how to install and maintain the Personal Job Scheduler (PJS) Release 3.1. It is intended for use by the system programmer responsible for installation of the PJS software. Also, the PJS Administrator, responsible for the daily operation and administration of PJS, should be familiar with the information contained in this manual.

Acknowledgments

PJS was written by Tim Henness

The original version of this manual was written by Matthew ???? and Tim Henness. Extensive revisions have been made by Tim Henness

PJS Software Copyright and License

Comments and Suggestions

The author welcomes your comments and suggestions. He can be contacted at:

Tim Henness
Northrop Grumman IT
Internal Information Services
Bldg. 521-2
4101 Washington Ave.
Newport News, VA 23607

E-mail: Tim.Henness@ngc.com

Organization of This Manual

This manual contains the following chapters:

Chapter 1 "PJS Requirements" describes the hardware and software requirements for running PJS.

Chapter 2 "PJS Installation" describes how to install PJS 3.1, either as a new installation, or as an upgrade from a previous release. Instructions are given for both SMP/E and non-SMP/E installation.

Chapter 3 "The PJS Options Module" describes how to create the PJSOPT module, which defines certain insatallation options to PJS.

Chapter 4 "PJS Exits" describes how to write exit routines that can provide security and customize how PJS works.

Chapter 5 "PJS Security" describes how to secure PJS.

Chapter 6 "PJS Operator Commands" describes the operator commands that can be used to start, stop, and control the PJS System Task.

Chapter 7 "PJS Utilities" describes how to use the PJS Utilities that can be used to install and maintain PJS.

Appendix A "PJS TSO Commands" lists the PJS TSO commands and their security access requirements.

Appendix B "PJS Data Areas" documents the format of the PJS Request Queue records, as well as certain PJS control blocks.

Appendix C "Summary of Changes" documents the changes made for each release of PJS.

Related Publications

Other PJS Documentation

Personal Job Scheduler (PJS) User's Guide

Personal Job Scheduler (PJS) Messages and Codes

OS/390 and z/OS Documentation

MVS Initialization and Tuning Reference
OS/390 - SC28-1752, z/OS - SA22-7592

MVS System Commands
OS/390 - GC28-1781, z/OS - SA22-7627

TSO/E Command Reference
OS/390 - SC28-196, z/OS - SA22-7782

TSO/E Customization
OS/390 - SC28-1965, z/OS - SA22-7783

ISPF Planning and Customizing
OS/390 - SC28-1298, z/OS - GC34-4814

ISPF Dialog Developer's Guide and Reference
OS/390 - SC28-1273, z/OS - SC34-4821

ISPF Services Guide
OS/390 - SC28-1272, z/OS - SC34-4819

Communications Server: IP User's Guide and Commands
OS/390 - GC31-8514, z/OS - SC31-8780

Notational Symbols

The following conventions are used in command formats throughout this manual:

BOLD UPPERCASE	is used to display commands or keywords you must code exactly as shown, for example, **SEND FILENAME.TXT**.
italic lowercase	is used to display information you must supply, for example, **SEND** *filename.txt*.
Underscores	either show a default value in a command description, display a default value in a screen image, or represent a highlighted word in a screen image.
Brackets [ ]	mean that you can select one of the items enclosed by the brackets; none of the enclosed items is required.
Braces { }	mean that you must select one of the items enclosed by the braces.
Vertical Bar |	separates options. One vertical bar separates two options, two vertical bars separate three options, and so on. You can select only one of the options.
Ellipsis . . .	means that you can repeat the word or clause that immediately precedes the ellipsis.

Chapter 1: PJS Requirements

Software Requirements

PJS Release 3.1 is intended to run on any currently supported release of OS/390 or z/OS. It may also run on older releases, but no effort will be made to maintain compatibility with unsupported operating systems.

The sample installation exits may have other software requirements. These requirements are noted in the documentation for the individual exits.

Hardware Requirements

PJS has no special hardware requirements, other than those required by its prerequisite software.

PJS uses the Immediate and Relative Instructions Facility, and may not run on a processor that does not support these instructions.

DASD Storage Requirements

PJS requires approximately 450 tracks on a 3390 for the software. Installation with SMP/E will require approximately an additional 500 tracks for DLIBs and SMP/E control data sets. In addition, approximately 650 tracks for non-SMP/E installation, or 1000 tracks for SMP/E installation, are required during the installation process, but can be deleted after installation is complete.

The PJS Request Queue is a VSAM KSDS that holds the scheduling information for all PJS users. Space requirements will vary for each installation, but 1 or 2 cylinders should be a sufficient starting point in most cases.

The PJS JCL Spool is a PDS (or PDSE) and needs to be large enough to hold all of the saved JCL for all of the PJS users, plus extra space for expansion between PDS compression. The directory requires 1 block for every 7 members.

The PJS Message History Log requirements may very, depending on how many jobs are submitted by PJS and how long messages are retained. Each job submitted will require approximately 250 bytes. Multiply the number of jobs submitted in your

message retention period by 250, then add an additional amount for jobs that have been submitted during that period. (PJS will always retain messages for the last submit for a job request, regardless for the retention period.)

Chapter 2: PJS Installation

PJS Installation Overview

PJS Installation should take between one and three hours. An IPL is not necessarily required, however, it is usually desirable to authorize the PJS load libraries. Also you will probably want to put the PJS TSO command and link libraries in the system link list, although this is not absolutely necessary.

Sample JCL is provided in the EXTRACT member of the PJS Installation Package (see Task 2) and in the PJS Installation JCL Library (see Task 4). You should review all sample JCL before submitting and make any changes appropriate for your installation.

PJS can be installed with or without SMP/E. Select one of the checklists below for the type of installation you wish to perform.

New Installation with SMP/E

Figure 1 lists the steps necessary for a new installation of PJS Release 3.1 with SMP/E.

✓	Task #	Step Description
	1	Obtain the PJS Installation Package
	2	Extract the PJS Installation Package PDS
	3	Review the $COPYRT, $LICENSE, and $README Files
	4	Extract the PJS Installation Files
	5	Extract the PJS Documentation Files
	6	Allocate the PJS Software Data Sets
	7	Prepare the SMP/E Zones
	8	Receive, Apply, and Accept the PJS Function SYSMOD
	11	Create and Install the PJSOPT Module
	12	Create and Install the Installation Exits (Optional)
	13	Add PJS to the Link List or the LPA (Optional)
	14	Authorize the PJS Load Libraries
	15	Allocate and Initialize the PJS Request Queue
	17	Allocate and Initialize the PJS Message History Log (Optional)
	18	Allocate the PJS JCL Spool (Optional)
	19	Set Up the PJS/TSO Interface
	20	Set Up the PJS/ISPF Interface
	21	Create the PJS System Task Procedure
	22	Start the PJS System Task

Figure 1: Checklist for New Installation with SMP/E

Upgrade Installation with SMP/E

Figure 2 lists the steps necessary to upgrade to PJS Release 3.1 (from PJS Release 2.0, 2.1, or 2.2) with SMP/E.

✓	Task #	Step Description
	1	Obtain the PJS Installation Package
	2	Extract the PJS Installation Package PDS
	3	Review the $COPYRT, $LICENSE, and $README Files
	4	Extract the PJS Installation Files
	5	Extract the PJS Documentation Files
	6	Allocate the PJS Software Data Sets
	8	Receive, Apply, and Accept the PJS Function SYSMOD
	11	Create and Install the PJSOPT Module
	12	Create and Install the Installation Exits (Optional)
	16	Convert the PJS R2.x Request Queue
	17	Allocate and Initialize the PJS Message History Log (Optional)
	19	Set Up the PJS/TSO Interface (Review)
	22	Start the PJS System Task

Figure 2: Checklist for Upgrade Installation with SMP/E

New Installation without SMP/E

Figure 3 lists the steps necessary for a new installation of PJS Release 3.1 without SMP/E.

✓	Task #	Step Description
	1	Obtain the PJS Installation Package
	2	Extract the PJS Installation Package PDS
	3	Review the $COPYRT, $LICENSE, and $README Files
	4	Extract the PJS Installation Files
	5	Extract the PJS Documentation Files
	6	Allocate the PJS Software Data Sets
	9	Copy PJS from the PJS Installation Files
	10	Link-Edit the PJS Software
	11	Create and Install the PJSOPT Module
	12	Create and Install the Installation Exits (Optional)
	13	Add PJS to the Link List or the LPA (Optional)
	14	Authorize the PJS Load Libraries
	15	Allocate and Initialize the PJS Request Queue
	17	Allocate and Initialize the PJS Message History Log (Optional)
	18	Allocate the PJS JCL Spool (Optional)
	19	Set Up the PJS/TSO Interface
	20	Set Up the PJS/ISPF Interface
	21	Create the PJS System Task Procedure
	22	Start the PJS System Task

Figure 3: Checklist for New Installation without SMP/E

Upgrade Installation without SMP/E

Figure 4 lists the steps necessary to upgrade to PJS Release 3.1 (from PJS Release 2.0, 2.1, or 2.2) with SMP/E.

✓	Task #	Step Description
	1	Obtain the PJS Installation Package
	2	Extract the PJS Installation Package PDS
	3	Review the $COPYRT, $LICENSE, and $README Files
	4	Extract the PJS Installation Files
	5	Extract the PJS Documentation Files
	6	Allocate the PJS Software Data Sets
	9	Copy PJS from the PJS Installation Files
	10	Link-Edit the PJS Software
	11	Create and Install the PJSOPT Module
	12	Create and Install the Installation Exits (Optional)
	16	Convert the PJS R2.x Request Queue
	17	Allocate and Initialize the PJS Message History Log (Optional)
	19	Set Up the PJS/TSO Interface (Review)
	22	Start the PJS System Task

Figure 4: Checklist for Upgrade Installation without SMP/E

PJS Installation Tasks

Task 1: Obtain the PJS Installation Package

The PJS Installation Package is a single sequential file that contains all the elements of PJS. This file may be obtained as part of a public software distribution tape, such as the MVS CBT Tape, or downloaded from the Internet. If you obtained PJS from the MVS CBT Tape, or another distribution tape, you should refer to the documentation for the distribution tape for specific details of how to extract the PJS Installation Package.

If you download PJS it from the Internet, it may be contained within a Zip file. If this is the case you will need to extract the PJS Installation Package file from the Zip file. In any case whenever the PJS Installation Package, or the Zip file that contains it, is transmitted, typically with FTP, it must be done as BINARY data.

When the PJS Installation Package file (unzipped if necessary) is finally loaded to the OS/390 or z/OS host it is to be installed on, it must be allocated as a physical sequential data set (DSORG=PS), fixed-length records (RECFM=FB), with a logical record length of 80 bytes (LRECL=80), and any appropriate block size. When FTP is used these data set characteristics are typically set using the FTP 'SITE' command. Your installation may also require additional parameters on the 'SITE' command to specify how and where the data set should be allocated. For a complete list of all the options available see the *IBM Communications Server: IP User's Guide* for your release of OS/390 or z/OS.

The example in Figure 5shows the FTP commands that can be used to send the PJS Installation Package from a PC to an OS/390 or z/OS host.

```
C:\>FTP hostname
Connected to hostname.
220-FTPD1 IBM FTP CS V1R4 at hostname, 16:34:20 on 2005-01-04.
220 Connection will close if idle for more than 20 minutes.
User (hostname:(none)): userid
331 Send password please.
Password: password
230 userid is logged on.  Working directory is "userid.".
ftp> CD 'qual'
250 "qual." is the working directory name prefix.
ftp> LCD C:\dir
Local directory now C:\dir.
ftp> QUOTE SITE RECFM=FB LRECL=80 BLKSIZE=0 TRACKS PRI=450 SEC=75
200 SITE command was accepted
ftp> BINARY
200 Representation type is Image
ftp> PUT PJS310.XMIT
200 Port request OK.
125 Storing data set qual.PJS310.XMIT
250 Transfer completed successfully.
ftp: 17842560 bytes sent in 16.44Seconds 1085.12Kbytes/sec.
ftp> QUIT
221 Quit command received. Goodbye.
```

Figure 5: Sending the PJS Installation Package Using FTP

Task 2: Extract the PJS Installation Package PDS

The PJS Installation Package contains a partitioned data set that has been formatted as a TSO TRANSMIT file. This PDS, called the PJS Installation Package PDS may be extracted with the TSO 'RECEIVE' command with the 'INDATASET' parameter.

The example in Figure 6 shows the TSO 'RECEIVE' command that can be used to extract the PJS Installation Package PDS.

```
 READY
RECEIVE INDATASET('qual.PJS310.XMIT')
 INMR901I Dataset qual.DDNAME.PACKAGE from TCH03 on SYSTEME
 INMR906A Enter restore parameters or 'DELETE' or 'END' +
DATASET('qual.PJS310.PACKAGE') NEW
 INMR001I Restore successful to dataset 'qual.PJS310.PACKAGE'
 READY
```

Figure 6: Extracting the PJS Installation Package PDS

Your installation may also require additional parameters on the 'RECEIVE' command prompt response to specify how and where the data set should be allocated. For a complete list of all the options available see the *TSO/E Command Reference* for your release of OS/390 or z/OS.

Figure 7 lists the members of the PJS Installation Package PDS:

Member Name	Description	Format
$COPYRT	PJS Copyright Notice	Text
$LICENSE	GNU General Public License	Text
$README	"Getting Started" Instructions	Text
EXTRACT	Sample job to extract the PJS installation files	Text
PJSDOC	PJS Documentation	Zip
INSTALL	Sample Installation JCL Library	PDS
SMPMCS	SMP/E MCS Statements	Seq
JCLIN	SMP/E JCLIN	PDS
MACLIB	PJS Macro Library	PDS
SRCLIB	PJS Source Library	PDS
MODLIB	PJS Modules Library	PDS
TSOHELP	PJS TSO Help Library	PDS
ISPFPNL	PJS ISPF Panels Library	PDS
ISPFMSG	PJS ISPF Messages Library	PDS
ISPFTBL	PJS ISPF Tables Library	PDS
SAMPLIB	PJS Samples Library	PDS
UCRLIB	PJS User Contributed Routines	PDS

Figure 7: PJS Installation Package Members

The 'Format' column indicates the type of data in each member as follows:

Text = Text file
Zip = Zip file in TSO XMIT format
Seq = Sequential Data Set in TSO XMIT format
PDS = Partitioned Data Set in TSO XMIT format

The $COPYRT, $LICENSE, and $README members are text files that contain important information about PJS that should be reviewed before completing the installation. The EXTRACT member contains a sample job that can be used to create the PJS installation files from the PJS Installation Package. Each of the other members contains one of the PJS installation files that has been formatted as a TSO TRANSMIT file.

Task 3: Review the $COPYRT, $LICENSE, and $README Files

After extracting the PJS Installation Package you should carefully read the $COPYRT and $LICENSE members. These contain the copyright notice and license terms for using PJS. If you do not agree to these terms you will not be authorized to use PJS.

PJS is licensed under the GNU General Public License Version 2, or (at your option) any later version, as published by the Free Software Foundation. This license is widely used for many open-source software products, including Linux. A copy of the GNU GPL is also included in an appendix of this book. For more information about the GNU GPL see the Free Software Foundation website at http:/www.gnu.org/licenses/.

You should also read the $README member. This contains the latest information that might not have been included in the documentation.

Task 4: Extract the PJS Installation Files

The EXTRACT member of the PJS Installation Package PDS contains a job to extract the PJS installation files from the PJS Installation Package PDS. This job will run TSO as a background job and use the TSO 'RECEIVE' command with the 'INDATASET' parameter to extract each of the PJS installation files.

This job should be tailored to suit your installation standards, paying particular attention to the job card, and the allocation parameters on the TSO 'RECEIVE' command prompt responses. For a complete list of all the options available see the *TSO/E Command Reference* for your release of OS/390 or z/OS.

Figure 8 lists the PJS installation files that are created by the EXTRACT job.

Data Set Name	Description	Format
qual.PJS310.PJSDOC	PJS Documentation	Zip
qual.PJS310.INSTALL	Installation JCL Library	PDS
qual.PJS310.SMPMCS	SMP/E MCS Statements	Seq
qual.PJS310.NPJ3100.F1	SMP/E JCLIN	PDS
qual.PJS310.NPJ3100.F2	PJS Macro Library	PDS
qual.PJS310.NPJ3100.F3	PJS Source Library	PDS
qual.PJS310.NPJ3100.F4	PJS Modules Library	PDS
qual.PJS310.NPJ3100.F5	PJS TSO Help Library	PDS
qual.PJS310.NPJ3100.F6	PJS ISPF Panels Library	PDS
qual.PJS310.NPJ3100.F7	PJS ISPF Messages Library	PDS
qual.PJS310.NPJ3100.F8	PJS ISPF Tables Library	PDS
qual.PJS310.NPJ3100.F9	PJS Samples Library	PDS
qual.PJS310.UCRLIB	PJS User Contributed Routines	PDS

Figure 8: PJS Installation Files

These data sets are only required for the installation of PJS. Once installation is complete these data sets may be deleted, though it may be desirable to keep the INSTALL and UCRLIB data sets.

Task 5: Extract the PJS Documentation Files

If you download PJS it from the Internet as a Zip file, the PJS documentation may have been included as a separate file within that Zip file. If this is not the case you will need to get the PJS documentation from the PJS Installation Package.

In Task 4 you should have created a file named '*qual*.PJS310.PJSDOC'. This is a variable-length record sequential data set that is a Zip file containing the PJS documentation files. To extract and view these files you will need to download them to a workstation, typically with FTP. This transfer must be done in BINARY mode.

The example in Figure 9 shows the FTP commands that can be used to receive the PJS documentation Zip file to a PC from an OS/390 or z/OS host.

```
C:\>FTP hostname
Connected to hostname.
220-FTPD1 IBM FTP CS V1R4 at hostname, 16:41:20 on 2005-01-04.
220 Connection will close if idle for more than 20 minutes.
User (hostname:(none)): userid
331 Send password please.
Password: password
230 userid is logged on.  Working directory is "userid.".
ftp> CD 'qual.PJS310'
250 "qual.PJS310." is the working directory name prefix.
ftp> LCD C:\dir
Local directory now C:\dir.
ftp> BINARY
200 Representation type is Image
ftp> GET PJSDOC PJSDOC.ZIP
200 Port request OK.
125 Storing data set qual.PJS310.PJSDOC
250 Transfer completed successfully.
ftp: 3034126 bytes received in 3.70Seconds 820.92Kbytes/sec.
ftp> QUIT
221 Quit command received. Goodbye.
```

Figure 9: Downloading the PJS Documentation Zip File Using FTP

Once the file is on your workstation you will need to extract the documentation files using WinZip, or any other program that extracts Zip files. Once the documents have been extracted the documents shown in Figure 10 will be available.

File Name	Description	Format
pjsinst.html	PJS Installation Guide	HTML
pjsinst.pdf		PDF
pjsinst.sxw		OOo
pjsmsg.html	PJS Messages and Codes	HTML
pjsmsg.pdf		PDF
pjsmsg.sxw		OOo
pjsuser.html	PJS User's Guide	HTML
pjsuser.pdf		PDF
pjsuser.sxw		OOo

Figure 10: PJS Documentation Files

The 'Format' column indicates the type of document in each file as follows:

HTML = HTML Web Page. These files may be viewed with any standard web browser.

OOo = OpenOffice.org 1.1.4 Text Document. These files may be viewed and edited with OpenOffice.org Writer. OpenOffice.org versions for Windows, Macintosh, Linux (X86 and PPC) Solaris (Sparc and X86), and FreeBSD are available free of charge at http://www.openoffice.org.

PDF = Adobe Acrobat Portable Document File. To view these files you will need the free Adobe Acrobat Reader, available at http://www.adobe.com/products/acrobat/readstep2.html.

Task 6: Allocate the PJS Software Data Sets

The data sets required to install PJS depend on whether SMP/E is used. Figure 11 lists the PJS target data sets that are required for all installations. Figure 12 lists the PJS SMP/E distribution data sets that are required only if you will be using SMP/E to install PJS.

The space listed in the tables are minimum requirements. Be sure to add extra space to each data set, to allow for maintenance and expansion. If this is an upgrade installation and you are installing PJS into existing data sets, be sure to check for adequate free space. Some of the data sets may require significantly more space for earlier releases.

If you will be using SMP/E to install PJS, the sample job **ALLOCS** will allocate the required target and distribution data sets. If you will not be using SMP/E, the sample job **ALLOCN** will allocate only the required target data sets.

File Name	Data Set Name	File Description	RECFM	LRECL	# of 3390 Tracks	# of Dir Blks
PJSMAC	*qual*.PJS.MACLIB	PJS Macro Library	FB	80	25	9
PJSSRC	*qual*.PJS.SRCLIB	PJS Source Library	FB	80	217	28
PJSCMD	*qual*.PJS.CMDLIB	PJS TSO Commands Library	U	0	9	15
PJSLINK	*qual*.PJS.LINKLIB	PJS Link Library (Common Routines)	U	0	3	2
PJSLOAD	*qual*.PJS.LOADLIB	PJS Load Library (System Task and Utilities)	U	0	2	2
PJSHELP	*qual*.PJS.TSOHELP	PJS TSO Help Library	FB	80	4	7
PJSPLIB	*qual*.PJS.ISPFPNL	PJS ISPF Panels Library	FB	80	31	34
PJSMLIB	*qual*.PJS.ISPFMSG	PJS ISPF Messages Library	FB	80	4	9
PJSTLIB	*qual*.PJS.ISPFTBL	PJS ISPF Tables Library	FB	80	1	1
PJSSAMP	*qual*.PJS.SAMPLIB	PJS Samples Library	FB	80	11	2

Figure 11: PJS Target Libraries

You can combine the PJSCMD, PJSLINK, and PJSLOAD data sets, or combine these data sets with other data sets by changing the appropriate JCL DD statements and SMP/E DDDEFs. For example, the PJSCMD and PJSLINK DD names can reference SYS1.LPALIB, and the PJSLOAD DD name can reference a common APF authorized non-link-list data set.

The load modules contained in the PJSCMD and PJSLINK libraries are reentrant, so they can be included in the system Link Pack Area (LPA). If you decide to include these reentrant load modules in the LPA, you must perform an IPL with CLPA after the installation to load the modules into the LPA. These modules use about 12K of LPA (below the 16M storage line), and about 192K of ELPA (above the 16M storage line).

File Name	Data Set Name	File Description	RECFM	LRECL	# of 3390 Tracks	# of Dir Blks
APJSMAC	*qual*.PJS.APJSMAC	PJS Macro DLIB	FB	80	25	9
APJSSRC	*qual*.PJS.APJSSRC	PJS Source DLIB	FB	80	217	28
APJSMOD	*qual*.PJS.APJSMOD	PJS Module DLIB	U	0	23	24
APJSHELP	*qual*.PJS.APJSHELP	PJS TSO Help DLIB	FB	80	4	7
APJSPLIB	*qual*.PJS.APJSPLIB	PJS ISPF Panels DLIB	FB	80	31	34
APJSMLIB	*qual*.PJS.APJSMLIB	PJS ISPF Messages DLIB	FB	80	4	9
APJSTLIB	*qual*.PJS.APJSTLIB	PJS ISPF Tables DLIB	FB	80	1	1
APJSSAMP	*qual*.PJS.APJSSAMP	PJS Samples DLIB	FB	80	11	2

Figure 12: PJS SMP/E Distribution Libraries

Task 7: Prepare the SMP/E Zones

Note: Perform this task only if you are using SMP/E.

PJS can be installed into an existing SMP/E zone, such as the MVS zone, if there are no name conflicts. It can also be installed into a separate zone. If you wish to install PJS into an existing zone, the sample job **SMPDDDEF** will define the DDDEFs required by PJS. If you wish to install PJS into a new zone, the sample job **DEFCSI** will create and define a new SMP/E CSI for PJS.

Task 8: Receive, Apply, and Accept the PJS Function SYSMOD

Note: Perform this task only if you are using SMP/E.

Receive, apply, and accept the PJS function SYSMOD (FMID NPJ3100) using your normal SMP/E procedures.

Sample job **SMPRECF** will receive PJS into the global zone. Sample job **SMPAPLYF** will install PJS into the target zone and data sets. Sample job **SMPACPTF** will accept PJS into the distribution zone and data sets.

Task 9: Copy PJS from the PJS Installation Files

Note: Perform this task only if you are **not** using SMP/E.

Sample job **COPYPJS** will copy PJS from the PJS Installation Files to the appropriate PJS target data sets using IEBCOPY.

Task 10: Link-Edit the PJS Software

Note: Perform this task only if you are **not** using SMP/E.

Sample job **LINKPJS** will link-edit the PJS software. The TSO command procedures are link-edited into the PJSCMD library, the common reentrant subroutines are link-edited into the PJSLINK library, and the non-reentrant utilities and the PJS System Task programs are link-edited into the PJSLOAD library.

Task 11: Create and Install the PJSOPT Module

The PJS Options Module (PJSOPT) defines several installation options to PJS. The PJS Options module is fully documented in in Chapter 3.

If this is an upgrade installation, the PJSOPT source should be reviewed for any required changes. The PJSOPT module must then be reassembled and link-edited using the PJS Release 3.1 macro library. Do not attempt to use a PJSOPT module assembled with an earlier release of PJS.

If you are using SMP/E to install PJS, create a USERMOD to install PJSOPT into the target zone. The member **USERMOD** in the PJS Installation Library provides a sample SMP/E USERMOD for installing PJSOPT. This member should be copied under a different name, since the same sample is used for the installation exits.

Sample job **SMPRECU** will receive a USERMOD into the global zone. Sample job **SMPAPLYU** will install the USERMOD into the target zone. We recommend that you do ***not*** accept USERMODs for PJS.

If you are not using SMP/E to install PJS, place the source for PJSOPT into the PJS Source Library. The sample job **ASMOPT** can be used to assemble and link-edit the PJS Options Module.

Task 12: Create and Install the Installation Exits

Note: The use of installation exits is optional.

The PJS installation exits can be used to provide security, or to tailor PJS to an installations own requirements. The PJS installation exits are fully documented in Chapter 4.

If this is an upgrade installation, the PJS installation exit source should be reviewed for any required changes. The exit must then be reassembled and link-edited using the PJS Release 3.1 macro library. Do not attempt to use a PJS installation exit assembled for an earlier release of PJS.

If you are using SMP/E to install PJS, create a USERMOD to install the exit into the target zone. The member **USERMOD** in the PJS Installation Library provides a sample SMP/E USERMOD for installing an exit. This member should be copied under a different name, since the same sample is used for the PJS Options Module and all installation exits.

Sample job **SMPRECU** will receive a USERMOD into the global zone. Sample job **SMPAPLYU** will install the USERMOD into the target zone. We recommend that you do ***not*** accept USERMODs for PJS.

If you are not using SMP/E to install PJS, place the source for the exit into the PJS Source Library. The sample job **ASMEXIT** can be used to assemble and link-edit a PJS installation exit.

Task 13: Add PJS to the Link List or the LPA

Note: This step is optional, but recommended. Perform this task only if this is a new installation, or if it was not done when PJS was previously installed.

It is recommended that the PJS TSO Command Library (PJSCMD) and the PJS Link Library (PJSLINK) be placed in the System Link List. Alternatively, if desired for performance, you may place them in the Link Pack Area (LPA). If neither is done STEPLIB DD statements must be included in your TSO logon PROCs.

If desired, the PJS Load Library (PJSLOAD) can be placed in the System Link List, but there is little reason to do so. The PJSLOAD library cannot be placed in the LPA.

To add a data set to the system link list, include an '**LNKLST ADD**' statement in the PROG*xx* member of SYS1.PARMLIB. You can also add it to the system link list by adding it to the *LNKLSTxx* member of SYS1.PARMLIB. An IPL is required to make this change effective. You may also be able to dynamically update the system link list using the '**SETPROG LNKLST**' command.

To add a data set to the LPA, add it to the *LPALSTxx* member of SYS1.PARMLIB. An IPL (with CLPA) is required to make this change effective. You may also be able to dynamically update the LPA using the '**SETPROG LPA**' command.

Refer to the *MVS Initialization and Tuning Reference* and *MVS System Commands* for your system for detailed instructions.

Task 14: Authorize the PJS Load Libraries

Note: This step is optional, but highly recommended. Perform this task only if this is a new installation, or if it was not done when PJS was previously installed.

It is highly recommended that the PJS System Task run with APF authorization. If PJS runs without authorization, PJS cannot send messages to its users. In addition, the most commonly used security schemes (using the PJS Security Exit) require that the PJS system task be authorized.

The PJS System Task (PJSTASK) is link-edited with AC=1. The PJSTASK and its subroutines are installed in the PJS Load Library (PJSLOAD) and the PJS Link Library (PJSLINK). These libraries should be APF authorized. If these libraries are placed on the system link list, and the system link list is authorized, they may acquire

authorization from that fact. If the PJSLINK library is placed in the LPA, those routines will be considered authorized if they are run from the LPA. Otherwise, the data sets in which these libraries are placed must explicitly authorized.

To authorize a load library, include an '**APF ADD**' statement in the PROG*xx* member of SYS1.PARMLIB. You can also authorize it by adding it to the IEAAPF*xx* member of SYS1.PARMLIB. An IPL is required to make this change effective. You may also be able to dynamically authorize PJS using the '**SETPROG APF**' command.

Refer to the *MVS Initialization and Tuning Reference* and *MVS System Commands* for your system for detailed instructions.

Task 15: Allocate and Initialize the PJS Request Queue

Note: Perform this step only if this is a new installation.

The PJS Request Queue contains records defining the Job Requests, Calendars, and Events specified by PJS users, as well as other control records.

The PJS Request Queue is a VSAM KSDS. The PJS Request Queue must be defined with RECSZ(144,4084), KEYS(17,0), and SHR(2). Most other data set attributes may be changed to optimize VSAM processing. For most installations, 1 or 2 cylinders of space should be sufficient.

The PJS Request Queue is defined with IDCAMS and initialized with the PJSQINIT utility. The PJSQINIT utility is fully documented in **Chapter 7**.

The sample job **DEFPJSQ** can be used to define and initialize the PJS Request Queue data set.

Task 16: Convert the PJS R2.x Request Queue

Note: Perform this step only if this is an upgrade installation.

The format of the PJS Request Queue records has changed from previous releases. A utility is provided to convert PJS Release 2.x records to PJS Release 3.1 records.

The PJS Request Queue is converted using the PJSQCONV utility. The PJSQCONV utility is fully documented in Chapter 7.

The sample job **CONVPJSQ** can be used to perform the PJS Request Queue conversion. Be sure to back up the PJS Release 2.x Request Queue before running the conversion utility.

Task 17: Allocate and Initialize the PJS Message History Log

Note: This step is optional, but recommended.

The PJS Message History Log contains a record of the messages send from the PJS System Task to the TSO user for each job request processed.

The PJS Message History Log is a VSAM ESDS. The PJS Message History Log must be defined with NONINDEXED, RECSZ(132,288), REUSE, and SHR(2). Most other data set attributes may be changed to optimize VSAM processing. For most installations, 1 or 2 cylinders of space should be sufficient.

The PJS Message History Log is defined with IDCAMS and initialized with the PJSHINIT utility. The PJSHINIT utility is fully documented in Chapter 7.

The sample job **DEFHIST** can be used to define and initialize the PJS Message History Log data set.

Task 18: Allocate the PJS JCL Spool

Note: This step is optional, but recommended. Perform this step only if this is a new installation, or if it was not done when PJS was previously installed.

If the PJS JCL Save feature is to be used, the PJS JCL Spool data set must be allocated. The JCL to be saved will be copied to the PJS JCL Spool by the PJS TSO commands and the PJS ISPF interface.

The PJS JCL Spool is a partitioned data set. The PJS JCL Spool must be defined with RECFM=FB, LRECL=80, and any suitable block size. Each JCL member saved has a PDS directory entry. Since each directory entry has 22 bytes of user data, 7 directory entries will fit in each directory block. A PDSE may be used.

The sample job **DEFSPOOL** may be used to allocate the PJS Spool data set.

Task 19: Set Up the PJS/TSO Interface

Note: The list of TSO commands in Appendix A has been updated from previous releases.

To make the PJS/TSO interface available to the user, the PJS TSO Command Library (PJSCMD) and the PJS Link Library (PJSLINK) must be concatenated to the STEPLIB DD of the TSO logon procedure, included in the system link list, or placed in the system link pack area. The PJSHELP library should be concatenated to the SYSHELP DD of the TSO logon procedure.

If you have a security system that restricts TSO commands (like ACF2), you may need to define the PJS TSO commands to your security system. The PJS TSO commands that must be defined are listed in Appendix A.

For more information, please refer to *TSO/E Customization* and your security system documentation.

Task 20: Set Up the PJS/ISPF Interface

Note: Perform this step only if this is a new installation.

To make the ISPF interface available to the user, the PJS ISPF panels, messages, and tables must be made available to ISPF, and a means of selecting PJS must be provided to the user. Each of these may be accomplished in several ways. The best way depends on how your installation manages ISPF.

There are several ways to make the PJS ISPF panels, messages, and tables available to ISPF:

- Copy the PJS panels, messages and tables in existing ISPF libraries (such as SYS1.ISPPLIB, SYS1.ISPMLIB, and SYS1.ISPTLIB).
- Concatenate the PJS libraries to the data sets already allocated to the ISPF libraries (ISPPLIB, ISPMLIB, and ISPTLIB). This could be done either in the TSO procedure, or by a CLIST or REXX exec that executes prior to ISPF.
- Allocate the PJS libraries using the ISPF LIBDEF service. This may require a CLIST or REXX exec to execute prior to the ISPF SELECT for PJS.

Generally, to call the PJS ISPF interface, the following ISPF SELECT is required:

```
SELECT PGM(PJSISPF) PARM(option) NEWAPPL(PJS) NOCHECK
```

where *option* is the initial PJS option to be selected.

To modify an ISPF menu to call the PJS/ISPF interface, the following statements should be included in the **)PROC** section of the menu panel definition:

```
&ZQ = TRUNC(&ZCMD,'.')
&ZTRAIL = .TRAIL
&ZSEL = TRANS( &ZQ
               .    .    .
               n,'PGM(PJSISPF) PARM(&ZTRAIL) NEWAPPL(PJS) NOCHECK'
               .    .    .
               *,'?')
```

Figure 13: ISPF Menu Panel Modifications

Detailed and authoritative information on each method can be found in *ISPF Planning and Customizing*, *ISPF Dialog Developer's Guide and Reference* and *ISPF Dialog Services Guide*. The particular documents required depend on your release of OS/390 or z/OS.

Task 21: Create the PJS System Task Procedure

Note: Perform this step only if this is a new installation.

Place the start-up procedure for the PJS System Task into a system procedure library. A sample procedure is in member **PJSPROC** in the INSTALL library.

Task 22: Start the PJS System Task

Once the PJS system is installed and customized, the PJS System Task can be started. Procedures for starting, stopping, and controlling the PJS System Task are in Chapter 6.

It is recommended that the start command for the PJS System Task be placed in the MVS automatic start-up commands list (member COMMNDxx of SYS1.PARMLIB). For detailed information on how to do this, see the *MVS Initialization and Tuning Reference*.

Chapter 3: The PJS Options Module

One of the PJS installation tasks requires you to create the PJS Options Module (PJSOPT). PJSOPT is a non-executable CSECT that defines certain constants used by the PJS system. Use the following format to build the PJSOPT module:

```
PJSOPT macro specification
END
```

Figure 14: PJSOPT Module Source Format

A PJSOPT module is created by the PJSOPT macro. An assembler END statement must be included at the end.

The PJSOPT module should be assembled and link-edited into a load module of the same name and placed into the PJS Link Library (PJSLINK). The linkage-editor should specify the RENT attribute. The load module produced will be assigned the attributes AMODE=31, and RMODE=ANY.

PJSOPT Macro Format and Operands

The PJSOPT macro has the following format:

```
PJSOPT  PJSQDSN=data-set-name,
        [CMPTIME={7|nnn},]
        [DATEFMT={MDY|DMY},]
        [DISTIME={30|nnn},]
        [DOWNWRN={60|nnn},]
        [ENQNAME={PJSQUEUE|qname},]
        [ERRTIME={7|nnn},]
```

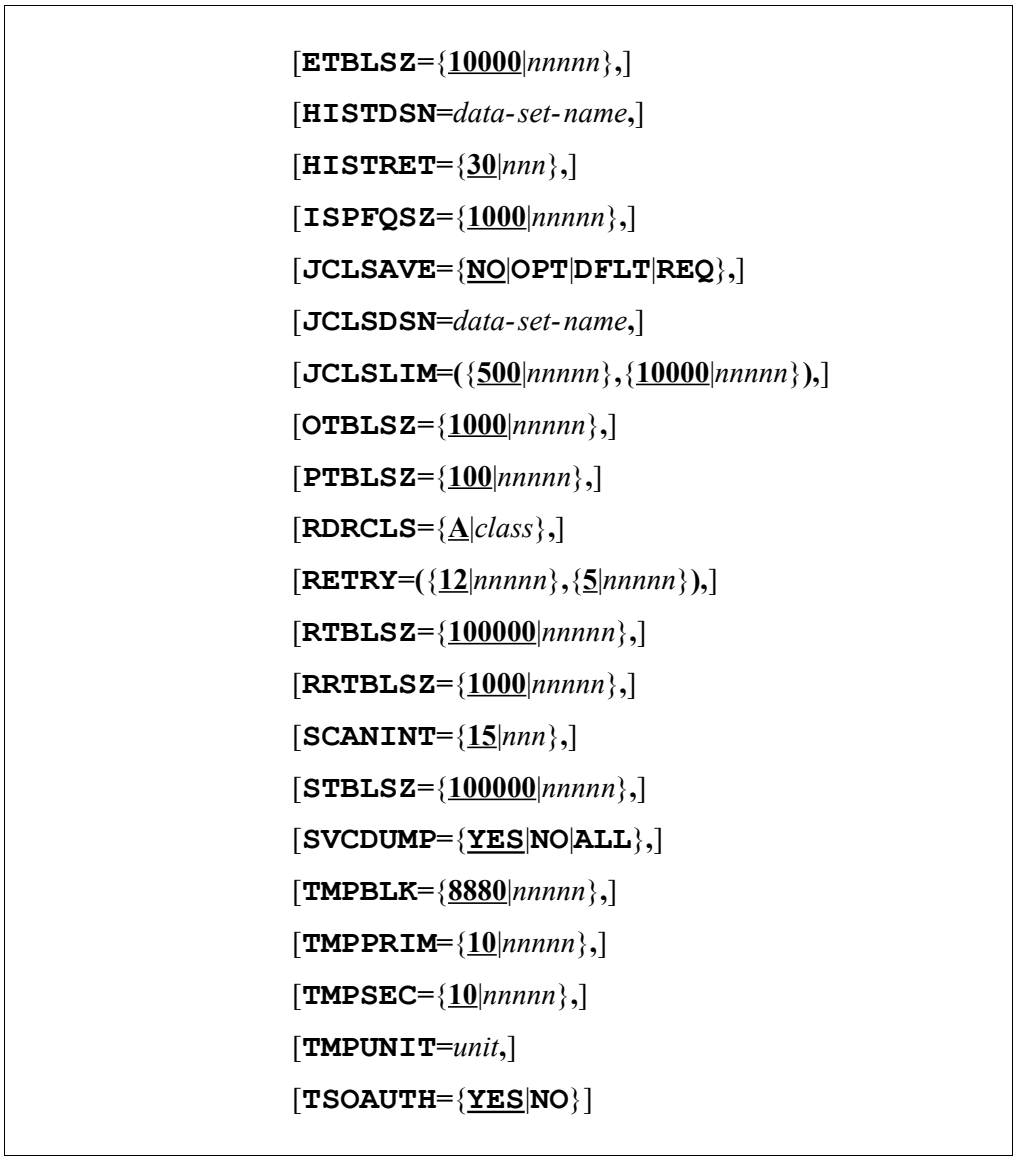


Figure 15: PJSOPT Macro Options

CMPTIME=*nnn*

specifies the number of days a job request can remain in COMPLETED status before it is deleted by the PJS Request Queue Maintenance Utility. A value of **0** indicates no limit. The default is **7** days.

DATEFMT={**MDY**|**DMY**}

specifies the format in which PJS is to display dates, and accept dates for input. **MDY** means dates are displayed and input in the *mm/dd/yyyy* format (commonly used in the U.S.). **DMY** means dates are displayed and input in the *dd/mm/yyyy* format (commonly used in Europe). The default is **MDY**.

DISTIME=*nnn*

specifies the number of days a job request can remain in DISABLED status before it is deleted by the PJS Request Queue Maintenance Utility. A value of **0** indicates no limit. The default is **30** days.

DOWNWRN=*nnn*

specifies the number of minutes the PJS System Task can be down before notifying the operator and prompting for the recovery option when PJS is restarted. The default is **60** minutes.

ENQNAME=*qname*

specifies the 8-character major QNAME that PJS will use when it serializes resources. The default is **PJSQUEUE**.

ERRTIME=*nnn*

specifies the number of days a job request can remain in ERROR status before it is deleted by the PJS Request Queue Maintenance Utility. A value of **0** indicates no limit. The default is **7** days.

ETBLSZ=*nnnnn*

specifies the number of entries to allocate for the PJS Request Queue Maintenance Utility Event Table. One entry is required for each Event in the PJS Request Queue. Each entry requires 17 bytes of virtual storage. The default is **10000** entries.

HISTDSN=*data-set-name*

specifies the data set name of the PJS Message History Log data set. If this parameter is omitted, the PJS Message History Log feature will not be available. There is no default.

HISTRET=*nnn*

specifies the number of days messages are to be retained in the PJS Message History Log before being deleted by the PJS Request Queue Maintenance Utility. The default is **30** days.

ISPFQSZ=*nnnnn*

specifies the number of entries to allocate for the PJS ISPF Interface List Job Requests Table, List Calendars Table, or List Events Table. One entry is required for each Job Request, Calendar, or Event selected by a list dialog. If this value is exceeded, the list is truncated and a system message is displayed. Each entry requires up to 256 bytes. The default is **1000** entries.

JCLSAVE={**NO**|**OPT**|**DFLT**|**REQ**}

specifies whether the PJS JCL Save Feature will be enabled. **NO** means that the PJS JCL Save Feature cannot be used. **OPT** means that the PJS JCL Save Feature can be used, but the default is NOSAVE. **DFLT** means that the PJS JCL Save Feature can be used, and the default is SAVE. **REQ** means that the PJS JCL Save Feature must be used. The default is **NO**.

JCLSDSN=*data-set-name*
specifies the data set name of the PJS JCL Spool data set. This parameter is required, unless **JCLSAVE=NO** is specified or defaulted. There is no default.

JCLSLIM=(*nnnnn***,***nnnnn***)**
specifies the maximum number of JCL records that can be stored in the PJS JCL Spool. The first value is the maximum for a single Job Request. The second value is the maximum for all Job Requests for an Owner. The default is **500** records per Job Request, and **10000** records per Owner.

OTBLSZ=*nnnnn*
specifies the number of entries to allocate for the PJS Request Queue Maintenance Utility Owner Table. One entry is required for each Owner in the PJS Request Queue. Each entry requires 8 bytes of virtual storage. The default is **1000** entries.

PJSQDSN=*data-set-name*
specifies the data set name of the PJS Request Queue. This parameter is required. There is no default.

PTBLSZ=*nnnnn*
specifies the number of entries to allocate for the PJS System Task Post/Reset Event Table. One entry is required for each Event posted within a PJS Short Request Queue Scan Interval (usually about 1 minute). Each entry requires 24 bytes of virtual storage. The default is **100** entries.

RDRCLS=*class*
specifies the SYSOUT Class to be used when allocating an internal reader. The default is **A**.

RETRY=(*nnnnn***,***nnnnn***)**
specifies how PJS os to retry a job submission when a temporary error (i.e. JCL data set in use) occurs. The first value is the maximum number of times the submit is to be retried before placing the request in the **ERROR** status. The second value is the time interval (in minutes) between each retry attempt. **RETRY=(0,0)** means that no retry is to be attempted. The default is **RETRY=(12,5)**.

RTBLSZ=*nnnnn*
specifies the number of entries to allocate for the PJS Request Queue Maintenance Utility Request Table. One entry is required for each Job Request in the PJS Request Queue. Each entry requires 32 bytes of virtual storage. The default is **100000** entries.

RRTBLSZ=*nnnnn*
specifies the number of entries to allocate for the PJS System Task Ready Request Table. One entry is required for each Job Request scheduled within a PJS Long Request Queue Scan Interval, a value set by the **SCANINT=** keyword discussed below. Each entry requires 32 bytes of virtual storage. The default is **1000** entries.

SCANINT=*nnnnn*

specifies the number of minutes in the PJS Long Request Queue Scan Interval. This interval is the time between Request Queue scans by the PJS System Task. Shorter times minimize the size of the PJS System Task Ready Request Table. Longer times reduce system overhead. In PJS Release 1.0, a small value meant a shorter time delay for new or changed requests with run times only slightly after the add or change time. In PJS Release 2.0, and later this is no longer an advantage. The default is **15** minutes.

STBLSZ=nnnnn

specifies the number of entries to allocate for the PJS Request Queue Maintenance Utility Spool Directory Table. One entry is required for each Job Request for which the JCL is saved in the PJS JCL Spool. Each entry requires 80 bytes of virtual storage. The default is **100000** entries.

SVCDUMP={**YES**|**NO**|**ALL**}

specifies whether an SVC dump is to be produced when PJS abends. Regardless of this option, an SVC dump can be produced only when the abending program is APF authorized. **YES** specifies that an SVC dump is to be produced for most PJS abends. (PJS has a table of abends for which dumps are not produced.) **ALL** specifies that an SVC dump is to be produced for all PJS abends, including those spacified in an internal PJS table as not requiring an SVC dump. **NO** specifies that SVC dumps are never to be produced. The default is **YES**.

TMPBLK=*nnnnn*

specifies the block size to be used when allocating a temporary data set to be used for listing or browsing JCL from the PJS JCL Spool. The default is **8880**.

TMPPRIM=*nnnnn*

specifies the number of blocks for the primary space allocation for a temporary data set to be used for listing or browsing JCL from the PJS JCL Spool. The default is **10** blocks.

TMPSEC=*nnnnn*

specifies the number of blocks for the secondary space allocation for a temporary data set to be used for listing or browsing JCL from the PJS JCL Spool. The default is **10** blocks.

TMPUNIT=*unit*

specifies the unit name to be used when allocating a temporary data set to be used for listing or browsing JCL from the PJS JCL Spool. A null value indicates that the user's default TSO allocation unit is to be used. The default is null.

TSOAUTH={YES|NO}

specifies whether the PJS ISPF interface is to invoke the PJS ISPF Interface TSO commands from an authorized environment. You may have to specify **YES** to secure the PJS Request Queue with RACF (or some other security interface) Program Access to Data Set; you may also need to specify the commands in the TSO Authorized Command List. (The PJS TSO Commands are listed in Appendix A). **NO** means that the PJS ISPF Interface Routine will be invoked from an unauthorized environment; this is a more efficient call. The default is **YES**.

PJSOPT Module Example

The following example shows how to use the PJSOPT and PJSAUTH macros to create the PJS Options Module (PJSOPT):

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
         PJSOPT PJSQDSN=PJS.QUEUE,                                      *
                HISTDSN=PJS.HISTORY,                                    *
                JCLSAVE=OPT,                                            *
                JCLDSN=PJS.JCLSPOOL,                                    *
                JCLSLIM=(200,1000),                                     *
                DISTIME=0
         END
```

Figure 16: PJSOPT Module Example

The name of the PJS Request Queue is specified as '**PJS.QUEUE**'. The PJS JCL Spool can be used (but is not required) with the spool data set name specified as '**PJS.JCLSPOOL**'. Users are limited to saving 200 records per job and 1000 per user. The name of the PJS Message History Log is specified as '**PJS.MSGHIST**'. Job requests in DISABLED status will not be deleted by the PJS Request Queue Maintenance Utility.

Chapter 4: PJS Exits

One of the optional installation subtasks in Chapter 2 enables you to modify PJS user exits so that PJS meets site requirements. The following table lists the exits provided by PJS:

Module Name	Module Description
PJSDYNX	PJS Installation Dynamic Variables Exit
PJSIDFX	PJS Installation Data Format Exit
PJSOPTX	PJS Options Exit
PJSSECX	PJS Security Exit
PJSSUBX	PJS Submit Exit

Figure 17: PJS Exits

The PJS Installation Dynamic Variables Exit enables you to define your own dynamic variables for inserting values into the JCL as it is submitted.

The PJS Installation Data Format Exit enables you to format the Installation Data Area from a PJS Job Request Record for display.

The PJS Options Exit enables you to dynamically set options at execution time.

The PJS Security Exit enables you to define and establish access control to PJS request records.

The PJS Submit Exit enables you to automatically modify JCL submitted through PJS and, if needed, to suppress the job.

The default exits perform no significant actions and have no adverse side effects. Even if you do not intend to modify any of these exits, we recommend that you read about them. Please pay particular attention to the section on the PJS Security Exit (PJSSECX); this section and Chapter 5 both contain a great deal of information on PJS site security issues.

Several sample exits are provided in the PJSSAMP library. Each exit is documented in the comments at the start of the program source.

The PJS Installation Dynamic Variables Exit (PJSDYNX)

PJSDYNX is an exit routine that enables the installation to define their own dynamic variables that can be inserted into the JCL as it is submitted. PJSDYNX is passed the Dynamic Value Name specified by the user in the job request. The exit will then compute the replacement value and place it in the area provided.

PJSDYNX is called by the PJS System Task just before the JCL for a job request that specifies a dynamic value is submitted. It is called before the PJS supplied dynamic values are computed, and if this routine provides a value, the PJS supplied routine will not be called. It is also called by the PJS TSO commands and the ISPF interface to verify that a dynamic value name is valid.

The PJSDYNX module must be link-edited with the RENT, AMODE=31, and RMODE=ANY attributes specified. The load module should be placed into the PJSLINK library.

PJSDYNX Input Parameters

The PJSDYNX exit is called using standard IBM linkage conventions. On entry, the registers will contain the following values:

R0	Unpredictable
R1	Address of parameter list
R2 - R11	Unpredictable
R12	Address of PJSVT
R13	Address of 72-byte save area
R14	Return address
R15	Address of PJSDYNX entry point

The parameter list pointer to by register 1 is a list of addresses. The last address in the list has the high-order bit set to '1', indicating the end of the list. The address will point to the following parameters:

Parm 1	PJS Vector Table (PJSVT).
Parm 2	16-character Dynamic Value Name.
Parm 3	Dynamic Value Area.
Parm 4	PJS Job Request Record being processed.

If the Dynamic Variable Value Area (Parm 3) is omitted, the exit should only set the return code indicating if the Dynamic Variable Name is valid.
The format of the Dynamic Variable Value Area (Parm 3) is as follows:

Bytes 0 - 1	Length of Value Area (set by calling program)
Bytes 2 - 3	Length of Value (to be set by the exit)
Bytes 4 - n	Value Area

The first length is the length of the value area (not including the length fields) and is set by the calling program. The second length is the actual length of the value, and is to be set by this routine.

PJSDYNX Return Codes

On return, the caller's registers (except register 15) must be restored. Register 15 should be set to one of the following return codes:

- **0** Value has not been set. If the dynamic variable name is a PJS defined dynamic variable, the PJS provided routine will be called.
- **4** Value has been set.

The PJS Installation Data Format Exit (PJSIDFX)

PJSIDFX is an exit routine that enables the installation to format the data in the installation data area of a PJS Job Request Record for display. For an example of how this information can be displayed, please refer to Chapter 3 in the *Personal Job Scheduler (PJS) User's Guide*. This exit will be called by the PJS TSO command processors and the PJS ISPF interface when a job request record is formatted for display.

The PJSIDFX module must be link-edited with the RENT, AMODE=31, and RMODE=ANY attributes specified. The load module should be placed into the PJSLINK library.

PJSIDFX Input Parameters

The PJSIDFX exit is called using standard IBM linkage conventions. On entry, the registers will contain the following values:

R0	Unpredictable
R1	Address of parameter list
R2 - R11	Unpredictable
R12	Address of PJSVT
R13	Address of 72-byte save area
R14	Return address
R15	Address of PJSIDFX entry point

The parameter list pointer to by register 1 is a list of addresses. The last address in the list has the high-order bit set to '1', indicating the end of the list. The address will point to the following parameters:

Parm 1	PJS Vector Table (PJSVT)
Parm 2	PJS Job Request Record being formatted
Parm 3	Formatted Installation Data Area

If the PJS Job Request Record (Parm 2) is omitted, the exit should place a heading for the installation data into the Formatted Installation Data Area (Parm 3). If the input PJS Job Request Record (Parm 2) is passed, the exit should place the formatted installation data for the record into the Formatted Installation Data Area (Parm 3). The PJS Job Request Record must **not** be changed by the exit.

The format of the Formatted Installation Data Area (Parm 3) is as follows:

Bytes 0 - 1	Length of Value Area (set by calling program)
Bytes 2 - 3	Length of Value (to be set by the exit)
Bytes 4 - n	Value Area

The first length is the length of the value area (not including the length fields) and is set by the calling program. The second length is the actual length of the value, and is to be set by this routine. The length of the value should generally be set to the same length for the heading and all data lines, since this length is used to determine break points for multi-line data and scroll lengths for the ISPF List Job Request dialog.

PJSIDFX Return Codes

On return, the caller's registers (except register 15) must be restored. Register 15 should be set to a return code of zero.

The PJS Installation Options Exit (PJSOPTX)

PJSOPTX is an exit routine that enables the installation to dynamically set installations options at execution time. For example, this exit can enable you to run multiple copies of PJS in some specialized environments.

This exit will be called by the PJS routines when the PJS Installation Options Module is loaded. The information in the options module is copied into the PJS Vector Table (PJSVT) and the exit called. The exit can then test any system control blocks available to it to change the installation options as desired.

The PJSOPTX module must be link-edited with the RENT, AMODE=31, and RMODE=ANY attributes specified. The load module should be placed into the PJSLINK library.

PJSOPTX Input Parameters

The PJSOPTX exit is called using standard IBM linkage conventions. On entry, the registers will contain the following values:

R0	Unpredictable
R1	Address of parameter list
R2 - R12	Unpredictable
R13	Address of 72-byte save area
R14	Return address
R15	Address of PJSOPTX entry point

The parameter list pointer to by register 1 is a list of addresses. The last address in the list has the high-order bit set to '1', indicating the end of the list. The address will point to the following parameters:

Parm 1	PJS Vector Table (PJSVT)

It should be noted that while the other exits are called using a standard 'CALL' (BALR R14, R15), this exit is called using the 'LINK' macro.

PJSOPTX Return Codes

On return, the caller's registers (except register 15) must be restored. Register 15 should be set to a return code of zero.

The PJS Installation Security Exit (PJSSECX)

Note: For more information on PJS security issues, please refer to Chapter 5: PJS Security.

PJSSECX is an exit routine that enables the installation to secure the PJS Request Queue and provide access control to the PJS functions and the request records. This exit will be called by several of the PJS programs at strategic points that can enable the installation to provide complete security. The specific calling point is identified by the function code parameter.

It is recommended that some type of security exit be written. The default exit will always approve all requests - it provides no security.

The PJSSECX module must be link-edited with the RENT, AMODE=31, and RMODE=ANY attributes specified. The load module should be placed into the PJSLINK library.

PJSSECX Input Parameters

The PJSSECX exit is called using standard IBM linkage conventions. On entry, the registers will contain the following values:

R0	Unpredictable
R1	Address of parameter list
R2 - R11	Unpredictable
R12	Address of PJSVT
R13	Address of 72-byte save area
R14	Return address
R15	Address of PJSSECX entry point

The parameter list pointer to by register 1 is a list of addresses. The last address in the list has the high-order bit set to '1', indicating the end of the list. The address will point to the following parameters:

Parm 1 Pointer to a 1-byte function code. This will contain one of the following values:

X'01'	Begin TSO Command Processing
X'02'	End TSO Command Processing
X'03'	TSO Command Abend
X'11'	Begin ISPF Interface Processing
X'12'	End ISPF Interface Processing
X'13'	ISPF Interface Abend
X'21'	Begin Batch Utility Processing
X'22'	End Batch Utility Processing
X'23'	Batch Utility Abend
X'31'	Begin System Task Processing
X'32'	End System Task Processing
X'33'	System Task Abend
X'41'	Begin Job Submit Processing
X'42'	End Job Submit Processing
X'43'	Job Submit Subtask Abend
X'71'	Open JCL Data Set
X'81'	Read Job Request Record
X'82'	Add Job Request Record
X'83'	Update Job Request Record
X'84'	Delete Job Request Record
X'85'	Add Job Request Early ID Check
X'86'	Update Job Request Early ID Check
X'91'	Read Calendar Record
X'92'	Add Calendar Record
X'93'	Update Calendar Record
X'94'	Delete Calendar Record
X'A1'	Read Event Record
X'A2'	Add Event Record
X'A3'	Post Event Record
X'A4'	Reset Event Record
X'A5'	Add Event Early ID Check
X'B1'	Read Global Variable Record
X'B2'	Add Global Variable Record
X'B3'	Update Global Variable Record
X'B4'	Delete Global Variable Record

Parm 2 PJS Vector Table (PJSVT)

The following parameter is passed for function codes X'41', X'42', X'43', X'81', X'82', X'83', X'84', X'85', and X'86':

Parm 3 PJSQ Job Request Record

Be aware that for function code X'85' the Job Request Record will not have been completely initialized. For function code X'86' the call is made prior to any changes. These calls are intended only to verify the user's authority to add or modify the record prior to prompting the user for additional input. These calls may not be made for every add or modify.

The following parameter is passed for function codes X'91', X'92', X'93', and X'94':

Parm 3 PJSQ Calendar Record

The following parameter is passed for function codes X'A1', X'A2', X'A3', X'A4', and X'A5':

Parm 3 PJSQ Event Record

Be aware that for function code X'A5' the Event Record will not have been completely initialized. This call is intended only to verify the user's authority to specify the event prior to completing the update. This call may not be made for every update.

The following parameter is passed for function codes X'B1', X'B2', X'B3', and X'B4':

Parm 3 PJSQ Global Variable Record

The following parameters are passed for function code X'71':

Parm 3 44-byte Data Set Name
Parm 4 8-byte Member Name
Parm 5 6-byte Volume Serial Number

PJSSECX Return Codes

On return, the caller's registers (except register 15) must be restored. Register 15 should be set to one of the following return codes:

0 Access Permitted
4 Access Denied

The PJS Installation Submit Exit (PJSSUBX)

PJSSUBX is an exit routine that enables the installation to examine and, if necessary, modify the submitted JCL or suppress the job. This exit will be called by the PJS each time a JCL image is to be passed to the internal reader.

The PJSSUBX module must be link-edited with the RENT, AMODE=31, and RMODE=ANY attributes specified. The load module should be placed into the PJSLINK library.

PJSSUBX Input Parameters

The PJSSUBX exit is called using standard IBM linkage conventions. On entry, the registers will contain the following values:

R0	Unpredictable
R1	Address of parameter list
R2 - R11	Unpredictable
R12	Address of PJSVT
R13	Address of 72-byte save area
R14	Return address
R15	Address of PJSSUBX entry point

The parameter list pointer to by register 1 is a list of addresses. The last address in the list has the high-order bit set to '1', indicating the end of the list. The address will point to the following parameters:

Parm 1	PJS Vector Table (PJSVT).
Parm 2	PJS Job Request Record being processed.
Parm 3	80-byte JCL image last read by the system.
Parm 4	80-byte JCL image to be inserted by the exit.

After the last JCL image has been passed to the internal reader, the exit will be called with Parm 3 omitted. This enables the exit to add any additional JCL statements required to the end of the job by generating a return code of 4.

The JCL image area pointed to by Parm 4 is initialized to blank characters when the exit is called. The JCL image placed in the area by the exit will be used only if the return code is set to 4.

PJSSUBX Return Codes

On return, the caller's registers (except register 15) must be restored. Register 15 should be set to one of the following return codes:

0 Pass the JCL image pointed to by Parm 3. If Parm 3 is passed, call the PJS Submit Exit again with the next JCL image. If Parm 3 is omitted, the submission is complete and the exit will not be called again.

4 Insert the JCL image pointed to by Parm 4. Call the exit again with the same JCL image.

8 Delete the JCL image pointed to by Parm 3. If Parm 3 is passed, call the PJS Submit Exit again with the next JCL image. If Parm 3 is omitted, the submission is complete and the exit will not be called again.

12 Suppress the job submission. A /*PURGE statement will be passed to the internal reader and the exit will not be called again.

How to Use the PJSSUBX Exit

The PJS Installation Submit Exit can modify the JCL being submitted by using the JCL image area passed by Parm 3 and Parm 4, and by setting the return code:

- To pass an input JCL image without change, set the return code to 0 (zero).
- To modify an input JCL image before submission, change the input JCL image in the area pointed to by Parm 3, then set the return code to 0 (zero).
- To insert a JCL image before the input JCL image, place the JCL image to be inserted into the area pointed to by Parm 4, then set the return code to 4.
- To delete a JCL image, set the return code to 8.
- To suppress the job submission, set the return code to 12. A /*PURGE statement will be passed to the internal reader.

The PJS Vector Table (PJSVT) contains four fullwords labeled PJSVT_USER*n*. The exit can use these fullwords to hold information or point to storage areas between calls to the exit. The exit must perform cleanup before it sets the return code to 0 (zero) or 4 when Parm 3 is omitted, i.e., the call after the final input JCL image. It must also perform cleanup when it sets the return code to 12 to purge the job.

Chapter 5: PJS Security

Some thought must be given to system security and integrity. Without proper precautions, it may be possible for an unauthorized user to use PJS to submit jobs with a different user-ID. Exit points and sample exits are provided to insure a completely secure environment. The specific steps required to provide adequate security will vary greatly from installation to installation, so detailed instructions cannot be given. Rather, this section is intended to give an overview of some of the things that should be considered. Although much of the discussion assumes a RACF environment, it should be apply to other security environments as well.

Securing the PJS Software

The first step to securing PJS is to secure the PJS software itself. All of the PJS libraries, in general, should be protected from unauthorized modifications just as you would protect any other production software system. The PJSLOAD library and the PJS Request Queue should be further secured. The programs contained in the PJSLOAD library are not generally required to be made available to users. The PJSLOAD library should therefore be in a data set with limited access.

The PJS system task (PJSTASK) must be authorized so that it can send messages to online TSO users via the TSO SEND command. Also, many security schemes (including that provided by the sample security exit provided) will require authorization. The PJS Request Queue Maintenance Utility (PJSQMNT) requires authorization to call IEBCOPY to compress the PJS JCL Spool data set. No other programs require authorization, and should not be link-edited with AC=1.

Securing the PJS Data Sets

Most of the effort to secure PJS will involve protecting the PJS Request Queue, the PJS JCL Spool, and the PJS Message History Log data sets from unauthorized access. This is complicated by the fact that the user must be able to update these data sets when using the PJS programs. The user must not, however, be able to update them by any other means. This can be accomplished using RACF program access control through a conditional access list.

To protect the data set with program access control, RACF profiles for the PROGRAM class must be created for the PJS TSO commands in the PJSCMD library (a generic profile name may be used for this purpose). The PROGRAM

profile(s) used must include the PJSCMD and PJSLINE libraries in the ADDMEM clause with the NOPADCHK option. A RACF profile for the PJS request queue data set must be created with a conditional access list granting access when the PJS programs in PJSCMD are being executed. Member RACFPROF in the PJSINST library contains sample RACF commands for creating the RACF profiles. Please note that this is an example only. Some of the statements may not be appropriate for your installation, and other RACF statements may be required. Refer to the appropriate RACF documentation for authoritative information.

The PJS TSO commands (listed in Appendix A) may also need to be defined in the TSO authorized command table (IKJTSO00 in SYS1.PARMLIB, or CSECT IKJEFTE2). (Although the PJS TSO commands do not require authorization, RACF program access control requires the secure environment TSO creates for a program in the authorized command list. Since the programs are not linked with AC=1, no APF authorization is actually conveyed to the program.) Member IKJTSO00 in the PJSINST library contains sample IKJTSO00 parameters to define the PJS TSO commands.

Securing the PJS Request Queue Records

The PJS Installation Security Exit can be used to define the type of access each user can have to each job request, calendar, event, and global variable record in the PJS Request Queue.

When you define access rules, remember that if a user can add or update job requests having an Owner-ID other than her or his own, that user may be able to submit jobs using the other user's User-ID. (This might depend on how you protect submitted batch jobs.) It is recommended that either users be permitted to add and update only job requests with their own User-ID, or that the PJS Installation Security Exit be written to record the RACF User-ID of the last user to update the request be recorded in the PJS Job Request record Installation Data Area, and use that User-ID for data set access and job submission.

Securing Batch Jobs Submitted by PJS

PJS submits batch jobs from the PJS System Task. The main task attaches a subtask to perform the actual submit. The submit subtask will terminate when the submit is complete. The PJS Installation Security exit is called at the beginning of the submit. The security exit can issue a RACINIT macro for the user owning the request record. This will update the submit task's TCB to point to the ACEE created by the RACINIT for the owning user. All JCL data set access will then be done under the user-ID of the job request owner. Alternatively, the PJS Installation Security Exit is also called when the JCL data set is opened.

The PJS system task submits the job exactly as it is in the JCL data set. The installation should take whatever steps are necessary to ensure that the job is run under the correct RACF User-ID and Group. In a JES2 environment, Exit 2 can be very useful for this purpose. Exit 2 has access to the RACF User-ID and Group of the task submitting the job. (This is in RIDSUSR and RIDSGRP in the JES2 DCT

for the reader. The DCT address is in PCEDCT in the JES2 PCE, which is in register 13 at entry to the exit.) The exit can force these fields into the job card for every batch job submitted in the system. Once it can be certain that every job has the correct USERID and GROUP parameters on the JOB card, batch jobs can be run without further password verification. Other schemes may also be possible.

Chapter 6: PJS Operator Commands

PJS operator commands are used by the system operator to start, stop, and control the activity of the PJS System Task. Operator commands are entered at the system console, or through any other facility for entering operator commands (such as SDSF). Specific details on how to enter operator commands can be found in *MVS System Commands*.

The START Command

The START command is used to start the PJS System Task. The command format is:

START *task-name*
S

Figure 18: START Command Format

After PJS has successfully started and is fully initialized, a message will be displayed on the console to notify the operator.

The STOP Command

The STOP command is used to stop the PJS System Task. There may be a short delay between the time the command is entered and the time the system actually shuts down. It is important that the task be allowed to finish its processing. Do not use the CANCEL command unless absolutely necessary. The command format is:

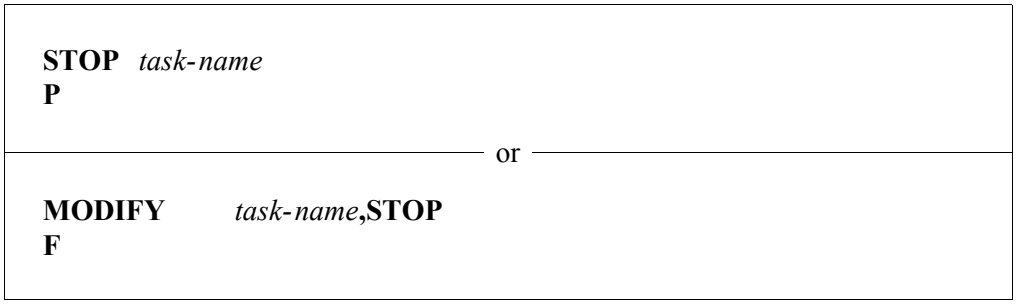


Figure 19: STOP Command Format

If PJS is currently processing a job request, PJS termination may be delayed until the job submission is complete. This delay is usually short.

The SCAN Command

The SCAN command is used cause an immediate scan of the PJS Request Queue. This should rarely be necessary, but may be convenient from time to time. The command format is:

Figure 20: SCAN Command Format

Chapter 7: PJS Utilities

The Personal Job Scheduler system provides several utilities to help install and maintain PJS. Only the Site Administrator, a systems programmer, the person who installs PJS, or a systems operator should use any of the PJS Utilities.

The following utilities will be discussed:

Utility Name	Utility Description
PJSHINIT	PJS Message History Log Initialization Utility
PJSQCONV	PJS Request Queue Conversion Utility
PJSQINIT	PJS Request Queue Initialization Utility
PJSQMNT	PJS Request Queue Maintenance Utility

Figure 21: PJS Utilities

The PJS Message History Log Initialization Utility (PJSHINIT) initializes a new PJS Message History Log data set.

The PJS Request Queue Initialization Utility (PJSQINIT) initializes a new PJS Request Queue data set.

The PJS Queue Conversion Utility (PJSQCONV) converts records from a PJS Version 2.x Request Queue to a PJS Version 3.1 Request Queue.

The PJS Request Queue Maintenance Utility (PJSQMNT) cleans up the PJS Request Queue and the PJS JCL Spool.

The PJS Message History Log Initialization (PJSHINIT) Utility

PJSHINIT is a utility that initializes a new PJS Message History Log data set. In most cases, a PJS Message History Log is initialized only during PJS installation.

Before you run **PJSHINIT**, the PJS Message History Log data set must be defined and allocated with IDCAMS. The PJS Message History Log is a VSAM ESDS. The PJS Message History Log must be defined with NONINDEXED, RECSZ(132,288), REUSE, and SHR(2). Most other data set attributes may be changed to optimize VSAM processing.

The PJS Message History Log requirements may very, depending on how many jobs are submitted by PJS and how long messages are retained. Each job submitted will require approximately 250 bytes. Multiply the number of jobs submitted in your message retention period by 250, then add an additional amount for jobs that have been submitted during that period. (PJS will always retain messages for the last submit for a job request, regardless for the retention period.) For a typical system 5 to 10 cylinders should be reasonable.

The PJS Message History Log is initialized by writing an initial message into an empty VSAM ESDS.

PJSHINIT has no input parameters.

The following **DD** statements are required:

PJSHIST is the PJS Message History Log data set.

SYSOUT is the message data set, usually **SYSOUT=***.

SYSABEND is the dump output data set, usually **SYSOUT=***.

The sample job **DEFHIST** in the PJS Installation Library can be used to define and initialize the PJS Message History Log data set.

The PJS Request Queue Conversion (PJSQCONV) Utility

PJSQCONV is a utility that converts PJS Version 2.x Request Queue records to a PJS Version 3.1 Request Queue data set. You can only use **PJSQCONV** when you upgrade from PJS 2.x to PJS 3.1.

Use the following procedure to convert the PJS Request Queue records:

1. Stop the PJS Release 2.x System Task.
2. Back up the PJS Release 2.x Request Queue data set.
3. Use **IDCAMS REPRO** to copy the PJS 2.x Request Queue to a sequential data set.
4. Use **IDCAMS** to define and allocate the PJS Version 3.1 Request Queue data set.
5. Use **PJSQINIT** to initialize the PJS Version 3.1 Request Queue. For more information, please refer to the section *The PJS Request Queue Initialization (PJSQINIT) Utility* on page 50.
6. Use **PJSQCONV** to convert the PJS Version 2.x Request Queue records to Release 3.1.
7. Start the PJS Release 3.1 System Task.

PJSQCONV has no input parameters.

The following **DD** statements are required:

SYSOUT is the message data set, usually **SYSOUT=***.

SYSABEND is the dump output data set, usually **SYSOUT=***.

The PJS Release 3.1 Request Queue must be initialized, in Step 4 above, but otherwise empty. The PJS Options Table (PJSOPT) must point to the PJS Release 3.1 Request Queue data set.

The sample job **CONVPJSQ** in the PJS Installation Library can be used to perform the PJS Request Queue conversion.

The PJS Request Queue Initialization (PJSQINIT) Utility

PJSQINIT is a utility that initializes a new PJS Request Queue data set. In most cases, a PJS Request Queue is initialized only during PJS installation.

Before you run **PJSQINIT**, the PJS Request Queue data set must be defined and allocated with IDCAMS. The PJS Request Queue is a VSAM KSDS. The new PJS Request Queue data set must be defined with RECSZ(144,4084), KEYS(17,0), and SHR(2). Most other data set attributes can be changed to optimize VSAM processing. For most sites, 1 or 2 cylinders of space should be sufficient.

The PJS Request Queue is initialized by writing a Queue Control Record into an empty VSAM KSDS.

PJSQINIT has no input parameters.

The following **DD** statements are required:

PJSQ is the PJS Request Queue data set.

SYSOUT is the message data set, usually **SYSOUT=***.

SYSABEND is the dump output data set, usually **SYSOUT=***.

The sample job **DEFPJSQ** in the PJS Installation Library can be used to define and initialize the PJS Request Queue data set.

The PJS Request Queue Maintenance (PJSQMNT) Utility

PJSQMNT is a utility that performs several maintenance functions on the PJS Request Queue, the PJS Message History Log, and the PJS JCL Spool. This utility should be run periodically to eliminate out-of-date records from these PJS data sets.

PJSQMNT performs the following functions:

- Deletes old job requests.
- Deletes unreferenced event records.
- Deletes unreferenced owner records.
- Deletes old messages from the PJS Message History Log.
- Deletes unreferenced PJS JCL Spool members.
- Re-synchronizes PJS JCL Spool record counts.
- Compresses the PJS JCL Spool data set.

Deletion of old job request records is controlled by the **CMPTIME**, **DISTIME**, and **ERRTIME** options in the PJS Options Table (PJSOPT).

Deletion of old messages is controlled by the **HISTRET** option in the PJS Options Table (PJSOPT).

PJSQMNT enqueues the PJS Request Queue while it runs. No other access is allowed when **PJSQMNT** runs. Any attempted accesses will be delayed until **PJSQMNT** completes. There is no need to stop the PJS System Task when **PJSQMNT** runs.

You can use PJS to schedule **PJSQMNT** to run at relatively quiet times of day. Since PJS is usually used to run jobs off-hours, it may be a good idea to run this utility late in the morning or early in the afternoon.

PJSQMNT has no input parameters.

The following **DD** statements are required:

SYSOUT is the message data set, usually **SYSOUT=***.

SYSABEND is the dump output data set, usually **SYSOUT=***.

The sample job **PJSQMNT** in the PJS Installation Library can be used to run the PJS Request Queue Maintenance Utility.

Appendix A: PJS TSO Commands

The following table lists the TSO commands and the access level required to the PJS Request Queue and the PJS JCL Spool data sets. Several of these commands are used internally by the PJS/ISPF interface, and are not documented in the *PJS User Guide*.

Command Name	Program Name	PJSQ Access	JCL Spool Access	Message History Log Access	Description
PJREQADD PJRA PJADD PJA	PJSTJA	Update	Update	None	TSO Add Job Request
PJREQMOD PJRM PJMODIFY PJMOD PJM	PJSTJM	Update	Update	None	TSO Modify Job Request
PJREQDEL PJRD PJDELETE PJDEL PJD	PJSTJD	Update	Update	None	TSO Delete Job Request
PJREQLST PJRL PJLIST PJL	PJSTJL	Read	Read	None	TSO List Job Requests
PJREQHST PJRH PJLIST PJL	PJSTJH	Read	None	Read	TSO List Job Request History
SET	PJSTJSET	None	None	None	TSO Add/Modify Job Request SET Subcommand Processor
RESET	PJSTJRST	None	None	None	TSO Add/Modify Job Request RESET Subcommand Processor

Command Name	Program Name	PJSQ Access	JCL Spool Access	Message History Log Access	Description
EVENT	PJSTJEVN	None	None	None	TSO Add/Modify Job Request EVENT Subcommand Processor
VARIABLE	PJSTJVAR	None	None	None	TSO Add/Modify Job Request VARIABLE Subcommand Processor
LIST	PJSTJLST	None	None	None	TSO Add/Modify Job Request LIST Subcommand Processor
PJCALADD PJCA	PJSTCA	Update	None	None	TSO Add Calendar
PJCALMOD PJCM	PJSTCM	Update	None	None	TSO Modify Calendar
PJCALDEL PJCD	PJSTCD	Update	None	None	TSO Delete Calendar
PJCALIST PJCL	PJSTCL	Read	None	None	TSO List Calendars
PJEVPOST PJEP	PJSTEP	Update	None	None	TSO Post Event
PJEVRSET PJER	PJSTER	Update	None	None	TSO Reset Event
PJEVLIST PJEL	PJSTEL	Read	None	None	TSO List Events
PJVARADD PJVA	PJSTVA	Update	None	None	TSO Add Global Variable
PJVARMOD PJVM	PJSTVM	Update	None	None	TSO Modify Global Variable
PJVARDEL PJVD	PJSTVD	Update	None	None	TSO Delete Global Variable
PJVARLST PJVL	PJSTVL	Read	None	None	TSO List Global Variables

Command Name	Program Name	PJSQ Access	JCL Spool Access	Message History Log Access	Description
PJSITCA	PJSITCA	Update	None	None	Command processors used internally by the ISPF Interface
PJSITCD	PJSITCD	Update	None	None	
PJSITCG	PJSITCG	Read	None	None	
PJSITCJ	PJSITCJ	Read	None	None	
PJSITCL	PJSITCL	Read	None	None	
PJSITCM	PJSITCM	Update	None	None	
PJSITEG	PJSITEG	Read	None	None	
PJSITEJ	PJSITEJ	Read	None	None	
PJSITEL	PJSITEL	Read	None	None	
PJSITEP	PJSITEP	Update	None	None	
PJSITER	PJSITER	Update	None	None	
PJSITJA	PJSITJA	Update	Update	None	
PJSITJD	PJSITJD	Update	Update	None	
PJSITJG	PJSITJG	Read	None	None	
PJSITJH	PJSITJH	Read	None	Read	
PJSITJJ	PJSITJJ	Read	Read	None	
PJSITJL	PJSITJL	Read	None	None	
PJSITJM	PJSITJM	Update	Update	None	
PJSITVA	PJSITCA	Update	None	None	
PJSITVD	PJSITVD	Update	None	None	
PJSITVG	PJSITVG	Read	None	None	
PJSITVJ	PJSITVJ	Read	None	None	
PJSITVL	PJSITVL	Read	None	None	
PJSITVM	PJSITVM	Update	None	None	

Appendix B: PJS Data Areas

The following sections contain information about the PJS Options Table, the PJS Vector Table, the PJS Request Queue Entry, the PJSQ Queue Control Record, the PJSQ Owner Record, the PJSQ Job Request Record, the PJSQ Calendar Record, the PJSQ Event Record, and the PJS Message History Log Record. This information may change from release to release.

The PJS Options Table

DSECT Name: PJSOPT
Macro ID: PJSOPT
Size: 352 Bytes
Function: The PJS Options Table contains installation-defined constants that are used throughout the PJS system.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSOPT	PJS Options Table.
0	(0)	CHARACTER	8	PJSOPT_TBL_ID	PJSOPT Table Identifier. =CL8'PJSOPT'
8	(8)	CHARACTER	8	PJSOPT_DATE	PJSOPT Generation Date.
16	(10)	CHARACTER	5	PJSOPT_TIME	PJSOPT Generation Time.
21	(15)	CHARACTER	8	PJSOPT_FMID	PJSOPT Generation Macro FMID.
29	(1D)	CHARACTER	8	PJSOPT_RMID	PJSOPT Generation Macro RMID.
37	(25)		3	RESERVED	Reserved for future use.
40	(28)	ADDRESS	4	PJSOPT_PJSAUTH	Pointer to the Authorization Table.
44	(2C)		4	RESERVED	Reserved for future use.
48	(30)	CHARACTER	44	PJSOPT_PJSQDSN	PJS Request Queue data set name.
92	(5C)	CHARACTER	44	PJSOPT_JCLSDSN	JCL Spool data set name.
136	(88)	CHARACTER	44	PJSOPT_HISTDSN	Message History Log data set name.
180	(B4)		44	RESERVED	Reserved for future use.
224	(E0)	CHARACTER	8	PJSOPT_TMPUNIT	Temporary data set unit name.
232	(E8)	CHARACTER	8	PJSOPT_ENQNAME	QNAME for serialization.
240	(F0)		8	RESERVED	Reserved for future use.
248	(F8)	BINARY	4	PJSOPT_SCANINT	PJSQ long scan interval time.
252	(FC)	BINARY	4	PJSOPT_DOWNWRN	System Task down warning time.
256	(100)	BINARY	4	PJSOPT_RETRY_NUM	Job submit retry max number.
260	(104)	BINARY	4	PJSOPT_RETRY_INT	Job submit retry interval.

Offsets Dec	Hex	Type	Len	Name	Description
264	(108)	BINARY	4	PJSOPT_CMPTIME	COMPLETE job request hold time.
268	(10C)	BINARY	4	PJSOPT_DISTIME	DISABLED job request hold time.
272	(110)	BINARY	4	PJSOPT_ERRTIME	ERROR job request hold time.
276	(114)	BINARY	4	PJSOPT_HISTRET	Message History Log retention period.
280	(118)		8	RESERVED	Reserved for future use.
288	(120)	BINARY	4	PJSOPT_JCLSLIM1	JCL Spool limit for request.
292	(124)	BINARY	4	PJSOPT_JCLSLIM2	JCL Spool limit for owner.
296	(128)	BINARY	4	PJSOPT_TMPBLK	Temporary data set blocksize.
300	(12C)	BINARY	4	PJSOPT_TMPPRIM	Temporary data set primary allocation.
304	(130)	BINARY	4	PJSOPT_TMPSEC	Temporary data set secondary allocation.
308	(134)		4	RESERVED	Reserved for future use.
312	(138)	BINARY	4	PJSOPT_ISPFQSZ	ISPF List Table size.
316	(13C)	BINARY	4	PJSOPT_PTBLSZ	PJSTASK Post/Reset Table size.
320	(140)	BINARY	4	PJSOPT_RRTBLSZ	PJSTASK Ready Request Table size.
324	(144)	BINARY	4	PJSOPT_OTBLSZ	PJSQMNT Owner Table size.
328	(148)	BINARY	4	PJSOPT_RTBLSZ	PJSQMNT Request Table size.
332	(14C)	BINARY	4	PJSOPT_ETBLSZ	PJSQMNT Event Table size.
336	(150)	BINARY	4	PJSOPT_STBLSZ	PJSQMNT Spool Member Table size.
340	(154)		4	RESERVED	Reserved for future use.
344	(158)	CHARACTER	1	PJSOPT_RDRCLS	INTRDR SYSOUT Class.
345	(159)	BITSTRING	1	PJSOPT_FLAGS	Option Flags.
		1...		PJSOPT_TSOAUTH	Call TSO commands from an authorized environment.
		.1..		PJSOPT_JCLSAVEN	JCL 'NOSAVE' may be used.
		..1.		PJSOPT_JCLSAVES	JCL 'SAVE' may be used.
		...1		PJSOPT_JCLSAVED	JCL 'SAVE' is default.
		.... 1...		PJSOPT_SVCDUMP	Produce SVC dump for abends
346	(15A)	BINARY	1	PJSOPT_DATEFMT	Date format
		X'00'		PJSOPT_DFMT_MDY	mm/dd/yyyy

Offsets Dec	Hex	Type	Len	Name	Description
		X'01'		PJSOPT_DFMT_DMY	dd/mm/yyyy
347	(15B)	CHARACTER	2	PJSOPT_YR2000	Minimum 2-digit year for 19xx
347	(15D)		3	RESERVED	Reserved for future use.
352	(160)			PJSOPT_LEN	Length of PJSOPT Table.

The PJS Vector Table

DSECT Name: PJSVT
Macro ID: PJSVT
Size: 688 Bytes
Function: The PJS Vector Table contains run time global pointers and flags. The PJSVT is created by the main routine and then passed to each subroutine and subtask it calls.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSVT	PJS Vector Table.
0	(0)	CHARACTER	8	PJSVT_TBL_ID	PJSVT Table Identifier. CL8'PJSVT'
					Installation options. May be modified by the Installation Options Exit (PJSOPTX).
8	(8)	CHARACTER	44	PJSVT_PJSQDSN	PJSQ Data Set Name.
52	(34)	CHARACTER	44	PJSVT_JCLSDSN	JCL Spool Data Set Name.
96	(60)	CHARACTER	44	PJSVT_HISTDSN	Message History Log Data Set Name.
140	(8C)		44	RESERVED	Reserved for future use.
184	(B8)	CHARACTER	8	PJSVT_TMPUNIT	Temporary data set unit name.
192	(C0)	CHARACTER	8	PJSVT_ENQNAME	QNAME for serialization.
200	(C8)		8	RESERVED	Reserved for future use.
208	(D0)	BINARY	4	PJSVT_SCANINT	PJSQ long scan interval time.
212	(D4)	BINARY	4	PJSVT_DOWNWRN	System Task down warning time.
216	(D8)	BINARY	4	PJSVT_RETRY_NUM	Job submit retry max number.
220	(DC)	BINARY	4	PJSVT_RETRY_INT	Job submit retry interval.
224	(E0)	BINARY	4	PJSVT_CMPTIME	COMPLETE job request hold time.
228	(E4)	BINARY	4	PJSVT_DISTIME	DISABLED job request hold time.
232	(E8)	BINARY	4	PJSVT_ERRTIME	ERROR job request hold time.
236	(EC)	BINARY	4	PJSVT_HISTRET	Message History Log retention period.
240	(F0)		8	RESERVED	Reserved for future use.

Offsets Dec	Hex	Type	Len	Name	Description
248	(F8)	BINARY	4	PJSVT_JCLSLIM1	JCL Spool limit for request.
252	(FC)	BINARY	4	PJSVT_JCLSLIM2	JCL Spool limit for owner.
256	(100)	BINARY	4	PJSVT_TMPBLK	Temporary data set blocksize.
260	(104)	BINARY	4	PJSVT_TMPPRIM	Temporary data set primary allocation.
264	(108)	BINARY	4	PJSVT_TMPSEC	Temporary data set secondary allocation.
268	(10C)		4	RESERVED	Reserved for future use.
272	(110)	BINARY	4	PJSVT_ISPFQSZ	ISPF List Table size.
276	(114)	BINARY	4	PJSVT_PTBLSZ	PJSTASK Post/Reset Table size.
280	(118)	BINARY	4	PJSVT_RRTBLSZ	PJSTASK Ready Request Table size.
284	(11C)	BINARY	4	PJSVT_OTBLSZ	PJSQMNT Owner Table size.
388	(120)	BINARY	4	PJSVT_RTBLSZ	PJSQMNT Request Table size.
292	(124)	BINARY	4	PJSVT_ETBLSZ	PJSQMNT Event Table size.
296	(128)	BINARY	4	PJSVT_STBLSZ	PJSQMNT Spool Member Table size.
300	(12C)		4	RESERVED	Reserved for future use.
304	(130)	CHARACTER	1	PJSVT_RDRCLS	INTRDR SYSOUT Class.
305	(131)	BITSTRING	1	PJSVT_OPT_FLAGS	Option Flags.
		1...		PJSVT_OPT_TAUTH	Call TSO commands from an authorized environment.
		.1..		PJSVT_OPT_JSAVEN	JCL 'NOSAVE' may be used.
		..1.		PJSVT_OPT_JSAVES	JCL 'SAVE' may be used.
		...1		PJSVT_OPT_JSAVED	JCL 'SAVE' is default.
		...1		PJSVT_OPT_JSAVED	JCL 'SAVE' is default.
		.... 1...		PJSVT_OPT_SDUMP	SVC dump for selected abends.
		.... .1..		PJSVT_OPT_SDUMPA	SVC dump for all abends.
306	(132)	BINARY	1	PJSVT_DATEFMT	Date format
		X'00'		PJSVT_DFMT_MDY	mm/dd/yyyy
		X'01'		PJSVT_DFMT_DMY	dd/mm/yyyy
307	(133)	CHARACTER	2	PJSVT_YR2000	Minimum 2-digit year for 19xx
309	(135)		3	RESERVED	Reserved for future use.

Offsets Dec	Hex	Type	Len	Name	Description
		Installation data areas. May be used by installation exits.			
312	(138)	BINARY	4	PJSVT_USER1	Installation use.
316	(13C)	BINARY	4	PJSVT_USER2	Installation use.
320	(140)	BINARY	4	PJSVT_USER3	Installation use.
324	(144)	BINARY	4	PJSVT_USER4	Installation use.
		System environment data areas. Not to be modified by installation exits.			
328	(148)	CHARACTER	8	PJSVT_LEVEL	PJS Version/Release/Level.
336	(150)	CHARACTER	8	PJSVT_PGMNAME	Main program name.
344	(158)	CHARACTER	1	PJSVT_ENV	System environment indicator
		C'I'		PJSVT_ENV_ISP	ISPF interface.
		C'J'		PJSVT_ENV_JOB	Batch utility job.
		C'S'		PJSVT_ENV_STC	PJS System Task.
		C'T'		PJSVT_ENV_TSO	TSO command processor.
345	(159)	BITSTRING	1	PJSVT_SYS_FLAGS	System flags.
		1...		PJSVT_SYS_APF	APF authorized environment.
346	(15A)		2	RESERVED	Reserved for future use.
348	(15C)	ADDRESS	4	PJSVT_CPPL	Pointer to TSO CPPL.
352	(160)		8	RESERVED	Reserved for future use.
		Pointers to routines. Not to be modified by installation exits.			
360	(168)	ADDRESS	4	PJSVT_PJSALLOC	Pointer to Dynamic Allocation Service Routine.

Offsets Dec	Hex	Type	Len	Name	Description
364	(16C)	ADDRESS	4	PJSVT_PJSCALFM	Pointer to Calendar Format Service Routine.
368	(170)	ADDRESS	4	PJSVT_PJSCALID	Pointer to Calendar-ID Input Service Routine.
372	(174)	ADDRESS	4	PJSVT_PJSCALU	Pointer to Calendar Utility Service Routine.
376	(178)	ADDRESS	4	PJSVT_PJSDATIM	Pointer to Date/Time Service Routine.
380	(17C)	ADDRESS	4	PJSVT_PJSDYN	Pointer to Dynamic Variable Service Routine.
384	(180)	ADDRESS	4	PJSVT_PJSEVNFM	Pointer to Event Format Service Routine.
388	(184)	ADDRESS	4	PJSVT_PJSEVNID	Pointer to Event-ID Input Service Routine.
392	(188)	ADDRESS	4	PJSVT_PJSFREQ	Pointer to Calculate Next Run Date/Time Service Routine.
396	(18C)	ADDRESS	4	PJSVT_PJSHIO	Pointer to Message History Log Data Set I/O Service Routine.
400	(190)	ADDRESS	4	PJSVT_PJSHSTFM	Pointer to Message History Log Format Service Routine.
404	(194)	ADDRESS	4	PJSVT_PJSJIO	Pointer to JCL Data Set I/O Service Routine.
408	(198)	ADDRESS	4	PJSVT_PJSJRQFM	Pointer to Job Request Format Service Routine.
412	(19C)	ADDRESS	4	PJSVT_PJSJRQID	Pointer to Job Request-ID Input Service Routine.
416	(1A0)	ADDRESS	4	PJSVT_PJSMSG	Pointer to Message Format Service Routine.
420	(1A4)	ADDRESS	4	PJSVT_PJSMSGC	Pointer to Console Message Service Routine.
424	(1A8)	ADDRESS	4	PJSVT_PJSMSGI	Pointer to ISPF Message Service Routine.
428	(1AC)	ADDRESS	4	PJSVT_PJSMSGH	Pointer to Message History Log Message Service Routine.
432	(1B0)	ADDRESS	4	PJSVT_PJSMSGP	Pointer to Print Message Service Routine.
436	(1B4)	ADDRESS	4	PJSVT_PJSMSGT	Pointer to TSO Terminal Message Service Routine.
440	(1B8)	ADDRESS	4	PJSVT_PJSMSGU	Pointer to TSO Send Message Service Routine.
444	(1BC)	ADDRESS	4	PJSVT_PJSQIO	Pointer to PJSQ I/O Service Routine.
448	(1C0)	ADDRESS	4	PJSVT_PJSSIO	Pointer to JCL Spool I/O Service Routine.

Offsets Dec	Hex	Type	Len	Name	Description
452	(1C4)	ADDRESS	4	PJSVT_PJSSPOOL	Pointer to Copy JCL to Spool Service Routine.
456	(1C8)	ADDRESS	4	PJSVT_PJSTEMP	Pointer to Copy JCL to Temporary Data Set Service Routine.
460	(1CC)	ADDRESS	4	PJSVT_PJSVARFM	Pointer to Global Variable Format Service Routine.
464	(1D0)	ADDRESS	4	PJSVT_PJSVARID	Pointer to Global Variable-ID Input Service Routine.
468	(1D4)	ADDRESS	4	PJSVT_PJSWTOR	Pointer to Get Operator Reply Service Routine.
472	(1D8)		16	RESERVED	Reserved for future use.
488	(1E8)	ADDRESS	4	PJSVT_PJSABOPT	Pointer to Abend Options Table
492	(1EC)	ADDRESS	4	PJSVT_PJSESTA2	Pointer to Second Level ESTAE Routine
496	(1F0)	ADDRESS	4	PJSVT_PJSTJEST	Pointer to TSO Job Request Subcommand Processor ESTAE Routine
500	(1F4)	ADDRESS	4	PJSVT_PJSTSOE	Pointer to TSO Command Processor ESTAE Routine
504	(1F8)	ADDRESS	4	PJSVT_PJSUTILE	Pointer to Batch Utility ESTAE Routine
508	(1FC)		8	RESERVED	Reserved for future use.
516	(204)	ADDRESS	4	PJSVT_PJSTJCMD	Pointer to Process TSO Job Request Subcommands Routine
520	(208)	ADDRESS	4	PJSVT_PJSTJEND	Pointer to TSO Job Request END Subcommand Processor
524	(20C)		4	RESERVED	Reserved for future use.
528	(210)	ADDRESS	4	PJSVT_PJSDYNX	Pointer to Installation Dynamic Variables Exit Routine.
532	(214)	ADDRESS	4	PJSVT_PJSIDFX	Pointer to Installation Data Exit Routine.
536	(218)	ADDRESS	4	PJSVT_PJSSECX	Pointer to Installation Security Exit.
540	(21C)	ADDRESS	4	PJSVT_PJSSUBX	Pointer to Installation Submit Exit.
544	(220)		8	RESERVED	Reserved for future use.
552	(228)	ADDRESS	4	PJSVT_ISPLINK	Pointer to ISPF Service Routine.
556	(22C)		4	RESERVED	Reserved for future use.

Offsets Dec	Hex	Type	Len	Name	Description
		Execution time parameters. Not to be modified by installation exits.			
560	(230)	ADDRESS	4	PJSVT_MSGDEST	Pointer to default message destination list.
564	(234)	BINARY	4	PJSVT_TEST_DATE	Used for testing.
568	(238)	BINARY	4	PJSVT_TEST_TIME	Used for testing.
572	(23C)		4	RESERVED	Reserved for future use.
		PJSMSGH data areas. Not to be modified by installation exits.			
576	(240)	BITSTRING	1	PJSVT_MSGH_FLAGS	PJSMSGH status flags.
		1...		PJSVT_MSGH_ACT	PJSMSGH is active.
577	(241)		7	RESERVED	Reserved for future use.
		PJSMSGP data areas. Not to be modified by installation exits.			
584	(248)	ADDRESS	4	PJSVT_PRTDCB	Pointer to printer DCB.
588	(24C)	BINARY	4	PJSVT_PRTDCB_LN	Length of printer DCB.
592	(250)	BITSTRING	1	PJSVT_PRT_FLAGS	Printer file status flags.
		1...		PJSVT_PRT_OPEN	Printer data set open.
		.1..		PJSVT_PRT_CLOSED	Printer data set closed.
593	(251)		7	RESERVED	Reserved for future use.
		PJSQIO data areas. Not to be modified by installation exits.			
600	(258)	ADDRESS	4	PJSVT_PJSQIOD	Pointer to PJSQIO Data Area.

Offsets Dec	Hex	Type	Len	Name	Description
604	(25C)	ADDRESS	4	PJSVT_PJSQIOD_LN	Length of PJSQIO Data Area.
608	(260)	ADDRESS	4	PJSVT_PQCTL_REC	Pointer to PJSQ Queue Control record.
612	(264)	ADDRESS	4	PJSVT_PQOWN_REC	Pointer to PJSQ Owner record.
616	(268)	ADDRESS	4	PJSVT_PQJRQ_REC	Pointer to PJSQ Job Request record.
620	(26C)	ADDRESS	4	PJSVT_PQCAL_REC	Pointer to PJSQ Calendar record.
624	(270)	ADDRESS	4	PJSVT_PQEVN_REC	Pointer to PJSQ Event record.
628	(274)	ADDRESS	4	PJSVT_PQVAR_REC	Pointer to PJSQ Global Variable record.
632	(278)	BITSTRING	1	PJSVT_PJSQ_FLAGS	PJSQ file status flags.
		1...		PJSVT_PJSQ_ALLOC	PJSQ data set allocated.
		.1..		PJSVT_PJSQ_OPEN	PJSQ data set open.
633	(279)		7	RESERVED	Reserved for future use.

PJSHIO data areas. Not to be modified by installation exits.

640	(280)	ADDRESS	4	PJSVT_PJSHIOD	Pointer to PJSHIO Data Area.
644	(284)	ADDRESS	4	PJSVT_PJSHIOD_LN	Length of PJSHIO Data Area.
648	(288)	BITSTRING	1	PJSVT_HIST_FLAGS	Message History Log file status flags.
		1...		PJSVT_HIST_ALLOC	Message History Log allocated.
		.1..		PJSVT_HIST_OPEN	Message History Log open.
649	(289)		7	RESERVED	Reserved for future use.

PJSSIO data areas. Not to be modified by installation exits.

656	(290)	ADDRESS	4	PJSVT_PJSSIOD	Pointer to PJSSIO Data Area.
660	(294)	ADDRESS	4	PJSVT_PJSSIOD_LN	Length of PJSSIO Data Area.
664	(298)	BITSTRING	1	PJSVT_SPL_FLAGS	JCL Spool file status flags.

Offsets Dec	Hex	Type	Len	Name	Description
		1...		PJSVT_SPL_ALLOC	JCL Spool allocated.
		.1..		PJSVT_SPL_OPENI	JCL Spool open for input.
		..1.		PJSVT_SPL_OPENO	JCL Spool open for output.
		...1		PJSVT_SPL_OPEND	JCL Spool directory open.
665	(299)		7	RESERVED	Reserved for future use.

PJSJIO data areas. Not to be modified by installation exits.

672	(2A0)	ADDRESS	4	PJSVT_PJSJIOD	Pointer to PJSJIO Data Area.
676	(2A4)	ADDRESS	4	PJSVT_PJSJIOD_LN	Length of PJSJIO Data Area.
680	(2A8)	BITSTRING	1	PJSVT_JCL_FLAGS	JCL Data Set file status flags.
		1...		PJSVT_JCL_ALLOC	JCL Data Set allocated.
		.1..		PJSVT_JCL_OPEN	JCL Data Set open.
681	(2A9)		7	RESERVED	Reserved for future use.
688	(2B0)			PJSVT_LEN	Length of PJS Vector Table.

The PJS Request Queue Entry

DSECT Name: PJSQE
Macro ID: PJSQE
Size: Variable
Function: This DSECT describes the format of the PJS Request Queue Entry. This DSECT is used to map a general PJSQ record without the specific record type fields. This basically consists of the record length and the key. The data portion is referred to only by a single data area. The fields for each record type are described by other DSECTs.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSQE	PJS Request Queue Entry.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PJSQE_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PJSQE_KEY	Record Key.
4	(4)	BINARY	1	PJSQE_TYPE	Record Type.
		X'01'		PJSQE_TYPE_CNTL	Queue Control Record.
		X'02'		PJSQE_TYPE_OWNER	Owner Record.
		X'03'		PJSQE_TYPE_JOB	Job Request Record.
		X'04'		PJSQE_TYPE_CAL	Calendar Record.
		X'05'		PJSQE_TYPE_EVNT	Event Record.
		X'06'		PJSQE_TYPE_VAR	Global Variable Record.
5	(5)	CHARACTER	8	PJSQE_OWNERID	Owner-ID of this record.
13	(D)	CHARACTER	8	PJSQE_RECID	Record-ID of this record.
		The following fields define the record type dependent area. This area is redefined for each record type.			

Offsets Dec	Hex	Type	Len	Name	Description
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PJSQE_VERSION	Record format version
		X'0301'		PJSQE_VERS_CUR	Current record format version
24	(18)	GROUP	4064	PJSQE_DATA	Record type dependent data area.
4088	(FF8)			PJSQE_LEN	Maximum length of PJSQ record.

The PJSQ Queue Control Record

DSECT Name: PJSQCTL
Macro ID: PJSQE
Size: 37 bytes
Function: This DSECT describes the format of the PJSQ Queue Control Record. There is one Queue Control record for the PJS Request Queue. It contains global information of concern to the entire Request Queue.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSQCTL	PJSQ Queue Control Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PQCTL_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PQCTL_KEY	Record Key.
4	(4)	BINARY	1	PQCTL_TYPE	Record Type (Always X'01').
5	(5)	CHARACTER	8	PQCTL_OWNERID	Owner-ID (Always X'00').
13	(D)	CHARACTER	8	PQCTL_RECID	Record-ID (Always X'00').
		The following fields define the record type dependent area.			
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PQCTL_VERSION	Record format version
24	(18)	BINARY	4	PQCTL_ACT_DATE	System Task last active date
28	(1C)	BINARY	4	PQCTL_ACT_TIME	System Task last active time

Offsets Dec	Hex	Type	Len	Name	Description
32	(20)	BINARY	4	PQCTL_NXT_JCLNUM	Next JCL Spool member number.
36	(24)	BITSTRING	1	PQCTL_UPD_FLAGS	Update flags.
		1...		PQCTL_UPD_JRQ	Job Request has been updated.
		.1..		PQCTL_UPD_CAL	Calendar has been updated.
		..1.		PQCTL_UPD_EVNT	Event has been updated.
37	(25)		11	RESERVED	Reserved.
48	(30)			PQCTL_LEN	Length of PJSQ Queue Control record.

The PJSQ Owner Record

DSECT Name: PJSQOWN
Macro ID: PJSQE
Size: 44 bytes
Function: This DSECT describes the format of the PJSQ Owner Record. It contains information for each Owner-ID with records stored in the PJS Request Queue.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSQOWN	PJSQ Owner Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PQOWN_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PQOWN_KEY	Record Key.
4	(4)	BINARY	1	PQOWN_TYPE	Record Type (Always X'02').
5	(5)	CHARACTER	8	PQOWN_OWNERID	Owner-ID.
13	(D)	CHARACTER	8	PQOWN_RECID	Record-ID (Always X'00').
		The following fields define the record type dependent area.			
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PQOWN_VERSION	Record format version
24	(18)	BINARY	4	PQOWN_USE_DATE	Date of last Owner update.
28	(1C)	BINARY	4	PQOWN_USE_TIME	Time of last Owner update.
32	(20)	BINARY	4	PQOWN_SPOOLNUM	Number of JCL records on spool.

Offsets Dec	Hex	Type	Len	Name	Description
36	(24)	BINARY	2	PQOWN_NXT_REQNUM	Next Request Number to assign.
38	(26)		6	RESERVED	Reserved.
44	(2C)			PQOWN_LEN	Length of PJSQ Owner record.

The PJSQ Job Request Record

DSECT Names: PJSQJRQ
PJSQJRQC
PJSQJRQE
PJSQJRQI
Macro ID: PJSQE
Size: variable, 288 bytes minimum, 4084 bytes maximum.
Function: These DSECTs describe the format of the PJSQ Job Request Record. It contains information for each Job Request entered by the users. The record consists of a basic record area, followed by one or more variable length segments. A Calendar segment is present if calendar scheduling is used, an Event segment is present if event scheduling is used, a Job Request Variables segment is present if variable substitution is used, and an Installation Data Area is present if it was created by the Installation Security Exit (PJSSECX). DSECT PJSQJRQ describes the format of the basic Job Request Record. DSECT PJSQJRQC describes the format of the Calendar Segment. DSECT PJSQJRQE describes the format of the Event Segment. DSECT PJSQJRQV describes the format of the Job Request Variables segment. DSECT PJSQJRQI describes the format of the installation data area.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name**	**Description**
0	(0)			PJSQJRQ	PJSQ Job Request Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PQJRQ_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PQJRQ_KEY	Record Key.
4	(4)	BINARY	1	PQJRQ_TYPE	Record Type (Always X'03').
5	(5)	CHARACTER	8	PQJRQ_OWNERID	Owner-ID.
13	(D)	CHARACTER	8	PQJRQ_REQNAME	Request Name.

Offsets Dec	Hex	Type	Len	Name	Description
				The following fields define the record type dependent area.	
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PQJRQ_VERSION	Record format version
24	(18)	BINARY	2	PQJRQ_CAL_OFF	Offset to Calendar Table.
26	(1A)	BINARY	2	PQJRQ_CAL_NUM	Number of Calendar Table entries.
28	(1C)	BINARY	2	PQJRQ_EVNT_OFF	Offset to Event Table.
30	(1E)	BINARY	2	PQJRQ_EVNT_NUM	Number of Event Table entries.
32	(20)	BINARY	2	PQJRQ_VAR_OFF	Offset to Variable Table.
34	(22)	BINARY	2	PQJRQ_VAR_NUM	Number of Variable Table entries.
36	(24)	BINARY	2	PQJRQ_INST_OFF	Offset to Installation Data Area.
38	(26)	BINARY	2	PQJRQ_INST_LEN	Length of Installation Data Area.
40	(28)		24	RESERVED	Reserved.
64	(40)	BINARY	1	PQJRQ_STATUS	Job Request processing status.
		X'01'		PQJRQ_STAT_WAIT	Waiting for next run time.
		X'02'		PQJRQ_STAT_SUB	Currently in submit processing.
		X'03'		PQJRQ_STAT_ERR	Error encountered.
		X'04'		PQJRQ_STAT_CMPL	Completed.
		X'05'		PQJRQ_STAT_DIS	Disabled.
		X'06'		PQJRQ_STAT_HOLD	System hold.
65	(41)		3	RESERVED	Reserved.
68	(44)	CHARACTER	50	PQJRQ_DESC	Job Request description.
118	(76)		10	RESERVED	Reserved.
128	(80)	BINARY	1	PQJRQ_JSAVE	JCL Save option.
		X'00'		PQJRQ_JSAVE_NO	JCL is not saved in Spool.
		X'01'		PQJRQ_JSAVE_YES	JCL is saved in Spool.
		X'02'		PQJRQ_JSAVE_REFR	JCL refresh pending.

Offsets Dec	Hex	Type	Len	Name	Description
		X'03'		PQJRQ_JSAVE_DEL	JCL delete pending.
129	(81)		3	RESERVED	Reserved.
132	(84)	CHARACTER	44	PQJRQ_DSNAME	JCL Data Set Name.
176	(B0)	CHARACTER	8	PQJRQ_MEMBER	JCL Member Name.
184	(B8)	CHARACTER	8	PQJRQ_SPOOLMEM	JCL Spool Member Name.
192	(C0)	BINARY	4	PQJRQ_SPOOLNUM	Number of JCL records on spool.
196	(C4)		12	RESERVED	Reserved.
208	(D0)	CHARACTER	8	PQJRQ_NTFY_USER	Notify Userid.
216	(D8)	CHARACTER	1	PQJRQ_NTFY_LVL	Notify Message Level.
		C'I'		PQJRQ_NTFY_INFO	Send Informational messages and higher.
		C'W'		PQJRQ_NTFY_WARN	Send Warning messages and higher.
		C'E'		PQJRQ_NTFY_ERROR	Send Error messages and higher.
		C'X'		PQJRQ_NTFY_NONE	Do not send any messages.
217	(D9)		7	RESERVED	Reserved.
224	(E0)	BINARY	4	PQJRQ_WNDW_TIME	Submit Window Time.
228	(E4)	BINARY	1	PQJRQ_WNDW_OPT	Submit Window Failure Option.
		X'01'		PQJRQ_WNDW_DIS	Place in DISABLED status.
		X'02'		PQJRQ_WNDW_ERR	Place in ERROR status.
		X'03'		PQJRQ_WNDW_SKIP	Reschedule for next period.
229	(E5)		3	RESERVED	Reserved.
232	(E8)	BINARY	2	PQJRQ_RETRY_NUM	Number of submit retries.
234	(EA)		14	RESERVED	Reserved.
248	(F8)	BINARY	4	PQJRQ_STAT_DATE	Status Date.
252	(FC)	BINARY	4	PQJRQ_STAT_TIME	Status Time.

Offsets Dec	Hex	Type	Len	Name	Description
256	(100)	BINARY	4	PQJRQ_LAST_DATE	Last Submit Date.
260	(104)	BINARY	4	PQJRQ_LAST_TIME	Last Submit Time.
264	(108)	BINARY	4	PQJRQ_NEXT_DATE	Next Submit Date.
268	(10C)	BINARY	4	PQJRQ_NEXT_TIME	Next Submit Time.
272	(110)	BINARY	4	PQJRQ_START_DATE	Starting Date.
276	(114)	BINARY	4	PQJRQ_START_TIME	Starting Time.
280	(118)	BINARY	4	PQJRQ_END_DATE	Ending Date.
284	(11C)	BINARY	4	PQJRQ_END_TIME	Ending Time.
288	(120)	BINARY	1	PQJRQ_FREQ	Job Submit Frequency Type.
		X'01'		PQJRQ_FREQ_ONCE	Once.
		X'02'		PQJRQ_FREQ_PRD	Periodic.
		X'03'		PQJRQ_FREQ_WKDAY	Day-of-Week.
		X'04'		PQJRQ_FREQ_EOM	End-of-Month.
		X'05'		PQJRQ_FREQ_CAL	Calendar.
289	(121)	BINARY	1	PQJRQ_PRD_UNIT	Periodic Units.
		X'01'		PQJRQ_PRD_YRS	Years.
		X'02'		PQJRQ_PRD_MOS	Months.
		X'03'		PQJRQ_PRD_WKS	Weeks.
		X'04'		PQJRQ_PRD_DAYS	Days.
		X'05'		PQJRQ_PRD_HRS	Hours.
		X'06'		PQJRQ_PRD_MINS	Minutes.
290	(122)	BINARY	2	PQJRQ_PRD_QTY	Periodic Quantity.
292	(124)	BITSTRING	1	PQJRQ_WKDAY	Day-of-Week to submit job.
		1...		PQJRQ_WKDAY_SUN	Sunday.
		.1..		PQJRQ_WKDAY_MON	Monday.
		..1.		PQJRQ_WKDAY_TUE	Tuesday.
		...1		PQJRQ_WKDAY_WED	Wednesday.
		.... 1...		PQJRQ_WKDAY_THU	Thursday.
		.... .1..		PQJRQ_WKDAY_FRI	Friday.
		.... ..1.		PQJRQ_WKDAY_SAT	Saturday.
293	(125)	BINARY	1	PQJRQ_EOM_DAYS	Days before End-of-Month.
294	(126)		10	RESERVED	Reserved.

Offsets Dec	Hex	Type	Len	Name	Description
304	(130)	GROUP	3784	PQJRQ_SEGMENTS	Record segment area.
4088	(FF8)			PQJRQ_LEN	Maximum length of PJSQ Job Request record.
		The following fields define the Calendar Segment. The Calendar Segment is a table, each entry of which contains information on a calendar referenced by the Job Request.			
0	(0)			PJSQJRQC	Calendar Table Segment.
0	(0)	GROUP	17	PQJRQ_CAL_ID	Calendar-ID.
0	(0)	BINARY	1	PQJRQ_CAL_TYPE	Calendar Type (Always X'04').
1	(1)	CHARACTER	8	PQJRQ_CAL_OWNID	Calendar Owner-ID.
9	(9)	CHARACTER	8	PQJRQ_CAL_NAME	Calendar Name.
17	(11)		3	RESERVED	Reserved.
20	(14)			PQJRQ_CAL_ELEN	Length of Calendar Table Entry.
		3		PQJRQ_CAL_MAX	Maximum number of Calendar Table Entries.
		The following fields define the Event Segment. The Event Segment is a table, each entry of which contains information on an event referenced by the Job Request.			
0	(0)			PJSQJRQE	Event Table Segment.
0	(0)	GROUP	17	PQJRQ_EVNT_ID	Event-ID.
0	(0)	BINARY	1	PQJRQ_EVNT_TYPE	Event Type (Always X'05').
1	(1)	CHARACTER	8	PQJRQ_EVNT_OWNID	Event Owner-ID.
9	(9)	CHARACTER	8	PQJRQ_EVNT_NAME	Event Name.
17	(11)	BITSTRING	1	PQJRQ_EVNT_FLAG	Job Request Event flags.
		1...		PQJRQ_EVNT_PREP	Event may be 'Preposted'.
18	(12)		2	RESERVED	Reserved.

Offsets Dec	Hex	Type	Len	Name	Description
20	(14)	BINARY	4	PQJRQ_EVNT_DATE	Date Event Posted.
24	(18)	BINARY	4	PQJRQ_EVNT_TIME	Time Event Posted.
28	(1C)			PQJRQ_EVNT_ELEN	Length of Event Table Entry.
		50		PQJRQ_EVNT_MAX	Maximum number of Event Table Entries.

The following fields define the Job Request Variables Segment. The Job Request Variables Segment is a table, each entry of which contains information on a variable substitution to be performed when the job is submitted.

0	(0)			PJSQJRQV	Job Request Variables Segment.
0	(0)	BINARY	2	PQJRQ_VAR_SRCH_COL1	Search start column.
2	(2)	BINARY	2	PQJRQ_VAR_SRCH_COL2	Search end column.
4	(4)	BINARY	2	PQJRQ_VAR_SHFT_COL	Shift end column.
6	(6)	BINARY	1	PQJRQ_VAR_SHFT_TYPE	Shift type.
		X'00'		PQJRQ_VAR_SHFT_NONE	No shift.
		X'01'		PQJRQ_VAR_SHFT_ALL	Shift all characters.
		X'02'		PQJRQ_VAR_SHFT_SPC	Shift to/from spaces.
7	(7)	BINARY	1	PQJRQ_VAR_REPL_TYPE	Replacement Value type.
		X'01'		PQJRQ_VAR_REPL_LIT	Value is Literal.
		X'02'		PQJRQ_VAR_REPL_VAR	Value is Global Variable.
		X'03'		PQJRQ_VAR_REPL_DYN	Value is Dynamic Value.
8	(8)		4	RESERVED	Reserved.
12	(C)	BINARY	2	PQJRQ_VAR_TGT_LEN	Target String length.
14	(E)	CHARACTER	8	PQJRQ_VAR_TGT	Target String.
22	(16)	GROUP	82	PQJRQ_VAR_REPL_VALUE	Replacement Value.
22	(16)	BINARY	2	PQJRQ_VAR_LIT_LEN	Literal Replacement Value string length.
24	(18)	CHARACTER	80	PQJRQ_VAR_LIT	Literal Replacement Value string.

Offsets Dec	Hex	Type	Len	Name	Description
22	(16)	GROUP	17	PQJRQ_VAR_ID	Global Variable Replacement Value ID
22	(16)	BINARY	1	PQJRQ_VAR_TYPE	Global Variable Type (Always X'06').
23	(17)	CHARACTER	8	PQJRQ_VAR_OWNID	Global Variable Owner-ID.
31	(1F)	CHARACTER	8	PQJRQ_VAR_NAME	Global Variable Name.
22	(16)	CHARACTER	16	PQJRQ_VAR_DYNAM	Dynamic Replacement Value name.
104	(68)			PQJRQ_VAR_ELEN	Length of Job Request Variable Table Entry.
		20		PQJRQ_VAR_MAX	Maximum number of Job Request Variable Table Entries.

The following fields define the Installation Data Area. The Installation Data Area is a variable length area defined by the installation and set by the Installation Security Exit (PJSSECX).

0	(0)			PJSQJRQI	Installation Data Area.
0	(0)	CHARACTER	256	PQJRQ_INST_DATA	Installation Data Area.

The PJSQ Calendar Record

DSECT Name: PJSQCAL
Macro ID: PJSQE
Size: variable, 88 bytes minimum, 4088 bytes maximum.
Function: This DSECT describes the format of the PJSQ Calendar Record. It contains a variable length bitstring of selected dates for the calendar.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSQCAL	PJSQ Calendar Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PQCAL_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PQCAL_KEY	Record Key.
4	(4)	BINARY	1	PQCAL_TYPE	Record Type (Always X'04').
5	(5)	CHARACTER	8	PQCAL_OWNERID	Owner-ID.
13	(D)	CHARACTER	8	PQCAL_CALNAME	Calendar Name.
		The following fields define the record type dependent area.			
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PJCAL_VERSION	Record format version
24	(18)	CHARACTER	50	PQCAL_DESC	Calendar description.
74	(4A)		6	RESERVED	Reserved.
80	(50)	BINARY	4	PQCAL_START_DATE	Date on which calendar starts.

Offsets Dec	Hex	Type	Len	Name	Description
84	(54)	BITSTRING	1	PQCAL_FLAGS	Update flags.
		1...		PQCAL_UPDATE	Calendar has been updated.
85	(55)		1	RESERVED	Reserved.
86	(56)	BINARY	2	PQCAL_DATES_LEN	Length of Calendar Dates bit string.
88	(58)	BITSTRING	4000	PQCAL_DATES	Calendar Dates. Each bit is one day, starting with the Start Date.
4088	(FF8)			PQCAL_LEN	Maximum length of PJSQ Calendar record.

The PJSQ Event Record

DSECT Name: PJSQEVN
Macro ID: PJSQE
Size: 33 bytes
Function: This DSECT describes the format of the PJSQ Event Record. It contains a flag that indicates when the Job Request Events are to be posted or reset.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSQEVN	PJSQ Event Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PQEVN_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PQEVN_KEY	Record Key.
4	(4)	BINARY	1	PQEVN_TYPE	Record Type (Always X'05').
5	(5)	CHARACTER	8	PQEVN_OWNERID	Owner-ID.
13	(D)	CHARACTER	8	PQEVN_EVNTNAME	Event Name.
		The following fields define the record type dependent area.			
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PJEVN_VERSION	Record format version
24	(18)	BINARY	4	PQEVN_POST_DATE	Date event last posted.
28	(1C)	BINARY	4	PQEVN_POST_TIME	Time event last posted.
32	(20)	BITSTRING	1	PQEVN_FLAGS	Update flags.
		1...		PQEVN_JOB_POST	Job Request Event post pending.

Offsets Dec	Hex	Type	Len	Name	Description
		.1..		PQEVN_JOB_RESET	Job Request Event reset pending.
33	(21)			PQEVN_LEN	Length of PJSQ Event record.

The PJSQ Global Variable Record

DSECT Names: PJSQVAR
Macro ID: PJSQE
Size: variable, 84 bytes minimum, 164 bytes maximum.
Function: This DSECT describes the format of the PJSQ Global Variable Record. It contains a variable substitution replacement value which can be refered to be one or more job request records.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSQVAR	PJSQ Global Variable Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PQVAR_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record key.			
4	(4)	GROUP	17	PQVAR_KEY	Record Key.
4	(4)	BINARY	1	PQVAR_TYPE	Record Type (Always X'06').
5	(5)	CHARACTER	8	PQVAR_OWNERID	Owner-ID.
13	(D)	CHARACTER	8	PQVAR_VARNAME	Variable Name.
		The following fields define the record type dependent area.			
21	(15)		1	RESERVED	Reserved.
22	(16)	BINARY	2	PJVAR_VERSION	Record format version
24	(18)	CHARACTER	50	PQVAR_DESC	Global Variable description.
74	(4A)		6	RESERVED	Reserved.

Offsets Dec	Hex	Type	Len	Name	Description
80	(50)	BINARY	1	PQVAR_REPL_TYPE	Replacement Value type.
		X'01'		PQVAR_REPL_LIT	Value is Literal.
		X'02'		PQVAR_REPL_VAR	Value is Global Variable.
		X'03'		PQVAR_REPL_DYN	Value is Dynamic Value.
81	(51)		1	RESERVED	Reserved.
82	(52)	GROUP	82	PQVAR_REPL_VALUE	Replacement Value.
82	(52)	BINARY	2	PQVAR_LIT_LEN	Literal Replacement Value string length.
84	(54)	CHARACTER	80	PQVAR_LIT	Literal Replacement Value string.
82	(52)	GROUP	17	PQVAR_VAR_ID	Global Variable Replacement Value ID
82	(52)	BINARY	1	PQVAR_VAR_TYPE	Global Variable Type (Always X'06').
83	(53)	CHARACTER	8	PQVAR_VAR_OWNID	Global Variable Owner-ID.
91	(5B)	CHARACTER	8	PQVAR_VAR_NAME	Global Variable Name.
99	(63)			PQVAR_VAR_LEN	Length of Variable value record
82	(52)	CHARACTER	16	PQVAR_DYNAM	Dynamic Replacement Value name.
98	(62)			PQVAR_DYNAM_LEN	Length of Dynamic value record
164	(A4)			PQVAR_LEN	Maximum length of PJSQ Global Variable record.

The PJS Message History Log Record

DSECT Names: PJSHIST
Macro ID: PJSHIST
Size: variable, 40 bytes minimum, 296 bytes maximum.
Function: This DSECT describes the format of the PJS Message History Log Record. It contains a record of the messages sent to the user by the PJS System Task.

Offsets Dec	Hex	Type	Len	Name	Description
0	(0)			PJSHIST	PJS Message History Log Record.
		The record descriptor word does not appear in the VSAM data set.			
0	(0)	BINARY	2	PJSHIST_RECLEN	Record length (including this field).
2	(2)		2	RESERVED	Reserved.
		The following fields define the record data area.			
4	(4)	BINARY	4	PJSHIST_DATE	Message Date.
8	(8)	BINARY	4	PJSHIST_TIME	Message Time.
12	(C)	GROUP	17	PJSHIST_REQID	Job Request-ID.
29	(1D)	CHARACTER	8	PJSHIST_MSG_ID	Message ID.
37	(25)		1	RESERVED	Reserved.
38	(26)	BINARY	2	PJSHIST_MSG_LEN	Message Text length.
40	(28)	CHARACTER	256	PJSHIST_MSG_TEXT	Message Text.
296	(128)			PJSHIST_LEN	Maximum length of PJS Message History Log record.

Appendix C: Summary of Changes

Changes for PJS Release 3.1

Functional Changes

- PJS can now modify the JCL for a job request as it is being submitted. The user can specify a string appearing in the JCL that is to be replaced, and the replacement value. A replacement value may be a literal string, a **'Global Variable'** (which specifies a replacement value outside of the job request that uses it), or a **'Dynamic Value'** (which is computed at job submit time).
- A **PJS Message History Log** is provided to record messages sent to the user by the **PJS System Task**.
- The **Job Request-ID** can be specified by the user, instead of being generated by PJS. The Request Number is replaced by an alphanumeric **Request Name**. If a request name is not specified by the user when a job request is added, a numeric **Request Number** will still be generated by PJS.
- A 50-character **Job Request Description** may be specified for a job request. This description is used for documentation purposes only, and does not affect PJS processing in any way.
- A **Notify Message Level** may be specified for a job request, to specify the minimum message severity level for messages sent to the user by the PJS System Task. Messages not sent can still be retrieved from the **PJS Message History Log**.
- A **Notify Userid** may be specified for a job request, to specify the TSO Userid to which the PJS System Task is to send messages about the job request. The default is the **Owner-ID** of the job request.
- The user can now explicitly specify that the **Next Run Date and Time** is to be recalculated, without changing the **Start Date and Time**, **End Date and Time**, or **Frequency**.
- The PJS ISPF job request dialog panels have been reorganized to make room for new options, and to display more complete information about a job request as it is being added/modified. Several new sub-dialog panels have been implemented.

The **Add Job Request** and **Modify Job Request** panels now display summary information about the job request, while most job request parameters are updated on sub-dialog panels.

- The PJS **Job Request Add** and **Job Request Modify** TSO commands have been changed to use a sub-command processing mode. The job request to be added or modified is created or read into storage when the **PJREQADD** or **PJREQMOD** command is entered. One or more sub-commands are then used to update the job request. When the job request is complete, the **END** sub-command is entered to complete the add or modify.
- When the **Next Run Date and Time** is recalculated by the ISPF interface, it is recalculated immediately, rather than waiting until the add of modify is completed.
- A **Display Job Request Internal Information** ISPF panel is provided to display internal information about a job request. This information can be valuable in debugging problems.
- The **Job Request List** TSO command DETAIL format has been changed to include the Added Status Change Date/Time, Spool Member, Spool Record Count, and Retry Count.
- When a **Calendar-ID** is entered on the **Specify Job Request Frequency** panel, the Calendar-ID will be checked for validity immediately, rather than waiting until the add or modify is completed.
- A 50-character **Calendar Description** may be specified for a calendar. This description is used for documentation purposes only, and does not affect PJS processing in any way.
- Security access to Job Requests, Calendars and Events can now be checked when they are entered by the suer, instead of waiting until the add or modify is completed.
- The PJS messages have been renumbered. PJS messages are now numbered **PJS***nnnn* where *nnnn* is a 4 digit number.

Installation Changes

- The PJS System Task will now periodically record the current date and time in the PJS Request Queue. If the PJS Request Queue is damaged and restored from a backup tape, the PJS System task can detect that a long period of time has passed since PJS last processed the PJS Request Queue. The operator will be prompted for a recovery option. If the JOB Request Queue was restored from a backup, job requests that were scheduled to be submitted between the backup and the current date and time (which may have already been submitted before the PJS Request Queue was lost) can be placed in the HOLD status, to prevent them from being processed again until the user can 'clean-up' the job requests.
- PJS abend handling has been improved. When an abend occurs in the PJS System Task while submitting a job, the abend will be analyzed to determine if the PJS System Task can continue processing with the next job request, or if the

PJS System Task should terminate. PJS will optionally produce an SVC dump when the abend occurs in an APF authorized environment. The PJS TSO command processors and the PJS utilities are now link-edited with AC=1. The PJS Installation Security Exit will be called when an abend occurs to allow the exit to clean-up any resources it has acquired.

- The PJS Request Queue records have been reformatted. A utility is provided to convert the PJS Release 2.x PJS Request Queue to the Release 3.1 format at installation time.
- Sample security and submit exits are provided to implement security in an ACF2 environment.
- The **DOWNWRN**, **HISTDSN**, **HISTRET**, **RTBLSZ**, and **SVCDUMP** parameters have been added to the PJS Installation Options Table.
- PJS may use system services that are not available on unsupported operating systems. OS/390 R10 is the earliest supported release. Compatibility with earlier releases has not been established.
- PJS is now using the Immediate and Relative Instructions Facility. PJS may not run hardware that does not support these instructions. It is believed that all such hardware is now out of support.
- The PJS documentation is now maintained with OpenOffice.org 1.1.4, rather than Microsoft Word. OpenOffice.org versions for Windows, Macintosh, Linux (X86 and PPC) Solaris (Sparc and X86), and FreeBSD are available free of charge at http://www.openoffice.org. The documentation is also provided in Adobe Acrobat PDF, and HTML formats.
- The internal formats of the PJS Options Table (**PJSOPT**) and the PJS Vector Table (**PJSVT**) have changed. The PJS Options Table and any user exits must be reassembled.

Problems Fixed

- The PJS Request Queue Maintenance Utility was deleting Job Requests in the ERROR and DISABLED status based on the Last Submit Date. This is frequently earlier than the date the Job Request was placed in the ERROR or DISABLED status. As a result these records were being deleted prematurely.

  A ‘Status Change Date and Time’ has been added to the Job Request record. The PJS Request Queue Maintenance Utility will now delete Job Requests based on this date, rather than the Last Submit Date.
- PJS APAR OPJ0034 (List Events for Owner does not list Events when the corresponding Owner record does not exist) is fixed correctly.

  The Job Request Add and Modify processes are changed to check that any added Events have Owner records. If not one will be created.

Changes for PJS Release 2.1.4

Functional Changes

- The CPU authorization checking has been removed.
- Commands to display copyright and license information have been added to the ISPF menu.

Installation Changes

- PJS is now distributed as a single file that can be distributed as part of a public software distribution tape, such as the MVS CBT Tape, or downloaded from the internet.
- PJS is now installed with full source code. All macros required for reassembling PJS (other than IBM macros that are shipped with MVS) are included.
- The sample installation JCL has been moved from the SMP/E JCLIN PDS to a separate PDS.
- The PJS documentation is now shipped as a zip file that can be downloaded to a workstation and expanded. The documentation is shipped in Microsoft Word, Adobe Acrobat PDF, and HTML files.
- The PJS Request Queue Conversion Utility (PJSQCONV) has been removed.

Changes for PJS Release 2.1.3

Functional Changes

- The default century for 2-digit years is changed from 19*xx* to 20*xx*.

Installation Changes

- PJS Documentation is now included on the installation tape in PDF (Adobe Acrobat Reader) format.

PJS R2.1 PTF's Incorporated

PPJ0029 ISPF dialog error with message **ISPP100** when HELP command entered from Delete Job Request panel.

PPJ0030 PJS System Task is not calling the Installation Security Exit (**PJSSECX**) for 'Begin System Task' and 'End System Task'.

PPJ0031 Documentation in the header comments of the sample Installation Security Exits (**PJSSECX**, **PJSSECX1**, and **PJSSECX2**) is incorrect.

OPJ0034 List Events for Owner does not list Events when the corresponding Owner record does not exist.

PPJ0035 Scope of enqueue on PJS Request Queue changed from **SYSTEM** to **SYSTEMS**.

PPJ0036 Scope of dequeue on PJS Request Queue changed from **SYSTEM** to **SYSTEMS**.

Changes for PJS Release 2.1

Functional Changes

- PJS can process dates in the *dd*/*mm*/*yyyy* format (commonly used in Europe), in addition to the *mm*/*dd*/*yyyy* format (commonly used in the U.S.). The format used is controlled by the new **DATEFMT** parameter of the **PJSOPT** macro.
- PJS ISPF commands may be abbreviated to 2 characters.
- A PJS Tutorial Index is provided.
- The name of the Batch Event Post Utility is changed from '**PJSEVENT**' to '**PJSPOST**'. The old name, '**PJSEVENT**', is still supported as an alias of '**PJSPOST**'.
- A new Batch Event Reset Utility (**PJSRESET**) is available. This utility complements the Batch Event Post Utility (**PJSPOST**).
- The Batch Event Post Utility (**PJSPOST**) and the Batch Event Reset Utility (**PJSRESET**) are described in the PJS ISPF Tutorial.

Installation Changes

- A User Contributed Routines library is provided on file 11 of the installation tape. These routines are provided by PJS users and distributed on an 'as-is' basis. They are not supported by NNS Information Services, nor have they been tested by NNS. Use of these routines is at the customers own risk.
- The installation options are checked for validity at program initialization time. In addition, the PJS System Task will display the current option values.
- The default for the **TSOAUTH** parameter of the **PJSOPT** macro is changed from '**NO**' to '**YES**'. '**NO**' is rarely needed, while '**YES**' is frequently required.
- The PJS Request Queue Maintenance Utility (**PJSQMNT**) will treat a zero value for the **CMPTIME**, **DISTIME**, or **ERRTIME** options as indicating 'no limit'.
- A sample PJS Options Exit (**PJSOPTX1**) is provided that will allow an installation to run multiple copies of PJS on a single system.
(PJS 2.0 PTF PPJ0017)

- The internal formats of the PJS Options Table (**PJSOPT**) and the PJS Vector Table (**PJSVT**) have changed. The PJS Options Table and any user exits must be reassembled.

Problems Fixed

- When the PJS System Task attempted to submit a job, and the JCL data set was in use by another task, the job request was failed with message **PJS401E**, and the job request was placed in the '**ERROR**' status.

  The PJS System Task is changed to requeue the job request and retry the submission at a later time. The maximum number of attempts and the interval between each attempt is controlled by the new **RETRY** parameter of the **PJSOPT** macro.

- When the PJS System Task attempted to submit a job, and no internal readers were available, the job request was failed with message **PJS401E**, and the job request was placed in the '**ERROR**' status.

  The PJS System Task is changed to wait until an internal reader is available. These is no limit to how long PJS will wait.

- After the PJS TSO commands (including those used by the PJS ISPF interface) terminated, the PJS Request Queue data set remained allocated.

  The PJS Request Queue data set is automatically deallocated when the TSO commands terminate. The PJS System Task will continue to leave the PJS Request Queue data set allocated.

- When the PJS Queue Maintenance Utility (**PJSQMNT**) was run, and the PJS JCL Spool was not used by the installation, message **PJS401E** and abend **2103** resulted.

  PJSQMNT determines if the PJS JCL Spool is in use by testing PJSVT_JCLSDSN for nulls, but the **PJSOPT** macro sets the PJS JCL Spool Data Set Name to spaces. **PJSQMNT** is changed to check the PJSVT_OPT_JSAVES bit of PJSVT_OPT_FLAGS (set by the **JCLSAVE** option), instead of PJSVT_JCLSDSN to determine if the JCL Spool data set is in use.
  (Supersedes PJS 2.0 PTF PPJ0023)

- The PJS System Task did not correctly check the Ready Request Table Entry against the current system time to determine if a request was ready to be submitted. Also, after a job request was successfully processed, its Ready Request Table Entry was not updated. The result was that every minute **PJSTASKJ** was called to process each job request in the Ready Request Table. Although **PJSTASKJ** performed its own check, the result was a great deal of unnecessary activity against the PJS Request Queue.

  The PJS System Task is fixed to correctly check the Ready Request Table Entry against the current system time. Also, **PJSTASKJ** is fixed to update the Ready Request Table Entry to reflect the new submit date (if any) of the job request.

- The use of attributes in the PJS ISPF panels was not always consistent. Also, for some panels, the use of the 'skip' attribute after some input fields is desirable.

  The use of attributes in the PJS ISPF panels has been standardized. Input fields with no existing value will be filled with underscores ('_'). The attribute terminating most input fields will have the 'noskip' attribute. Some selection fields will have 'skip', where this facilitates data entry.

- When the PJS Tutorial was entered using the '**T**' option on the PJS Main Menu panel, at the end of the tutorial, the IBM ISPF tutorial was entered, instead of returning to the PJS Tutorial main panel.

  The PJS ISPF Interface is changed to correctly set the ISPF Tutorial control variables **ZHTOP** and **ZHINDEX** in the shared variable pool.

PJS R2.0 PTF's Incorporated

PPJ0018 Message **PJS749I** incorrectly displayed by PJS Queue Maintenance Utility (**PJSQMNT**).

PPJ0020 ISPF List panels do not display national characters (**$**, **#**, or **@**) in dynamic area.

PPJ0021 ISPF Copy Calendar function does not set the new Calendar-ID.

PPJ0022 ISPF List panels are being formatted with nulls (X'00') embedded in the lines.

PPJ0024 Message **PJS234E** issued by PJS System Task with sample security exit **PJSSECX** or **PJSSECX2**.

PPJ0025 Abend **U2706** issued by PJS TSO commands **PJREQADD** and **PJREQMOD**.

PPJ0026 Invalid years displayed by PJS ISPF calendar dialogs.

PPJ0027 Message **PJS254E** issued by PJS TSO command **PJEVRSET**.

PPJ0028 System console messages have the wrong route and descriptor codes.

GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program

or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C)  yyyy  name of author

This program is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the
Free Software Foundation; either version 2 of the License, or (at your
option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
'show w'.
This is free software, and you are welcome to redistribute it under
certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front

cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2 or
any later version published by the Free Software Foundation; with no
Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  A
copy of the license is included in the section entitled "GNU Free
Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the Front-
Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

PJS Installation Guide