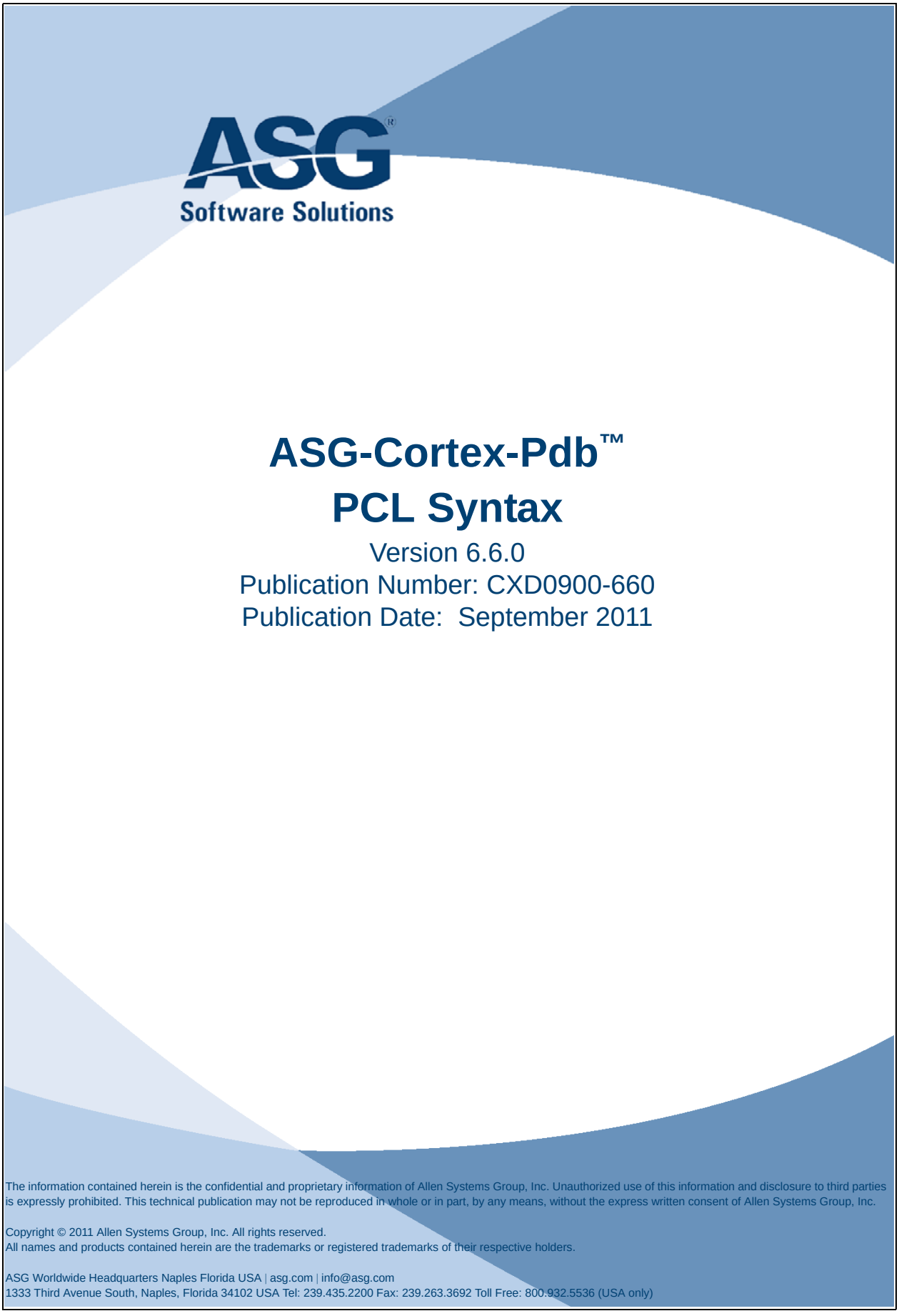

ASG-Cortex-Pdb™
PCL Syntax

Version 6.6.0
Publication Number: CXD0900-660
Publication Date: September 2011

ASG Worldwide Headquarters Naples Florida USA | asg.com | info@asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 239.435.2200 Fax: 239.263.3692 Toll Free: 800.932.5536 (USA only)

Contents

Preface

This *ASG-Cortex-Pdb PCL Syntax* provides a quick reference summary of the Production Control Language (PCL) syntax used in ASG-Cortex-Pdb (herein called Cortex-Pdb).

Cortex-Pdb is geared to standardizing production, thus contributing to the automation of IT centers. The Pdb Repository is used to describe and document user applications by means of the PCL. The Pdb compiler is used to generate JCL procedures and all related production materials.

This publication is intended for administrators and PCL coders responsible for maintaining the Pdb Repository and operating Cortex-Pdb.

About this Publication

This publication is organized according to the PCL zones of the Pdb Repository. Each chapter briefly describes the PCL statement(s) that you code in the corresponding PCL zone(s). This publication consists of these chapters:

- Chapter 1, "PCL Statements for Defining Jobsets," provides the syntax for the PCL statements and operands you can use to describe your applications as Pdb jobsets.
- Chapter 2, "PCL Statements for Defining Programs," provides the syntax for the PCL statements you can use to define your programs.
- Chapter 3, "PCL Statements for Defining TP Transactions," provides the syntax for the TRANS and TRANSEND statements used to define a TP transaction.
- Chapter 4, "PCL Statement for Declaring Files and File Patterns," provides the syntax for the DCLF statement used to declare the files and file patterns that are referenced by your jobsets.
- Chapter 5, "PCL Statement for Declaring Reports and Report Patterns," provides the syntax for the DCLR statement used to declare the report and report patterns that are referenced by your jobsets.
- Chapter 6, "PCL Statements for Defining Patterns," provides the syntax for the PATTERN, DEFAULT, and PATEND statements used to define PCL patterns.
- Chapter 7, "PCL Statements for Defining Filesets," provides the syntax for the PCL statements you can use to define filesets.

- Chapter 8, "PCL Statements for Describing Parameters," provides the syntax for the PARMS statements used to describe variable parameters.
- Chapter 9, "PCL Statements for Declaring Variables," provides the syntax for the DCLV statements used to describe jobset variables.
- Chapter 10, "PCL Statement for Defining Symbols," provides the syntax for the DEFINE statement you can use to define symbols.
- Chapter 11, "Statements for User-defined Objects," explains how you can define user-defined objects.

Related Publications

The documentation library for Cortex-Pdb consists of these publications (where *nn* represents the product version number):

- *ASG-Cortex-Pdb User's Guide* (CXD0200-*nn*) describes the Cortex-Pdb subsystem and provides you with the information necessary to use Cortex-Pdb (e.g., to generate process flows).
- *ASG-Cortex-Pdb Customization Guide* (CXD2800-*nn*) describes the installation steps for Cortex-Pdb and customization tasks specific to Cortex-Pdb.
- *ASG-Cortex-Pdb Implementation Guide* (CXD2300-*nn*) describes the functions of Cortex-Pdb and provides the information you require for implementing the product.
- *ASG-Cortex Security Manager for Pdb Administrator's Guide* (CXT2100-*nn*) provides a detailed description of how to install and administrate Cortex Security Manager for Pdb, the optional add-on package that provides services for controlling access to the objects and functions of Cortex-Pdb.
- *ASG-Cortex-Pdb JCL Integration Option* (CXB0200-*nn*) describes the functions of the Cortex-Pdb JCL Integration Option and provides the information you require for implementing the product.
- *ASG-Cortex-Pdb Toolkit* (CXD2500-*nn*-KIT) provides a description of the principles behind the Cortex-Pdb Toolkit, as well as the syntax of the language used to describe your own process flow generator.
- *ASG-Cortex Installation and Customization Guide* (CXX0300-*nn*) discusses how to install, customize, and maintain Cortex-Env, the base function that provides general-purpose services to all Cortex installations. These services include installation functions, utilities, and service routines. It also tells you how to install and subsequently extend your initial Cortex product mix. For this reason, it provides information on how to install and operate Cortex-Emcee, a piece of Cortex-Env that you need if you install any of these products:
  - Cortex-Plan and the optional Cross-MVS feature, which allows Cortex-Plan to

manage operations in a multi-MVS environment

— Any of the Cortex Windows Interfaces

— Cortex Security Manager for Pdb

- *ASG-Cortex Windows Interfaces Installation Guide* (CXE0300-*nn*) describes how to install and use the Cortex Windows Interfaces for Cortex-Pdb, Cortex-Prep, and Cortex-Plan. Cortex Windows Interfaces allow you to access Cortex services through a Windows workstation in client/server mode.
- *ASG-Cortex Glossary* (CXX2500-*nn*-GLO) provides a comprehensive list and description of the specific terms used with Cortex.

Some PCL extensions are described in these additional ASG-Cortex publications:

- *ASG-Cortex-Calendar and ASG-Cortex-Autoprep Administrator's Guide* (CXC2100-*nn*) discusses the variables and parameters that are managed by Cortex-Autoprep.

Note:

To obtain a specific version of a publication, contact ASG Customer Support.

Publication Conventions

ASG uses these conventions in technical publications:

Convention	Usage
Arrow (▸)	Used in a procedure to indicate commands within menus. Also used to denote a one-step procedure.
Bold	Indicates that case-sensitive usage is required for a directory, path, file, dataset, member, database, program, command, or parameter name. ▸ Verify the settings in the **asg.conf** file.

Convention	Usage
Capitalization	For system element names, this varies according to the product interface and its operating environment. Mainframe file names use uppercase, for example: ▸ Allocate a JSOPTEM member in the JLRCL library. Windows file names use mixed case, for example: ▸ Create a text file named SECLIST.txt in the C:\Program Files\ASG\config directory. UNIX file names use mixed case, for example: ▸ Edit the **databaseID.ACC** file in the **/database** directory. Typical product and operating system elements include: • Directory, path, file, dataset, member, database, program, command, and parameter names. • Window, field, field group, check box, button, panel (or screen), and option labels. • Names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown, such as code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.
Underline	Denotes a cursor-selectable field or line.
Vertical separator bar (|) with underline	Indicates options available with the default value underlined (e.g., Y|N).
Paired square brackets ([ ])	Optional elements in syntax descriptions (e.g., `CZXG0GGs[,ALL]`).

Companion Products

This table lists the ASG-Cortex products and the names by which ASG publications refer to them:

ASG Product Name	ASG Product Herein Name
ASG-Cortex	Cortex
ASG-Cortex-Autoprep	Cortex-Autoprep
ASG-Cortex-Calendar	Cortex-Calendar
ASG-Cortex-Env	Cortex-Env
ASG-Cortex-OMS	Cortex-OMS
ASG-Cortex-Pdb	Cortex-Pdb
ASG-Cortex-Pdb JCL Integration Option	Cortex-Pdb JCL Integration Option
ASG-Cortex-Plan	Cortex-Plan
ASG-Cortex-Prep	Cortex-Prep
ASG-Cortex Security Manager for Pdb	Cortex Security Manager for Pdb
ASG-Cortex Windows Interfaces	Cortex Windows Interfaces

Worldwide Customer Support

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. ASG provides all levels of support during normal business hours and emergency support during non-business hours.

You can access support information at http://www.asg.com/support/support.asp.

ASG Third-party Support. ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Intelligent Support Portal (ISP)

The ASG Intelligent Support Portal (ISP) provides online support at http://isp.asg.com.

Log on to the ISP with this information:

Customer ID = *NNNNNNNNN*

Password = *XXXXXXXXXX*

where:

NNNNNNNNN is your customer ID supplied by ASG Product Distribution.

XXXXXXXXXX is your unique password supplied by ASG Product Distribution.

If you do not have your logon information, contact your local support center.

This table outlines the support response times you can expect:

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

Product Support Policy

ASG fully supports the current release and one previous release of each of its products. ASG will temporarily support an older release, for up to six months, to provide time for you to upgrade.

Once programming support for a product release is withdrawn, ASG will no longer supply new fixes for problems nor accept enhancement requests for that release. When a vendor announces the end of support for system software or a hardware configuration on which ASG products rely, ASG will make a similar announcement regarding the support plans for its products. ASG's support for problems affected by system software release levels will terminate when the vendor no longer supports their hardware or software. Announcements regarding support plans for various products can be found on ASG's Web site.

Support for Field-developed Interfaces (FDIs) developed by ASG's Professional Services staff is described in *ASG Professional Services FDI Support Guide* that can be found on the ASG Support Web site in the Guide to Support section. This document describes how FDIs are supported by ASG Customer Support and ASG Worldwide Professional Services.

ASG Documentation/Product Enhancements

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/asp/emailproductsuggestions.asp.

Include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions, include the publication number located on the publication's front cover.

PCL Statements for Defining Jobsets

This chapter provides the syntax for the PCL statements and operands you use to define your applications as Cortex-Pdb jobsets.

JOBSET Statement Syntax

Functional attributes:

```
jobset ID JOBSET[ALT=(alternate-jobset,...)]
                [APPL=application|$NULL]
                [BACKOUT=JOBSET|CHORE|STEP|NONE]
                [FREQ=frequency]
                [OWNER=owner-ID]
                [PATKW=(varname1=value1,...)]
                [PATTERN=patset-name](1)
                [SYSTEM=system]
                [TITLE='title']
                [USE=(standards-ID,...)|NONE]
                [USERDATA=user-data]
                [WAIT=(eventname,...)]
```

MVS technical attributes:

```
                [ACCT=accounting-information]
                [ADDRSPC=VIRT|REAL]
                [BYTES=([n][,CANCEL|DUMP|WARNING])]
                [CARDS=([n][,CANCEL|DUMP|WARNING])]
                [CCSID=ccsid]
                [CLASS=class]
                [COND=((n,comp)[,(n,comp)]...)]
                [GROUP=group-name]
                [JESLOG=option]
                [LINES=([n][,CANCEL|DUMP|WARNING])]
                [MEMLIMIT=limit]
                [MSGCLASS=class]
                [MSGLEVEL=([statements]messages])]
                [NOTIFY=userid]
                [PAGES=([n][,CANCEL|DUMP|WARNING])]
                [PASSWORD=password]
                [PERFORM=n]
                [PGMR=programmer-information]
                [PRTY=prty-info]
                [RD=R|RNC|NR|NC]
                [REGION=nK|nM]
                [SCHENV=schenv-name]
                [SECLABEL=value]
```

```
                    [TIME=([minutes][,seconds])|NOLIMIT]
                    [USER=userid]
```

(1) For PCL statements for defining patterns, see "PCL Statements for Defining Patterns" on page 29.

JOB Statement Syntax

Functional attributes:

```
job ID JOB [BYPASS=YES|NO]
           [COMREG=YES|NO]
           [HDR=(YES,[0|n])|NO)]
           [PARMS=(name=default,...)]
           [PATKW=(varname1=value1,...)]
           [PATTERN=patjob-name](1)
           [SYSTEM=system]
           [TEXT=text-name]
           [USE=(standards-ID,...)|NONE]
           [USERDATA=user-data]
```

MVS technical attributes:

```
           [ACCT=accounting-information]
           [ADDRSPC=VIRT|REAL]
           [BYTES=([n][,CANCEL|DUMP|WARNING])]
           [CARDS=([n][,CANCEL|DUMP|WARNING])]
           [CCSID=ccsid]
           [CLASS=class]
           [COND=((n,comp)[,(n,comp)]...)]
           [GROUP=group-name]
           [JESLOG=option]
           [LINES=([n][,CANCEL|DUMP|WARNING])]
           [MEMLIMIT=limit]
           [MSGCLASS=class]
           [MSGLEVEL=([statements][,messages])]
           [NAME=jobname]
           [NOTIFY=userid]
           [PAGES=([n][,CANCEL|DUMP|WARNING])]
           [PASSWORD=password]
           [PERFORM=n]
           [PGMR=programmer-information]
           [PRTY=prty-info]
           [RD=R|RNC|NR|NC]
           [REGION=nK|nM]
           [RESTART=(*|stepname
                     |stepname.procstepname[,checkid])]
           [SCHENV=schenv-name]
           [SECLABEL=value]
           [TIME=([minutes][,seconds])|NOLIMIT]
           [USER=userid]
```

(1) For PCL statements for defining job patterns, see "PCL Statements for Defining Patterns" on page 29.

JOBTP Statement Syntax

Functional attributes:

```
job ID JOBTP [COMREG=YES|NO]
             [HDR=(YES,[0|n])|NO)]
             [PARMS=(name=default,...)]
             [PATKW=(varname1=value1,...)]
             [PATTERN=patjob-name](1)
             [SYSTEM=system]
             [TEXT=text-name]
             [USE=(standards-ID,...)|NONE]
             [USERDATA=user-data]
```

MVS technical attributes:

```
             [ACCT=accounting-information]
             [ADDRSPC=VIRT|REAL]
             [BYTES=([n][,CANCEL|DUMP|WARNING])]
             [CCSID=ccsid
             [CLASS=class]
             [COND=((n,comp)[,(n,comp)]...)]
             [GROUP=group-name]
             [JESLOG=option]
             [LINES=([n][,CANCEL|DUMP|WARNING])]
             [MEMLIMIT=limit]
             [MSGCLASS=class]
             [MSGLEVEL=([statements][,messages])]
             [NAME=jobname]
             [NOTIFY=userid]
             [PAGES=([n][,CANCEL|DUMP|WARNING])]
             [PASSWORD=password]
             [PERFORM=n]
             [PGMR=programmer-information]
             [PRTY=prty-info]
             [REGION=nK|nM]
             [SECLABEL=value]
             [TIME=([minutes][,seconds])|NOLIMIT]
             [USER=userid]
```

(1) For PCL statements for defining job patterns, see "PCL Statements for Defining Patterns" on page 29.

CALL Statement Syntax

Functional attributes:

```
CALL    NAME=trans-ID(1)
             [USERDATA=user-data]
```

(1) For PCL statements for defining TP transactions, see "PCL Statements for Defining TP Transactions" on page 19.

STEP Statement Syntax

Functional attributes:

```
step ID STEP [LANG=(lang1[,lang2],... )]
             [NO=(parm-name,...)]
             [PATKW=(varname1=value1,...)]
             [PATTERN=patstep-name](1)
              PGM=pgm-name|*
             [RSTRT=WARM|COLD|SAME|EVER|NONE]
             [SORT=(recl,recnb[,K|M])|MERGE]
             [SPGM=(name,...)]
             [TEXT=text-name]
             [USE=(standards-ID,...)|NONE]
             [USERDATA=user-data]
             [WKNB=number-of-sort-workfiles]
             [YES=(parm-name,...)]
```

MVS technical attributes:

```
             [ACCT=acct-info]
             [ADDRSPC=VIRT|REAL]
             [CCSID=ccsid]
             [COND1=(n,comp[,ref])|EVEN|ONLY].....
             [COND8=(n,comp[,ref])|EVEN|ONLY]
             [DPRTY=dprty-info]
             [DYNAMNBR=n]
             [MEMLIMIT=limit]
             [NAME=stepname]
             [PARM=parm-info](2)
             [PERFORM=n]
             [RD=R|RNC|NC|NR]
             [REGION=nK|nM]
             [TIME=([minutes][,seconds])|NOLIMIT]
```

Old technical attributes:

```
             [DATE=date-info]
```

```
[DIALOG=NO|YES]
[UPSI=upsi-info]
```

(1) For PCL statements for defining mono-step patterns, see "PCL Statements for Defining Patterns" on page 29.
(2) For referring to files from a PARM operand, see "Symbolic File ID Notation" on page 14.

IMSSTEP Statement Syntax

Functional attributes specific to IMSSTEP:

```
step ID IMSSTEP[BMPIN=name]              (IMS-DC/BMP)
               [BMPOUT=name]             (IMS-DC/BMP)
               [BUF=n]
               [CMPAT=N|Y]               (CICS Shared Database)
               [DBD=name]
               [DFHENV=name]
               [DLIENV=name]
               [LANG=(lang1[,lang2],...)]
               [LOG=NO|(YES,file-ID)|OMIT](1)
               [PARMSUP=parm-info]
                PGM=pgm-name
               [PSB=name]
               [SSA=n]                   (CICS Shared Database)
               [TYPE=DBB|DLI|ULU|UDR|BMP|DFH]
               [VSAMP=member-name]
               [VSAM1='dfsvsamp-info-1']
               ....
               [VSAM20='dfsvsamp-info-20']
```

Other functional attributes:

```
               [NO=(parm-name,...)]
               [PATKW=(varname1=value1,...)]
               [PATTERN=patstep-name](2)
               [RSTRT=WARM|COLD|SAME|EVER|NONE]
               [SORT=(recl,recnb[,K|M])|MERGE]
               [SPGM=(name,...)]
               [TEXT=text-name]
               [USE=(standards-ID,...)|NONE]
               [USERDATA=user-data]
               [WKNB=number-of-sort-workfiles]
               [YES=(parm-name,...)]
```

MVS technical attributes:

```
               [ACCT=acct-info]
               [ADDRSPC=VIRT|REAL]
               [CCSID=ccsid]
               [COND1=(n,comp[,ref])|EVEN|ONLY]
               ....
               [COND8=(n,comp[,ref])|EVEN|ONLY]
               [DPRTY=dprty-info]
               [DYNAMNBR=n]
```

```
[MEMLIMIT=limit]
[NAME=stepname]
[PERFORM=n]
[RD=R|RNC|NC|NR]
[REGION=nK|nM]
[TIME=([minutes][,seconds])
      |NOLIMIT
```

Old technical attributes:

```
[DATE=date-info]
[DIALOG=NO|YES]
[UPSI=upsi-info]
```

Log file attributes (functional):

```
[DATACLAS=dataclass]
[GEN=0|+n|-n]
[MODE=O|X|N]
[PASS=YES|NO]
```

Log file attributes (technical):

```
[AVGREC=U|K|M]
[BLKSIZE=blksize]
[BLKSZLIM=blkszlim]
[DEN=0|1|1|2|3|4]
[DISP=(disp-info)]
[DSN=dataset-name]
[EATTR=OPT|NO]
[EROPT=SKP|ACC|ABE]
[FREE=CLOSE|END]
[LABEL=(label-info)]
[LIKE=dsname]
[MGMTCLAS=mgmtclass]
[MSVGP=group]
[OPTCD=codes]
[REFDD=ddname]
[SECMODEL=(profile-name
          [,GENERIC])]
[SPACE=(space-info)]
[STORCLAS=storclass]
[SUBSYS=(subsys-info)]
[TRTCH=E|T|C|ET|COMP|NOCOMP]
[UNIT=(unit-info)]
[VOL=(volume-info)]
[XJCL=(keyword=value,...)]
```

(1) For PCL statements for declaring log files, see "PCL Statement for Declaring Files and File Patterns" on page 21.

(2) For PCL statements for defining mono-step patterns, see "PCL Statements for Defining Patterns" on page 29.

SORT Statement Syntax

Functional attributes:

```
step ID SORT [LANG=(lang1[,lang2],...)]
             [NO=(parm-name,...)]
             [PATKW=(varname1=value1,...)]
             [PATTERN=patstep-name](1)
             [RSTRT=WARM|COLD|SAME|EVER|NONE]
             [SORT=(recl,recnb[,K|M])]
             [TEXT=text-name]
             [USE=(standards-ID,...)|NONE]
             [USERDATA=user-data]
             [WKNB=number-of-sort-workfiles]
             [YES=(parm-name,...)]
```

MVS technical attributes:

```
             [ACCT=acct-info]
             [ADDRSPC=VIRT|REAL]
             [CCSID=ccsid]
             [COND1=(n,comp[,ref])|EVEN|ONLY]
             .....
             [COND8=(n,comp[,ref])|EVEN|ONLY]
             [DPRTY=dprty-info]
             [DYNAMNBR=n]
             [MEMLIMIT=limit]
             [NAME=stepname]
             [PARM=parm-info]
             [PERFORM=n]
             [RD=R|RNC|NC|NR]
             [REGION=nK|nM]
             [TIME=([minutes][,seconds])|NOLIMIT]
```

Old technical attributes:

```
             [DATE=date-info]
             [DIALOG=NO|YES]
             [UPSI=upsi-info]
```

(1) For PCL statements for defining mono-step patterns, see "PCL Statements for Defining Patterns" on page 29.

MERGE Statement Syntax

Functional attributes:

```
step ID MERGE [LANG=(lang1[,lang2],...)]
              [NO=(parm-name,...)]
              [PATKW=(varname1=value1,...)]
              [PATTERN=patstep-name](1)
```

```
[RSTRT=WARM|COLD|SAME|EVER|NONE]
[TEXT=text-name]
[USE=(standards-ID,...)|NONE]
[USERDATA=user-data]
[YES=(parm-name,...)]
```

MVS technical attributes:

```
[ACCT=acct-info]
[ADDRSPC=VIRT|REAL]
[CCSID=ccsid]
[COND1=(n,comp[,ref])|EVEN|ONLY]
.....
[COND8=(n,comp[,ref])|EVEN|ONLY]
[DPRTY=dprty-info]
[DYNAMNBR=n]
[MEMLIMIT=limit]
[NAME=stepname]
[PARM=parm-info]
[PERFORM=n]
[RD=R|RNC|NC|NR]
[REGION=nK|nM]
[TIME=([minutes][,seconds])|NOLIMIT]
```

Old technical attributes:

```
[DATE=date-info]
[DIALOG=NO/YES]
[UPSI=upsi-info]
```

(1) For PCL statements for defining mono-step patterns, see "PCL Statements for Defining Patterns" on page 29.

PATTERN Statement Syntax

Functional attributes:

```
name    PATTERN NAME=pattern-name (1)
                [PATKW=(varname1=value1,...)] (2)
                [USE=(standards-ID,...)|NONE]
```

(1) For PCL statements for defining multi-step patterns, see "PCL Statements for Defining Patterns" on page 29.

(2) For referring to files from PATKW values, see "Symbolic File ID Notation" on page 14.

Caution! Do not confuse the PATTERN statement that calls a multi-step pattern with the PATTERN statement that begins a pattern definition (see "PCL Statements for Defining Patterns" on page 29).

FILE Statement Syntax

Functional attributes:

```
[ddname] FILE  [DATACLAS=dataclass]
               [GEN=0|+n|-n|ALL]
               [LDD=long-name]
               [LINK=YES|NO|WEAK]
               [MBR=member-name]
                MODE=(I|O|OI|U|X|N|WK[,I|O|OI|U|X|N)]
               [NAME=file-ID](1)
               [PASS=YES|NO]
               [REST=(YES,member-name)|NO]
               [USE=(standards-ID,...)|NONE]
               [USERDATA=user-data]
               [ACCODE=accode]
```

MVS technical attributes:

```
               [AMP=(suboperand)|
                ('suboperand','...')]
               [AVGREC=U|K|M]
               [BLKSIZE=blksize]
               [BLKSZLIM=blkszlim]
               [BUFL=buffer-length]
               [BUFNO=n]
               [BUFOFF=buffer-offset|L]
               [CCSID=ccsid]
               [CHKPT=EOV]
               [DEN=0|1|2|3|4]
               [DISP=(disp-info)]
               [DSN=dataset-name]
               [DSNTYPE=PDS|LIBRARY|PIPE|HFS|EXTREQ|EXTPREF|
                        BASIC|LARGE]
               [DSORG=PS[U]|DA[U]|PO[U]]
               [DYNAM=YES|NO]
               [EATTR=OPT|NO]
               [EROPT=SKP|ACC|ABE]
               [FILEDATA=BINARY|TEXT]
               [FREE=CLOSE|END]
               [KEYLEN=key-length]
               [KEYOFF=n]
               [LABEL=(label-info)]
               [LGSTREAM=name]
               [LIKE=dsname|'":file-idf""']
               [LRECL=lrecl|X]
               [MGMTCLAS=mgmtclass]
               [MSVGP=group]
               [OPTCD=codes]
               [OVERSTEP=stepname]
               [PATH='pathname']
               [PATHDISP=(pathdisp1,pathdisp2)|NONE]
               [PATHMODE=(pathmode-info)]
               [PATHOPTS=(pathopts-info)|NONE]
               [PROTECT=YES|NO]
               [RECFM=F|V|U[B][S][A|M]|D|NONE]
               [RECORG=KS|ES|RR|LS]
               [REFDD=ddname]
```

```
[RLS=rls]
[SECMODEL=(profile-name[,GENERIC])]
[SPACE=(space-info)]
[STORCLAS=storclass]
[SUBSYS=(subsys-info)]
[TRTCH=E|T|C|ET|COMP|NOCOMP]
[UNIT=(unit-info)]
[VOL=(volume-info)]
[XJCL=(keyword=value,...)]
```

(1) For PCL statements for declaring files, see "PCL Statement for Declaring Files and File Patterns" on page 21.

FILESET Statement Syntax

Fileset functional attributes:

```
[name] FILESET   NAME=fileset-name(1)
                 [FILKW=(varname1=value1,...)]
                 [USE=(standards-ID,...)|NONE]
```

File-related functional attributes:

```
[DATACLAS=dataclass]
...
[MODE=(I|O|OI|U|X|N|WK[,I|O|OI|U|X|N)]]
...
[USERDATA=user-data]
```

File-related MVS technical attributes:

```
[AVGREC=U|K|M]]
...
[OVERSTEP=stepname]
...
[VOL=(volume-info)]
[XJCL=(keyword=value,...)]
```

(1) For PCL statements for defining filesets, see "PCL Statements for Defining Filesets" on page 33

Note:

The functional attributes (except NAME) and MVS technical attributes (except DSN) of FILESET are identical with those of FILE (see "FILE Statement Syntax" on page 9).

LINK Statement Syntax

Functional attributes:

```
ddname    LINK    MODE=O|I
                  NAME=link-name
                 [USE=(standards-ID,...)|NONE]
                 [USERDATA=user-data]
                 [WEAK=YES|NO]
```

RESOURCE Statement Syntax

Functional attributes:

```
ddname RESOURCE   EXCL=YES|NO]
                  NAME=resource-name
                 [NBR=number]
                 [USE=(standards-ID,...)|NONE]
                 [USERDATA=user-data]
```

REPORT Statement Syntax

Functional attributes:

```
ddname REPORT  [NAME=(report-ID,...)](1)
               [USE=(standards-ID,...)|NONE]
               [LDD=long-name]
               [USERDATA=user-data]
```

MVS technical attributes:

```
               [BLKSIZE=blksize]
               [BURST=burst-info]
               [CHARS=chars-info]
               [COPIES=copies-info]
               [DEST=name]
               [DSID=name]                                  (3540)
               [EROPT=SKP|ACC|ABE]
               [FCB=(fcb-ID[,ALIGN|VERIFY])]                (3211)
               [FLASH=flash-info]
               [FREE=CLOSE|END]
               [HOLD=YES|NO]
```

```
                [LRECL=lrecl]
                [MODIFY=modify-info]
                [OPTCD=codes]
                [OUTLIM=n]
                [OUTPUT=output-reference]
                [OVERSTEP=stepname]
                [RECFM=F|V|U[B][S][A|M]]
                [SEGMENT=value]
                [SPIN=value]
                [SUBSYS=subsys-info]
                [SYSOUT=(class[,pgm[,fno]])]
                [UCS=(ucs-ID[,FOLD[,VERIFY]])]          (1403)
                [XJCL=(keyword=value,... )]
```

(1) For PCL statements for declaring reports, see "PCL Statement for Declaring Reports and Report Patterns" on page 27.

RPTSET Statement Syntax

Functional attributes:

```
[name] RPTSET [NAME=reportset-name](1)
              [RPTKW=(varname1=value1,...)]
              [USE=(standards-ID,...)|NONE]
              [USERDATA=user-data]
```

MVS technical attributes:

```
              [BLKSIZE=blksize]
              [BURST=burst-info]
              [CHARS=chars-info]
              [COPIES=copies-info]
              [DEST=name]
              [DSID=name]                               (3540)
              [EROPT=SKP|ACC|ABE]
              [FCB=(fcb-ID[,ALIGN|VERIFY])]             (3211)
              [FLASH=flash-info]
              [FREE=CLOSE|END]
              [HOLD=YES|NO]
              [LRECL=lrecl]
              [MODIFY=modify-info]
              [OPTCD=codes]
              [OUTLIM=n]
              [OUTPUT=output-reference]
              [OVERSTEP=stepname]
              [RECFM=F|V|U[B][S][A|M]]
              [SEGMENT=value]
              [SPIN=value]
              [SUBSYS=subsys-info]
              [SYSOUT=(class[,pgm[,fno]])]
              [UCS=(ucs-ID[,FOLD[,VERIFY]])]            (1403)
              [XJCL=(keyword=value,... )]
```

(1) For PCL statements for declaring reports, see "PCL Statement for Declaring Reports and Report Patterns" on page 27.

DATA Statement Syntax

Functional attributes:

```
[ddname]   DATA   [*]
                  [CLASS=VAR|fixed-parameter-class]
                  [DIALOG=member-name]
                  [MBR=member-name]
                  [PARMS=member-name] (1)
                  [USE=(standards-ID,...)|NONE]
                  [USERDATA=user-data]
```

MVS technical attributes:

```
                  [DLM=delimiter] (2)
                  [FREE=CLOSE|END]
                  [OVERSTEP=stepname]
                  [XJCL=(keyword=value,... )]
```

(1) For PCL statements for describing parameters, see "PCL Statements for Describing Parameters" on page 35.

(2) For use with option JCLTYPE=NOPROC only.

DATA Statement Syntax for Fixed Parameters

Functional attributes:

```
[ddname]   DATA   [*]
                  [CLASS=fixed-parameter-class]
                  [MBR=member-name]
                  [USE=(standards-ID,...)|NONE]
                  [USERDATA=user-data]
```

MVS technical attributes:

```
                  [DLM=delimiter] (1)
                  [FREE=CLOSE|END]
                  [OVERSTEP=stepname]
                  [XJCL=(keyword=value,... )]
```

(1) For use with option JCLTYPE=NOPROC only.

Note:

A DATA * statement must be followed by a fixed-parameter stream and terminated by a DATAEND statement with no operands.

Symbolic File ID Notation

In a fixed-parameter stream delimited by DATA * and DATAEND statements, this syntax refers to a file described by means of a DCLF statement (see "PCL Statement for Declaring Files and File Patterns" on page 21):

```
":file-ID[,MODE=I|(x,y)][,LINK=YES|NO|WEAK]"[old-dsn]"
```

where:

file-ID represents the NAME attribute of the FILE statement.

MODE represents the MODE attribute of the FILE statement.

LINK represents the LINK attribute of the FILE statement.

old-dsn represents the DDSN attribute of the FILE statement.

For further details, see "FILE Statement Syntax" on page 9.

This syntax also can be used in these PCL attributes:

- PARM attribute of the STEP statement (see "STEP Statement Syntax" on page 4)
- PATKW attribute values of the PATTERN statement (see "PATTERN Statement Syntax" on page 8)

DATA Statement Syntax for Variable Parameters

Functional attributes:

```
[ddname]  DATA  CLASS=VAR
                [DIALOG=member-name]
                [LDD=long-name]
                [MBR=member-name]
                [PARMS=member-name](1)
                [USE=(standards-ID,...)|NONE]
                [USERDATA=user-data]
```

MVS technical attributes:

```
                [FREE=CLOSE|END]
                [OVERSTEP=stepname]
                [XJCL=(keyword=value,... )]
```

(1) For PCL statements for describing parameters, see "PCL Statements for Describing Parameters" on page 35.

IF, ELSE, and ENDIF Statements Syntax

MVS technical attributes:

```
[name] IF [(] relational-expression [)] THEN
.
. action when relational-expression is true
.
[name] ELSE
.
. action when relational-expression is false
.
[name] ENDIF
```

+USEROBJ Directive Syntax

Functional attribute:

```
+USEROBJ zone-name(object-name)
```

For PCL statements for defining user-defined objects, see "Statements for User-defined Objects" on page 49.

END Statement Syntax

The END statement, which has no operands, optionally terminates a jobset definition.

PCL Statements for Defining Programs

This chapter lists the PCL statements you use to define your programs as Cortex-Pdb objects.

A program definition starts with one of these heading statements:

Statement	For Details
STEP	See “STEP Statement Syntax” on page 4
IMSSTEP	See “IMSSTEP Statement Syntax” on page 5
SORT	See “SORT Statement Syntax” on page 7
MERGE	See “MERGE Statement Syntax” on page 7

The heading statement can then be followed by one or more of these statements:

Statement	For Details
FILE	See “FILE Statement Syntax” on page 9
FILESET	See “FILESET Statement Syntax” on page 10
LINK	See “LINK Statement Syntax” on page 11
RESOURCE	See “RESOURCE Statement Syntax” on page 11
REPORT	See “REPORT Statement Syntax” on page 11
DATA	See “DATA Statement Syntax” on page 13

Note:

You can use program definitions as step models for creating jobset definitions.

PCL Statements for Defining TP Transactions

This chapter provides the PCL statements you use to define your TP transactions as Cortex-Pdb objects.

TRANS Statement Syntax

Functional attributes:

```
trans ID  TRANS    [LANG=(lang1[,lang2],...)]
                   [PGM=pgm-name]
                   [TITLE='title']
                   [USERDATA=user-data]
```

The TRANS statement can be followed by one or more of these statements:

Statement	For Details
FILE	See “FILE Statement Syntax” on page 9
FILESET	See “FILESET Statement Syntax” on page 10
LINK	See “LINK Statement Syntax” on page 11
RESOURCE	See “RESOURCE Statement Syntax” on page 11
REPORT	See “REPORT Statement Syntax” on page 11
DATA	See “DATA Statement Syntax” on page 13

The TP transaction definition can then be followed by CALL statements (see “CALL Statement Syntax” on page 4) and must terminate with a TRANSEND statement with no operands.

TRANSEND Statement Syntax

The TRANSEND statement, which has no operands, terminates a TP transaction definition.

PCL Statement for Declaring Files and File Patterns

This chapter provides the syntax for the PCL statement and operands you use to declare your files and file patterns as Cortex-Pdb objects.

DCLF Statement Syntax

Functional attributes:

```
file ID DCLF     [CLASS=file-class]
                 [DATACLAS=dataclas]
                 [KEYL=key-length]
                 [KEYOF=key-offset]
                 [KEYU=M|U]
                 [ORG=S|P|D|E|K|R|L|X|T|V|H]
                 [OWNER=owner-info]
                 [PATDCLF=patdclf-name](1)
                 [RECF=F|V|U]
                 [RECL=80|(max[,average])]
                 [RECNB=estimate]
                 [RFREQ=receive-frequency]
                 [SEC=percentage]
                 [SFREQ=send-frequency]
                 [SHR=0|1|2|3]
                 [SIZE=estimate-Kbytes]
                 [SYNC=YES|NO]
                 [TARGET=target-file-ID]
                 [USERDATA=user-data]
```

MVS technical attributes:

```
                 [AMP=(suboperand)|
                      ('suboperand','...')|
                      'suboperand,...']
                 [AVGREC=U|K|M]
                 [BLKSIZE=blksize|0]
                 [BLKSZLIM=blkszlim]
                 [BUFL=bufl]
                 [BUFNO=bufno]
                 [BUFOFF=bufoff|L]
                 [CCSID=ccsid]
                 [DEN=0|1|2|3|4]
                 [DSN=dataset-name]
                 [DSNTYPE=PDS|LIBRARY|PIPE|HFS|EXTREQ|
                          EXTPREF|BASIC|LARGE]
                 [DSORG=PS[U]|DA[U]|PO[U]]
                 [EATTR=OPT|NO]
```

```
[EROPT=ABE|ACC|SKP]
[FILEDATA=BINARY|TEXT]
[GDGNB=max-catlg-capacity]
[KEYLEN=key-length]
[LABEL=(label-info)]
[LRECL=lrecl|X|0]
[MGMTCLAS=mgmtclas]
[MSVGP=group]
[OPTCD=codes]
[PATH='pathname']
[PATHMODE=pathmode-info]
[PATTERN=pattern-dscb-name]
[RECFM=F|V|U[B][S][A|M]|D|NONE]
[RKP=key-position]
[SMS=YES|NO]
[SPACE=(space-info)]
[STORCLAS=storclas]
[TRTCH=E|T|C|ET|COMP|NOCOMP]
[UNIT=(unit-info)]
[VOL=(volume-info)]
[XJCL=(keyword=value,...,...)]
```

Old technical attributes:

```
[DAMP=(amp-info)]
[DAVGREC=U|K|M]
[DBLK=blksize]
[DBLKL=blkszlim]
[DCAT=catalog-label]
[DDATACL=dataclas]
[DDEN=0|1|2|3|4]
[DDSN=vse-label|
      mvs-dataset-name]
[DDSNTYPE=PDS|LIBRARY]
[DDSORG=PS|DA|ISAM|VSAM]
[DEXTENT=first-extent-trk]
[DFILSEQ=file-sequence-no]
[DGDGNB=max-catlg-capacity]
[DLBLTYP=AL|AUL|LTM|BLP|SUL|NSL|NL]
[DMGMTCL=mgmtclas]
[DPRIME=primary-trks]
[DRECFM=F|V|U|[B][S][A|M]]
[DRETPD=retention]
[DSECND=secondary-trks]
[DSMS=YES|NO]
[DSTORCL=storclas]
[DSYNC=YES|NO]
[DTRTCH=E|T|C|ET|COMP|NOCOMP]
[DUNIT=unit-name]
[DVOLSER=first-volume-serial]
```

(1) In file definitions only (i.e., *not* in a file pattern).

Frequency Codes

This table shows the frequency codes that are used in coding the DCLF statement:

Code	Meaning
D	Daily
W	Weekly
M	Monthly
Q	Quarterly
Y	Yearly
RQ	On request. This is the default value. For calculation purposes, it is assumed to be quarterly.
nf	This means that the frequency *f* is multiplied by *n* (ranging from 1 through 9).
fn	This means that the frequency *f* is divided by *n* (ranging from 1 through 9).

Examples

This table shows a few examples of specific frequency codes:

Code	Meaning
FREQ=M	Once a month
FREQ=W	Once a week
FREQ=W2	Every two weeks
FREQ=2W	Twice a week

File Classes

This table shows the classification of files defined in Cortex-Pdb:

Type	Class	Meaning
Master Files	DM	Direct Master file: Permanent disk file updated directly
	SM	Sequential Master file: Permanent sequential file, updated by new generation rewrite
Interface Files	XS	Cross-set file: Inter-jobset file
	SS	Set-to-Set file: File reused in the following run of the same jobset
	TS	Trans-Set file: Inter-jobset file with a lifetime that is short in relation to its receiving frequency (i.e., rapidly consumed inter-jobset file)
	BU	Backup file
	LG	Log file: IMSSTEP log file
External Files	XD	External Data file: File with a source/destination that is outside the computer center
	CD	Collected Data file: File containing input data for a jobset
	ED	Edited Data file
Intermediate Files	XJ	Cross-Job file: Inter-job (but within-jobset) file
	XP	Cross-Program file: Inter-step (but within-job) file
	WK	Work file: Temporary file within a single step
	RD	Restart Data file: Job restart file
	TD	Transaction Data file: Data for modification of master file

Alternate Classes

The Cortex-Pdb administrator can define alternate classes of this form:

```
classn
```

where *n* ranges from 2 through 9.

These are examples of alternate classes:

Class	Meaning
DM2	Second class for Disk Master files
BU3	Third class for Backup files

PCL Statement for Declaring Reports and Report Patterns

This chapter provides the syntax for the PCL statement and operands you use to declare your reports and report patterns as Cortex-Pdb objects.

DCLR Statement Syntax

Functional attributes:

```
report ID DCLR   [PATDCLR=patdclr-name] (1)
                 [keyword=value] (2)
                 [CZXSCOPE=JOB|STEP] (3)
```

MVS technical attributes:

```
                 [keyword=value] (2)
```

(1) In report definitions only (i.e., *not* in a report pattern).

(2) All functional and technical attributes, except PATDCLR, are defined by the Cortex-Pdb administrator.

(3) If your IT center uses the sample report compiler supplied with Cortex-Pdb, CZXSCOPE is the functional attribute that indicates whether Cortex-Pdb is to generate JOB- or STEP-level OUTPUT statements. The Cortex-Pdb administrator can cancel this attribute.

PCL Statements for Defining Patterns

This chapter provides a quick reference summary for defining these patterns:

- Jobset patterns
- Job patterns
- Mono-step patterns
- Multi-step patterns

Note:

For file and report pattern definitions, see "PCL Statement for Declaring Files and File Patterns" on page 21 and "PCL Statement for Declaring Reports and Report Patterns" on page 27.

This table summarizes the contents of pattern definitions:

Statement	Jobset Pattern	Job Pattern	Mono-step Pattern	Multi-Step Pattern
Heading PATTERN statement	Optional (1)	Optional (1)	Optional (1)	Mandatory
DEFAULT statement(s)	Optional	Optional	Optional	Optional
SETVAR statement(s)	Optional	Optional	Optional	Optional
Next statement	JOBSET	JOB or JOBTP	STEP, IMSSTEP, SORT, MERGE, or PROC (2)	STEP, IMSSTEP, SORT, MERGE, or PATTERN
Contents (3)	One PCL jobset	One PCL job	One PCL step (2)	One or more PCL step(s)
Trailing PATEND statement	None	None	None	Mandatory

(1) Cortex-Pdb for Windows renders the heading PATTERN statement mandatory.

(2) The contents of a mono-step pattern definition that represents an external procedure can consist of a single PROC statement with no operands.

(3) For the other PCL statements that make up a pattern definition, see "PCL Statements for Defining Jobsets" on page 1.

PATTERN Statement Syntax

Functional attribute:

```
PATTERN [REQOP=(varname,...)]
```

Caution! Do not confuse the PATTERN statement that optionally begins a pattern definition with the PATTERN statement that calls a multi-step pattern from a jobset definition (see "PATTERN Statement Syntax" on page 8).

DEFAULT Statement Syntax

Functional attribute:

```
DEFAULT    varname=value,...
```

SETVAR Statement Syntax

Functional attribute:

```
varname    SETVAR    operand
```

PROC Statement Syntax

The PROC statement, which has no operand, represents an external procedure in a mono-step pattern definition.

PATEND Statement Syntax

The PATEND statement, which has no operand, terminates a multi-step pattern definition.

PCL Statements for Defining Filesets

This table shows the PCL statements you use to define your filesets as Cortex-Pdb objects:

Statement	For Details
FILE	See “FILE Statement Syntax” on page 9.
LINK	See “LINK Statement Syntax” on page 11.
RESOURCE	See “RESOURCE Statement Syntax” on page 11.
REPORT	See “REPORT Statement Syntax” on page 11.
DATA	See “DATA Statement Syntax” on page 13.

PCL Statements for Describing Parameters

This chapter provides the PCL statements you use to describe your parameters as Cortex-Pdb objects.

Syntax of PARMS Statement and Substatements

Syntax for Cortex-Prep

This is the syntax of PARMS statements and substatements for use in an installation where Cortex-Autoprep is *not* available:

Functional attributes:

```
PARMS

 LINE

  FLD    [position|1]
         [JUST=L|R]
        |SOURCE=(varname[,date-translation]
        |          [,edited-date-picture])
        |SOURCE=(varname
        |          [,substr-start|1
        |          [,substr-length]])
        |SOURCE='literal'
```

Syntax for Cortex-Autoprep Only

This is the syntax of PARMS statements and substatements for use with Cortex-Autoprep:

Functional attributes:

```
PARMS

 LINE   [COND=(varname
               [,'date-expression'],
               EQ|NE,'attribute')](1)

        [COND=(varname,EQ|NE,
               varname|'literal')](1)


  FLD   [position|1]
        [JUST=L|R]

        [COND=(varname
               [,'date-expression'],
               EQ|NE,'attribute')](1)

        [COND=(varname,EQ|NE,
               varname|'literal')](1)

        SOURCE=(varname[,date-translation]
                [,edited-date-picture])
        SOURCE=(varname,
                SHIFT='date-expression'
                [,edited-date-picture])(1)
        SOURCE=(varname
                [,substr-start|1
                [,substr-length]])
        SOURCE='literal'
```

(1) For use with Cortex-Autoprep only.

Note:

For PCL statements for declaring variables, see Chapter 9, "PCL Statements for Declaring Variables," on page 39.

Date Translation

This table shows the syntax of date translation codes in SOURCE operands:

Code	Meaning
FDY (or PJA)	First day of the year
LDY (or DJA)	Last day of the year
FDQ (or PJT)	First day of the quarter
LDQ (or DJT)	Last day of the quarter
FDM (or PJM)	First day of the month
LDM (or DJM)	Last day of the month

- For the syntax of edited date picture elements in SOURCE operands, see "Edited Date Picture" on page 42.
- For the syntax of date expressions in SOURCE and COND operands, see "Date Expressions" on page 42.
- For a list of system variables for use with Cortex-Autoprep only, see "System Variables" on page 44.

PCL Statements for Declaring Variables

This chapter provides the PCL statements you use to declare your variables.

DCLV Statement Syntax

Syntax for Cortex-Prep

This is the syntax of the DCLV statement for use in an installation where Cortex-Autoprep is *not* available:

Functional attributes:

```
varname DCLV [DEF=default-value]
             [PROMPT=(line1[,line2,..,line5])]
             [TYPE=(DATE,date-picture)]
             [CAPS=ON|OFF]
             [VER=(ver1[,ver2[,ver3]])]
```

Syntax for Cortex-Autoprep Only

This is the syntax of the DCLV statement for use with Cortex-Autoprep:

Functional attributes:

```
varname DCLV [DEF=default-value]
             [TYPE=(DATE,edited-date-picture)]
             [VER=(ver1[,ver2[,ver3]])]
```

```
[SOURCE='literal'|varname](1)
[SOURCE=('literal'|varname,...,
         'literal'|varname)](1)
[SOURCE=(varname,
         SHIFT='date-expression')](1)
[SOURCE=(varname
         [,substr-start|1
         [,substr-length]])](1)
```

```
[COND=(varname
       [,'date-expression'],
       EQ|NE,'attribute')](1)
[COND=(varname,EQ|NE,
       varname|'literal')](1)
```

(1) For use with Cortex-Autoprep only.

Date Picture

This table shows the syntax of date picture elements in TYPE operands:

Picture	Meaning
DD (or JJ)	Day number in the month (2 digits)
MM	Month number (2 digits)
CC	First 2 digits of the year (century indicator)
YY (or AA)	Last 2 digits of the year
QQQ	Julian date (3 digits, day number in the year)

Verification Criteria for Variables

This table shows the syntax of verification criteria for variables in VER operands:

Keyword	Meaning
`NONBLANK`	A value must be specified for the variable.
`ALPHA`	The value must be alphabetical.
`NUM`	The value must be numeric.
`HEX`	The value must be hexadecimal (0 through 9, A through F).
`(PICT,'pict-string')`	The value must match the specifications of picture `pict-string`, that is: `C` Any character `A` Any alphabetical character (A through Z) `9` or `N` Any numeric character (0 through 9) `X` Any hexadecimal character (0 through 9, A through F) *a special character* with no symbolic meaning attached to it. For example: `(PICT,'A/NNC')` where the first character must be alphabetical, followed by a slash (/), 2 digits, and finally any character (5 characters in all).
`NAME`	The value must comply with MVS conventions for naming members (up to 8 alphanumeric characters, the first one being alphabetical).
`DSNAME`	The value must comply with TSO conventions for naming datasets (refer to TSO documentation).
`(RANGE,lower,upper)`	This numeric value must come between the specified lower and upper limits.
`(LIST,value1,...)`	The only possible values for the variable are those that are listed.

Edited Date Picture

This table shows the syntax of edited date picture elements in SOURCE and TYPE operands:

Picture	Meaning
DD (or JJ)	Day number in the month (2 digits)
DAY	English name for the day (10 characters)
JRN	French name for the day (10 characters)
QQQ	Julian date (3 digits, day number in the year)
MM	Month number (2 digits)
MTH	English name for the month (10 characters)
MOI	French name for the month (10 characters)
WW	Week number (2 digits)
CC	First 2 digits of the year (century indicator)
YY (or AA)	Last 2 digits of the year
ANN	Year (4 digits)

Date Expressions

This table shows the syntax of date expressions in SOURCE and COND operands:

Date Expression	Meaning
++ OPEN	The next open day
++ MONDAY	The next Monday
++ MONTH_1ST	The next first day of a month
++ APRIL	The next day of April
++ DAY	The next day

This table shows the operators in date expressions of SOURCE and COND operands:

Operator	Meaning
+	The next (in the broad sense)
++	The next (in the strict sense)
-	The previous (in the broad sense)
--	The previous (in the strict sense)
+-	The nearest (+ if equality)
-+	The nearest (- if equality)

This table shows the meaning of the number of shifts in the date expressions of SOURCE and COND operands:

Number of Shifts	Meaning
+ 2 DAY	Plus 2 days
- 3 MONDAY	Less 3 Mondays

This table shows the various date attributes that can be used in the date expressions of SOURCE and COND operands:

Type	Attributes
System attributes	OPEN, CLOSED, HOLIDAYS DAY, MONTH_1ST, QUARTER_1ST, JANUARY_1ST MONDAY through SUNDAY JANUARY through DECEMBER
User attributes	Any user-defined attributes
Attribute combinations	NOT *attribute* *attribute* AND *attribute* *attribute* OR *attribute*

System Variables

This table lists the system variables that are available in SOURCE operands only when you are using Cortex-Autoprep:

System Variable	Meaning
The jobset's planning date:	
ZXXPDATC	Planning date in the CCYY/MM/DD format and its components below
ZXXPDATE	Planning date in the YY/MM/DD format and its components below
ZXXPCC	Designates the 2-digit century
ZXXPYY	Designates the 2-digit year
ZXXPMM	Designates the 2-digit month
ZXXPDD	Designates the 2-digit day
ZXXPQQQ	Designates the 3-digit Julian date
The current day's date:	
ZXXCDATE	Current date in the CCYY/MM/DD format and its components below
ZXXDATE	Current date in the YY/MM/DD format and its components below
ZXXCC	Designates the 2-digit century
ZXXYY	Designates the 2-digit year
ZXXMM	Designates the 2-digit month
ZXXDD	Designates the 2-digit day
ZXXCJDD	Current date in the CCYY.DDD format
ZXXJDD	Current date in the YY.DDD format
Additional jobset planning information:	
ZXXPRUN	Run number
ZXXPNUM	Sequence number in the day

System Variable	Meaning
ZXXUSER	Job setup operator

PCL Statement for Defining Symbols

DEFINE Statement Syntax

This is the syntax of the DEFINE statement for defining symbols:

Functional attribute:

```
DEFINE     symbol='value',...
```

Statements for User-defined Objects

A user-defined object definition can include these PCL elements:

- Fixed-parameter streams (see "DATA Statement Syntax" on page 13)
- PCL comments that hold directives or documentary data for other software products
- JCL statements, such as OUTPUT or INCLUDE

Note:

ASG recommends that you avoid using user-defined objects to store PCL objects that are part of the Cortex-Pdb model.