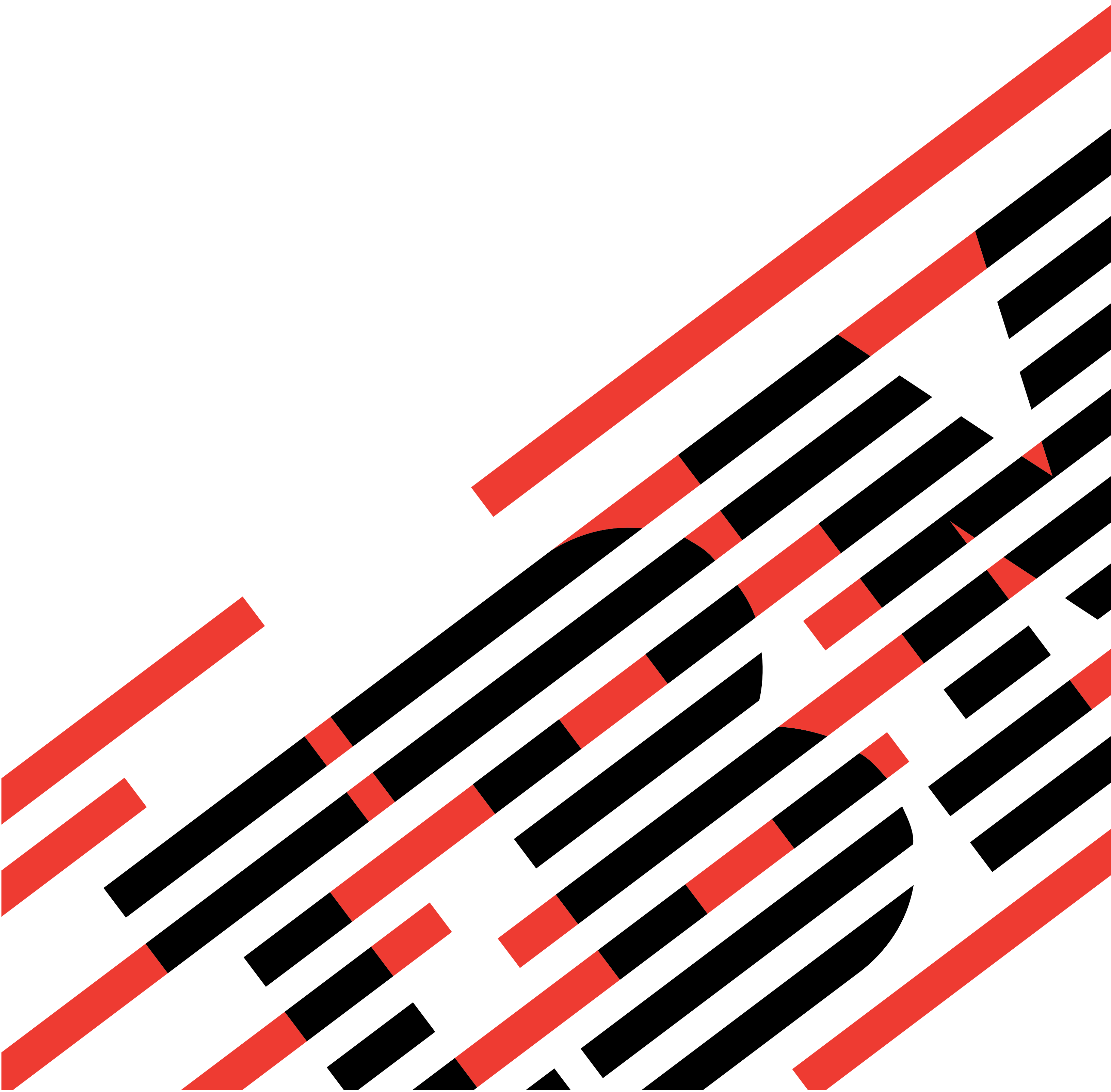
IBM

eserver

iSeries

Control Language (CL) Concepts

Version 5 Release 3



IBM

eserver®

iSeries

Control Language (CL) Concepts

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in Appendix A, “Notices,” on page 125.

Fourth Edition (May 2004)

This edition applies to version 5, release 3, modification 0 of Operating System/400 (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

CL concepts and reference

This section describes concepts and provides reference information you may need when using Control Language (CL) commands.

»

About control language (CL)

This topic describes all control language (CL) commands that are part of the base operating system, as well as commands that are optionally installable parts of the base operating system. It also includes many commands for related licensed program products that can be used on iSeries(TM) servers.

Before using CL commands, you should be familiar with the concepts discussed in this topic. On occasion, you may need to refer to other IBM(R) books or topics for more specific information about a particular topic.

See Print these topics to print this information and to find related information.

Following is additional general information about this topic:
- Who should use commands (page 1)
- How this information is organized (page 1)
- Code disclaimer information (page 1)

Who should use commands

CL commands are typically used by programmers, data processing managers, and system administrators. Through OS/400(R) command menus, powerful command prompter tools, and online command help, non-technical users can also use CL commands. To use commands, you should have a general understanding of OS/400. If you want to combine several CL commands together to create a CL program, some background in programming is helpful. **CL Programming** has more information on writing CL programs and defining your own CL commands.

How this information is organized

In the CL command finder, you can search for commands by product, by command name, by descriptive name, or by part of the name. You also can search for new commands and changed commands.

Code disclaimer information

This topic contains programming examples.

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar functions tailored to your own specific needs.

All sample code is provided by IBM for illustrative purposes only. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

All programs contained herein are provided to you "AS IS" without any warranties of any kind. The implied warranties of non-infringement, merchantability, and fitness for a particular purpose are expressly disclaimed.

«

Commands

You can use Control Language (CL) commands to request system functions. See the following for command concepts.

-
- "Command names"
- "Commands operating on OS/400 objects" on page 3
- "Commands operating on multiple objects" on page 3
- "Command description format" on page 4
- "Command parts" on page 8
- "CL programs" on page 22
- "Command definition statements" on page 38
- "Parameter values used for testing and debugging" on page 40

Command names

Most command names consist of a combination of the verb and the object being acted on: (command = verb + object acted on). For example, you can create, delete, or display a library; so the verb abbreviations CRT, DLT, and DSP are joined to the abbreviation for library, LIB. The result is three commands that can operate on a library: CRTLIB, DLTLIB, and DSPLIB.

The conventions for naming the combination verb and object commands are as follows:
- The primary convention (as just shown) is to use three letters from each word in the descriptive command name to form the abbreviated command name that is recognized by the system.
- The secondary convention is to use single letters from the ending word or words in the command title for the end of the command name, such as the three single letters DLO on the DLTDLO (Delete Document Library Object) command.
- An occasional convention is to use single letters in the middle of the command name (usually between commonly used three-character verbs and objects), such as the letters CL in the CRTCLPGM (Create CL Program) command.

Some command names consist of the verb only, such as the MOV (Move) command, or an object only, such as the DATA (Data) command.

» For a complete list of object types, see Object types used by commands containing the OBJTYPE parameter (page 69) table.

For a list of all abbreviations used in command (and keyword) names, see the Abbreviations of CL Commands and Keywords appendix in *CL Programming* « .

A few commands have an OS/400(R) command name, and can also be called using one or more alternate names that may be familiar to users of systems other than the OS/400 system. An alternate name is known as an **alias**, such as the name CD is an alias for the CHGCURDIR (Change Current Directory) command.

Commands operating on OS/400 objects

Each of the OS/400 object types has a set of commands that operates on that object type. Most OS/400 object types have commands that perform the following actions:

- Create (CRT): Creates the object and specifies its attributes.
- Delete (DLT): Deletes the object from the system.
- Change (CHG): Changes the attributes and/or contents of the object.
- Display (DSP): Displays the contents of the object. Display commands cannot be used to operate on objects.
- Work with (WRK): Works with the attributes and/or contents of the object. Unlike display commands, work commands allow users to operate on objects and modify applications.

For additional information, see "Commands operating on multiple objects."

Commands operating on multiple objects

In addition to the commands that operate on single object types, there are commands that operate on several object types. These commands are more powerful because they can operate on several objects of different types at the same time. For example:

- » Display object description (DSPOBJD or DSPLNK): Displays the common attributes of an object.
- Move object (MOVOBJ or MOV): Moves an object from one library or directory to another.
- Rename object (RNMOBJ or RNM): Specifies the new name of an object.
- Save object (SAVOBJ or SAV): Saves an object and its contents on tape, optical media, or in a save file.
- Restore object (RSTOBJ or RST): Restores a saved version of the object from tape, optical media, or from a save file. «

» See the "Commands Operating on Multiple Object Types (where object identified by object name, library and type)" and Commands Operating on Multiple Object Types (where object identified by path name) (page "Commands Operating on Multiple Object Types (where object identified by path name)" on page 4) tables to see a list of the commands that perform an action on many of the object types. « Some of the commands, such as the Move Object (MOVOBJ) command, operate on only one object at a time, but that object can be any one of several OS/400(R) object types. For example, the MOVOBJ command can move a file or a job description.

You can also refer to the Object Types Used by Commands Containing the OBJTYPE Parameter (page 69) table in "OBJTYPE parameter" on page 69 to see how these multiple-object commands affect specific object types.

For additional information, see "Commands operating on OS/400 objects."

Commands Operating on Multiple Object Types (where object identified by object name, library and type)

Item	Actions	Identifier
Object	ALC, CHK, CPR, DCP, DLC, DMP, MOV, RNM, RST, SAV, WRK, CRTDUP, SAVCHG	OBJ
Object Access	SET	OBJACC
Object Auditing	CHG	OBJAUD

Item	Actions	Identifier
Object Authority	DSP, EDT, GRT, RVK	OBJAUT
Object Description	DSP	OBJD
» Object Journaling	CHG, END, STR	JRNOBJ «
Object Lock	WRK	OBJLCK
Object Owner	CHG, WRK	OBJOWN
Object Primary Group	CHG, WRK	OBJPGP

Commands Operating on Multiple Object Types (where object identified by path name)

Item	Actions	Identifier
Object	CPY, MOV, RNM, RST, SAV	not applicable
Object Auditing	CHG	AUD
Object Authority	CHG, DSP, WRK	AUT
Object Description	DSP, WRK	LNK
Object Integrity	CHK	OBJITG
Object Journaling	END, STR	JRN
Object Owner	CHG	OWN
Object Primary Group	CHG	PGP

«

Command description format

Each command description follows the same format and includes the parts discussed in the following paragraphs. » At the beginning of the command description documentation, there are links to the Parameters, Examples, and Error messages sections. «

It should be noted that, because a command is an OS/400$^{(R)}$ object, each command can be authorized for specific users or authorized for use by the public (all users authorized in some way to use the system). Because this is true for nearly every command, it is not stated in each command description. The *iSeries Security Reference* book contains additional information about IBM$^{(R)}$-supplied user profiles and the commands authorized for each.

»

Environment and threadsafe classification

At the very top of the command description documentation are environment and threadsafe classifications. **Where allowed to run:** indicates in which environments the command can be run. **Threadsafe:** indicates whether a command is threadsafe.

For more details about environment, see "Environment" on page 5. For more details about threadsafe classifications, see "Threadsafe classification" on page 6. «

Command description

The general description of the command follows the environment and threadsafe classification. It briefly explains the function of the command and any relationships it has with a program or with other commands. If there are restrictions on the use of the command, they are described under the heading "Restrictions."

Parameters

The **Parameters** section provides a parameter summary table. The parameter summary table shows all the parameters and values that are valid for the command. Possible values are indicated in the Choices column. The default value, as shipped by IBM, is underlined in the Choices column. The default values are used by the system for parameters or parts of parameters that are not coded.

See "Parameter summary table" on page 7 for more details.

Parameter descriptions

Parameter descriptions follow the parameter summary table. Parameter descriptions are presented in the same order as the parameters are listed in the parameter summary table. Each parameter description includes an explanation of the function of the parameter, followed by a description of each possible parameter value. The default parameter value, if there is one, is usually described first and is shown as an underlined heading at the beginning of the text that describes the value.

The description of each parameter explains what the parameter means, what it specifies, and the dependent relationships it has with other parameters in the command. When the parameter has more than one value, the information that applies to the parameter as a whole is covered first, then the specific information for each of the values is described after the name of each value.

Command coding examples

The **Examples** section provides at least one coded example for the command. Where necessary, several examples are provided for commands with many parameters and several logical combinations.

For clarity, examples are coded in keyword form only. The same examples could be coded either in positional form or in a combination of keyword and positional forms, for commands that support one or more positional parameters.

See Code disclaimer information (page 1) for information pertaining to code examples.

Error messages

The **Error messages** section lists error messages that can be issued for the command.

Environment

Where allowed to run indicates in which environments the command can be entered. This is the same information that is shown in the output of the Display Command (DSPCMD) command, which reflects what was specified for the ALLOW parameter when the command definition object was created. The "Where allowed to run" value includes the symbolic special values specified for the ALLOW parameter and a brief description that explains the environments where the command is allowed to run.

The majority of commands are created with ALLOW(*ALL); *ALL is also the shipped default value for the ALLOW parameter. In this case, the description will be "All environments (*ALL)".

For commands that must be run interactively, the ALLOW values specified when the command was created are usually (*INTERACT *IPGM *IREXX *EXEC) or (*INTERACT *IPGM *IMOD *IREXX *EXEC).

In these two cases, the description shown will be “Interactive environments (*INTERACT *IPGM *IREXX *EXEC)” or “Interactive environments (*INTERACT *IPGM *IMOD *IREXX *EXEC)”.

For commands that are created to be run only in a compiled CL or interpreted REXX program, the ALLOW values specified when the command was created are usually (*BPGM *IPGM *BREXX *IREXX) or (*BPGM *IPGM *BMOD *IMOD *BREXX *IREXX). In these two cases, the description shown will be “Compiled CL program or interpreted REXX (*BPGM *IPGM *BREXX *IREXX)” or “Compiled CL or interpreted REXX (*BPGM *IPGM *BMOD *IMOD *BREXX *IREXX)”.

If the combination of values specified for the ALLOW parameter when the command was created is not one of the above combinations, a bulleted list is shown that gives a brief description of each value that was specified.

- Batch job (*BATCH)
- Interactive job (*INTERACT)
- Batch ILE CL module (*BMOD)
- Interactive ILE CL module (*IMOD)
- Batch program (*BPGM)
- Interactive program (*IPGM)
- Batch REXX procedure (*BREXX)
- Interactive REXX procedure (*IREXX)
- Using QCMDEXEC, QCAEXEC, or QCAPCMD API (*EXEC)

Note: Some command definition objects shipped as part of OS/400$^{(R)}$ are not intended to be used as CL commands. For example, the CMD and PARM command definition objects are used in command definition source. These special-purpose command objects will not have any “Where allowed to run” information. »

Threadsafe classification

The threadsafe classification indicates whether a command is threadsafe. Each command has a threadsafe classification. The three types of threadsafe classifications are as follows:

- Threadsafe: Yes

  This classification indicates that you can safely call the command simultaneously in multiple threads without restrictions. This classification also indicates that all functions called by this command are threadsafe.
- Threadsafe: Conditional

  This classification indicates that not all functions provided by the command are threadsafe. The Restrictions section of the command provides information relating to thread safety limitations. Many commands are classified conditionally threadsafe because either some underlying system support is not threadsafe or the command can cause an exit point to be called. Some conditionally threadsafe commands may deny access under some circumstances. The command restriction section describes the conditions that cause the command to deny access.
- Threadsafe: No

  This classification indicates that the command is not threadsafe and should not be used in a multithreaded program. While some thread unsafe commands may deny access, most thread unsafe commands do not. A diagnostic message, CPD000D, may be sent to the job log to indicate that a non-threadsafe command has been called. Whether or not the diagnostic message CPD000D is sent to the job log depends on the “multithreaded job action” attribute of the command; that attribute can be determined by using the Display Command (DSPCMD) command. The possible values and actions are:
  - *SYSVAL - Action is based on system value QMLTTHDACN
  - *RUN - Command runs. No messages are sent.
  - *MSG - Diagnostic message CPD000D is sent to the job log. The command runs.

– *NORUN - Diagnostic message CPD000D is sent to the job log, and escape message CPF0001 is sent. The command does not run.

If the command is run, the results are unpredictable.

Note: Some command definition objects shipped as part of OS/400$^{(R)}$ are not intended to be used as CL commands. For example, the CMD and PARM command definition objects are used in command definition source. These special-purpose command objects will not have any "Threadsafe" information.

« »

Parameter summary table

The parameter summary table summarizes parameters and values for CL commands. The parameter summary table replaces the syntax diagrams used in past releases.

See the following topics for information about parameter summary table format.
- Keyword column (page 7)
- Description column (page 7)
- Choices column (page 7)
- Notes column (page 7)

Keyword column

This column shows the *parameter keyword* name. Every CL command parameter has a keyword name associated with it. When you are viewing the command documentation using a browser, you can click on the keyword name to link to the start of the information for the parameter within the command documentation file.

Description column

This column shows the prompt text defined for the parameter, a parameter qualifier, or a parameter element. *Qualifiers* are normally used for qualified object names or qualified job names. *Elements* are used to define multiple input fields for a single parameter. The description for a qualifier or element contains the qualifier or element number within the parameter.

Choices column

This column shows the possible values for the parameter, qualifier, or element.
- *Predefined values*, also known as special values, are listed in this column. Predefined values usually begin with an asterisk (*) or Q, followed by all uppercase letters.
- If the parameter, qualifier, or element allows *user-defined values*, a description of the parameter type appears in italics, for example *Name*.
- *Default values* may be defined for optional parameters. Default values are shown in bold, underlined text, for example ***NO**.
- For complex parameters that have multiple qualifiers or elements, or if the parameter or element supports a list of values, any *single value* choices are identified. Single value choices may be used only once.
- *Repeated values* can be specified for some parameters. For repeated values, this column indicates the number of allowed repetitions.

Notes column

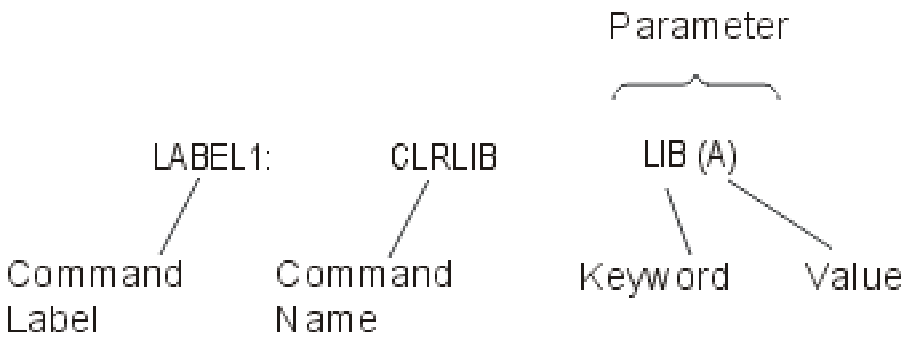
This column shows additional information about each parameter.

- “Required” appears in this column to indicate *required parameters*, parameters for which you are always required to specify an input value.
- “Optional” appears in this column to indicate *optional parameters*, parameters for which no input value is required.
- “Key” appears in this column to indicate *key parameters*, which are used by commands that have prompt override programs.
- “Positional” appears in this column if the parameter is allowed to be specified positionally (without the associated parameter keyword) in a command string. The parameter’s positional number appears following “Positional”.

«

Command parts

This figure illustrates the parts of a command, which includes: command label (optional), command name (mnemonic), and one or more parameters. The parameter includes a keyword and a value.



For information about the parts of a command, see the following:
- “Command label”
- “Command name” on page 9
- “Command parameters” on page 9

In addition, see the following for additional information about command coding:
- “Command syntax” on page 18
- “Command delimiters” on page 18
- “Command continuation” on page 19
- “Entering comments in commands” on page 20
- “Summary of general command coding rules” on page 20
- “Double-byte character text in CL commands” on page 22

For additional information about commonly used parameters, see “Commonly used parameters: Expanded descriptions” on page 51.

Command label

Command labels identify particular commands for branching purposes in a CL program. Labels can also be used to identify statements in CL programs that are being debugged. They can identify statements used either as breakpoints or as starting and ending statements for tracing purposes.

» The label is typed just before the command name. The standard rules for specifying simple names (*SNAME) apply. The label is immediately followed by a colon. Blanks are allowed, though not required, between the colon and the command name. The label can contain as many as 10 characters, in addition to the colon. « START: and TESTLOOP: are examples of command labels.

Command labels are not required, but a label can be placed on any command. For labels that are placed on commands that cannot be run (for example, the Declare CL Variable (DCL) command), when the program branches to that label, the next command following the label is run. If the command following the label cannot be run, the program will move to the next command that can be run. Similarly, you can specify only one label on a line. If a command is not located on that line, the program will jump to the next command that can be run.

To specify multiple labels, each additional label must be on a separate line preceding the command as shown:

```
LABEL1:
LABEL2:  CMDX
```

No continuation character (+ or -) is allowed on the preceding label lines.

Command name

The command name identifies the function that will be performed by the program that is called when the command is run. The command name is an abbreviation of the command description. For example, the name MOVOBJ identifies the CL command (Move Object) that moves an object from one library to another. Like other objects, a command name can be optionally qualified by a library name.

For more information about object names and how CL commands are named, see “Simple and qualified object names” on page 46 and “Object naming rules” on page 48.

Command parameters

Most CL commands have one or more *parameters* that specify the objects and values used to run the commands. When a command is entered, the user supplies the command object name, the parameter keyword names, and the parameter values used by the command. The number of parameters specified depends upon the command. Some commands (like the DO (Do) command and the ENDBCHJOB (End Batch Job) command) have no parameters, and others have one or more.

The specification of a group of values on one parameter is described under “List of values” on page 16.

In this topic, the word *parameter* usually refers to the combination of the parameter keyword and its value. For example, the Move Object (MOVOBJ) command has a parameter called OBJ that requires an object name to be specified. OBJ is the parameter keyword, and the name of the object is the value entered for the OBJ parameter.

See the following topics:

- “Required, optional, and key parameters”
- “Parameters in keyword and positional form”
- “Parameter values” on page 11

Required, optional, and key parameters: A command can have parameters that must be coded (required parameters) and parameters that do not have to be coded (optional parameters). Optional parameters are usually assigned a system-defined default value if another value is not specified for the parameter when the command is entered.

A command can also have *key parameters* which are the only parameters shown on the display when a user prompts for the command. After values are entered for the key parameters, the remaining parameters are shown with actual values instead of the default values (such as *SAME or *PRV).

Parameters in keyword and positional form: You can specify parameters in CL using keyword (page 9) form, positional (page 10) form, or in a combination (page 10) of the two.

Parameters in keyword form

A parameter in *keyword form* consists of a keyword followed immediately by a value (or a list of values separated by blank spaces) enclosed in parentheses. You cannot use blanks between the keyword and the left parenthesis preceding the parameter value. You can place blanks between the parentheses and the parameter value. For example, LIB(MYLIB) is a keyword parameter specifying that MYLIB is the name of the library that is used in some way, depending upon the command in which this LIB parameter is used.

When command parameters are all specified in keyword form, they can be placed in any order. For example, the following two commands are the same:

```
CRTLIB   LIB(MYLIB)  TYPE(*TEST)
CRTLIB   TYPE(*TEST)  LIB(MYLIB)
```

Parameters in positional form

A parameter in *positional form* does not have its keyword coded; it contains only the value (or values, if it is a list) whose function is determined by its position in the parameter set for that command. The parameter values are separated from each other and from the command name by one or more blank spaces. Because there is only one positional sequence in which parameters can be coded, the positional form of the previous CRTLIB (Create Library) command first example is:

```
CRTLIB   MYLIB  *TEST
```

If you do not want to enter a value for one of the parameters, the predefined value *N (null) can be entered in that parameter's position. The system recognizes *N as an omitted parameter, and either assigns a default value or leaves it null. In the previous CRTLIB command second example, if you coded *N instead of *TEST for the TYPE parameter, the default value *PROD is used when the command is run, and a production library named MYLIB is created. The description of the CRTLIB command contains the explanation for each parameter.

Notes:
- Parameters cannot be coded in positional form beyond the positional coding limit. If you attempt to code parameters in positional form beyond that point, the system returns an error message.
- » Using positional form in your CL program source may save time when writing the program, but will be more difficult for you or someone else to maintain. Commands written using keyword form are generally easier to understand and enhance. «

Combining keyword and positional forms

A command may also have its parameters coded in a combination of keyword and positional forms. The following examples show three ways to code the Declare CL Variable (DCL) command.

Keyword form:

```
DCL   VAR(&QTY)  TYPE(*DEC)  LEN(5)  VALUE(0)
```

Positional form:

```
DCL   &QTY  *DEC  5  0
```

Positional and keyword forms together:

```
DCL   &QTY  *DEC  VALUE(0)
```

In the last example, because the optional LEN parameter was not coded, the VALUE parameter *must* be coded in keyword form.

Note: You cannot specify parameters positionally after a parameter specified in keyword form.

For additional information, see "Summary of general command coding rules" on page 20.

Parameter values: A parameter value is user-supplied information used during the running of a command. An individual value can be specified in any one of the following:

- "Constant value" (its actual value): The types of constants are character string (includes names, date and hexadecimal values), decimal, and logical.
- » "Variable name" on page 15 (the name of the variable containing the value): The types of variables are character string (includes names), decimal, logical, and integer. Decimal and logical values must match the type of value expected for the parameter. Character variables can specify any type of value. For example, if a decimal value is expected, it can be specified by a character variable as well as by a decimal variable.«
- "Expressions" on page 16 (the value used is the result of evaluating an expression): The types of expressions are arithmetic, character string, relational, and logical. Expressions can be used as a value for parameters in commands in CL programs only.
- "List of values" on page 16 (a list of values is one or more values that can be specified for a parameter): Not all parameters can accept a list of values. A *list parameter* can be defined to accept a specific set of multiple values that can be of one or more types. Values in the list must be separated by one or more blanks. Each list of values is enclosed by parentheses, indicating that the list is treated as a single parameter. Parentheses are used even when a parameter is specified in positional form. To determine whether a list can be specified for a parameter, and what kind of list it can be, refer to the parameter description under the appropriate command description.

A parameter can specify one or a group of such values, depending on the parameter's definition in a command. If a group of values is specified, the parameter is called a *list parameter* because it can contain a list of values.

On commands with key and positional parameters, values can be specified in keyword form, positional form, or a combination of both forms. Parameter values must be enclosed in parentheses if any of the following conditions are true:

- A keyword precedes the value.
- The value is an expression.
- A list of values is specified.

Note: If only one value is specified for a list, no parentheses are required.

For additional information, see "Summary of general command coding rules" on page 20.

Constant value: A constant value is an actual numeric value or a specific character string whose value does not change. Three types of constants can be used by the control language: character string (quoted or unquoted), decimal, and logical.

For more information about constant value, see the following:

- Character string (page 11)
- Decimal values (page 14)
- Logical values (page 15)

Character string

A *character string* is a string of any EBCDIC characters (alphanumeric and special) that are used as a value, including date (page 14) and hexadecimal (page 14) values. A character string can have two forms: quoted string or unquoted string. Either form of character string can contain as many as 5000 characters.

A *quoted* character string is a string of alphanumeric and special characters that are enclosed in apostrophes. For example, 'Credit limit has been exceeded' is a quoted character string.

The quoted string is used for character data that is not valid in an unquoted character string. For example, user-specified text can be entered in several commands to describe the functions of the commands. Those descriptions must be enclosed in apostrophes if they contain more than one word because blanks are not allowed in an unquoted string.

An *unquoted* character string is a string consisting of only alphanumeric characters and the special characters that are shown in the *Unquoted String* column in the Quoted and Unquoted Character Strings (page 12) table. The table summarizes the main EBCDIC characters that are valid in unquoted and quoted character string values. An X in the last column indicates that the character on the left is valid; refer to the specific notes following the figure that indicate why the character is valid as described. The special characters allow the following to be unquoted character string values:

- Predefined values (* at the beginning)
- Qualified object names (/)
- Generic names (* at the end)
- Decimal constants (+, -, ., and ,)

Any of these unquoted strings can be specified for parameters defined to accept character strings. In addition, some parameters are defined to accept predefined values, names, or decimal values either singly or in combinations.

Quoted and Unquoted Character Strings

Name of Character	Character	Unquoted String	Quoted String
Ampersand	&	See Note 5	X
Apostrophe	’	See Note 7	-
Asterisk (*)	*	See Notes 5, 6	X
At sign	@	X	X
Blank			X
Colon	:		X
Comma	,	See Note 1	X
Digits	0-9	See Note 1	X
Dollar sign	$	X	X
Equal	=	See Notes 5, 8	X
Greater than	>	See Notes 5, 8	X
Left parenthesis	(	See Note 4	X
Less than	<	See Notes 5, 8	X
Letters (lowercase)	a-z	See Note 2	X
Letters (uppercase)	A-Z	X	X
Minus	-	See Notes 1, 5	X
Not	¬	See Notes 5, 8	X
Number sign	#	X	X
Percent	%		X
Period	.	See Notes 1, 11	X
Plus	+	See Notes 1, 5	X
Question mark	?		X
Quotation marks	“ ”	See Note 10	X

Name of Character	Character	Unquoted String	Quoted String
Right parenthesis	)	See Note 4	X
Semicolon	;		X
Slash	/	See Notes 3, 5	X
Underscore	_	See Note 9	X
Vertical bar	|	See Notes 5, 8	X

Notes:

1. An unquoted string of all numeric characters, an optional single decimal point (. or ,), and an optional leading sign (+ or -) are valid unquoted strings. Depending on the parameter attributes in the command definition, this unquoted string is treated as either a numeric or character value. On the CALL command or in an expression, this unquoted string is treated as a numeric value; a quoted string is required if character representation is desired. Numeric characters used in any combination with alphanumeric characters are also valid in an unquoted string.
2. In an unquoted string, lowercase letters are translated into uppercase letters unless the string is specified for a parameter that has the attribute CASE(*MIXED).
3. A slash can be used as a connector in qualified names and path names.
4. In an unquoted string, parentheses are valid when used to delimit keyword values and lists, or in expressions to indicate the order of evaluation.
5. In an unquoted string, the characters +, -, *, /, &, |, ¬, <, >, and = are valid by themselves. If they are specified on a parameter that is defined in the command definition with the EXPR(*NO) attribute, they are treated as character values. If they are specified on a parameter that is defined in the command definition with the EXPR(*YES) attribute, they are treated as expression operators.
6. In an unquoted string, the asterisk is valid when followed immediately by a name (such as in a predefined value) and when preceded immediately by a name (such as in a generic name). For further information on unquoted strings in expressions, see “Expressions” on page 16.
7. Because an apostrophe within a quoted string is paired with the opening apostrophe (delimiter) and is interpreted as the ending delimiter, an adjacent pair of apostrophes (’’) must be used inside a quoted string to represent an apostrophe that is not a delimiter. When characters are counted in a quoted string, a pair of adjacent apostrophes is counted as a single character.
8. In an unquoted string, the characters <, >, =, ¬, and | are valid in some combinations with another character in the same set. Valid combinations are: <=, >=, ¬=, ¬>, ¬<, ||, |<, and |>. If the combination is specified on a parameter that is defined in the command definition with the EXPR(*NO) attribute, it is treated as a character value. If it is specified on a parameter that is defined in the command definition with the EXPR(*YES) attribute, it is treated as an expression operator.
9. In an unquoted string, the underscore is not valid as the first character or when used by itself.
10. Quotation marks are used to delimit a quoted name.
11. A period is valid in a basic name, except as the first character.

The following are examples of quoted string constants:

Constant	Value
’1,2,’	1,2,
’DON’’T’	DON’T
’24 12 20’	24 12 20

The following are examples of unquoted strings:

Constant	**Meaning**
CHICAGO	CHICAGO
FILE1	FILE1
*LIBL	Library list
LIBX/PGMA	Program PGMA in library LIBX
1.2	1.2

More information and examples can be found in "Character string expressions" on page 30.

Date values

A date value is a character string that represents a date. Its format is specified by the system value QDATFMT. The length of the date value varies with the format used and whether a separator character is used. For example, if no separator character is used, the length of a date in a Julian format is five characters, and the length of a date in a non-Julian format is six characters. If a separator character is used, the length will be greater. More information on system value QDATFMT is in the System values topic.

The system value QDATSEP specifies the optional separator character that can be used when the date is entered. If a separator character is used, the date must be enclosed in apostrophes. For additional information on system value QDATSEP, see the *Work Management* book on the V5R1 Supplemental Manuals Web site.

A date value can be specified for the parameters of type *DATE. A year value equal to or greater than 40 indicates a year from 1940 through 1999. A year value less than 40 indicates a year from 2000 through 2039. For additional information on parameter value *DATE, see the "PARM (Parameter) statement" on page 39 description.

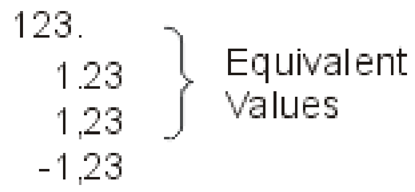
Hexadecimal values

A hexadecimal value is a constant made up of a combination of the hexadecimal digits A through F and 0 through 9. All character strings except names, dates, and times can be specified in hexadecimal form. To specify a hexadecimal value, the digits must be specified in multiples of two, be enclosed in apostrophes, and be preceded by an X. Examples are: X'F6' and X'A3FE'.

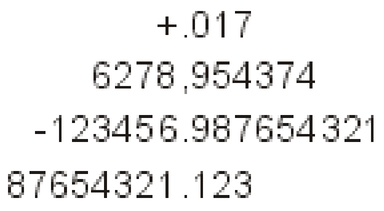
Note: Care should be used when entering hexadecimal values in the range of 00 through 3F, or the value FF. If these characters are shown or printed, they may be treated as device control characters producing unpredictable results.

Decimal values

A decimal value is a numeric string of one or more digits, optionally preceded by a plus (+) or minus (-) sign. A decimal value can contain a maximum of 15 digits, of which no more than nine can follow the decimal point (which can be either a comma or a period). Therefore, a decimal value can have no more than 17 character positions including the plus or minus sign and decimal point (if any). The following are examples of decimal values.


123.
1.23
1,23
-1,23

Equivalent Values


+.017
6278,954374
-123456.987654321
87654321.123

Logical values

A logical value is a single character (1 or 0) enclosed in apostrophes. It is often used as a switch to represent a condition such as on or off, yes or no, and true or false. When used in expressions, it can be optionally preceded by *NOT or

¬

. The following are examples of logical values:

Constant	Value	Meaning
’0’	0	Off, no, or false
’1’	1	On, yes, or true

Floating-point constants

A floating-point constant is a representation of a number that consists of:

- A significand sign: The significand sign may be + or -. The significand sign is optional; it is assumed to be + if no sign is specified.
- A significand: The significand must contain a decimal point. The maximum number of digits that can be specified for the significand is 253; however, only the first 17 significant digits are used.
- An exponent character: The exponent character must be E.
- An exponent sign: The exponent sign must be + or -. The significand sign is optional; it is assumed to be + if no sign is specified.
- An exponent: The exponent must be an integer; numbers 0 through 9 are valid. The maximum number of digits that can be specified is three.

All floating-point constants are stored as double-precision values. No blanks are allowed between any of the parts of a floating-point constant, and the parts must be in the order listed above.

Some commands have parameters for which floating-point constants can be specified:

- » Call Program (CALL) or Call Procedure (CALLPRC) command: You can use the PARM parameter to pass a floating-point constant to a called program. Any program you call must receive a floating-point constant as a double precision value.«
- Change Program Variable (CHGPGMVAR) command: You can use the VALUE parameter to change a floating-point variable in a program.
- Copy File (CPYF) command: You can use floating-point construction in the FROMKEY, TOKEY, and INCREL parameters to select which records are copied from a database file.

For more information about floating-point constants, see the DDS information in the Programming topic.

Variable name: A *variable* contains a data value that can be changed when a program is run. The variable is used in a command to pass the value that it contains at the time the command is run. The change in value can result if one of the following conditions occur: the value is received from a data area, a display device file field, or a message; the value is passed as a parameter; a Change Variable (CHGVAR) command is run in the program; or another program that is called changes the value before returning it.

The variable name identifies a value to be used; the name points to where the actual data value is. Because CL variables are valid only in CL programs, they are often called *CL program variables* or, simply, CL variables. CL variable names must begin with an ampersand (&).

CL variables can be used to specify values for almost all parameters of CL commands. When a CL variable is specified as a parameter value and the command containing it is run, the current value of the variable is used as the parameter value. That is, the variable value is passed as if the user had specified the value as a constant.

Because it is generally true that CL variables can be used for most parameters of commands in CL programs, the command descriptions usually do not mention CL variables. For parameters that are restricted to constants only (such as in the DCL command), to CL variables only (such as all of the parameters of the Retrieve Job Attributes (RTVJOBA) command), or to specific types of variables (such as on the RTVJOBA or Retrieve Message (RTVMSG) command), the individual parameter descriptions specify those limitations. Otherwise, if the command is allowed in a CL program, CL variables can be used in place of a value, even with parameters that accept only predefined values. For example, a KEEP parameter having only predefined values of *YES and *NO can have a CL variable specified instead; its value can then be changed to *YES or *NO, depending on its value when the command is run.

A CL variable must contain only one value; it may not contain a list of values separated by blanks.

The value of any CL program variable can be defined as one of the following types:

- Character: A character string that can contain a maximum of 9999 characters. The character string can be coded in quoted or unquoted form, but only the characters in the string itself are stored in the variable.
- Decimal: A packed decimal value that can contain a maximum of 15 digits, of which no more than nine can be decimal positions.
- Logical: A logical value of '1' or '0' that represents on/off, true/false, or yes/no.
- » Integer: A two-byte or four-byte binary integer value that can be either signed (value may be positive or negative) or unsigned (value is always positive). «

If value is:	**CL variable can be declared as:**
Name	Character
Date or time	Character
Character string	Character
Numeric	» Decimal or integer or character «
Logical	Logical or character

Expressions: An expression is a group of constants or variables, separated by operators, that produces a single value. The operators specify how the values are combined to produce the single value or result. The operators can be arithmetic, character string, relational, or logical. » The constants or variables can be character, decimal, integer, or logical. « For example, the expression (&A + 1) specifies that the result of adding 1 to the value in the variable &A is used in place of the expression.

Character string expressions can be used in certain command parameters defined with EXPR(*YES) in CL programs. An expression can contain the built-in functions %BINARY (or %BIN), %SUBSTRING (or %SST), and %SWITCH, which are covered in detail in "Expressions in CL commands" on page 29. The types of expressions and examples of each are described there.

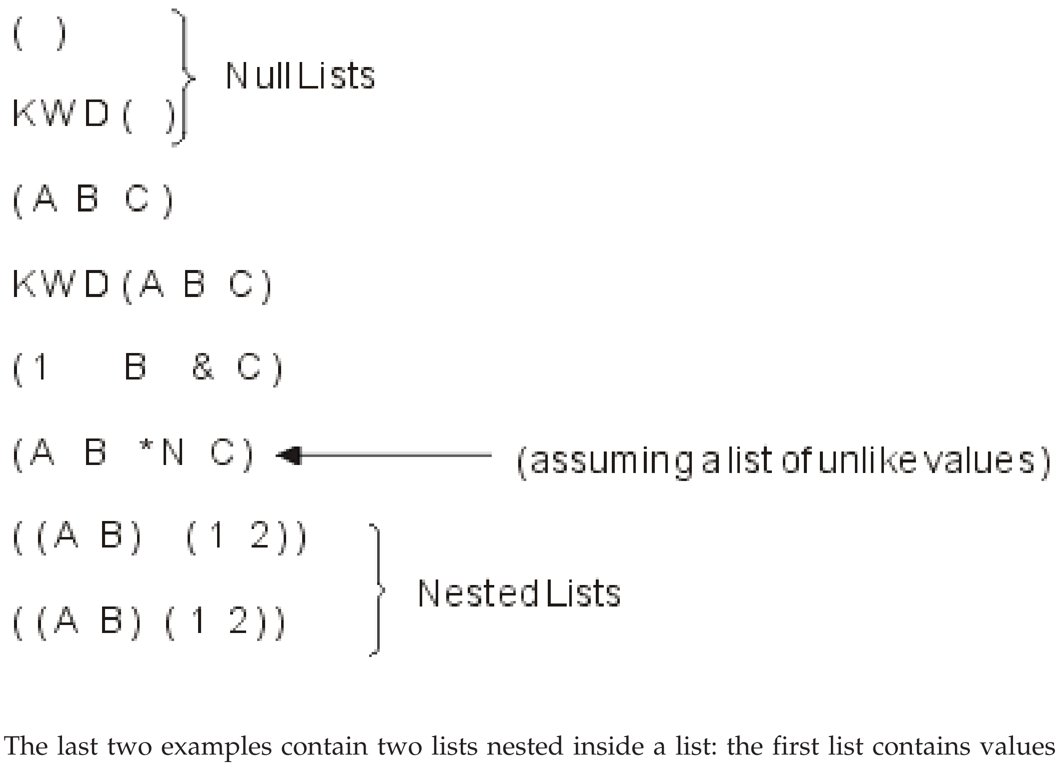
List of values: A list of values is one or more values that can be specified for a parameter. Not all parameters can accept a list of values. A *list parameter* can be defined to accept a specific set of multiple values that can be of one or more types. Values in the list must be separated by one or more blanks. Each list of values is enclosed by parentheses, indicating that the list is treated as a single parameter. Parentheses are used even when a parameter is specified in positional form. To determine whether a list can be specified for a parameter, and what kind of list it can be, refer to the parameter description under the appropriate command description.

A list parameter can be defined to accept a list of multiple like values (a simple list) or a list of multiple unlike values (a mixed list). Each value in either kind of list is called a *list element*. List elements can be constants, variables, or other lists; expressions are not allowed.

- A *simple list* parameter accepts one or more values of the type allowed by a parameter. For example, (RSMITH BJONES TBROWN) is a simple list of three user names.
- A *mixed list* parameter accepts a fixed set of separately defined values that are in a specific order. Each value can be defined with specific characteristics such as type and range. For example, LEN(5 2) is a mixed list in which the first element (5) gives the length of a field and the second element (2) gives the number of decimal positions in that field.
- For many parameters defined to accept lists, predefined single values can be specified in place of a list of values. One of these single values can be the default value, which can be either specified or assumed if no list is specified for a simple or mixed list. To determine what defaults are accepted for a given list parameter, refer to the description of the parameter in the description of the command for which the parameter is defined and used.

  Note: *N cannot be specified in a simple list, but it can be specified in a mixed list. Also, individual parameters passed on the CALL and CALLPRC commands cannot be lists.
- The maximum level of nesting of lists inside lists is three, including the first. These are indicated by three nested levels of parentheses.

The following are examples of lists:



```
( )
KWD( )
(A B C)
KWD(A B C)
(1    B   & C)
(A  B  *N  C)
((A B)   (1 2))
((A B) (1 2))
```

The last two examples contain two lists nested inside a list: the first list contains values of A and B, and the second list contains values of 1 and 2. The space between the two nested lists is not required. Blanks are the separators between the values inside each nested list, and the sets of parentheses group the nested values into larger lists.

Command syntax

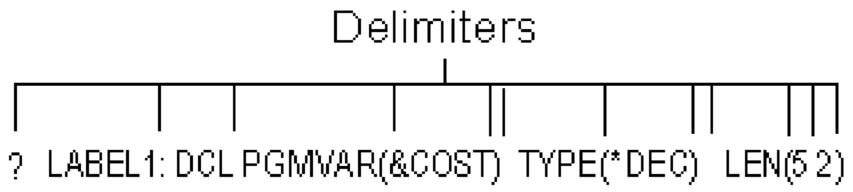
Commands have the following general syntax. The brackets indicate that the item within them is optional; however, the parameter set may or may not be optional, depending upon the requirements of the command.

```
[//] [?] [label-name:][library-name/]command-name
[parameter-set]
```

Note: The // is valid only for a few batch job control commands, such as the DATA command. The // identifies those types of commands sent to the spooling reader that reads the batch job input stream.

Command delimiters

Command delimiters are special characters or spaces that identify the beginning or end of a group of characters in a command. Delimiters are used to separate a character string into the individual parts that form a command: command label, command name, parameter keywords, and parameter values. Parameter values can be constants, variable names, lists, or expressions. The following diagram shows various delimiters for a command:



The following delimiters are used in the OS/400(R) control language:

- The colon (:) ends the command label, and it can be used to separate the command label from the command name.
- Blank spaces separate the command name from its parameters and separate parameters from each other. They also separate values in a list. Multiple blanks are treated as a single blank except in a quoted character string or comment enclosed in apostrophes. A blank *cannot* separate a keyword and the left parenthesis for the value.
- Parentheses ( ) are used to separate parameter values from their keywords, to group lists of values, and to group lists within lists.
- Slashes (/) connect the parts of a qualified name or the parts of a path name.
  - For a qualified object name, the two parts are the library qualifier and the object name (LIBX/OBJA). Qualified object names are described in “Simple and qualified object names” on page 46.
  - For a path name, the parts are the directory or directories searched and the object name ('/Dir1/Dir2/Dir3/ObjA').
- Either a period or a comma can be used as a decimal point in a decimal value (3.14 or 3,14). Only one decimal point is allowed in a value.
- Apostrophes specify the beginning and ending of a quoted character string, which is a combination of any of the 256 extended binary-coded decimal interchange code (EBCDIC) characters that can be used as a constant. For example, 'YOU CAN USE $99@123.45 ()*></ and lowercase letters' is a valid quoted string that is a constant.

  Because an apostrophe inside a quoted string is paired with the opening apostrophe (delimiter) and is interpreted as the ending delimiter, an apostrophe inside a quoted string must be specified as two apostrophes. A pair of adjacent apostrophes used this way is counted as a single character.
- A special character is used to separate a date into three parts: month, day, and year (two parts for Julian dates: year and day). The special characters that may be used as date separators are the slash

(/), the hyphen (-), the period (.), a blank ( ), and the comma (,). The special character used to code as separators in a command date must be the same as the special character specified as the date separator for the job.

- The characters /* and */ can indicate the beginning and ending of a comment, or can be used in a character string. To begin a comment, the characters /* must begin in the first position of the command, be preceded by a blank, or be followed by either a blank or an asterisk. If the characters /* or */ occur in a later position of a command, they will usually be enclosed in apostrophes and can represent, for example, all objects in the current directory for a path name. For more information on path names, see the Integrated File Systems topic.
- A question mark (?) preceding the command name indicates that the command is prompted. If the command is specified with a label, the question mark may either precede the label, or it may follow the label and precede the command name.

  Within a CL program, when a question mark precedes a command name, a prompt display is presented. You can enter parameter values not specified on the command in the program.

  Prompting characters may be put into a command in two forms. A single question mark (?) may be coded before the command name (either before or after the command label in a CL program) to cause the entire command to be prompted. Selective prompt characters (?? or ?*) may be coded before any parameter keyword to cause that parameter to be prompted when the command is run.

  If a question mark is entered before the command name on the command entry display, the effect is the same as pressing the F4 (Prompt) key after the command is entered.

  Within a CL program, when a question mark precedes the command name, a prompt display is presented. This display is of the same format as that presented when pressing the F4 key from the command entry display. Parameters of the command for which the program has coded values are shown for informational purposes, but the user cannot change the values supplied by the program. Parameters for which no value was coded are shown as input fields so you can enter values to be used in processing the command.

  Selective prompting allows you to identify specific command parameters to be prompted. To call selective prompting, the characters ??, ?*, or ?- are coded immediately preceding the keyword name of the parameter(s) to be prompted. More information on prompting is available in the *CL Programming* book.

  Notes:

  – Selective prompting is not allowed with command string (*CMDSTR) parameters.

  – Parameters of the command that are preceded by the characters ?* are shown, but you cannot change the values that are supplied by the program. Parameters preceded by the characters ?? are shown as input fields containing the values coded in the program or command defaults so you can enter or change the values used in processing the command. Parameters preceded by the characters ?- are omitted from the display. All selectively prompted parameters must be coded in keyword or keyword-with-value form. Several parameters may be selectively prompted within one command. When selective prompting is called, only keywords that are immediately preceded by the selective prompt characters are prompted. All other parameters are processed using the values as coded on the command or, if not coded, using command defaults.

    Either form of prompting, but not both, is allowed on a single command in a CL program. If the character ? precedes the command name and selective prompt characters (except ?-) precede any keyword, an error message is returned and the program is not created.

For additional information, see “Summary of general command coding rules” on page 20.

Command continuation

Commands can be entered in free format. This means that a command does not have to begin in a specific location on a coding sheet or on the display. A command can be contained entirely in one record, or it can be continued on several lines or in several records. Whether continued or not, the total command length cannot exceed 32,702 characters. Either of two special characters, the plus sign (+) or the minus sign (-), is entered as the last nonblank character on the line to indicate that a command is continued. Blanks immediately *preceding* a + or - sign are always included; blanks immediately following a + or - in the *same record* are ignored. Blanks in the *next record* that precede the first nonblank character in the record are ignored when + is specified but are included when - is specified.

The + is generally useful between parameters or values. At least one blank must precede the sign when it is used between separate parameters or values. The difference between the plus and minus sign usage is particularly important when continuation occurs inside a quoted character string.

The example that follows shows the difference. See Code disclaimer information (page 1) for information pertaining to code examples.

```
CRTLIB LIB(XYZ) TEXT('This is CONT+
   INUED')

CRTLIB LIB(XYZ) TEXT('This is CONT-
   INUED')

For + : CRTLIB LIB(XYZ) TEXT('This is CONTINUED')

For - : CRTLIB LIB(XYZ) TEXT('This is CONT   INUED')
```

Notes:

- The minus sign causes the leading blanks on the next line to be entered.
- Use continuation characters + and - in CL programs only. An error occurs if + or - is used on a command entry display.
- The characters + and - are used for multiple-command examples, but not for single-command examples.

Entering comments in commands

Comments can be inserted either inside or outside a command’s character string wherever a blank is permitted. However, because a continuation character must be the last nonblank character of a line (or record), comments *may not* follow a continuation character on the same line.

For readability, it is recommended that each comment be specified on a separate line preceding or following the command it describes, as shown here:

```
MOVOBJ   OBJA  TOLIB(LIBY)
       /* Object OBJA is moved to library LIBY. */
DLTLIB  LIBX
       /* Library LIBX is deleted. */
```

Comments can include any of the 256 EBCDIC characters. However, the character combination */ should not appear within a comment because these characters end the comment. To begin a comment, the characters /* must be placed in the first position of the command, be preceded by a blank, or be followed by either a blank or an asterisk.

Summary of general command coding rules

This section contains a summary of general information needed to properly code CL commands.

Delimiters

- Blanks are the basic separators between the parts of a command:
  - Between command label and command name (not required, because the colon [:] is the delimiter).
  - Between command name and first parameter, and between parameters.
  - Between values in a list of values (not required between ending and beginning parentheses that enclose nested lists inside a list).
  - Between the slashes and the name or label of some job control commands, like // DATA (not required).

- Blanks *cannot* separate a parameter's keyword from the left parenthesis preceding its values. When a keyword is used, parentheses must be used to enclose the values; blanks *can* occur between the parentheses and the values. For example, KWD(A   ) is valid.
- Multiple blanks are treated as a single blank, unless they occur in a quoted string or a comment.
- A colon must immediately follow a command label. Only one label can be used on any command (LABEL1: DCLF).
- Apostrophes must be used to specify the beginning and end of a quoted character string. If a character string contains special characters, such as blanks, apostrophes are required. If an apostrophe must be used in the quoted string, two apostrophes must be entered consecutively to indicate that it is an apostrophe and not the end of the quoted string.
- Parentheses must be used:
  - On parameters that are specified (coded) in keyword form
  - To group multiple values in a single list, in a positional parameter, or around expressions
  - To indicate a list (of none, one, or several elements) nested inside *another* list
- Sets of parentheses inside parentheses can be entered as long as they are paired, up to the maximum of five nested levels in logical expressions or three nested levels in lists of values.
- Comments can appear wherever blanks are permitted, except after a continuation character on the same line or record.
- A plus or minus sign at the end of a line indicates that the command is continued on the following line. Blanks following a + or - sign in the same record are ignored; blanks in the next record that precede the first nonblank character are ignored when + is specified and included when - is specified. One blank must precede the + sign when it is used between separate parameters or values.

Parameters

- All required parameters must be coded.
- If an optional parameter is not coded, the system uses its default value if the parameter has one. » A default value is indicated by showing the value as bold and underlined text in the Choices column of the parameter summary table.«
- Words or abbreviations specified in capital letters in the command and parameter descriptions must be coded as shown. This is true of all command names, the keywords of parameters (if used), and many parameter values. Lowercase letters not coded in quoted strings or in comments are translated to uppercase letters. Lowercase letters specified for values on parameters defined as CASE(*MIXED) are not translated to uppercase letters.
- If there are key parameters, the values for the key parameters must be entered on the prompt before the remaining parameters will be shown. » The Notes column in the parameter summary table indicates which parameters, if any, are key parameters.«
- » Parameters cannot be coded in positional form past the positional parameter limit defined in the command object. The Notes column in the parameter summary table indicates which parameters may be specified in positional form.«

Parameter Values

- The first character in all names must be an alphabetic character (A through Z, $, #, @, or a double quotation mark (")). Names must not exceed 10 characters (CL variable names and built-in function names can have 11 characters maximum, including the preceding & or % characters). In some commands, the names of objects can be specified in qualified form (library-name/object-name) or in path name form (directory-name/object name).
- Predefined values that begin with an asterisk can be used only for the purposes intended, unless they are included in comments or quoted strings. These include predefined parameter values (*ALL, for example), symbolic operators (*EQ, for example), and the null value (*N).

- In a CL program, a variable can be specified for all parameters, except where that is explicitly restricted. The user-specified value of the variable is passed as if it had been specified on the command.
- In a CL program, a character string expression can be specified for any parameter defined with EXPR(*YES). The resulting value of the expression is passed as if the value had been specified on the command.
- Null (omitted) values are specified with the characters *N, which mean that no value was specified and the default value, if one exists, should be used. *N is needed only when another value following the omitted value is being specified as a positional parameter or an element in a list.
- Either a comma or a period can be used to indicate a decimal point in a numeric value. The decimal point is the only special character allowed between digits in the numeric string; there is no delimiter for indicating thousands.
- When repetition is indicated for a parameter:
  - A predefined value is not coded more than once in a series of values.
  - As many user-defined values (like names or numeric limits) can be entered as there are different values or names, up to the maximum number of repetitions allowed. For example, if a parameter description states ″Up to 20 repetitions,“ you can specify a maximum of 20 values.

Note: When you are using parameters that have the same name in different commands, the meaning of (and the values for) that parameter in each command may be somewhat different. Refer to the correct command description for the explanation of the parameter you are using. For some parameters, you can also refer to “Commonly used parameters: Expanded descriptions” on page 51 for both general information about a parameter and an expanded description of its values coded in commands.

Double-byte character text in CL commands

You can use double-byte character data anywhere in a CL command that descriptive text can be used.

Enter double-byte character text as follows:

1. Begin the double-byte character text with an apostrophe (′).
2. Enter a shift-out character.
3. Enter the double-byte character text.
4. Enter a shift-in character.
5. End the double-byte character text with an apostrophe (′).

For example, to enter the double-byte character literal ABC, enter the following, where $\boxed{\text{so}}$ represents the shift-out character and $\boxed{\text{si}}$ represents the shift-in character:

′ $\boxed{\text{so}}$ ABC $\boxed{\text{si}}$ ′

Limit the length of a double-byte character text description of an object to 14 double-byte characters, plus the shift control characters, to make sure that the description is properly displayed and printed.

CL programs

A Control Language (CL) program is a program that is created from source statements consisting entirely of control language commands.

See the following for concepts relating to CL programs:

- “CL character sets and values” on page 23

- “Naming within commands” on page 25
- “Expressions in CL commands” on page 29

CL character sets and values

This section explains the usage of the EBCDIC character sets, special characters, and IBM(R) -defined fixed values called predefined values.

See the following:
- “Character sets”
- “Special character use”
- “Predefined values” on page 25

Character sets: The control language uses the extended binary-coded decimal interchange code (EBCDIC) character set. For convenience in describing the relationship between characters used in the control language and those in the EBCDIC character set, the following control language categories contain the EBCDIC characters shown:

Category	Characters Included
Alphabetic[1]	26 letters (A through Z), $, #, and @
Numeric	10 digits (0-9)
Alphanumeric[1,2]	A through Z, 0 through 9, $, #, @, period (.), and underscore (__)
Special Characters	All other EBCDIC characters (for those having special uses in CL, see “Special character use”)

Notes:

1 Lowercase letters (a through z) are accepted, but they are translated into the corresponding uppercase letters by the system except when they are included within a quoted character string or a comment, or they are specified for a value on a parameter that has the character (*CHAR) or the path name (*PNAME) attribute for its TYPE parameter and the mixed case (*MIXED) attribute for its CASE parameter in the command definition.

2 The underscore (__) is an alphanumeric connector that can be used in OS/400(R) CL to connect words or alphanumeric characters to form a name (for example, PAYLIB__01). This use of the underscore might not be valid in other high-level languages.

The first three categories contain the characters that are allowed in quoted and unquoted character strings, in comments, and in CL names, such as in names of commands, labels, keywords, variables, and OS/400 objects. Special characters in the last category can only be used in quoted character strings and comments; they cannot be used in unquoted strings. However, some have special syntactical uses when coded in the proper place in CL commands. These uses are given in the “Special character use” chart.

Special character use: The following special EBCDIC characters are used by the CL in various ways. They are most frequently used as delimiters (which are covered in “Command delimiters” on page 18) and as symbolic operators in expressions. For more information about symbolic operators, see:
- “Symbolic operators” on page 24
- the discussion about the types of expressions in “Operators in expressions” on page 33

Special characters can be used only in these special ways or inside quoted character strings or comments. The special characters, as shown in the following table, have assigned meanings when coded in control language commands:

Delimiters

Name	Symbol	Meanings
Apostrophe	′ ′	Single apostrophe delimiters indicate the beginning and end of a quoted character string (a constant).

Name	**Symbol**	**Meanings**
Begin and end comment	/* */	Indicates the beginning and end of a comment.
Blank	b[1]	Basic delimiter for separating parts of a command (label, command name, and its parameters), and for separating values inside lists.
Colon	:	Ending delimiter for command labels. Separates parts of time values.[3]
Comma	,	In many countries, used as decimal point in numeric values. Separates parts of date values.[2]
Left and right parentheses	( )	Grouping delimiter for lists and parameter values, and for evaluating the order of expressions.
Period	.	Decimal point. Used to separate the name and extension of a document and folder name and to separate the parts of date values.[2]
Quote	" "	Start of a quoted object name.
Slash	/	Connects parts of qualified names or path names.
Slashes	//	Identifying characters used in positions 1 and 2 of BCHJOB, ENDBCHJOB, and DATA commands in the job stream. Also, used as a default delimiter on inline data files.

Notes:

1 Because this character does not resolve in the online version of this book, b is used to represent a blank space only when the character cannot be clearly explained in another manner.

2 Valid only when the job date separator value specifies the same character.

3 Valid only when the job time separator value specifies the same character.

Symbolic operators: The following characters are used as symbolic operators in CL commands.

Name	**Symbol**	**Meanings**
And	&	Symbolic logical operator for AND.
Asterisk	*	Multiplication operator. Indicates a generic name when it is the last character in the name. Indicates OS/400(R) reserved values (predefined parameter values and expression operators) when it is the first character in a string.
Concatenation	|>, |<, and ||[3]	Character string operator (indicates both values are to be joined). See "Expressions" on page 16 for more information on the differences between these concatenation operators.
Equal	=	Symbolic *equal* relational operator.
Greater than	>	Symbolic *greater than* relational operator.
Less than	<	Symbolic *less than* relational operator.
Minus (hyphen)	-	Subtraction operator, command continuation operator, and negative signed value indicator. Separates parts of date values.[1]
Not	¬[2]	Symbolic NOT relational operator.
Or	|[3]	Symbolic logical operator for OR.
Plus	+	Addition operator, command continuation character, and positive signed value indicator.
Slash	/	Division operator. Separates parts of date values.[1] Used as the separator between parts of a qualified name.

Notes:

1 Valid only when the job date separator value specifies the same character.

2 In some character sets, including the multinational character set, the character ^ replaces the ¬ character. Either ^ or *NOT can be used as the logical NOT operator in those character sets.

3 In some character sets, including the multinational character set, the character ! replaces the | character. Either ! or *OR can be used as the logical OR operator, and either || or *CAT can be used as the concatenation operator in those character sets.

Note: The symbolic operators can also be used in combinations as listed in the chart under "Operators in expressions" on page 33.

Symbolic operators: Other uses

Symbolic operators can also be used in the following ways:

Name	Symbol	Meanings
Ampersand	&	Identifies a CL variable name when it is the first character in the string.
Percent	%	Identifies a built-in system function when it is the first character in the string.
Question mark	?	Specifies a prompt request when it precedes a command name or keyword name.

Predefined values: Predefined values are IBM^R-defined fixed values that have predefined uses in the CL and are considered to be reserved in the OS/400$^{(R)}$ system. Predefined values have an asterisk (*) as the first character in the value followed by a word or abbreviation, such as *ALL or *PGM. The purpose of the * in predefined values is to prevent possible conflicts with user-specified values, such as object names. Each predefined value has a specific use in one or more command parameters, and each is described in detail in the command description.

Some predefined values are used as operators in expressions, such as *EQ and *AND. The predefined value *N is used to specify a null value and can be used to represent any optional parameter. A *null value* (*N) indicates a parameter position for which no value is being specified; it allows other parameters that follow it to be entered in positional form. To specify the characters *N as a character value (not as a null), the string must be enclosed in apostrophes ('*N') to be passed. Also, when the value *N appears in a CL program variable at the time it is run, it is always treated as a null value.

Naming within commands

The type of name you specify in the OS/400$^{(R)}$ control language determines the characters you can use to specify a name. For certain types of names, there are restrictions on the use of certain characters to represent the name. Those types of names are *NAME, *SNAME, and *CNAME.

Note: For a description of how to specify these names when you use command definitions to create commands, see the PARM (parameter) and ELEM (element) statements in "Command definition statements" on page 38.

The characters allowed for the *NAME, *SNAME, and *CNAME names and the rules you use to specify them are shown in "Allowable Characters for *NAME, *SNAME, and *CNAME" on page 26.

» See the following for more information about naming within commands:

- See "Folder and document names" on page 27 for information about how to specify names for folders and documents.
- See "Path names (*PNAME)" on page 49 for information about specifying names for objects in the integrated file system. «

Allowable Characters for *NAME, *SNAME, and *CNAME:

Type of Name	First Character	Other Characters	Min. Length	Max. Length
*NAME[1]	`A-Z, $, #, @`	`A-Z, 0-9, $, #, @, _, .`	1	256
*SNAME[1]	`A-Z, $, #, @`	`A-Z, 0-9, $, #, @, _`	1	256
*CNAME[1]	`A-Z, $, #, @`	`A-Z, 0-9, $, #, @`		
Quoted name[2]	`"`[3]	Any except blank, *, ?, ', ", X'00'-X'3F', and X'FF'	3	256

Notes:

1 The system converts lowercase letters to uppercase letters.
2 Double quotes can only be used for basic names (*NAME).
3 Both the first and last characters must be a double quote (").

***NAME (Basic Name):** Every basic name can begin with the characters A-Z, $, #, or @ and can be followed by up to nine characters. The remaining characters can include the same characters as the first but can also include numbers 0-9, underscores (_), and periods (.). Lowercase letters are changed to uppercase letters by the system. Basic names used in IBM(R)-supplied commands can be no longer than 10 characters. However, in your own commands, you can define parameters of type *NAME (specified on the TYPE parameter of the PARM or ELEM statements) with up to 256 characters.

Examples of basic names are shown below:

```
A987@.442#    ONE_NAME    LIB_0690    $LIBX
```

Names can be entered in quoted or unquoted form. If you use the quoted form, the following rules and considerations also apply:

***NAME (Basic Name in Quoted Form):** Every quoted name must begin and end with a quotation mark ("). The middle characters of a quoted name can contain any character except , *, ?, ', ", hex 00 through 3F, or hex FF, and is delimited by a slash. Quoted names allow you to use graphic characters in the name. The quoted form of basic names used in IBM-supplied commands can be no longer than 8 characters (not including the double quotes). In your own commands, you can define parameters of type *NAME in quoted form with up to 254 characters (not including the double quotes).

Note: Only basic names can be used in quoted form.

Examples of quoted names are shown below:

```
"A"        "AA%abc"        "ABC%%abc"
```

When you use quoted names, you should be aware of certain restrictions:

- Code points in a name might not be addressable from all keyboards.

- Characters in a quoted name might not be valid in a high-level language.
- The System/38(TM) environment supports only simple (*SNAME) names. If other characters are used, the objects cannot be accessed as System/38 environment objects.
- Names that are longer than eight characters cannot be accessed by the System/36TM environment unless control language overrides are used.
- A Structured Query Language (SQL) name that contains a period must be specified in an SQL statement in quotation marks.

If a name enclosed in quotation marks is a valid unquoted basic name, the quotation marks are removed. Thus, ”ABC“ is equivalent to ABC. Because the quotation marks are removed, they are not included in the length of the name. ”ABCDEFGHIJ“ is, therefore, a valid name on IBM* commands even though it is longer than 10 characters.

***SNAME (Simple Name):** Simple names are the same as *unquoted* basic names but with one exception: periods (.) cannot be used. Simple names are used for CL variables, labels, and keywords to simplify the syntax of the control language.

Some examples of simple names are as follows:

```
NEWCMD    LIB_2
```

***CNAME (Communications Name):** Communications names are the same as *unquoted* basic names with the following exceptions:

1. Periods (.) and underscores (_) cannot be used.
2. For IBM commands, *CNAME is limited to 8 characters.

An example of a communications name is shown below:

```
APPN3@@
```

Note: Because restricted character sets are sometimes used by other IBM systems, use caution when choosing names that use the special characters #, $, and @. These special characters might not be on the remote system’s keyboard. The names that may be exchanged with the remote systems include the following:

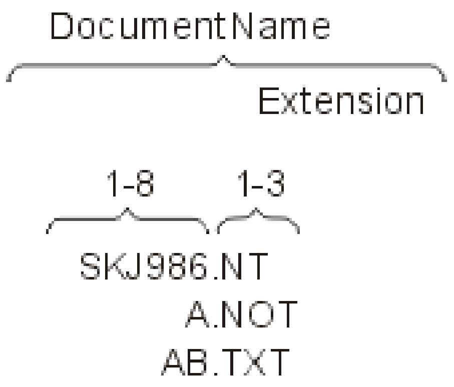
- Network IDS
- Location names
- Mode names
- Class-of-service names
- Control point names

Folder and document names: Folder names should describe the contents of a folder. The names must be unique and should be easy to type, as well as descriptive to a user. To find a particular folder on the system and change a document stored in it, you must either supply the folder name or select it from a list of names.

Document names should describe the contents of the document. You should give careful consideration to the names you use to help you find the document later. The names must be unique in the folder and should be easy to type, as well as descriptive.

The name you use for a folder or a document must follow these rules:

- The name must be unique within a folder.
- A document or folder name can be 1 to 12 characters long, including an optional extension. If no extension is included, a document or folder name can have a maximum of eight characters. If an extension is included, the extension must start with a period and can have up to three additional characters. An extension in the document name allows you to identify the document by using specific information that can help you do a selective listing of documents on your system.



- A document or folder name can include any single-byte EBCDIC character *except for the following* special characters that the system uses for other purposes:

Character	Special uses
Asterisk (*)	Multiplication operator, indicates generic names, and indicates OS/400(R) reserved values
Slash (/)	Division operator, delimiter within system values, and separates parts of qualified object names
Question Mark (?)	Initiates requests for system help

- When a folder is stored in another folder, both folder names are used, separated by a slash (/). That combination of names is called a **folder path**. For example, if a folder named FOLDR2 is stored in FOLDR1, the path for FOLDR2 is FOLDR1/FOLDR2. FOLDR1 is the **first-level folder**. FOLDR2 is the **next-level folder**. The name of a single folder can be 1 to 12 characters long, including an optional extension. A folder path can contain a maximum of 63 characters.

  Folder names should not begin with Q because the system-supplied folder names begin with Q. The following are examples of permitted folder names and folder paths:

```
@LETTERS
FOLDER.PAY
PAYROLL/FOLDER.PAY
#TAX1/FOLD8.TAX/$1988/PAYROLL/FOLDER.PAY
```

Notes:

1. In CL commands, folder path names must be enclosed in apostrophes to prevent the system from processing them as qualified (library/object) names. If an apostrophe is to be part of the name, it must be specified as two consecutive apostrophes.
2. A number of CL commands act on either documents or folders, and some act on both. The abbreviation DLO (document library object) is used when referring to either a document or folder.
3. In CL commands, folder and document names must be enclosed in apostrophes if they contain characters that are CL delimiters.
4. The system does not recognize graphic characters; it recognizes only code points and uses the following assumptions:
   - All folder and document names are encoded using single-byte EBCDIC code pages. Since code points hex 41 through FE represent graphic characters in those code pages, they are the only code points that can be used in folder and document names.
   - Code points hex 5C, 61, and 6F represent the asterisk (*), slash (/), and question mark (?) respectively, and cannot be used in folder and document names.
   - The code points for lowercase letters in English (hex 81 through 89, 91 through 99, and A2 through A9) are converted to the code points for uppercase letters (C1 through C9, D1 through D9, and E2 through E9, respectively).

More information on code pages that are supported by OS/400 is in Chapter 5 of the *Local Device Configuration* book and in the description of the CRTDEVDSP (Create Device Description (Display)) command.

In addition to the folder and document names previously described, folders and documents are internally classified in the system by their system object names. These are 10-character names derived from date/time stamps, and, while they are generally not known to the user, they may be specified on some CL commands by specifying *SYSOBJNAM for the folder or document name and by specifying the system object name in a separate parameter.

For more information, see "Naming within commands" on page 25.

Expressions in CL commands

A character string expression can be used for any parameter, element, or qualifier defined with EXPR(*YES) in the command definition object. Any expression can be used as a single parameter in the Change Variable (CHGVAR) and If (IF) commands. An expression in its simple form is a single constant, a variable, or a built-in function. An expression usually contains two operands and an operator that indicates how the expression is to be evaluated. Two or more expressions can be combined to make a complex expression.

The following types of expressions are supported in CL programs:

- "Arithmetic expressions"`(&VAR + 15)`
- "Character string expressions" on page 30`(SIX || TEEN)`
- "Relational expressions" on page 32 `(&VAR > 15)`
- "Logical expressions" on page 32`(&VAR & &TEST)`

A *complex* expression contains multiple operands, operators that indicate what operation is performed on the operands, and parentheses to group them. Only one operator is allowed between operands, except for the + and - signs when they immediately precede a decimal value (as a signed value), and the *NOT operator when it is used in a logical expression.

No complex expression can have more than five nested levels of parentheses, including the outermost (required) level.

Arithmetic and character string expressions can be used together in a complex expression if they are used with relational and logical operators; for example: `(A=B&(1+2)=3)`. A pair of arithmetic expressions or a pair of character string expressions can be compared within a relational expression. Also, relational expressions can be used within a logical expression

Arithmetic expressions: » The operands in an arithmetic expression must be decimal constants or decimal or integer CL variables. « An arithmetic operator (only in symbolic form) must be between the operands. The results of all arithmetic expressions are decimal values, which may be stored in a CL variable.

Note: The division operator (/) must be preceded by a blank if the operand that precedes it is a variable name. (For example, `&A /2`, *not* `&A/2`.) All other arithmetic operators may optionally be preceded or followed by a blank.

» Arithmetic operands can be signed or unsigned; that is, each operand (whether it is a numeric constant or a decimal or integer CL variable) can be immediately preceded by a plus (+) or minus (-) sign, but a sign is not required. « When used as a sign, no blanks can occur between the + or - and its value. For example, a decimal constant of 23.7 can be expressed as +23.7 or -23.7 (signed) or as 23.7 (unsigned).

The following are examples of arithmetic expressions:

(&A + 1)	(&A + &B -15)
(&A - &F)	(&A+&B-15)
(&A + (-&B))	

If the last nonblank character on a line is a + or -, it is treated as a continuation character and not as an arithmetic operator.

Character string expressions: The operands in a character string expression must be quoted or unquoted character strings, character variables, or the substring (%SUBSTRING or %SST) built-in function. The value associated with each variable or built-in function must be a character string. The result of concatenation is a character string.

There are three operators that can be used in character string expressions. These operators concatenate (or join) two character strings, but each has a slightly different function. They are:

- *CAT (concatenation, symbol ||)

  ***CAT Operator**

  The *CAT operator concatenates two character strings. For example:
  `ABC *CAT DEF` becomes `ABCDEF`

  Blanks are included in the concatenation. For example:
  `'ABC '`
  `*CAT 'DEF '` becomes `'ABC DEF '`
- *BCAT (concatenation with blank insertion, symbol |>)

  ***BCAT Operator**

  The *BCAT operator truncates all trailing blanks in the first character string; one blank is inserted, then the two character strings are concatenated. Leading blanks on the second operand are not truncated. For example:
  `ABC *BCAT DEF` becomes `ABC DEF`
  `'ABC ' *BCAT DEF` becomes `'ABC DEF'`
- *TCAT (concatenation with trailing blank truncation, symbol |<)

  ***TCAT Operator**

  The *TCAT operator truncates all trailing blanks in the first character string, then the two character strings are concatenated. All leading blanks on the second operand are not truncated. For example:
  `ABC *TCAT DEF` becomes `ABCDEF`
  `'ABC ' *TCAT DEF` becomes `'ABCDEF'`
  `ABC *TCAT ' DEF'` becomes `'ABC DEF'`
  `'ABC '*TCAT ' DEF'` becomes `'ABC DEF'`

All blanks that surround the concatenation operator are ignored, but at least one blank must be on each side of the reserved value operator (*CAT, *BCAT, or *TCAT). If multiple blanks are wanted in the expression, a quoted character string (a character string enclosed within apostrophes) must be used.

See the following examples for more information about character string expressions.
- "Example: Character string expressions"
- "Example: Using character strings and variables" on page 31

Example: Character string expressions: The following are examples of string expressions. Assume the following variables:

Variable	**Value**
&AA	`'GOOD '`
&BB	`'REPLACEMENT'`
&CC	`'ALSO GOOD'`

Variable	Value
&DD	'METHOD'

Expression	Result
(&AA || &BB)	GOOD REPLACEMENT
(&AA||&BB)	GOOD REPLACEMENT
(&AA *CAT &BB)	GOOD REPLACEMENT
(&CC |> &DD)	ALSO GOOD METHOD
(&CC *BCAT &DD)	ALSO GOOD METHOD
(A *CAT MOUSE)	AMOUSE
('A ' *CAT MOUSE)	A MOUSE
(FAST *CAT MOUSE)	FASTMOUSE
('FAST ' *BCAT MOUSE)	FAST MOUSE
('FAST ' *TCAT MOUSE)	FASTMOUSE
('AB' *CAT 'CD')	ABCD
('AB' *BCAT 'CD')	AB CD
('AB' *TCAT 'CD')	ABCD
(%SST(&AA 1 5) *CAT (%SST(&BB 3 5))	GOOD PLACE
(%SST(&CC 1 9) *BCAT (%SST(&BB 3 5))	ALSO GOOD PLACE
(&AA *CAT ' TIME')	GOOD TIME
(&CC *BCAT TIME)	ALSO GOOD TIME

For another example using character strings and variables:
- "Example: Using character strings and variables"

Example: Using character strings and variables: See Code disclaimer information (page 1) for information pertaining to code examples.

The following example shows how several character variables and character strings can be concatenated to produce a message for a work station operator. The example assumes that the variables &DAYS and &CUSNUM were declared as character variables, not decimal variables.

```
DCL     VAR(&MSG)TYPE(*CHAR)     LEN(100)
         *
         *
CHGVAR      &MSG   ('Customer' *BCAT &CUSNAMD  +
              *BCAT'Account Number' *BCAT +
              &CUSNUM *BCAT 'is overdue by' +
              *BCAT &DAYS *BCAT 'days.')
```

After the appropriate variables have been substituted, the resulting message might be:

```
Customer ABC COMPANY Account Number 12345
is overdue by 4 days.
```

If the variables &DAYS and &CUSNUM had been declared as decimal variables, two other CHGVAR commands would have to change the decimal variables to character variables before the concatenation could be performed. If, for example, two character variables named &DAYSALPH and &CUSNUMALPH were also declared in the program, the CHGVAR commands would be:

```
CHGVAR  &DAYSALPH  &DAYS

CHGVAR  &CUSNUMALPH  &CUSNUM
```

Then instead of &DAYS and &CUSNUM, the new variables &DAYSALPH and &CUSNUMALPH would be specified in the CHGVAR command used to concatenate all the variables and character strings for &MSG.

For another example using character string expressions:
- "Example: Character string expressions" on page 30

Relational expressions: The operands in a relational expression can be arithmetic or character string expressions; they can also be logical constants and logical variables. Only two operands can be used with each relational operator. The data type (arithmetic, character string, or logical) must be the same for the pair of operands. The result of a relational expression is a logical value '0' or '1'.

Refer to the table under "Operators in expressions" on page 33 for the meanings of the relational operators, which can be specified by symbols (=, >, <, >=, <=, ¬=, ¬>, ¬<) or their reserved values (*EQ, *GT, *LT, *GE, *LE, *NE, *NG, *NL).

If an operation involves character fields of unequal length, the shorter field is extended by blanks added to the right.

Arithmetic fields are compared algebraically; character fields are compared according to the EBCDIC collating sequence.

When logical fields are compared, a logical one ('1') is greater than logical zero ('0'). Symbolically, this is ('1' > '0').

The following are examples of relational expressions:

```
(&X *GT 25)
(&X > 25)
(&X>25)

(&NAME *EQ GSD)
(&NAME *EQ &GSD)
(&NAME *EQ 'GSD')
(&BLANK *EQ ' ')
```

Logical expressions: The operands in a logical expression consist of relational expressions, logical variables, or constants, separated by logical operators. Two or more of these types of operands can be used in combinations, making up two or more expressions within expressions, up to the maximum of five nested levels of parentheses. The result of a logical expression is a '0' or '1' that can be used as part of another expression or saved in logical variables.

The logical operators used to specify the relationship between the operands are *AND and *OR (as reserved values), and & and | (as symbols). The AND operator indicates that both operands (on either side of the operator) have to be a certain value to produce a particular result. The OR operator indicates that one or the other of its operands can determine the result.

The logical operator *NOT (or ¬) is used to negate logical variables or logical constants. All *NOT operators are evaluated before the *AND or *OR operators are evaluated. All operands that follow *NOT operators are evaluated before the logical relationship between the operands is evaluated.

The following are examples of logical expressions:

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C < 1) & (&TIME *GT 1430))
((&C<1)&(&TIME>1430))

(&A *OR *NOT &B)
(&TOWN *EQ CHICAGO *AND &ZIP *EQ 60605)
```

Two examples of logical expressions used in the IF command are:

```
IF &A CALL PROG1
IF (&A *OR &B) CALL PROG1
```

Operators in expressions: Operators are used in expressions to indicate an action to be performed on the operands in the expression or the relationship between the operands. There are four kinds of operators, one for each of the four types of expressions:

- Arithmetic operators `(+, -, *, /)`
- Character operator `(||, |>, |<)`
- Logical operators `(&, |, <img src="notsym.gif" alt="not symbol">)`
- Relational operators `(=, >, <, >=, <=, <img src="notsym.gif" alt="not symbol">=, <img src="notsym.gif" alt="not symbol">>, <img src="notsym.gif" alt="not symbol"><)`

Each operator must be between the operands of the expression in which it is used; for example, `(&A + 4)`. Operators can be specified as a predefined value (for example, *EQ) or as a symbol (for example, =).

- All predefined value operators must have a blank on each side of the operator:

  `(&VAR *EQ 7)`
- Except for the division operator (/), symbolic operators need no blanks on either side. For example, either `(&VAR=7)` or `(&VAR = 7)` is valid.

  Where the division operator *follows* a variable name, the division operator must be preceded by a blank. For example, `(&VAR / 5)` or `(&VAR /5)` is valid; `(&VAR/5)` is not valid.

The following character combinations are the predefined values and symbols that represent the four kinds of operators; they should not be used in unquoted strings for any other purpose.

Predefined values and symbols representing the four kinds of operators

Predefined Value	Predefined Symbol	Meaning	Type
	+	Addition	Arithmetic operator
	-	Subtraction	Arithmetic operator
	*	Multiplication	Arithmetic operator
	/	Division	Arithmetic operator
*CAT	||[1]	Concatenation	Character string operator
*BCAT	|>[1]	Blank insertion with concatenation	Character string operator
*TCAT	|<[1]	Blank truncation with concatenation	Character string operator
*AND	&	AND	Logical operator
*OR	|[1]	OR	Logical operator
*NOT	¬[2]	NOT	Logical operator
*EQ	=	Equal	Relational operator
*GT	>	Greater than	Relational operator
*LT	<	Less than	Relational operator

Predefined Value	Predefined Symbol	Meaning	Type
*GE	>=	Greater than or equal	Relational operator
*LE	<=	Less than or equal	Relational operator
*NE	¬ =[2]	Not equal	Relational operator
*NG	¬ >[2]	Not greater than	Relational operator
*NL	¬ <[2]	Not less than	Relational operator

Notes:

1 In some national character sets and in the multinational character set, the character | (hexadecimal 4F) is replaced by the character ! (exclamation point). Either ! or *OR can be used as the OR operator and either || or *CAT, !> or *BCAT, and !< or *TCAT can be used for concatenation in those character sets.

2 In some national character sets and in the multinational character set, the character ¬ (hexadecimal 5F) is replaced by the character *. Either * or *NOT can be used as the NOT operator in those character sets.

Priority of operators when evaluating expressions: When multiple operators occur in an expression, the expression is evaluated in a specific order depending upon the operators in the expression. Parentheses can be used to change the order of expression evaluation. The following table shows the priority of all the operators used in expressions, including signed decimal values.

Priority	Operators
1	signed (+ and -) decimal values, *NOT, ¬
2	* , /
3	+, - (when used between two operands)
4	*CAT, ||, *BCAT, |>, *TCAT, |<
5	*GT, *LT, *EQ, *GE, *LE, *NE, *NG, *NL, >, <, =, >=, <=, ¬ =, ¬ >, ¬ <
6	*AND, &
7	*OR, |

A priority of 1 is the highest priority (signed values are evaluated first); a priority of 7 is the lowest priority (OR relationships are evaluated last). When operators with different priority levels appear in an expression, operations are performed according to priorities.

When operators of the *same* priority appear in an expression, operations are performed from left to right within the expression. Parentheses can always be used to control the order in which operations are performed. The value of a parenthetical expression is determined from the innermost level to the outermost level, following the priorities stated above within matching sets of parentheses.

Built-in functions for CL: CL provides the following built-in functions:

- "%BINARY"
- "%SUBSTRING" on page 36
- "%SWITCH" on page 37

%BINARY: The binary (%BINARY) built-in function operates on a character string that is contained in a CL character variable.

%BINARY or %BIN can be used in expressions and as either operand (receiver) of the Change Variable (CHGVAR) command. See the CHGVAR command description for more information.

Notes:

- The binary built-in function can also be used on command parameters that are defined as numeric (*DEC, *INT2 and *INT4) and EXPR(*YES) has been specified.
- » Using CL integer variables is an alternative to using the binary built-in function. See *CL Programming* for information. «

Sample %BINARY syntax

The syntax of the binary built-in function is:

```
>>-&BINARY(-character-variable-name-+-------------------------+--)->
                                    '-starting-position-+-2-+-'
                                                        '-4-'

>--------------------------------------------------------------><
```

The binary built-in function treats the contents of the specified CL character variable, starting at the position specified for a length of 2 or 4 characters, as a signed binary integer.

When the binary built-in function is used with the VAR parameter on the CHGVAR command, the decimal number or arithmetic expression in the VALUE parameter is converted to a 2-byte or 4-byte signed binary integer. A decimal fraction is not included.

If the starting position and length are not specified, then a starting position of 1 and the length of the character variable specified is used. The length of the character variable must be declared as 2 or 4.

The following are examples of how the %BINARY built-in function can be used.

See Code disclaimer information (page 1) for information pertaining to code examples.

Example 1: Converting binary to decimal

```
    DCL  VAR(&N)  TYPE(*DEC)  LEN(3 0)
    DCL  VAR(&B2)  TYPE(*CHAR)  LEN(2)  VALUE(X'0012')
    CHGVAR  &N  %BINARY(&B2)
```

The content of character variable &B2 is treated as a 2-byte signed binary number and is converted to its decimal equivalent of 18. It is then assigned to the decimal variable &N.

Example 2: Converting decimal to binary

```
    CHGVAR  %BIN(&B2)  &N
```

The number contained in the decimal variable &N is converted to a 2-byte signed binary number and is placed in the first and second bytes of the character variable &B2.

Example 3: Used within an arithmetic expression

```
    CHGVAR  &N  VALUE(%BIN(&B2) + 4)
```

The contents of character variable &B2 is treated as a 2-byte signed binary integer and is converted to its decimal equivalent of 18. The decimal number 4 is then added and the sum, 22, is assigned to the decimal variable &N.

Example 4: Converting decimal to binary with truncation

```
    CHGVAR  %BINARY(&B2)  VALUE(122.567)
```

The number 122.567 is truncated to the whole number 122 and is then converted to a 2-byte signed binary integer and assigned to the character variable &B2. Character variable &B2 will then contain the hexadecimal equivalent of X'007A'.

Example 5: Converting a negative number

```
    DCL  VAR(&B4)  TYPE(*CHAR)  LEN(4)
      CHGVAR  %BIN(&B4)  VALUE(-45)
```

The value -45 is converted to a 4-byte signed binary integer assigned to the character variable &B4. Character variable &B4 then contains the hexadecimal equivalent of X'FFFFFFD3'.

Example 6: Used on the IF command

```
    IF  COND(%BIN(&B4) *EQ 0)  THEN(GOTO ENDIT)
```

The content of character variable &B4 is treated as a 4-byte signed binary integer and is compared to the decimal number 0. If they are equal, the command following the label ENDIT is run. If they are not equal, the command following the IF command is run.

Example 7: Varying length character string to CPP

```
    PGM  PARM(&P ... )

      DCL  VAR(&P)  TYPE(*CHAR)  LEN(202)

      DCL  VAR(&L)  TYPE(*DEC)  LEN(5 0)
      DCL  VAR(&C)  TYPE(*CHAR)  LEN(200)
          *
          *
          *
      CHGVAR  &L  %BINARY(&P 1 2)
      CHGVAR  &C  %SST(&P 3 &L)
          *
          *
          *
    ENDPGM
```

This program is the command processing program CPP for a command with a first parameter defined with the attributes TYPE(*CHAR), LEN(200) and VARY(*YES). The first two bytes of character variable &P contain the length of the parameter as *INT2, a 2-byte signed binary integer. The character string specified on the command starts in position 3 of the variable &P. The maximum length of the character string is 200 characters.

The first CHGVAR command retrieves the length from the first two character positions of variable &P and treats the 2 bytes as a signed binary integer. The bytes are converted to the decimal equivalent of the signed binary integer, and are assigned to the decimal variable &L.

The second CHGVAR command retrieves the contents of the parameter by making variable &P a substring and assigning it to variable &C.

Example 8: Converting binary to integer

```
 DCL  VAR(&N)  TYPE(*INT)
    DCL  VAR(&B2) TYPE(*CHAR)  LEN(2)  VALUE(X'0012')
    CHGVAR  &N  %BINARY(&B2)
```

The content of character variable &B2 is treated as a 2-byte signed binary number and is converted to its integer equivalent of 18. It is then assigned to the integer variable &N.

%SUBSTRING: The substring built-in function operates on a character string that is contained in a CL character variable or in a local data area. %SUBSTRING or %SST can be used in expressions and as either operand (receiver) of the Change Variable (CHGVAR) command. For more information, see the description of the CHGVAR command. This built-in function can be coded as either %SUBSTRING or %SST.

Sample %SUBSTRING syntax

The syntax of the substring built-in function is:

```
>>-&SST(-+-*LDA--------------------+--starting-position--length---)->
         '-character-variable-name-'

>--------------------------------------------------------------><
```

This built-in function produces a substring from the contents of the specified CL character variable or local data area. The substring begins at the specified starting position in the value and continues for the length specified. For example:

```
%SST(&TEST  5  3)
```

In this example, a portion of the variable &TEST is referenced. That position (or substring) is 3 characters long and begins with the fifth character position. If &TEST contains `ABCDEFGHIJ`, the resulting substring will be `EFG`.

CL variables can also be used to specify the starting position and the length values in the function. For example:

```
CHGVAR   &X  %SST(*LDA  &B  &C)
```

The value of the character variable named &X is to be replaced by the value in the job's local data area, starting at the position obtained from variable &B and continuing for the length specified by the value in &C.

```
RTVJOBA   SWS(&JOBSWS)

CHGVAR   VAR(&CURSW4)  VALUE(%SST(&JOBSWS  4  1))
```

In this example, the Retrieve Job Attributes (RTVJOBA) command is used to retrieve the current value of the job's eight job switches. The CHGVAR command is then used to extract the current value of the fourth job switch only and store it in the variable &CURSW4. If the value of the eight job switches retrieved in &JOBSWS is `10010000`, the second 1 would be stored in &CURSW4.

%SWITCH: The built-in function %SWITCH tests one or more of the eight job switches in the current job and returns a logical value of 1 or 0. If every job switch tested by %SWITCH has the value indicated, the result is a 1 (true); if any switch tested does not have the value indicated, the result is a 0 (false).

The 8-character mask is used to indicate which job switches are tested, and what value each switch is tested for. Each position in the mask corresponds with one of the eight job switches in a job. Position 1 corresponds with job switch 1, position 2 with switch 2, and so on. Each position in the mask can be specified as one of three values: 0, 1, or X.

0	The corresponding job switch is tested for a 0 (off).
1	The corresponding job switch is tested for a 1 (on).
X	The corresponding job switch is not tested. The value in the switch does not affect the result of %SWITCH.

Sample %SWITCH syntax

The syntax of the switch built-in function is:

```
>>-&SWITCH(--)-------------------------------------------------><
```

If `%SWITCH(0X111XX0)` is specified, job switches 1 and 8 are tested for 0s, switches 3, 4, and 5 are tested for 1s, and switches 2, 6, and 7 are not tested. If each job switch contains the value (1 or 0 only) shown in the mask, the result of %SWITCH is true (1).

Function %SWITCH can be used in the Change Variable (CHGVAR) and If (IF) commands. On the CHGVAR command, it can be used in place of a logical variable in the VALUE parameter. On the IF command, it can be used in the COND parameter as the logical expression to be tested.

The following two examples show how the same mask can be used to control a branch in a program (the IF command), or to set the value of a variable (the CHGVAR command).

```
IF   COND(%SWITCH(0X111XX0))  THEN(GOTO C)
```

```
CHGVAR   VAR(&A)  VALUE(%SWITCH(0X111XX0))
```

If job switches 1, 3, 4, 5, and 8 respectively contain 0, 1, 1, 1, and 0 respectively when `%SWITCH(0X111XX0)` is specified in the IF command, the result is true and the program branches to the command having label C. If one or more of the switches tested do not have the values indicated in the mask, the result is false and the branch does not occur. If the same mask is used in the CHGVAR command and the result is true, the variable &A is set to a ’1’; if the result is false, &A is set to a ’0’. Note that &A must be declared as a logical variable.

Monitoring messages: Monitorable messages are those *ESCAPE, *STATUS, and *NOTIFY messages that can be issued by each CL command that can be used in a program. You can use this information to determine which messages you want to monitor for in your program.

Using the Monitor Message (MONMSG) CL command, you can monitor for one or more messages and then specify (on the MONMSG command) what action you want taken when any of those messages are issued by the commands(s) being monitored.

Documentation for each command contains a section that lists the monitorable messages issued by the command. You can display documentation for a command using the CL command finder.

If you have a V4R2 or later system, you can refer to the online help for an individual command to obtain its monitorable message information. To view the online command help on an iSeries$^{(TM)}$ computer, type the command name on a command line and press F1 (Help). The error message information follows the brief description of the purpose for the command.

Refer to Chapter 8 of the *CL Programming* book for information concerning messages that are sent to the QSYSMSG message queue.

Command definition statements

The OS/400$^{(R)}$ operating system lets users define a command that calls a program to perform some function. Users can define commands by using command definition statements. The defined command can include the following:

- Keyword notation parameters for passing data to programs
- Default values for omitted parameters
- Parameter validity checking so the program performing the function will have correct input
- Prompt text for prompting interactive users

For about command definition statements, see the following:

- “Creating user-defined commands” on page 39
- “CMD (command) statement” on page 39
- “PARM (Parameter) statement” on page 39
- “ELEM (element) statement” on page 39
- “QUAL (Qualifier) statement” on page 40
- “DEP (dependent) statement” on page 39

- “PMTCTL (Prompt Control) statement”

Creating user-defined commands

Users can define a command by entering command definition statements into a source file and running a Create Command (CRTCMD) command using the source file as input. The **command definition statement** of each command contains one or more **command statements**.

One and only one “CMD (command) statement” must be somewhere in the source file. A “PARM (Parameter) statement” must be provided for each parameter that appears on the command being created. » Complex parameters can be defined by using “ELEM (element) statement” and “QUAL (Qualifier) statement” on page 40 to define the parts of the parameter. « If any special keyword relationships need checking, the “DEP (dependent) statement” is used to define the relationships. The DEP statement can refer only to parameters that have been previously defined. These statements can appear in any order. “PMTCTL (Prompt Control) statement” can be used to selectively prompt command parameters.

See the *CL Programming* book for a complete description of how to use these statements to define a command.

Only one command can be defined in each source member in the source file. The CRTCMD command is run to create the command definition object from the command definition statements in one source file member. Other users can then be authorized to use the new command by the Grant Object Authority (GRTOBJAUT) command or the Edit Object Authority (EDTOBJAUT) command.

CMD (command) statement

The Command (CMD) statement specifies the prompt text for the command being created. The CMD statement can be anywhere in the source file referred to by the Create Command (CRTCMD) command; one and only one CMD statement must be used in the source file, even if no prompt text is specified for the created command.

See the Command Definition (CMD) command for more information.

DEP (dependent) statement

The Dependent (DEP) statement defines a required relationship between parameters and parameter values that must be checked. This relationship can refer to either the specific value of a parameter or parameters or to the required presence of parameters.

See the Dependent Definition (DEP) command for more information.

ELEM (element) statement

Element (ELEM) statements are used to define the elements of a mixed list (list elements) parameter on a command. A list parameter is a parameter that accepts multiple values that are passed together as consecutive values pointed to by a single keyword.

See the Element Definition (ELEM) command for more information.

PARM (Parameter) statement

The Parameter (PARM) statement defines a parameter of a command being created. A parameter is the means by which a value is passed to the command processing program (CPP). One PARM statement must be used for each parameter that appears in the command being defined.

See the Parameter Definition (PARM) command for more information.

PMTCTL (Prompt Control) statement

The Prompt Control (PMTCTL) statement specifies a condition that is tested to determine if prompting is done for the parameters whose PARM statement refers to the PMTCTL statement.

See the Prompt Control Definition (PMTCTL) command for additional information.

QUAL (Qualifier) statement

The Qualifier (QUAL) statement describes one part of a qualified name. If a name is the allowed value of a parameter or list element defined in a PARM or ELEM statement, it can be changed to a qualified name by using a QUAL statement for each qualifier used to qualify the name.

See the Qualifier Definition (QUAL) command for more information.

Parameter values used for testing and debugging

This section contains expanded descriptions of the program variable, basing pointer, subscript, and qualified-name parameter values. These values can be specified on the Add Breakpoint (ADDBKP), Add Trace (ADDTRC), Change High-Level Language Pointer (CHGHLLPTR), Change Program Variable (CHGPGMVAR), and Display Program Variable (DSPPGMVAR) commands.

For expanded descriptions, see the following:
- "Program-variable description"
- "Basing-pointer description"
- "Subscript description" on page 41
- "Qualified-name description" on page 41

Program-variable description

```
Program Variable

                                .-------------------.
                                V            (1)   |
>>-qualified-name---subscript------,subscript-------+---------><
```

Note:
1. A maximum of 14 repetitions

The program variable must be enclosed in apostrophes if it contains special characters. Up to 132 characters can be specified for a program variable name. This includes any subscripts, embedded blanks, parentheses, and commas. It does not include the enclosing apostrophes when special characters are used. Some examples are:

```
COUNTA
'VAR1(2,3)'
'A.VAR1(I,3,A.J,1)'
'VAR1 OF A(I,3,J OF A)'
'&LIBNAME'
```

Basing-pointer description

```
Basing Pointer

                                .-------------------.
                                V            (1)   |
>>-qualified-name---subscript------,subscript-------+---------><
```

Note:
1. A maximum of 14 repetitions

The basing pointer must be enclosed in apostrophes if it contains special characters. Up to 132 characters can be specified for a basing pointer name. This includes any subscripts, embedded blanks, parentheses, and commas. It does not include the enclosing apostrophes when special characters are used. Some examples are:

```
        PTRVAR1
        'ABC.PGMPTR(5,B.I)'
```

If more than one basing pointer is specified for a variable, the list of basing pointers must be enclosed in parentheses. When multiple basing pointers are specified, they must be listed in order, from the first basing pointer to the last, when used to locate the variable. In the example below, the PTR_1 basing pointer is the first basing pointer used to locate the variable; it either must have a declared basing pointer, or it must not be a based variable. The address contained in the PTR_1 pointer is used to locate the A.PTR_2 pointer (which must be declared as a based pointer variable). The contents of the A.PTR_2 pointer are used to locate the PTR_3 pointer array (which must also be declared based), and the contents of the specified element in the last pointer array are used to locate the variable. An example is:

```
        ('PTR_1' 'A.PTR_2' 'PTR_3(1,B.J)')
```

Subscript description

```
Subscript

   .-integer-number--.
>>-+-qualified-name--+-----------------------------------------><
   '-*---------------'
```

An integer number contains from 1 through 15 digits with an optional leading sign (either plus or minus). A decimal point is not allowed in an integer-number subscript. If a decimal point is specified, the subscript value is not interpreted as the correct numeric value by the system, and an error message is returned.

An asterisk (*) can be used to request a single-dimensional cross-section display of an array program variable. An asterisk can only be specified for a subscript on the primary variable (not on a basing pointer) for the PGMVAR keyword on the Add Break Point (ADDBKP), Add Trace (ADDTRC), and Display Program Variable (DSPPGMVAR) commands. In addition, if the variable has multiple dimensions, only one of the subscript values can be an asterisk. An example of a request to display an array cross-section is:

```
        DSPPGMVAR  PGMVAR('X1(*,5,4)')
```

This display shows the values of all elements of the array that have the second subscript equal to five, and the third subscript equal to four.

Qualified-name description

```
Qualified-Name

>>--+-/ODV-number-----------------------------+---------------><
    |             .------------------------.  |
    |             V                (1)    |  |
    '-variable-+----+-OF-+--variable-------+-+-'
               |    '-IN-'                 |
               |  .----------------.       |
               |  V        (1)    |       |
               '---variable-------+----------'
```

Note:

1. A maximum of 19 repetitions

Some high-level languages may allow you to declare more than one variable with the same fully qualified name (although you generally are not able to refer to these variables in the high-level language program after they are declared). If you attempt to refer to such a variable using an OS/400(R) test facility command, the system selects one of the variables and uses it for the operation. No error is reported when a duplicate fully qualified name is selected.

Rules for qualified name description

- An ODV number is a slash (/) followed by 1 to 4 hexadecimal digits (0 through 9, and A through F).
- The variable-name must be the name of a variable in the program. This name must be specified the same way in the high-level language. Some high-level languages introduce qualifier variable names in addition to the ones you specified in the source for your program. See the appropriate high-level language manual for more information about variable names.
- Blanks must separate the variable-names from the special words OF and IN.
- When a period is used to form a qualified name, no blanks can appear between it and the variable-names.
  - The ordering of the variable names must follow these rules:
    - For qualified names that contain no embedded period, the variable names are assumed to be specified from the lowest to the highest levels in the structure.
    - For qualified names that contain one or more embedded periods, the variable names are assumed to be specified from the highest to the lowest levels in the structure.
  - When an ODV number is not used for the qualified name, enough qualifier variable names must be specified so that a single unique variable can be identified in the program. Whether the qualified name is a simple name (only one variable name specified) or a name with multiple qualifier variable names, the variable in the program is uniquely identified if either of the following conditions is true (these conditions may require you to specify more qualifier variable names for OS/400 test facility commands than you need to specify in the high-level language program to uniquely select a program variable):
    - A variable is uniquely identified if there is one and only one variable in the program with a set of qualifier variables matching the qualified variable name specified.
    - A variable is uniquely identified in the program if it has exactly the same set of qualifier variables as the qualifier variable names specified. When the complete set of qualifiers is specified, the variable name is said to be *fully qualified*. A variable that is a *fully qualified* match for the qualified-name is selected even if there are other variables with names that match the qualified name but have additional qualifier variables which were not specified.

OS/400 objects

Operating System/400(R) (OS/400(R)) objects provide the means through which all data processing information is stored and processed by OS/400. An **OS/400 object** is a named unit that exists (occupies space) in storage, and on which operations are performed by the operating system.

CL commands perform operations on the Operating System/400 objects. Several types of OS/400 objects are created and used in the control language.

OS/400 objects have the following in common:
- Objects have a set of descriptive attributes that are defined when the object is created.
- Objects have to be used by the system to perform a specific function must be specified in the CL command that performs that function.
- Objects have a set of attributes that describe it and give the specific values assigned for those attributes.
- Generally, objects are independent of all other objects. However, some objects must be created before other objects can be created; for example, a logical file cannot be created if the physical file it must be based on does not exist.
- Objects must be created before other operations that use the object are performed. Descriptions of the create commands (those commands that begin with the letters CRT) give more information about the object types that they create.
- Every OS/400 object that is used by the control language has a name. The object name specified in a CL command identifies which object is used by the operating system to perform the function of the command.

- Objects have either a simple, qualified, or generic name.

For additional information about OS/400 objects, see the following:
- "Library objects"
- "External object types"
- "Simple and qualified object names" on page 46
- "Generic object names" on page 46
- "Object naming rules" on page 48
- Predefined Values and Default Library Locations for OS/400 Object Types table (page 43)

Library objects

Many objects are grouped in special objects called libraries. "External object types" provides information about various object types and their default libraries.

Some objects, which use the integrated file system, are located in directories and can be found by using path names or object name patterns instead of searching libraries. You can also use these directories to locate objects. For more information about integrated file system commands, see the Integrated File Systems topic.

External object types

The following table lists the predefined values for all the OS/400(R) external object types.

When an object is created and a library qualifier can be specified but is not, the object is stored in the user's current job library, as shown in the Default User Library column. The user profile for each user specifies the user's current library. The current library will be QGPL if it is not specified otherwise. The other objects, identified by N/A in the Default User Library column, cannot be stored in user-provided libraries. » With the exception of the DMPSYSOBJ (Dump System Object) command, you cannot specify the object type in the format shown in the Hexadecimal Format column with commands. «

Predefined Values and Default Library Locations for External OS/400 Object Types

Value	Object Type	Hexadecimal Format	Default User Library
*ALRTBL	Alert table	0E09	*CURLIB
*AUTL	Authorization list	1B01	QSYS
*BLKSF	Block special file	1E05	N/A
*BNDDIR	Binding directory	1937	*CURLIB
*CFGL	Configuration list	1918	QSYS
*CHRSF	Character special file	1E06	N/A
*CHTFMT	Chart format	190D	*CURLIB
*CLD	C/400R locale description	190B	*CURLIB
*CLS	Class	1904	*CURLIB
*CMD	Command	1905	*CURLIB
*CNNL	Connection list	1701	QSYS
*COSD	Class-of-service description	1401	QSYS
*CRG	Cluster resource group	192C	QUSRSYS
*CRQD	Change request description	0E0F	*CURLIB
*CSI	Communications side information	1935	*CURLIB

Value	Object Type	Hexadecimal Format	Default User Library
*CSPMAP	Cross-system product map	1922	*CURLIB
*CSPTBL	Cross-system product table	1923	*CURLIB
*CTLD	Controller description	1201	QSYS
*DDIR	Distributed file directory	1F02	N/A
*DEVD	Device description	1001	QSYS
*DIR	Directory	0C01	N/A
*DOC	Document	190E	QDOC
*DSTMF	Distributed stream file	1F01	N/A
*DTAARA	Data area	190A	*CURLIB
*DTADCT	Data dictionary	1920	library with same name as data dictionary
*DTAQ	Data queue	0A01	*CURLIB
*EDTD	Edit description	1908	QSYS
*EXITRG	Exit registration	1913	QUSRSYS
*FCT	Forms control table	0E04	*CURLIB
*FIFO	First-in-first-out special file	1E07	N/A
*FILE	File	1901	*CURLIB
*FLR	Folder	1912	QDOC
*FNTRSC	Font resources	1926	*CURLIB
*FNTTBL	Font mapping table	192B	*CURLIB
*FORMDF	Form definition	1928	*CURLIB
*FTR	Filter	0E0B	*CURLIB
*GSS	Graphics symbol set	190C	*CURLIB
*IGCDCT	Double-byte character set (DBCS) conversion dictionary	0E06	*CURLIB
*IGCSRT	Double-byte character set (DBCS) sort table	191A	*CURLIB
*IGCTBL	Double-byte character set (DBCS) font table	1910	N/A
*IMGCLG	Image Catalog	192E	QUSRSYS
*IPXD	Internetwork packet exchange description	191E	QSYS
*JOBD	Job description	1903	*CURLIB
*JOBQ	Job queue	0E01	*CURLIB
*JOBSCD	Job schedule	0E0C	*CURLIB
*JRN	Journal	0901	*CURLIB
*JRNRCV	Journal receiver	0701	*CURLIB
*LIB	Library	0401	QSYS
*LIND	Line description	1101	QSYS
*LOCALE	Locale	1921	*CURLIB
*MEDDFN	Media definition	191C	*CURLIB
*MENU	Menu description	1916	*CURLIB

Value	Object Type	Hexadecimal Format	Default User Library
*MGTCOL	Management collection	192D	NA, or QPFRDATA if library specified using QYPSCSCA API
*MODD	Mode description	1501	QSYS
*MODULE	Compiler unit	0301	*CURLIB
*MSGF	Message file	0E03	*CURLIB
*MSGQ	Message queue	1902	*CURLIB
*M36	AS/400(R) Advanced 36(R) machine	1E04	*CURLIB
*M36CFG	AS/400 Advanced 36 machine configuration	1924	*CURLIB
*NODGRP	Node group	192A	*CURLIB
*NODL	Node list	0E0E	*CURLIB
*NTBD	NetBIOS description	1914	QSYS
*NWID	Network interface description	1601	QSYS
*NWSD	Network server description	1D01	QSYS
*OUTQ	Output queue	0E02	*CURLIB
*OVL	Overlay	1929	*CURLIB
*PAGDFN	Page definition	1936	*CURLIB
*PAGSEG	Page segment	1927	*CURLIB
*PDFMAP	Portable Document Format map	0E11	*CURLIB
*PDG	Print Descriptor Group	1930	*CURLIB
*PGM	Program	0201	*CURLIB
*PNLGRP	Panel group definition	1915	*CURLIB
*PRDAVL	Product availability	1933	QSYS
*PRDDFN	Product definition	191B	QSYS
*PRDLOD	Product load	191D	QSYS
*PSFCFG	Print Services Facility(TM) configuration	1925	*CURLIB
*QMFORM	Query management form	1932	*CURLIB
*QMQRY	Query management query	1931	*CURLIB
*QRYDFN	Query definition	1911	QGPL
*RCT	Reference code translate table	0E08	QGPL
*SBSD	Subsystem description	1909	*CURLIB
*SCHIDX	Search index	0E07	QGPL
*SOCKET	Local socket	1E03	N/A
*SPADCT	Spelling aid dictionary	1C01	QGPL
*SQLPKG	Structured Query Language package	0202	*CURLIB
*SQLUDT	User-defined SQL type	191F	*CURLIB
*SRVPGM	Service program	0203	*CURLIB
*SSND	Session description	0E05	QGPL
*STMF	Bytestream file	1E01	N/A
*SVRSTG	Server storage space	1917	QUSRSYS

Value	Object Type	Hexadecimal Format	Default User Library
*SYMLNK	Symbolic link	1E02	N/A
*S36	System/36TM machine description	1919	QGPL
*TBL	Table	1906	*CURLIB
» *TIMZON	Time zone description	192F	QSYS «
*USRIDX	User index	0E0A	*CURLIB
*USRPRF	User profile	0801	QSYS
*USRQ	User queue	0A02	*CURLIB
*USRSPC	User space	1934	*CURLIB
*VLDL	Validation list	0E10	*CURLIB
*WSCST	Workstation user customization object	1938	*CURLIB

Simple and qualified object names

The name of a specific object that is located in a library can be specified as a simple name or as a qualified name. A *simple object name* is the name of the object only. A *qualified object name* is the name of the library where the object is stored followed by the name of the object. In a qualified object name, the library name is connected to the object name by a slash (/).

Either the simple name or the qualified name of an object can be specified if the object exists in one of the libraries named in the job's library list; the library qualifier is optional in this case. A qualified name *must* be specified if the named object is not in a library named in the library list.

Note: Although a job name also has a qualified form, it is not a qualified object name because a job is not an OS/400(R) object. A job name is qualified by a user name and a job number, not by a library name. For more information about the JOB parameter, refer to "JOB parameter" on page 62.

Example

The following table shows how simple and qualified object names are formed.

Name Type	Name Syntax	Example
Simple object name	object-name	OBJA
Qualified object name	library-name/object-name	LIB1/OBJB

Generic object names

A *generic object name* may refer to more than one object. That is, a generic name contains one or more characters that are the first group of characters in the names of several objects; the system then searches for all the objects that have those characters at the beginning of their names and are in the libraries named in the library list. A generic name is identified by an asterisk (*) as the last character in the name.

A quoted generic name consists of a generic name enclosed in quotation marks. Unlike quoted names, if there are no special characters between the quotation marks, the quotation marks are not removed. The generic name "ABC*" would cause the system to search for objects whose name begins with *"ABC.*

A generic name can also be qualified by a library name. If the generic name is qualified, the system will search only the specified library for objects whose names begin with that generic name.

Note: A generic name also can be qualified by one or more directories if it is a path name. In a path name, letters can be specified before and after the asterisk (*). For more information on path names, see the Integrated File System topic.

When you specify a generic name, the system performs the required function on all objects whose names begin with the specified series of characters. You must have the authority required to perform that function on every object the generic name identifies. If you do not have the required authority for an object, the function is not performed and a diagnostic message is issued for each instance that the attempted generic function failed. A completion message is issued for each object the generic function operates on successfully. You must view the online low-level messages to see the completion messages. Once the entire generic function is completed, a completion message is issued that states that all objects were operated on successfully. If one or more objects could not be successfully operated on an escape message is issued. If an override is in effect for a specific device file, the single object name specified on the override, rather than the generic name, is used to perform the operation.

You may not be able to use a generic name for delete, move, or rename commands if the library containing the objects is already locked. A search for generic object names requires a more restrictive lock on the library containing the objects than a search for full object names. The more restrictive lock is necessary to prevent another user from creating an object with the same name as the generic search string in the library while the delete, move, or rename command is running. You can circumvent this problem by using the full name of the objects instead of a generic name. Or you can end the job or subsystem that has a lock on the library.

Note: Use the WRKOBJLCK (Work with Object Locks) command to determine which jobs or subsystems have a lock on the library.

For some commands, a library qualifier can be specified with the generic name to limit the scope of the operation. For example, a Change Print File (CHGPRTF) command with `FILE(LIB1/PRT*)` performs the desired operation on printer files beginning with `PRT` in library `LIB1` only; printer files in other libraries are not affected.

The limitations associated with the various library qualifiers are as follows:
- *library-name:* The operation is performed on generic object names only in the specified library.
- *LIBL: The operation is performed on generic object names in the library list associated with the job that requested the generic operation.
- *CURLIB: The operation is performed on generic object names in the current library.
- » *ALL: The operation is performed on generic object names in all libraries for which you are authorized.«
- *USRLIBL: The operation is performed on generic object names in the user part of the library list for the job.
- *ALLUSR: The operation is performed on all nonsystem libraries (libraries that do not start with the letter Q), with some exceptions. » See Generic library names in the APIs topic for the complete list of libraries included for *ALLUSR. «

  Note: A different library name, of the form QUSRVxRxMx, can be created by the user for each release that IBM^R supports. VxRxMx is the version, release, and modification level of the library.

Example: Generic object name

Name Type	Name Syntax	Example
Simple generic name	generic-name*	OBJ*
Qualified generic name	library-name/generic-name*	LIB1/OBJ*
Quoted generic name	"generic-name*"	"ABC*"

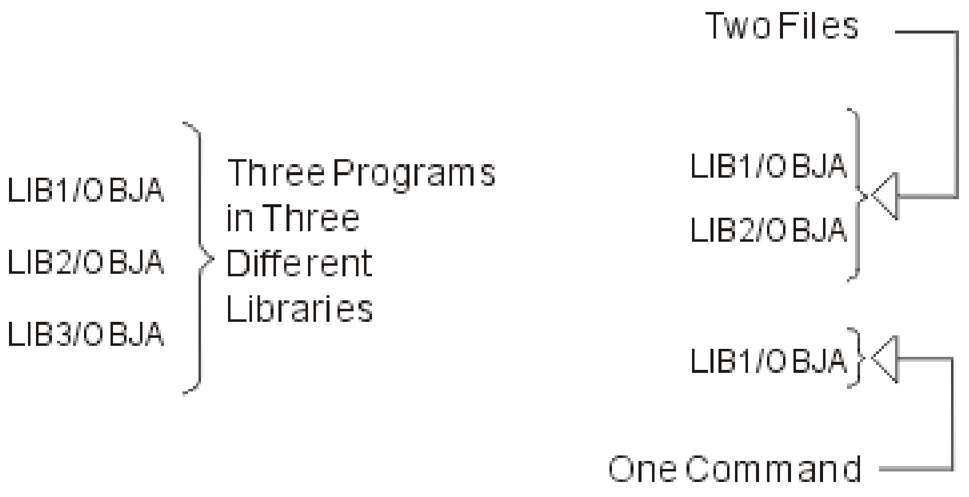
Object naming rules

» The standard rules for all names are described in “Naming within commands” on page 25. In addition, the following rules are used to name all OS/400$^{(R)}$ objects used in control language commands. The parameter summary table for each CL command shows whether a simple object name, a qualified name, or a generic name can be specified. «

- *Naming a Single Object*: In the name of a single object, each part (the simple name and the library qualifier name) can have a maximum of 10 characters. For more information on specifying objects, see “Generic object names” on page 46.
- *Naming a User-Created Object*: To distinguish a user-created object from an IBM$^{(R)}$-supplied object, you should not begin user-created object names with Q because the names of all IBM-supplied objects (except commands) begin with Q. Although you can use as many as 10 characters in CL object names, you may need to use fewer characters to be consistent with the naming rules of the particular high-level language that you are also using. Also, the high-level language might not allow underscores in the naming rules. For example, RPG limits file names to eight characters and does not allow underscores.
- *Naming a Generic Object*: In a generic name, a maximum of nine alphanumeric characters can be used, not including the asterisk (*) that must immediately follow the last character. For more information on using generic names, see “Generic object names” on page 46.

  INV and INV* are valid values where a generic name is accepted. When the name INV is specified, only the object INV is referenced. When the quoted generic name INV* is specified, objects that begin with INV are referred to, such as INV, INVOICE, INVENTORY, and INVENPGM1. When the quoted generic name “INV*” is specified, objects that begin with “INV” are referred to, such as “INV%1” and “INV>.”
- *Object Library Qualifier Limitations*: No library qualifier can be specified with the object name if the object being created is a library, user profile, line description, controller description, device description, mode description, class-of-service description, or configuration list. A library name can never be qualified because a library cannot be placed in a library. The other object types (*USRPRF, *LIND, *CTLD, *DEVD, *MODD, *COSD, and *CFGL) appear to be types that exist only in the QSYS library. When only the name of an object of these object types is accepted, a library qualifier cannot be specified with the object name. On the Display Object Description (DSPOBJD) command, where any object name is accepted, QSYS can be specified.
- *Library List Qualifiers*: The predefined value *LIBL (and others, such as *CURLIB and *ALLUSR) can be used in place of a library name in most commands. *LIBL indicates that the libraries named in the job’s library list are used to find the object named in the second part of the qualified name.
- *Duplicate Object Names*: Duplicate names are not allowed for objects of the same type in the same library.

  Two objects with the same name cannot be stored in the same library unless their object types are different. Two objects named OBJA can be stored in the library LIBx only if, for example, one of the objects is a program and the other is a file. The following combinations of names and object types could all exist on the system at the same time.



If more than one library contains an object with the same name (and both libraries are in the same library list) and a library qualifier is not specified with the object name, the first object found by that name is used. Therefore, when you have multiple objects with the same name, you should specify the library name with the object name or ensure that the appropriate library occurs first in the library list. For example, if you are testing and debugging and choose not to qualify the names, ensure that your test library precedes your production library in the library list.

Default libraries

In a qualified object name, the library name is usually optional. If an optional library qualifier is not specified, the default given in the command's description is used (usually either *CURLIB or *LIBL). If the named object is being created, the current library is the default; when the object is created, it is placed either in the current library or in the QGPL (the general purpose library) if no current library is defined. For objects that already exist, *LIBL is the default for most commands, and the job's library list is used to find the named object. The system searches all of the libraries currently in the library list until it finds the object name specified.

For additional information about object naming, see the following:
- "Path names (*PNAME)"
- "Generic names (*GENERIC)" on page 51
- "Additional rules for unique names" on page 51

Related information:
- "Naming within commands" on page 25
- "Folder and document names" on page 27

Path names (*PNAME)

A path name is a character string that can be used to locate objects in the integrated file system. The string can consist of one or more elements, each separated by a slash (/) or back slash (\). Each element is typically a directory or equivalent, except for the last element, which can be a directory, another object such as a file, or a generic presentation of an object or objects to be located.

The / and \ characters and nulls cannot be used in the individual components of the path name because the / and \ characters are used as separators. The name may or may not be changed to uppercase, depending on whether the file system containing the object is case-sensitive and whether the object is being created or searched for. If the parameter is defined as CASE(*MONO) (the default), any values that are not enclosed in single quotes will be changed to uppercase by the command analyzer.

A / or \ character at the beginning of a path name means that the path begins at the top most directory, the "root" (/) directory. If the path name does not begin with a / or \ character, the path is assumed to begin at the current directory of the user entering the command.

The path name must be represented in the CCSID currently in effect for the job. If the CCSID of the job is 65535, the path name must be represented in the default CCSID of the job. Hard-coded path names in programs are encoded in CCSID 37. Therefore, the path name should be converted to the job CCSID before being passed to the command. The maximum length of the path name character string on the CL commands is 5000 characters.

When operating on objects in the QSYS.LIB file system, the component names must be of the form name.object-type; for example:

```
'/QSYS.LIB/PAY.LIB/TAX.FILE'
```

Path names must be enclosed in apostrophe (′) marks when entered on a command line if they contain special characters. These marks are optional when path names are entered on displays. If the path name includes any quoted strings or special characters; however, the enclosing ′′ marks must be included. The following are rules for using special characters:

- A tilde (~) character followed by a slash or backslash at the beginning of a path name means that the path begins at the home directory of the user entering the command.
- A tilde (~) character followed by a user name and then a slash or backslash at the beginning of a path name means that the path begins at the home directory of the user identified by the user name.
- In some commands, an asterisk (*) or a question mark (?) can be used in the last component of a path name to search for patterns of names. The * tells the system to search for names that have any number of characters in the position of the * character. The ? tells the system to search for names that have a single character in the position of the ? character.
- To avoid confusion with OS/400(R) special values, path names cannot start with a single asterisk (*) character. To perform a pattern match at the beginning of a path name, use two asterisks (**).

  Note: This only applies to relative path names where there are no other characters before the asterisk.
- The path name must be enclosed in apostrophes (′) or quotation marks (") if any of the following characters are used in a component name:
  - Asterisk (*)
  - Question mark (?)
  - Apostrophe (′)
  - Quotation mark (")
  - Tilde (~), if used as the first character in the first component name of the path name (if used in any other position, the tilde is interpreted as just another character)

    This practice is not recommended because the meaning of the character in a command string could be confused and it is more likely that the command string will be entered incorrectly.
- Do not use a colon (:) in path names. It has a special meaning within the system.
- The processing support for commands and associated user displays does not recognize code points below hexadecimal 40 as characters that can be used in command strings or on displays. If these code points are used, they must be entered as a hexadecimal representation, such as the following:

  ```
  crtdir dir(X'02')
  ```

  Therefore, use of code points below hexadecimal 40 in path names is not recommended. This restriction applies only to commands and associated displays, not to APIs. » In addition, a value of hexadecimal 00 is not allowed in path names. «

For further information on device names, see Specifying the device name in the Systems management, Backup and recovery topic.

For further information on path names, see the Integrated File System topic in the Information Center.

Generic names (*GENERIC)

A generic name is one that contains at least one initial character that is common to a group of objects, followed by an asterisk. (The asterisk identifies the series of common characters as a generic name; otherwise, the system interprets the series of characters as the name of a specific object).

For more information on *GENERIC names, see the “Generic object names” on page 46 section in this topic.

Additional rules for unique names

Additional rules involving special characters (as an extra character) that apply to the following types of names are:

- A *command label* must be immediately followed by a colon (:). Blanks can follow the colon, but none can precede it. A command label name cannot be a quoted name.
- A CL *variable name* must be preceded by an ampersand (&) to indicate that it is a CL variable used in a CL program.
- A *built-in function name* must be preceded by a percent sign (%) to indicate that it is an IBM$^{(R)}$-supplied built-in function that can be used in an expression. A built-in function name cannot be a quoted name.

These special characters are not part of the name; each is an additional character attached to a name (making a maximum of 11 characters) indicating to the system what the name identifies.

The names of OS/400$^{(R)}$ objects, CL program variables, system values, and built-in functions can be specified in the parameters of individual commands. Instead of specifying a constant value, a CL variable name can be used on most parameters in CL programs to specify a value that may change during the running of programs. It is the contents of the variable that identify the objects and variables that are used when the command is run.

Commonly used parameters: Expanded descriptions

This section contains the expanded descriptions of some of the parameters commonly used in the CL commands. The parameters included here meet one or both of these criteria:

- There is extensive information about how they are used.
- They are used in many of the CL commands (such as the AUT parameter), and the parameter description in the individual command description gives only the essential information.

The expanded descriptions of the applicable command parameters have been placed here for several reasons:

- to reduce the amount of material needed in the individual commands. Normally programmers familiar with a parameter’s main function do not need the details.
- to provide the supplemental information that is useful to programmers in some instances.

The format for this information is designed for easy reference and includes a general description of each parameter that explains its function, states the rules for its use, and provides other helpful information. The values that can be specified for each parameter are also listed. Each value is followed by an explanation of what it means and (possibly) in which commands it is used. Not all of the values appear in every command. Refer to the individual command descriptions for the specific use of the value in that command parameter.

See the following:

AUT parameter

You use the authority (AUT) parameter in create, grant, and revoke commands. It specifies the authority granted to all users of an object. It also specifies an authorization list that is used to secure the object. Four object types allow the AUT parameter to contain an authorization list: LIB, PGM, DTADCT, and FILE. Public authority is an OS/400(R) object attribute that controls the base set of rights to that object for all users having access to the system. These rights can be extended or reduced for specific users. If you specify an authorization list, the public authority in the authorization list is the public authority for the object. The owner of an object has all authority to the object at its creation.

If the object is created as a private object or with the limited authority given to all users, the owner can grant more or less authority to specific users by specifically naming them and stating their authority in the Grant Object Authority (GRTOBJAUT) command. The owner also can withdraw specific authority from specific users, or from all users (publicly authorized and/or specifically authorized) by using the Revoke Object Authority (RVKOBJAUT) command or the Edit Object Authority (EDTOBJAUT) command.

The *iSeries Security Reference* book has a complete description of security provisions and applicable rights of use by object type.

Values allowed

***LIBCRTAUT:** The public authority for the object is taken from the value on the CRTAUT parameter of the target library (the library that is to contain the object). The public authority is determined when the object is created. If the CRTAUT value for the library changes after the object is created, the new value does not affect any existing objects.

***USE:** You can perform basic operations on the object, such as running a program or reading a file. The user cannot change the object. *USE authority provides object operational authority, read authority, and execute authority.

***CHANGE:** You can perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. You can change and perform basic functions on the object. Change authority provides object operational authority and all data authority.

***ALL:**You can perform all operations except those limited to the owner or controlled by authorization list management authority. Your can control the object's existence, specify the security for the object, change the object, and perform basic functions on the object. You also can change ownership of the object.

***EXCLUDE:** You cannot access the object.

***EXECUTE:** You can run a program or procedure or search a library or directory.

authorization-list-name: Specify the name of the authorization list whose authority is used.

CLS parameter

The class (CLS) parameter identifies the attributes that define the run time environment of a job. The following attributes are defined in each class:

- Run priority: A number that specifies the priority level assigned to all jobs running that use the class. The priority level is used to determine which job, of all the jobs competing for system resources, is run next. The value can be 1 through 99, where 1 is the highest priority (all jobs having a 1 priority are run first).
- Time slice: The maximum amount of processor time that the system allows the job to run when it is allowed to begin. The time slice indicates the amount of time needed for the job to accomplish a meaningful amount of work (the time used by the system for reading auxiliary storage is not charged against the time slice). When the time slice ends, the job waits while other queued jobs of the same or higher priority are allowed to run (up to the time specified in their time slices); then the job is given another time slice.
- Purge value: Indicates whether the job step is eligible to be moved from main storage to auxiliary storage while the job is waiting for some resource before it can continue, or when its time slice is used up and equal or higher priority jobs are waiting.
- Default wait time: The default amount of time that the system waits for the completion of an instruction that performs a wait. This wait time applies to times when an instruction is waiting for a system action, not to the time an instruction is waiting for a response from a user. Normally, this would be the amount of time you are willing to wait for the system before ending the request. If the wait time is exceeded, an error message is passed to the job. This default wait time applies only when a wait time is not specified in the CL command that causes the wait.

  The wait time used for allocating file resources is specified in the file description and can be overridden by an override command. It specifies that the wait time specified in the class object is used. If file resources are not available when the file is opened, the system waits for them until the wait time ends.

  Note: The class attributes apply to each routing step of a job. Most jobs have only one routing step, but if the job is rerouted (because of something like the Remote Job or Transfer Job command) the class attributes will be reset.
- Maximum CPU time: The maximum amount of processor time (the sum of the time slices if they differ, or time slice period multiplied by number of time slices if they are equal) allowed for a job's routing step to complete processing. If the job's routing step is not completed in this amount of time, it is ended, and a message is written to the job log.
- Maximum temporary storage: The maximum amount of temporary storage that can be used by a job's routing step. This temporary storage is used for the programs that run in the job, for the system objects used to support the job, and for temporary objects created by the job.

The system is shipped with a set of classes that define the attributes for several job processing environments. Other classes can be created by the Create Class (CRTCLS) command; any class can be displayed or deleted by the respective Display Class (DSPCLS) and Delete Class (DLTCLS) commands.

Values allowed

qualified-class-name: Specify the name of the class, optionally qualified by the name of the library in which the class is stored. If the class name is not qualified and the CLS parameter is in the CRTCLS command, the class object is stored in *CURLIB; otherwise, the library list (*LIBL) is used to find the class name.

The following classes (by name) are supplied with the system:

QGPL/QBATCH
For use by batch jobs

QSYS/QCTL
For use by the controlling subsystem

QGPL/QINTER
For use by interactive jobs

QGPL/QPGMR
For use by the programming subsystem

QGPL/QSPL
For use by the spooling subsystem printer writer

QGPL/QSPL2
For general spooling use in the base system pool

COUNTRY parameter

The Country parameter specifies the country or region code part of the X.400^R O/R name. An ISO 3166 Alpha-2 code or an ITU-T country or region code can be specified. (The ITU-T country or region code is the data country or region or geographical area code published in the “International Numbering Plan for Public Data Networks,” Recommendation X.121 (09/92), by the ITU-T (formerly CCITT). The following table contains a list of the possible country or region codes that can be specified.

Values allowed

***NONE:** No country or region code is specified.

country-code: Specify an ISO 3166 Alpha-2 code or a CCITT (also known as ITU-2) country or region code from the following table.

ISO X.400 Country or Region Codes

Country or Region	**ISO 3166 Alpha-2 Code**	**ITU-T[1] Country or Region Code**
Afghanistan	AF	412
Albania	AL	276
Algeria	DZ	603
American Samoa	AS	544
Andorra	AD	
Angola	AO	631
Anguilla	AI	
Antarctica	AQ	
Antigua and Barbuda	AG	344
Argentina	AR	722
Armenia	AM	283

Country or Region	ISO 3166 Alpha-2 Code	ITU-T[1] Country or Region Code
Aruba	AW	362
Australia	AU	505
Austria	AT	232
Azerbaijan	AZ	400
Bahamas	BS	364
Bahrain	BH	426
Bangladesh	BD	470
Barbados	BB	342
Belarus	BY	257
Belgium	BE	206
Belize	BZ	702
Benin	BJ	616
Bermuda	BM	350
Bhutan	BT	
Bolivia	BO	736
Bosnia and Herzegovina	BA	
Botswana	BW	652
Bouvet Island	BV	
Brazil	BR	724
British Indian Ocean Terr.	IO	
Brunei Darussalam	BN	528
Bulgaria	BG	284
Burkina Faso	BF	613
Burundi	BI	642
Cambodia	KH	456
Cameroon	CM	624
Canada	CA	302, 303
Cape Verde	CV	625
Cayman Islands	KY	346
Central African Republic	CF	623
Chad	TD	622
Chile	CL	730
China	CN	460
Christmas Island	CX	
Cocos (Keeling) Islands	CC	
Colombia	CO	732
Comoros	KM	654
Congo	CG	629
Cook Islands	CK	548
Costa Rica	CR	712
Cote d'Ivoire	CI	612
Croatia	HR	
Cuba	CU	368
Cyprus	CY	280
Czech Republic	CZ	230
Denmark	DK	238
Djibouti	DJ	638
Dominica	DM	366
Dominican Republic	DO	370
East Timor	TP	
Ecuador	EC	740
Egypt	EG	602
El Salvador	SV	706
Equatorial Guinea	GQ	627

Country or Region	ISO 3166 Alpha-2 Code	ITU-T[1] Country or Region Code
Eritrea	ER	
Estonia	EE	248
Ethiopia	ET	636
Falkland Islands (Malvinas)	FK	
Faroe Islands	FO	288
Fiji	FJ	542
Finland	FI	244
France	FR	208, 209
France, Metropolitan	FX	
French Antilles		340
French Guiana	GF	742
French Polynesia	PF	547
French Southern Terr.	TF	
Gabon	GA	628
Gambia	GM	607
Georgia	GE	282
Germany	DE	262 - 265
Ghana	GH	620
Gibralter	GI	266
Greece	GR	202
Greenland	GL	290
Grenada	GD	352
Guadeloupe	GP	
Guam	GU	535
Guatemala	GT	704
Guinea	GN	611
Guinea-Bissau	GW	632
Guyana	GY	738
Haiti	HT	372
Heard and Mc Donald Islands	HM	
Honduras	HN	708
China (Hong Kong S.A.R.)	HK	453, 454
Hungary	HU	216
Iceland	IS	274
India	IN	404
Indonesia	ID	510
Iran	IR	432
Iraq	IQ	418
Ireland	IE	272
Israel	IL	425
Italy	IT	222
Jamaica	JM	338
Japan	JP	440 - 443
Jordan	JO	416
Kazakhstan	KZ	401
Kenya	KE	639
Kiribati	KI	545
Korea, Democratic People's Republic	KP	467
Korea, Republic of	KR	450, 480, 481
Kuwait	KW	419
Kyrgyzstan	KG	437
Lao People's Democratic Rep.	LA	457
Latvia	LV	247

Country or Region	ISO 3166 Alpha-2 Code	ITU-T[1] Country or Region Code
Lebanon	LB	415
Lesotho	LS	651
Liberia	LR	618
Libyan Arab Jamahiriya	LY	606
Liechtenstein	LI	
Lithuania	LT	246
Luxembourg	LU	270
China (Macau S.A.R.)	MO	455
Macedonia [2]	MK [2]	
Madagascar	MG	646
Malawi	MW	650
Malaysia	MY	502
Maldives	MV	472
Mali	ML	610
Malta	MT	278
Marshall Islands	MH	
Martinique	MQ	
Mauritania	MR	609
Mauritius	MU	617
Mayotte	YT	
Mexico	MX	334
Micronesia	FM	550
Moldova, Republic of	MD	259
Monaco	MC	212
Mongolia	MN	428
Montenegro [2]	ME [2]	
Montserrat	MS	354
Morocco	MA	604
Mozambique	MZ	643
Myanmar	MM	414
Namibia	NA	649
Nauru	NR	536
Nepal	NP	429
Netherlands	NL	204, 205
Netherlands Antilles	AN	362
New Caledonia	NC	546
New Zealand	NZ	530
Nicaragua	NI	710
Niger	NE	614
Nigeria	NG	621
Niue	NU	
Norfolk Island	NF	
Northern Mariana Islands	MP	534
Norway	NO	242
Oman	OM	422
Pakistan	PK	410
Palau	PW	
Panama	PA	714
Papua New Guinea	PG	537
Paraguay	PY	744
Peru	PE	716
Philippines	PH	515
Pitcairn	PN	
Poland	PL	260

Country or Region	ISO 3166 Alpha-2 Code	ITU-T[1] Country or Region Code
Portugal	PT	268
Puerto Rico	PR	330
Qatar	QA	427
Reunion	RE	647
Romania	RO	226
Russian Federation	RU	250, 251
Rwanda	RW	635
St. Helena	SH	
St. Kitts and Nevis	KN	356
St. Lucia	LC	358
St. Pierre and Miquelon	PM	308
St. Vincent and the Grenadines	VC	360
Samoa, Western	WS	549
San Marino	SM	292
Sao Tome and Principe	ST	626
Saudi Arabia	SA	420
Senegal	SN	608
Serbia [2]	SP [2]	
Seychelles	SC	633
Sierra Leone	SL	619
Singapore	SG	525
Slovakia	SK	
Slovenia	SI	
Solomon Islands	SB	540
Somalia	SO	637
South Africa	ZA	655
South Georgia and the S.S.I	GS	
Spain	ES	214
Sri Lanka	LK	413
Sudan	SD	634
Suriname	SR	746
Svalbard and Jan Mayen Is.	SJ	
Swaziland	SZ	653
Sweden	SE	240
Switzerland	CH	228
Syrian Arab Republic	SY	417
Taiwan	TW	466
Tajikistan	TJ	436
Tanzania, United Republic of	TZ	640
Thailand	TH	520
Togo	TG	615
Tokelau	TK	
Tonga	TO	539
Trinidad and Tobago	TT	374
Tunisia	TN	605
Turkey	TR	286
Turkmenistan	TM	438
Turks and Caicos Islands	TC	376
Tuvalu	TV	
Uganda	UG	641
Ukraine	UA	255
United Arab Emirates	AE	424, 430, 431
United Kingdom	GB	234, 235, 236, 237
United States	US	310 - 316

Country or Region	ISO 3166 Alpha-2 Code	ITU-T[1] Country or Region Code
United States Minor Outlying Is.	UM	
Uruguay	UY	748
Uzbekistan	UZ	434
Vanuatu	VU	541
Vatican City State (Holy See)	VA	225
Venezuela	VE	734
Viet Nam	VN	452
Virgin Is. (Brit.)	VG	348
Virgin Is. (U.S.)	VI	332
Wallis and Futuna Is.	WF	543
Western Sahara	EH	
Yemen	YE	421, 423
Yugoslavia, territories of the former	YU	220
Zaire	ZR	630
Zambia	ZM	645
Zimbabwe	ZW	648

Notes:

1 This International Telecommunication Union (ITU) committee was formerly known as CCITT.

2 At the time of publication, the ISO 3166 Alpha-2 Code for this country or region could not be confirmed. Before using this code, be sure to confirm with the latest ISO 3166 standard.

FILETYPE parameter

The FILETYPE parameter specifies whether the database file description describes data records or source records. Further, it specifies whether each member of a database file being created is to contain data records or source records (statements). For example, the file could contain RPG source statements for an RPG program or data description source (DDS) statements for another device or database file.

Note: If you are creating a source type *physical* database file and are not providing field-level descriptions of the file (through data description specifications (DDS)), you can use either the Create Physical File (CRTPF) command or the Create Source Physical File (CRTSRCPF) command. However, the CRTSRCPF command is usually more convenient and efficient, because it is designed to be used to create source physical files. If DDS is provided when you are creating a source type database file, you should use the CRTPF command or the Create Logical File (CRTLF) command, which both have the SRCFILE and SRCMBR parameters for specifying source input.

Records in a source file must have at least three fields: the first two are the source sequence number field and the date field; the third field contains the source statement. These three fields are automatically provided by the OS/400(R) when a source file is created for which no DDS is provided; additional source fields can be defined in DDS. The length of the sequence number field must be six zoned digits with two decimal places. The length of the date field must be six zoned digits with no decimal places.

The source sequence number and date fields are added to the source record when:

- Records are read into the system.
- Records are created by the Source Entry Utility (which is part of the licensed Application Development* Tools program).

The fields are added when an inline data file (specified as the standard source file format) is read from the device. The spooling reader places a sequence number in the source sequence number field and sets up a zeroed date field.

If those fields already exist in records read from the device, they are not changed. If the records in a database file are in source format and are being read as an inline data file in data format, the source sequence number and date fields are removed.

For more information about data and source files, see the Database Programming topic in the Information Center.

Values allowed

***DATA:** The file created contains or describes data records.

***SRC:** The file created contains or describes source records. If the file is keyed, the 6-digit source sequence number field must be used as the key field.

FRCRATIO parameter

The force write ratio (FRCRATIO) parameter specifies the maximum number of records that can be inserted, updated, or deleted before they are forced into auxiliary (permanent) storage. The force write ratio ensures that all inserted, updated, or deleted records are written into auxiliary storage at least as often as this parameter specifies. In the event of system failure, the only records likely to be lost would be those that were inserted, updated, or deleted since the last force write operation.

The force write ratio is applied to all records inserted, updated, or deleted in the file through the open data path (ODP) to which the force write ratio applies. If two programs are sharing the file, SHARE(*YES), the force write ratio is not applied separately to the set of records inserted, updated, or deleted by each program. It is applied to any combination of records (from both programs) that equals the specified force write ratio parameter value. For example, if a force write ratio of 5 was specified for the file, any combination of five records from the two programs (such as four from one program and one from the other) forces the records to be written to auxiliary storage. If two or more programs are using the file through separate ODPs, the insertions, updates, and deletions from each program are accumulated individually for each ODP.

Each database file can have a force write ratio assigned to it. Logical files, which can access data from more than one physical file, can specify a more restrictive force write ratio (a smaller number of records) than that specified for the based-on physical files. However, a logical file cannot specify a *less* restrictive force write ratio. If a logical file specifies a less restrictive force write ratio than that specified for any of the physical files, the most restrictive force write ratio from the physical files is used for the logical file. For example, if the force write ratios of three physical files are 2, 6, and 8, the force write ratio of a logical file based on these physical files cannot be greater than 2. If no force write ratio is specified for the logical file, 2 is assumed. Thus, each time a program inserts, updates, or deletes two records in the logical file (regardless of which physical files are affected), those records are forced into auxiliary storage.

The FRCRATIO number overrides the SEQONLY number specified. For example, if you specify:

```
OVRDBF ... SEQONLY(*YES 20) FRCRATIO(5)
```

The value of 20 is overridden and a buffer of five records is used. When FRCRATIO(1) is used, a buffer still exists, but it contains only a single record.

Access paths associated with the inserted, updated, and deleted records are written to auxiliary storage only when all the records covered by the access path have been written to auxiliary storage. If only one ODP exists for the file, the access path is forced to auxiliary storage whenever a forced write occurs. If two or more ODPs to the file exist, the access path is written to auxiliary storage whenever all the inserted, updated, and deleted records for all the ODPs have been forced.

Notes:

- These rules apply only when a force write ratio of 2 or higher is specified. When a force write ratio of 1 is specified, the access path is not written to auxiliary storage until all the ODPs have been closed.
- If the file is being recorded in a journal, FRCRATIO(*NONE) should be specified. » For more information, see Journal management and system performance in the Systems Management topic. «

Values allowed

***NONE:** There is no specified ratio; the system determines when the records are written to auxiliary storage.

number-of-records-before-force: Specify the number of updated, inserted, or deleted records that are processed before they are explicitly forced to auxiliary storage.

IGCFEAT parameter

The IGCFEAT parameter specifies which double-byte character set (DBCS) table is used, according to device and language. The following table indicates the corresponding IGCFEAT parameter and DBCS font table for the double-byte character set device being configured.

DBCS Features Configurable on the IGCFEAT Parameter

Language/Device	Type of Physical DBCS Work Station	Configure as Type-Model	Configure with DBCS Feature
Japanese Display Stations	5295-001 Display	5555-B01	((2424J4 55FE))
	5295-002 Display	5555-B01	((2424J4 68FE))
	InfoWindow^R 3477-J Display	5555-B01, C01	((2424J4 68FE))
	PS/55 with 5250PC	5555-B01	((2424J4 68FE))
	PS/55* with graphics 5250PC	5555-G01	((2424J4 68FE))
	PS/55* with graphics 5250PC	5555-G02	((2424J4 68FE))
	PS/55 with 5250PC/2	5555-E01	((2424J0 (1)))
	3270-type Display	3279-0	((2424J0 (1)))
	PS/55 with iSeries Access	5555-B01	((2424J0 (1)))
Japanese 24x24 Printers	Attached to 5295-001 Display	5553-B01	((2424J1 55FE))
	Attached to 5295-002 Display	5553-B01	((2424J1 68FE))
	Attached to PS/55	5553-B01	((2424J1 68FE))
	5227-001 Printer	5553-B01	((2424J2 55FE))
	5327-001 Printer	5553-B01	((2424J2 68FE))
Japanese 32x32 Printers	5337-001 Printer	5553-B01	((3232J0 (1)))
	5383-200 Printer	5583-200	((3232J0 (1)))
Korean Display Stations	5250-Type Display	5555-B01	((2424K0 (1)))
	3270-Type Display	3279-0	((2424K0 (1)))
Korean 24x24 Printers	Attached to 5295 Display	5553-B01	((2424K0 (1)))
	Attached to PS/55	5553-B01	((2424K0 (1)))
	5227-002 Printer	5553-B01	((2424K2 52FE))
Traditional Chinese Display Stations	5250-Type Display	5555-B01	((2424C0))
	3270-Type Display	3279-0	((2424C0))
Traditional Chinese 24x24 Printers	Attached to 5295 Display	5553-B01	((2424C0))
	Attached to PS/55	5553-B01	((2424C0))
	5227-003 Printer	5553-B01	((2424C2 5CFE))
Simplified Chinese Display Stations	5250-Type Display	5555-B01	((2424S0))
	3270-Type Display	3279-0	((2424S0))
Simplified Chinese 24x24 Printers	Attached to PS/55	5553-B01	((2424S0))
	5227-005 Printer	5553-B01	((2424S2 6FFE))

JOB parameter

The JOB parameter specifies the name of the job to which the command is applied. The job name identifies all types of jobs on the system. Each job is identified by a qualified job name, which has the following format:

job-number/user-name/job-name

Note: Although the syntax is similar, job names are qualified differently than OS/400$^{(R)}$ object names.

The following list describes the pieces of the qualified job name:

- Job number: The job number is a unique 6-digit number that is assigned to each job by the system. The job number provides a unique qualifier if the job name is not otherwise unique. The job number can be determined by means of the Display Job (DSPJOB) command. If specified, the job number must have exactly six digits.
- User name: The user name identifies the user profile under which the job is to run. The user name is the same as the name of the user profile and contains a maximum of 10 alphanumeric characters. The name can come from one of several sources, again, depending on the type of job:
  - Batch job: The user name is specified on the SBMJOB command, or it is specified in the job description referenced by the BCHJOB or SBMJOB commands.
  - Interactive job: The user name is specified at sign-on, or the user name is provided from the default in the job description referred to by the work station's job entry.
  - Autostart job: The user name is specified in the job description referred to by the job entry for the autostart job.
- Job name: The job name can contain a maximum of 10 alphanumeric characters, of which the first character must be alphabetic. The name can come from one of three sources, depending on the type of job:
  - Batch job: The job name is specified on the Batch Job (BCHJOB) or Submit Job (SBMJOB) commands or, if not specified there, the unqualified name of the job description is used.
  - Interactive job: The job name is the same as the name of the device (work station) from which the sign-on was performed.
  - Autostart job: The job name is provided in the autostart job entry in the subsystem description under which the job runs. The job name was specified in the Add Autostart Job Entry (ADDAJE) command.

Commands only require that the simple name be used to identify the job. However, additional qualification must be used if the simple job name is not unique.

Duplicate job names

If a duplicate job name is specified in a command in an *interactive* job, the system displays all of the duplicates of the specified job name to the user in qualified form. The job names are displayed in qualified form along with the user name and job number so that you can further identify the job that is to be specified in a command. You can then enter he correct qualified job name.

If a duplicate job name is used in a command in a *batch* job, the command is not processed. Instead, an error message is written to the job log.

Values allowed

The JOB parameter can have one or more of the following values, depending upon the command:

***:** The job is the one in which the command is entered; that is, the command with JOB(*) specified on it.

***JOBD:** The simple job name is the unqualified name of the job description.

***NONE:** No job name is specified as in the Display Log (DSPLOG) command.

job-name: A simple job name is specified.

qualified-job-name: You must specify a qualified job name. If no job qualifier (user name and job number) is given, all of the jobs currently in the system are searched for the job name. If duplicates of the specified name are found, a qualified job name must be specified.

LABEL parameter

» The LABEL parameter specifies the data file identifier of the data file on tape used in input and output operations. « The data file can be in either the exchange format or the save/restore format.

Note: The device file commands are used for tapes that are in the exchange format only, *not* for those in the save/restore format; user-defined device files are not used in save/restore operations. «

» Each data file on tape has its data file identifier stored in its own file label. « The data file label (or header label) of each data file is stored on the tape just before the data in the file. That is, each file on the tape has its own header label and its own data records together as a unit, and one file follows another. In addition to the data file identifier, each label also contains other information about the file, such as the file sequence number, record and block attributes, and whether it is a multivolume data file.

Generally, the data file identifier is an alphanumeric character string that contains no more than 8 characters. » However, the maximum length actually depends on several things: what data format is used for the files and CL commands in which the identifiers are specified. « The unused portion of the file identifier field should be left blank.

The first character of the data file identifier must be alphabetic (A through Z, $, #, or @) and the rest of the characters *should* be alphanumeric (A through Z, 0 through 9, $, #, _, ., and @). You can use special characters if the identifier is enclosed in apostrophes. » However, if the tape is used on a system other than an iSeries(TM) system, the requirements for specifying identifiers on that system must be considered. «

» Tape data file identifiers «

Tape data file identifiers can have as many as 17 characters. However, if a tape is used on a system other than an iSeries system, a maximum of 8 characters, or a qualified identifier of no more than 17 characters, should be used. If more than 8 characters are used, the identifier should be qualified and enclosed in apostrophes so that no more than 8 characters occur in either part, and the parts are separated by a period; for example, LABEL('TAXES.JAN1980'). This limitation applies to the following commands: Create Tape File (CRTTAPF), Change Tape File (CHGTAPF), Override Tape File (OVRTAPF), and Display Tape (DSPTAP).

The data file identifier is put on the volume when the data file is put on the volume. » For input/output operations, the identifier can be specified in one of tape device file commands, or it can be passed as a parameter when the device file is opened by the high-level language program that uses the file. «

Save/Restore format

For tapes in the save/restore format, the identifier can have a maximum of 17 characters. If a library name is used to generate the label, the identifier cannot exceed 10 characters. You may specify a label other than a library name.

Values allowed

One of the following values can be specified for the LABEL parameter, depending upon the command.

***ALL:** Labels for all the data file identifiers in the specified tape volumes are shown on the display.

***NONE:** The data file identifier is not specified. It must be supplied before the device file (and/or database file) is opened to be used in the tape operation.

***SAME:** The data file identifier already present in the tape device file does not change.

data-file-identifier: Specify the identifier of the data file used or displayed with the device file description.

***LIB:** The file label is created by the system and the name of the library specified on the LIB parameter is used as the qualifier for the file name.

***SAVLIB:** The file label is created by the system, and the name of the library specified on the SAVLIB parameter is used as the qualifier for the file name.

LICOPT parameter

You use the Licensed Internal Code options (LICOPT) parameter to specify individual compile-time options. This parameter is intended for the advanced programmer who understands the potential benefits and drawbacks of each selected type of compiler option.

This table shows the strings that are recognized by the Licensed Internal Code option (LICOPT) parameter. These strings are not case sensitive, but they are shown as mixed case for readability.

LICOPT parameter strings

String	Description
AllFieldsVolatile	If set, treats all fields as volatile.
NoAllFieldsVolatile	If set, no fields are treated as volatile.
AllowBindingToLoadedClasses	Indicates that temporary class representations that were created as a result of defineClass calls within a running Java(TM) virtual machine may be tightly bound to other class representations within the same Java virtual machine.
NoAllowBindingToLoadedClasses	Indicates that temporary class representations that were created as a result of defineClass calls within a running Java virtual machine may not be tightly bound to other class representations within the same Java virtual machine.
AllowClassCloning	When multiple Java programs are generated for a JAR file, allows copies of classes from one program to be included in the generated code for another program. Facilitates aggressive inlining.
NoAllowClassCloning	Does not allow copies of classes from one program to be included in the generated code for another program.
AllowInterJarBinding	Allows tight binding to classes outside the class or JAR file being compiled. Facilitates aggressive optimizations.
NoAllowInterJarBinding	Does not allow tight binding to classes outside the class or JAR file being compiled. This overrides the presence of the CLASSPATH and JDKVER parameters on CRTJVAPGM.

String	Description
AllowMultiThreadedCreate	CRTJVAPGM uses multiple threads, if they are available, during creation. On multiprocessor systems this enables the use of more than one processor at a time, reducing the overall time required for a long CRTJVAPGM operation. However, the CRTJVAPGM will use more system resources, leaving fewer resources available for other applications.
NoAllowMultiThreadedCreate	Indicates that CRTJVAPGM performs as usual, using only one thread.
AnalyzeObjectLifetimes	Performs analysis using visible classes to determine which objects are short-lived. A short-lived object does not outlive the method in which it is allocated, and may be subject to more aggressive optimizations.
NoAnalyzeObjectLifetimes	Does not perform analysis of short-lived objects.
AllowBindingWithinJar	Indicates that class representations within a ZIP file or JAR file may be tightly bound to other class representations within the same ZIP file or JAR file.
NoAllowBindingWithinJar	Indicates that class representations within a ZIP file or JAR file may not be tightly bound to other class representations within the same ZIP file or JAR file.
AllowInlining	Tells the translator that it is permitted to inline local methods. This is the default for optimization levels 30 and 40.
NoAllowInlining	Does not tell the translator that it is permitted to inline local methods.
AssumeUnknownFieldsNonvolatile	When the attributes of a field in an external class cannot be determined, this parameter generates code by assuming that the field is non-volatile.
NoAssumeUnknownFieldsNonvolatile	When the attributes of a field in an external class cannot be determined, this parameter generates code by assuming that the field is volatile.
BindErrorHandling	Specifies what action should be taken if, as a result of honoring the AssumeUnknownFieldsNonvolatile, PreresolveExtRef, or PreLoadExtRef Licensed Internal Code option, the Java virtual machine class loader detects that a class representation contains method representations, which cannot be used in the current context.
BindInit	Uses bound call to local init methods.
NoBindInit	Does not use bound call to local init methods.
BindSpecial	Uses bound call to local special methods.
NoBindSpecial	Does not use bound call to local special methods.
BindStatic	Uses bound call to local static methods.
NoBindStatic	Does not use bound call to local static methods.
BindTrivialFields	Binds trivial field references during program creation.
NoBindTrivialFields	Resolves field references at first touch.
BindVirtual	Uses bound call to local final virtual methods.
NoBindVirtual	Does not use bound call to local final virtual methods.

String	Description
DeferResolveOnClass	Takes a string parameter that is presumed to be the name of a class (for example, java.lang.Integer). When you set PreresolveExtRef to optimization level 40, classes that are specified with DeferResolveOnClass are not in the preresolve operation. This is useful if some classes in unused paths in the code are not in the CLASSPATH. It allows you to use optimization level 40 regardless of this by specifying a ″DeferResolveOnClass=’somepath.someclass’″ for each missing class. Multiple DeferResolveOnClass entries are allowed.
DevirtualizeFinalJDK	Allows CRTJVAPGM to use knowledge of the standard JDK to devirtualize calls to those JDK methods that are known to be final methods or members of final classes. It is the default at optimization levels 30 and 40.
NoDevirtualizeFinalJDK	Does not allow CRTJVAPGM to use knowledge of the standard JDK to devirtualize calls to those JDK methods that are known to be final methods or members of final classes.
DevirtualizeRecursive	Causes special code to be generated in the case of some recursive methods and eliminates much of the overhead of the recursive method calls. However, additional checking logic is generated on initial entry to the recursive method, so performance may not improve in cases of shallow recursion.
NoDevirtualizeRecursive	Does not cause special code to be generated in the case of some recursive methods.
DisableIntCse	Causes certain common subexpression optimizations to be disabled when generating code for certain types of integer expressions. This may improve overall optimization by exposing other optimization opportunities to the Optimizing Translator.
NoDisableIntCse	Causes certain common subexpression optimizations to not be disabled when generating code for certain types of integer expressions. This generally results in better performing code at lower optimization levels.
DoExtBlockCSE	Performs extended basic block common subexpression elimination.
NoDoExtBlockCSE	Does not perform extended basic block common subexpression elimination.
DoLocalCSE	Performs local common subexpression elimination.
NoDoLocalCSE	Does not perform local common subexpression elimination.
EnableCseForCastCheck	If set, generates code for castcheck that can be DAGed to an earlier instance.
NoEnableCseForCastCheck	Is not set; does not generate code for castcheck that can be DAGed to an earlier instance.
ErrorReporting	Runtime error reporting field**: Provides the option to fail the compile when encountering verification or class format errors. 0=Report all errors immediately; 0=Report all errors immediately; 1=Defer reporting of bytecode verification errors; 2=Defer reporting of bytecode verification errors and class format errors to runtime.

String	Description
HideInternalMethods	Causes methods in cloned classes to be made internal, allowing the methods to be omitted if there are no references to them or if all references are inlined. The default is HideInternalMethods for optimization 40 and NoHideInternalMethods for optimization between 0 and 30.
InlineArrayCopy	Causes the inlining of the System.arraycopy method in some cases of scalar arrays.
NoInlineArrayCopy	Prevents the inlining of the System.arraycopy method.
InlineInit	Inlines init methods for java.lang classes.
NoInlineInit	Does not inline init methods.
InlineMiscFloat	Inlines miscellaneous float/double methods from java.lang.Math.
NoInlineMiscFloat	Does not inline miscellaneous float/double methods.
InlineMiscInt	Inlines miscellaneous int/long methods from java.lang.Math.
NoInlineMiscInt	Does not inline miscellaneous int/long methods.
InlineStringMethods	Permits inlining of certain methods from java/lang/String.
NoInlineStringMethods	Inhibits inlining of certain methods from java/lang/String.
InlineTransFloat	Inlines transcendental float/double methods from java.lang.Math.
NoInlineTransFloat	Does not inline transcendental float/double methods.
OptimizeJsr	Generates better code for ″jsr″ bytecodes that have a single target.
NoOptimizeJsr	Suppresses generation of better code for ″jsr″ bytecodes that have a single target.
PreloadExtRef	Indicates that referenced classes may be preloaded (without class initialization) upon method entry.
NoPreloadExtRef	Indicates that referenced classes may not be preloaded upon method entry. However, the PreresolveExtRef parameter overrides this setting and causes referenced classes to be preloaded and initialized.
PreresolveExtRef	Preresolves referenced methods at method entry.
NoPreresolveExtRef	Resolves method references at first touch. Use to resolve ″class not found″ exceptions on programs that run on other machines.
ProgramSizeFactor	When a JAR file may be large enough to require multiple Java programs, this numeric value (default 100) is used to determine how large each program can grow.
ShortCktAthrow	If set, attempt to short-circuit athrows.
NoShortCktAthrow	Is not set, does not attempt to short-circuit athrows.
ShortCktExSubclasses	If set, recognizes some subclasses of Exception and short-circuit them directly.
NoShortCktExSubclasses	If not set, does not recognize some subclasses of Exception and short-circuit them directly.

String	Description
StrictFloat	Inhibits floating-point optimizations that are not strictly compliant with the Java specification.
NoStrictFloat	Permits floating-point optimizations that are not strictly compliant with the Java specification.

The double asterisk (**) signifies that these strings require a numerical value for input in the syntax of stringname=number (with no spaces in between).

«

MAXACT parameter

The maximum activity level (MAXACT) parameter specifies the maximum number of jobs that can be concurrently started and that remain active through a job queue entry, communications entry, routing entry, or work station entry. A job is considered active from the time it starts running until it is completed. This includes time when:

- The job is actually being processed.
- The job is waiting for a response from a work station user.
- The job is started and available for processing but is not actually using the processor. For example, it might have used up its time slice and is waiting for another time slice.
- The job is started but is not available for processing. For example, it could be waiting for a message to arrive on its message queue.

Values allowed

***NOMAX:** There is no maximum number of jobs that can be active at the same time.

maximum-active-jobs: Specify a value that indicates the maximum number of jobs that can be concurrently active through this entry.

See the Work Management book on the V5R1 Supplemental Manuals Web site for a description of activity level controls.

OBJ parameter

The object (OBJ) parameter specifies the names of one or more objects affected by the command in which this parameter is used. » If the OBJ parameter identifies objects that must exist in an OS/400(R) library, all of the objects must be in one of the following, depending upon which command is used:

- the library specified in the LIB parameter,
- the SAVLIB parameter,
- the library qualifier in the OBJ parameter,
- or the library part of the path name in the OBJ parameter

«

On some commands, the generic name of a group of objects can be specified. To form a generic name, add an asterisk (*) after the last character in the common group of characters; for example, ABC*. If an * is not included with the name, the system assumes that the name is a complete object name.

Values allowed

Depending on the command, the following types of values can be specified on the OBJ parameter:

- *ALL
- Simple object name
- Qualified object name
- Generic object name
- Qualified generic object name
- Path name (For more information, see "Path names (*PNAME)" on page 49.)

OBJTYPE parameter

The object type (OBJTYPE) parameter specifies the types of OS/400[(R)] objects that can be operated on by the command in which they are specified. The object types that can be specified in the OBJTYPE parameter vary from command to command.

The object-related commands allow you to perform general functions on most objects without knowing the special commands related to the specific object type. For example, you could use the CRTDUPOBJ command to create a copy of a file or library instead of the specific commands CPYF (Copy File) or CPYLIB (Copy Library).

Object-related commands

This section lists commands containing the OBJTYPE parameter. See the information for the individual commands listed to find out which object types can be operated on using the commands. See "External object types" on page 43 for a list of the special values and their corresponding object types.

The following commands contain the OBJTYPE parameter but operate on only a few object types.

- CHKDLO operates on *DOC and *FLR.
- CPROBJ and DCPOBJ operate on *FILE, *MENU, *MODULE, *PGM, *PNLGRP, and *SRVPGM.
- CRTSQLPKG operates on *PGM and *SRVPGM.
- DSPPGMADP operates on *PGM, *SQLPKG, and *SRVPGM.
- DSPPGMREF operates on *PGM and *SQLPKG.
- RSTCFG operates on *CFGL, *CNNL, *COSD, *CTLD, *DEVD, *LIND, *MODD, and *NWID.
- SAVLICPGM operates on *LNG and *PGM.
- SETOBJACC operates on *FILE and *PGM.

The DSPLNK and WRKLNK commands operate on all object types.

The ALCOBJ and DLCOBJ commands also require that an object type value is specified. However, for these commands, the object type value is specified as one of four values (in a list of values) on the required parameter OBJ.

The following object-related commands operate on many object types.

Object	Object Authority	Save/Restore	Journal	Other
CHGOBJD	CHGOBJAUD	RSTOBJ	CHGJRNOBJ	DMPOBJ
CHKOBJ	CHGOBJOWN	SAVCHGOBJ	ENDJRNOBJ	DMPSYSOBJ
CRTDUPOBJ	CHGOBJPGP	SAVOBJ	STRJRNOBJ	PRTDSKINF
DSPOBJD	DSPOBJAUT			WRKOBJLCK
MOVOBJ	EDTOBJAUT			
RNMOBJ	GRTOBJAUT			
RTVOBJD	RVKOBJAUT			
WRKOBJ				

OUTPUT parameter

You use the OUTPUT parameter to specify whether the output from the display command will be shown on the display, printed, or written to an output file. Basically, the same information is provided in either form; only the format is changed as necessary to present the information in the best format for the device. For example, because there are more lines on a printed page than on a display, column headings are not repeated as often in printed output.

If the output is to be shown on the display, it will be sent to the work station that issued the display command. It will be shown in the format specified in the display device file used by that display command. A different device file is used for the output of each display command, and the file is different for displayed, printed, or written file output. In most cases, the name of the command is part of the file names of either type of device file.

If the output will be printed, it is spooled and an entry is placed on the job's output queue. The output can be printed depending on which device is specified in the Start Printer Writer (STRPRTWTR) command.

Note: Although the IBM(R)-supplied printer files are shipped with SPOOL(*YES) specified, they can be changed to SPOOL(*NO) by the Override with Printer File (OVRPRTF) and Change Printer File (CHGPRTF) commands.

If the OUTPUT parameter is not specified in the display command, the default value * is assumed. The output resulting from this value depends on the type of job that entered the command. The following table shows how the output is produced for interactive and batch jobs.

Output	Interactive Job	Batch Job
*	Displayed	Printed
*PRINT	Printed	Printed

Values allowed

***:** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT:** The output is printed with the job's spooled output.

***OUTFILE:** The only output is to be written to a specified database file.

PRTTXT parameter

The print text (PRTTXT) parameter specifies the text that appears at the bottom of listings and on separator pages. Print text is copied from the job attribute when the job enters the system. Print files that originate on another system do not use the print text on the target system. Print text exists as a job attribute (PRTTXT) for defining the print text of a specific job, and as a system value (QPRTTXT) for the default of jobs with *SYSVAL specified. QPRTTXT is the system-wide default for all jobs.

The print text can be up to 30 characters in length. The text should be centered in the form's width and printed in the overflow area. You should center the desired text within the 30 character field.

If the print text is not blank, the system prints 30 characters of text on the bottom of each page. This text normally follows the overflow line and is preceded by a blank line (if the form's length permits). If the user prints past the overflow line, the print text follows the last line of the user text, again preceded by a blank line when possible. If the overflow line is the last line of the form, the print text also prints on the last line of the form, which may result in the typing over of user text.

The print text for job and file separators is put on the first line of the separator page. A job separator contains print text of the job that created the separator at the time the file was printed. A file separator contains the same print text as the spooled file it precedes.

The print text can be specified for all job types. System and subsystem monitor jobs use the system value. Reader and writer jobs use the system value unless print text is changed in the QSPLxxxx job description associated with the reader or writer.

The print text is determined from several places by using the following hierarchical order. If print text is not specified in one place, the next place in the order is used.

The hierarchical order, beginning with the highest priority, is as follows:
- Override print file value
- Print file value
- Job attribute changed by the Change Job (CHGJOB) command
- Job attribute set by the Submit Job (SBMJOB) or Batch Job (BCHJOB) command
- Job description
- System value

Values allowed

For the system value QPRTTXT, any character string can be specified, with the exception of *SYSVAL. If *BLANK is specified, there will be no print text. For PRTTXT, some of the following values can be selected, depending on the command:

***SAME:** The print text does not change.

***CURRENT:** The print text is taken from the submitting job.

***JOBD:** The print text is taken from the job description under which the job is run.

***SYSVAL:** The print text is taken from the system value QPRTTXT.

***BLANK:** There is no text or blanks printed.

'print-text': Specify 30 characters of text. If there are blanks in the text, then apostrophes must be used around the entry. The text should be centered within the field for the text to be centered on the page.

REPLACE parameter

The replace (REPLACE) parameter is used on create commands. It specifies that the existing object, if one exists, is replaced by the object of the same name, library, and object type that is being created. The user of the new object is granted the same authority as for the object being replaced. If the object being replaced is secured by an authorization list, then the new object is secured by the same authorization list. The public authority of the new object is the same as the public authority of the replaced object. The AUT parameter from the create command is ignored. All private authorities from the replaced object are copied to the new object. The owner of the new object is *not* copied from the replaced object. The owner of the new object is the creator of the new object or the creator's group profile. Some objects such as panel groups, display files, and menus cannot be replaced if they are in use by the current job or another job.

If the object being created is a program or service program, then the user profile (USRPRF parameter) value from the replaced program is used. The user profile (USRPRF parameter) value from the Create Program or Create Service Program command is ignored. If the value of the user profile (USRPRF parameter) of the program or service program being replaced is *OWNER, then only the current owner of the program or service program being replaced can create the new program or service program that replaces the existing program or service program. If the owner of the existing object and the object being created do not match, the object is not created and message CPF2146 is sent.

If the object being created is a program or service program, then the use adopted authority (USEADPAUT) value from the replaced program or service program is used as long as the user creating the object can create programs/service programs with the USEADPAUT(*YES) attribute. The QUSEADPAUT system value determines whether or not users can create programs or service programs to use adopted authority. For example, if the existing object being replaced has USEADPAUT(*YES) and you do not have authority to create a program or service program that uses adopted authority, the program or service program created will have USEADPAUT(*NO). In this case, the USEADPAUT value was not copied. If you have authority to create programs or service programs that use adopted authority, the created program or service program will have the same USEADPAUT value as the program or service program being replaced. An informational message is sent which indicates whether the USEADPAUT value was copied to the object being replaced.

If the object being created is a file, and the default, or *YES, is specified on the REPLACE parameter, an existing device file other than save file and a DDM file with the same qualified name will be replaced by the new file. For example, an existing display file can be replaced by a new printer file, or tape file, etc.

Object management (*OBJMGT), object existence (*OBJEXIST), and read (*READ) authorities are required for the existing object to allow replacement of the existing object with a new object.

The existing object is renamed and moved to library QRPLOBJ or library QRPLxxxxx if the object resides on an Independent ASP (where ’xxxxx’ is the number of the primary ASP of the ASP group) when the creation of the new object is successful. The replaced object is renamed with a Q appended to a time stamp and moved to library QRPLOBJ or library QRPLxxxxx if the object resides on an Independent ASP.

Restriction: Programs can be replaced while they are being run; however, if the replaced program refers to the program message queue after the renaming of the replaced program to the Qtimestamp name, the program fails and an error message is sent stating that the program message queue is not found.

A database file, physical or logical, and a save file cannot be replaced by any file.

Library QRPLOBJ is cleared when an initial program load (IPL) of the system is done. Library QRPLxxxxx is cleared when the primary ASP of the ASP group is varied on.

Values allowed

***YES:** The system replaces the existing object with the new object being created that has the same name, library, and object type.

***NO:** The system does not replace the existing object that has the same name, library, and object type with the object being created.

Scheduling priority parameters (JOBPTY, OUTPTY, PTYLMT)

The scheduling priority parameters specify the priority values used by the system to determine the order in which the jobs and spooled files are selected for processing. Each job is given a scheduling priority that is used for both job selection and spooled file output. The job scheduling priority is specified by the JOBPTY parameter in commands like the Batch Job (BCHJOB), Submit Job (SBMJOB), Create Job Description (CRTJOBD), and Change Job Description (CHGJOBD) commands. The priority for producing the spooled output from a job is specified by the OUTPTY parameter in the same commands.

In addition, because every job is processed under a specific user profile, the priority for jobs can be limited by the PTYLMT parameter specified on the Create User Profile (CRTUSRPRF) and Change User Profile (CHGUSRPRF) commands. This parameter value controls the maximum job scheduling priority and output priority that *any* job running under a user profile can have; that is, the priority specified in the JOBPTY and OUTPTY parameters of any job command cannot exceed the priority specified in the PTYLMT parameter for that user profile. The scheduling priority is used to determine the order in which jobs are selected for processing and is not related to the process priority specified in the class object.

The three scheduling priority parameters specify the following:

- The PTYLMT parameter specifies the *highest* scheduling priority for *any* job that you submit. In the commands that affect the user profile, the PTYLMT parameter specifies the highest priority that can be specified in another JOBPTY parameter on commands relating to each specific job. You can specify a lower priority for a job on the command used to submit the job. If you specify a higher priority for JOBPTY in the BCHJOB or SBMJOB command than is specified for PTYLMT in the associated user profile, an error message is shown on the display and the maximum priority specified in PTYLMT is assumed. If a higher job priority is specified in the CHGJOB or CHGJOBD command, an error message is shown and the attributes are not changed.
- The JOBPTY parameter specifies the priority value to be used for a *specific* job being submitted. In the commands relating to a specific job being submitted, the JOBPTY parameter specifies the actual scheduling priority for the job.
- The OUTPTY parameter specifies the priority for producing the output from all spooled output files from the job. The priority value specified in the OUTPTY parameter determines the order in which spooled files are handled for output. The same value is applied to all the spooled files produced by the job.

The scheduling priority can have a value ranging from 0 through 9, where 1 is the highest priority and 9 is the lowest priority. Any job with a priority of 0 is scheduled for processing before all other jobs that are waiting and that have priorities of 1 through 9.

The priority parameters can be specified on the following commands.

Priority parameter	Commands on which it can be specified
JOBPTY	ADDJOBJS, BCHJOB, CHGJOB, CHGJOBD, CHGJOBJS, CRTJOBD, SBMJOB, SBMJOBJS
OUTPTY	ADDJOBJS, BCHJOB, CHGDKTF, CHGJOBD, CHGJOBJS, CHGJOB, CHGJOBD, CHGJOBJS, CHGPJ, CHGPRTF, CHGSPLFA, CRTDKTF, CRTJOBD, CRTPRTF, OVRDKTF, OVRPRTF, SBMJOB, SBMJOBJS
PTYLMT	CHGUSRPRF, CRTUSRPRF, RTVUSRPRF

Values allowed

Depending upon the command, one or more of the following values apply to the parameter.

5: If a value is not specified in the CRTUSRPRF command, five is the default value that is assumed for the priority limit for the user profile. That would be the highest priority that the user could specify for any job he submits for processing. If not specified in the CRTJOBD command, five is the default value for both the job scheduling priority and the output priority.

***SAME:** The priority assigned, or the highest priority that can be assigned, does not change.

***JOBD:** The scheduling priority for the job is obtained from the job description under which the job runs.

scheduling-priority: Specify a priority value ranging from 0 through 9, where 0 is the highest priority and 9 is the lowest priority. Priority 0 is allowed only on CHGJOB.

SEV parameter

The severity (SEV) parameter specifies the severity code that:

- Describes the level of severity associated with an error message.
- Indicates the minimum severity level that causes a message to be returned to a user or program.
- Causes a batch job to end.
- Causes processing of a command to end if a syntax error of sufficient severity occurs.

Note: The LOG parameter on some commands also uses these severity codes for logging purposes (to control which job activity messages and error messages are logged in the job log).

The severity code is a 2-digit number that can range from 00 through 99. The higher the value, the more severe or important the condition. The severity code of a message that is sent to a user indicates the severity of the condition described by the message. More than one message can have the same severity code. If a severity code is not specified for a predefined message, it is assumed to be 00 (information only).

You can specify a severity code for any message when it is defined by the Add Message Description (ADDMSGD) command. To change the severity code of a message, use the Change Message Description (CHGMSGD) command.

IBM^{R}-defined severity codes are used in all of the IBM$^{(R)}$-supplied messages that are shipped with the system.

00 - Information: A message of this severity is for information purposes only; no error was detected and no reply is needed. The message could indicate that a function is in progress or that it has reached a successful completion.

10 - Warning: A message of this severity indicates a potential error condition. The program may have taken a default, such as supplying missing input. The results of the operation are assumed to be what was intended.

20 - Error: An error has been detected, but it is one for which automatic recovery procedures probably were applied, and processing has continued. A default may have been taken to replace input that was in error. The results of the operation may not be valid. The function may be only partially complete; for example, some items in a list may be processed correctly while others may fail.

30 - Severe Error: The error detected is too severe for automatic recovery, and no defaults are possible. If the error was in source data, the entire input record was skipped. If the error occurred during program processing, it leads to an abnormal end of the program (severity 40). The results of the operation are not valid.

40 - Abnormal End of Program or Function: The operation has ended, possibly because it was unable to handle invalid data, or possibly because the user ended it.

50 - Abnormal End of Job: The job was ended or was not started. A routing step may have ended abnormally or failed to start, a job-level function may not have been performed as required, or the job may have been ended.

60 - System Status: A message of this severity is issued only to the system operator. It gives either the status of or a warning about a device, a subsystem, or the whole system.

70 - Device Integrity: A message of this severity is issued only to the system operator. It indicates that a device is malfunctioning or in some way is no longer operational. You may be able to restore system operation, or the assistance of a service representative may be required.

80 - System Alert: A message of this severity is issued only to the system operator. It warns of a condition that, although not severe enough to stop the system now, could become more severe unless preventive measures are taken.

90 - System Integrity: A message of this severity is issued only to the system operator. It describes a condition that renders either a subsystem or the whole system inoperative.

99 - Action: A message of this severity indicates that some manual action is required, such as specifying a reply or changing printer forms.

SPLNBR parameter

The spooled file number (SPLNBR) parameter is used when more than one spooled file is created by a job and the files all have the same name. The files are numbered, starting with 1, in the order that they are opened by the job. The job log is always the last file for a job.

A file number is generated for each file when it is opened within a job (when output records are produced) and it is used by the system as long as the job and/or the files are on the system. If the files are not uniquely named because they were opened more than once, this file number is used to specify which file (or group of records, if the complete file has not yet been produced) is acted upon by a CL command.

TEXT parameter

The TEXT parameter specifies the user-defined description that briefly describes the object being created or changed. The description can include up to 50 characters; if it is a quoted string (that is, enclosed in apostrophes), any of the 256 EBCDIC characters can be used. The apostrophes are not required if the string does not contain any blanks or other special characters. Any of the 50 character positions not filled by the specified description are padded with blanks.

The description is used to describe any of the OS/400$^{(R)}$ objects when the named object is shown on the display by the Display Object Description (DSPOBJD) command. Only objects for which object operational authority has been obtained can be displayed by a user. See the “OBJTYPE parameter” on page 69 description for a list of the OS/400 object types.

For commands that use a database source file to create some type of object, you can (by default) use the text from the source file member as the text for the newly-created object. For example, if you use the Create Control Language Program (CRTCLPGM) command to create a CL program, but you do not specify a description in the TEXT parameter, the text specified for the source file member (SRCMBR parameter) of the source file (SRCFILE parameter) is assumed as the descriptive text for the CL program.

Values allowed

Depending upon the command, one or more of the following values apply to the TEXT parameter.

***SRCMBRTXT:** For commands that create objects based on database source files only, the text is taken from the source member. If a device or an inline file is used for source input or if source is not used, the text is left blank.

***BLANK:** The user description of the object being created or changed is left blank.

***SAME:** The user-defined description does not change.

'description': Specify the description of the object being created or changed. Up to 50 characters enclosed in apostrophes (required for blanks and other special characters) can be specified to describe the object. If an apostrophe is one of the 50 characters, two apostrophes ('') must be used instead of one to represent the apostrophe character.

VOL parameter

The volume (VOL) parameter specifies the volume identifiers of the volumes used in a tape or optical operation. A tape volume consists of a tape cartridge or reel. An optical volume consists of a single side of an optical cartridge or a single CD-ROM. Optical cartridges are dual-sided and each side is a separate volume.

The volume identifier is the identifier stored on each tape or optical disk (in the volume label area) that it identifies. An inquiry message is sent to the system operator if a volume identifier is missing or out of order.

Tape volumes must be on the tape units in the same order as their identifiers are specified in the VOL parameter and as the device names are specified in the DEV parameter of the tape device file commands. However, if the tapes are read backward (a function supported in COBOL), the volumes must be in reverse order to that specified in the VOL parameter. Nevertheless, the device names are still specified in forward order in the DEV parameter.

In general, the rule for specifying tape volume identifiers is that as many as 6 characters, containing any combination of letters and digits, can be used. Special characters can be used if the identifier is enclosed in apostrophes. However, if the tape is used on a system other than an iSeries(TM) system, the requirements for specifying identifiers on that system must be considered.

Optical volume identifiers can be up to 32 characters long and can contain any combination of digits and upper case letters. Each optical volume identifier must be unique. No two optical volumes with the same identifier can be present on the system at the same time. A complete list of the rules for optical volume identifiers can be found in the *Optical Support* book.

For labeled tapes, the following rules apply:

- Characters: A maximum of 6 characters, or fewer, can be specified for each volume identifier. Alphabetic and numeric characters can be used in any order.
- Uniqueness: More than one volume can have the same identifier. You may have a file using the same identifier for several volumes; in this case, the system keeps track of the order internally with a sequence number written on the volumes. However, volume identifiers should be unique whenever possible.
- Order: When multiple volumes (with different identifiers) are used in a single operation, they must be in the same order as the volume identifiers specified in the VOL parameter.

Multivolume files

If multiple volumes (tapes) are used in an operation and all have the same volume identifier, that identifier must be specified in the VOL parameter once for each volume used. For example, if three tapes named QGPL are used in a save operation, VOL(QGPL QGPL QGPL) must be specified.

When a multivolume file on *tape* is processed and multiple tape units are used, the tape volumes must be placed in the tape devices in the same order as they are specified in the VOL parameter. For example, if five volumes and three tape units are used, they are mounted as follows: VOL1 on unit 1, VOL2 on unit 2, VOL3 on unit 3, VOL4 on unit 1, and VOL5 on unit 2.

Values allowed

***MOUNTED:** The volume currently placed in the device is used.

***NONE:** No volume identifier is specified.

***SAME:** Previously specified volume identification does not change.

***SAVVOL:** The system, using the save/restore history information, determines which tape volumes contain the most recently saved version. If the device specified in the DEV parameter of the restore command does not match the device of the most recently saved version of the object, an error message is returned to the user, and the function is ended. If the wrong volume is mounted in the unit specified by the command, a message is returned to the system operator that identifies the first volume that must be placed in the device before the restore operation can begin.

volume-identifier: Specify the identifiers of one or more volumes in the order in which they are put on the device and used. Each tape volume identifier contains a maximum of 6 alphanumeric characters. Each optical volume identifier contains a maximum of 32 characters. A blank is used as a separator character when listing multiple identifiers.

WAITFILE parameter

You use the WAITFILE parameter to specify the following:

- For the maximum number of seconds that a program waits for file resources to be allocated when the file is opened
- For session resources when the evoke function is issued for an APPC device
- For the device to be allocated when an acquire operation is performed to read the file

If the program must wait, it will be placed in a wait state until the resources are available or until the wait time expires. If two or more file resources are needed and are not available because they are being used by different system users, the acquisition of each resource might require a wait. This maximum is applied to each wait.

The length of the wait can be specified in this parameter, or the default wait time of the class that applies to the object can be used. If the file resources cannot be allocated in the specified number of seconds, an error message is returned to the program.

The file resources that must be allocated depend on the type of file being opened. File resources consist of the following.

- For device files that are not spooled (SPOOL(*NO)), the file resources include the file description and device description. Because the device description must be allocated, the device itself must also be available.
- For device files that are spooled (SPOOL(*YES)), the file resources include the file description, the specified output queue, and storage in the system for the spooled data. Because the data is spooled, the device description (and thus the device itself) need not be available.
- For database files, the file resources consist of the file and member data. The file's associated member paths are not accessed, and therefore, the system does not wait for them. A file open exception error can occur before the WAITFILE time has expired when an access path is not available (for example, when the access path is being rebuilt).

  The Allocate Object (ALCOBJ) command can be used to allocate specific file resources before the file is opened.

The session resources that were allocated for an APPC device conversation can be lost between the time the application issues a detach function or receives a detach indication and the time another evoke function is issued. If the session resource is lost, this parameter is used to determine the length of time that the system waits for another session resource.

Values allowed

***IMMED:** The program does not wait; when the file is opened, an immediate allocation of the file resources is required.

***CLS:** The default wait time specified in the class description is used as the wait time for the file resources to be allocated.

1-32767 : Specify the number of seconds that the program waits for the file resources to be allocated.

Database and device files used by CL commands

Many of the IBM(R)-supplied CL commands use database and device files during processing. This section provides a cross-reference between the commands and the IBM-supplied files used by the commands. All of the commands and files for *all* licensed programs that meet the following criteria are included:

- The types of files included are:
  - Database files: physical (PF) and logical (LF), including files with data and files used as model files (no data)
  - Device files: tape (TAPF) and printer (PRTF)
- The files are included only if a user might have a reason to use them, such as declaring them in a program so they can be overridden with another file. Two examples:
  - You want to change some attributes for a printer device file, such as the font to be used or the printed lines per inch (the FONT or LPI parameter).
  - You want to override the IBM-supplied file with your own output file (when valid).

The types of files **not** included in this section are:

- OS/400(R)-provided display (DSPF) device files, because they should not be changed or overridden.
- Most of the OS/400-provided database files used by *directory* commands, *document library object* commands, and *optical index database* files, because they cannot be overridden.

As mentioned, IBM-supplied physical (PF) or logical (LF) files that are used as **model files** for certain commands are included in the following table. (Examples are the model files listed under the DSPFD, DSPJRN, and STRPFRMON commands.) In most cases, these model files do not contain data; instead, they contain the *definitions* (or record formats) of the files to be created for storing the actual output data resulting from use of these commands. For the record formats of these files, you can display the file's description online or you can refer to the publication or soft-copy manual that documents that command.

Additionally, some files considered to be IBM-supplied files do not actually exist on the system until some function is used that requires the file and so creates it at that time.

The following notes describe how the table of commands and files are sorted and explain the meanings of the superscripts used in the table.

Notes for table:

1. The first column of the table lists the CL commands that use the OS/400-provided files shown in the third column. The table entries are in alphabetic order by *command name* first. If there is more than one library for a command, they are further ordered by the file *library name* and then by *file name* within each library.
2. **A superscript 1 (^{1})** following the description of a file indicates that the file is used only when the output from the command is directed to that *form* of output-by an output-related parameter on the command.
   - The superscript ^{1} is used at the end of *printer file* (PRTF) descriptions to show that use of the printer file is dependent on the job environment and the print-related value specified (or assumed) on the command. For commands with an OUTPUT parameter (primarily DSPxxx and WRKxxx commands), the output is printed either when OUTPUT(*) is specified in a batch job, or when OUTPUT(*PRINT) is specified in a batch or interactive job. For other commands, the output is printed if an *PRINT, *LIST, and/or *SRC value is specified on a different parameter.
   - The superscript ^{1} is used at the end of *database file* (PF or LF) descriptions to show that use of the database file is dependent on the print-related value specified (or assumed) on the command. For commands with an OUTPUT parameter (primarily DSPxxx and SAVxxx commands), the output is directed to the database file when OUTPUT(*OUTFILE) is specified. The same is true for other commands that specify a value similar to *OUTFILE on one of its parameters.
3. **A superscript 2 (^{2})** following a file's description indicates that the file is a *model file* and not an output file. As a model file, it defines the record format of the file created to contain the actual output.
4. Those files showing "user-lib" as the file library do not exist on the system until the user creates them. When the command is used, the file is created in the user's specified library with the file name shown.

Files Used by CL Commands (Part 1)

Command Name	File Library	File Name	File Type	File Usage
ADDDOCCVN	QUSRSYS	QAO1CRL	LF	Document conversion *logical* file for input or output.
	QUSRSYS	QAO1CVNP	PF	Document conversion *physical* file for input or output.
	QUSRSYS	QAO1DCVN	PRTF	Document conversion printer file.
ADDDSTQ	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
ADDDSTRTE	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
	QUSRSYS	QASNADSR	PF	SNADS routing table.
ADDDSTSYSN	QUSRSYS	QASNADSA	PF	SNADS secondary node ID table.
ADDNETJOBE	QUSRSYS	QANFNJE	PF	Network job entry database file.
ADDSOCE	QUSRSYS	QAALSOC	PF	Sphere of control file.
ADDTAPCTG	QUSRSYS	QATAMID	PF	Cartridge ID db file.
	QUSRSYS	QLTAMID	LF	Cartridge ID logical file.
	QUSRSYS	QATACGY	PF	Category db file.
	QUSRSYS	QLTACGY	LF	Category logical file.
ADDTCPHTE	QUSRSYS	QATOCHOST	PF	TCP/IP host file.
ADDTCPIFC	QUSRSYS	QATOCIFC	PF	TCP/IP configuration file.
ADDTCPPORT	QUSRSYS	QATOCPORT	PF	TCP/IP configuration file.
ADDTCPRSI	QUSRSYS	QATOCRSI	PF	TCP/IP configuration file.

Command Name	File Library	File Name	File Type	File Usage
ADDTCPRTE	QUSRSYS	QATOCRTE	PF	TCP/IP configuration file.
ADDTXTIDXE	QUSRSYS	QABBADMTB	PF	OfficeVision^{R} text search services administration table.
ANSQST	QSYS	QPQAPRT	PRTF	Q & A printer file.
ANZACCGRP	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QAPTPAGD	PF	Performance database input file of files and programs in a process access group (PAG).
	QPFR	QPPTPAG	PRTF	Printer file of PAG data containing environment, job, file, and program summary data.
ANZDBF	QPFR	QAPTAZDR	PF	Performance data collection file: analyze application database files data.
	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTANZD	PRTF	Performance printer file showing physical-to-logical and logical-to-physical database file relationships.
ANZDBFKEY	QPFR	QAPTAZDR	PF	Performance input file of analyze application database files data showing logical file key structures.
	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTANKM	PRTF	Performance printer file containing logical file key structure data.
	QPFR	QPPTANZK	PRTF	Performance printer file containing access path and record selection data.
ANZPFRDTA	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = ASYN, BSC, CIOP, CONF, DIOP, DISK, ECL, ETH, HDLC, IDLC, JOBS, LAPD, LIOP, MIOP, POOL, SYS, and X25.[2]
	QPFR	QPAVPRT	PRTF	Performance printer file containing the advisor report.[1]
ANZPGM	QPFR	QAPTAZPD	PF	Performance data collection file: analyze application programs data.
	QPFR	QPPTANZP	PRTF	Performance printer file showing program-to-file and file-to-program relationships.
APYJRNCHG	QSYS	QAJRNCHG	PF	Model output file for apply journaled changes.
APYJRNCHGX	QSYS	QAJRNCHG	PF	Model output file for apply journaled changes extend.
ASKQST	QSYS	QPQAPRT	PRTF	Q & A printer file.
CFGDSTSRV	QUSRSYS	QASNADSA	PF	SNADS secondary node ID table.
	QUSRSYS	QASNADSQ	PF	SNADS destination queues table.
	QUSRSYS	QASNADSR	PF	SNADS destination systems routing table.
CFGTCP	QUSRSYS	QATOCIFC	PF	TCP/IP interface file.
	QUSRSYS	QATOCRTE	PF	TCP/IP route file.
	QUSRSYS	QATOCTCPIP	PF	TCP/IP attributes file.
	QUSRSYS	QATOCPORT	PF	TCP/IP port restrictions file.
	QUSRSYS	QATOCRSI	PF	TCP/IP RSI file.

Command Name	File Library	File Name	File Type	File Usage
	QUSRSYS	QATOCHOST	PF	TCP/IP hostname file.
	QUSRSYS	QATOCPS	PF	TCP/IP services file.
CFGTCPSMTP	QUSRSYS	QATMSMTP	PF	TCP/IP SMTP file.
	QUSRSYS	QATMSMTPA	PF	TCP/IP SMTP file.
CHGDOCCVN	QUSRSYS	QAO1CRL	LF	Document conversion *logical* file for input or output.
CHGDTA	QIDU	QDTALOG	PRTF	Audit control log printer file.
	QIDU	QDTAPRT	PRTF	Record and accumulator total printer file.
	QSYS	QPDZDTALOG	PRTF	DFU run-time audit log.
	QSYS	QPDZDTAPRT	PRTF	DFU run-time printer data file.
CHGDSTQ	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
CHGDSTRTE	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
	QUSRSYS	QASNADSR	PF	SNADS routing table.
CHGFTPA	QUSRSYS	QATMFTP	PF	TCP/IP FTP configuration file.
CHGHTTPA	QUSRSYS	QATMHTTP	PF	TCP/IP HTTP file.
CHGLPDA^{R}	QUSRSYS	QATMLPD	PF	TCP/IP LPD configuration file.
CHGPOPA	QUSRSYS	QATMPOPA	PF	POP server configuration file.
CHGPRB	QUSRSYS	QASXNOTE	PF	Problem log user notes file.
	QUSRSYS	QASXPROB	PF	Problem log problem file.
CHGQSTDB	QSYS	QPQAPRT	PRTF	Q & A printer file.
CHGSMTPA	QUSRSYS	QATMSMTP	PF	TCP/IP SMTP configuration file.
CHGTAPCTG	QUSRSYS	QATAMID	PF	Cartridge ID database file.
	QUSRSYS	QATAMID	LF	Cartridge ID logical database file.
CHGTCPA	QUSRSYS	QATOCTCPIP	PF	TCP/IP attributes file.
CHGTCPHTE	QUSRSYS	QATOCHOST	PF	TCP/IP host file.
CHGTCPIFC	QUSRSYS	QATOCIFC	PF	TCP/IP interface file.
CHGTCPRTE	QUSRSYS	QATOCRTE	PF	TCP/IP routes file.
CHGTELNA	QUSRSYS	QATMTELN	PF	TCP/IP TELNET configuration file.
CHKTAP	QSYS	QSYSTAPE	TAPF	Printer file for tape output.
CMPJRNIMG	QSYS	QPCMPIMG	PRTF	Compare journal images printer file.
CPYF	QSYS	QSYSPRT	PRTF	Copy file printer file.[1]
CPYFRMDKT	QSYS	QSYSPRT	PRTF	Copy file printer file.[1]
CPYFRMQRYF	QSYS	QSYSPRT	PRTF	Copy file printer file.[1]
CPYFRMTAP	QSYS	QSYSPRT	PRTF	Copy file printer file.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for input and output.
CPYPFRDTA	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = ASYN, BSC, BUS, CIOP, CONF, DIOP, DISK, DMPT, ECL, ETH, HDLC, IOBS, JOBS, LIOP, MIOP, POOL, RESP, RWS, SBSD, SYS, TSK, and X25.[2]
	QSYS	QSYSTAP	TAPF	Tape device file used for input and output.

Command Name	File Library	File Name	File Type	File Usage
CPYSRCF	QSYS	QSYSPRT	PRTF	Copy file printer file.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
CRTBNDC	QGPL	QCSRC	PF	C/400[R] source default input file.
CRTBNDCBL	QGPL	QCBLLESRC	PF	ILE COBOL/400[R] source default input file.
	QSYS	QSYSPRT	PF	ILE COBOL/400[R] source listing printer file.
CRTBNDCL	QGPL	QCLSRC	PF	CL source default input file.
	QSYS	QSYSPRT	PRTF	CL source listing printer file.[1]
CRTBNDRPG	QGPL	QRPGLESRC	PF	ILE RPG/400[R] source default input file
	QSYS	QSYSPRT	PRTF	ILE RPG/400[R] source listing printer file.
CRTCBLMOD	QGPL	QCBLLESRC	PF	ILE COBOL/400[R] source default input file.
	QSYS	QSYSPRT	PF	ILE COBOL/400[R] source listing printer file.
CRTCBLPGM	QGPL	QLBLSRC	PF	COBOL/400(R) source default input file.
	QSYS	QSYSPRT	PRTF	COBOL/400(R) source listing printer file.[1]
CRTCLD	QGPL	QCLDSRC	PF	C locale description default source input file.
	QSYS	QSYSPRT	PRTF	C locale description source listing printer file.[1]
CRTCLMOD	QGPL	QCLSRC	PF	CL source default input file.
	QSYS	QSYSPRT	PRTF	CL source listing printer file.[1]
CRTCLPGM	QGPL	QCLSRC	PF	CL source default input file.
	QSYS	QSYSPRT	PRTF	CL source listing printer file.[1]
CRTCMD	QGPL	QCMDSRC	PF	Command definition source default input file.
	QSYS	QSYSPRT	PRTF	Command definition source listing printer file.
CRTCMOD	QGPL	QCSRC	PF	C/400[R] source default input file.
	QSYS	QSYSPRT	PRTF	C/400[R] source listing printer file.[1]
CRTDSPF	QGPL	QDDSSRC	PF	DDS source default input file.
	QSYS	QPDDSSRC	PRTF	DDS source listing printer file.[1]
CRTICFF	QGPL	QDDSSRC	PF	DDS source default input file.
	QSYS	QPDDSSRC	PRTF	DDS source listing printer file.[1]
CRTLF	QGPL	QDDSSRC	PF	DDS source default input file.
	QSYS	QPDDSSRC	PRTF	DDS source listing printer file.[1]
CRTMNU	QGPL	QMNUSRC	PF	Default menu source input file.
	QSYS	QSYSPRT	PRTF	Menu source listing printer file.[1]
CRTMSGFMNU	QGPL	QDDSSRC	PF	DDS source created for menu by $BMENU.
	QGPL	QS36DDSSRC	PF	DDS source created for menu by $BMENU.
	QSSP	QPUTMENU	PRTF	$BMENU source listing printer file.
	QSYS	QSYSPRT	PRTF	Pascal source listing printer file.[1]
CRTPF	QGPL	QDDSSRC	PF	DDS source default input file.
	QSYS	QPDDSSRC	PRTF	DDS source listing printer file.[1]
CRTPGM	QSYS	QSYSPRT	PRTF	Program source listing printer file.[1]
CRTPNLGRP	QGPL	QPNLSRC	PF	Default panel group source input file.
	QSYS	QSYSPRT	PRTF	Panel group source listing printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
CRTPRTF	QGPL	QDDSSRC	PF	DDS source default input file.
	QSYS	QPDDSSRC	PRTF	DDS source listing printer file.[1]
CRTQSTDB	QSYS	QAQA00xxxx	LF	Q & A database model files.[2]
	QSYS	QAQA00xxxx	PF	Q & A database model files.[2]
CRTQSTLOD	QSYS	QPQAPRT	PRTF	Q & A printer file.
CRTRJEBSCF	QRJE	QRJESRC	PF	DDS source file used to create RJE BSC file.
CRTRJECFG	QRJE	QRJESRC	PF	DDS source file used to create RJE BSC or RJE communications file.
CRTRJECMNF	QRJE	QRJESRC	PF	DDS source file used to create RJE communications file.
CRTRPGMOD	QGPL	QRPGLESRC	PF	ILE RPG/400^R source default input file.
	QSYS	QSYSPRT	PRTF	ILE RPG/400^R source listing printer file.[1]
CRTRPGPGM	QGPL	QRPGSRC	PF	RPG source default input file.
	QSYS	QSYSPRT	PRTF	RPG source listing printer file.[1]
CRTRPTPGM	QGPL	QRPGSRC	PF	RPG source default input file.
	QSYS	QSYSPRT	PRTF	RPG source listing printer file.[1]
CRTSQLC	QGPL	QCSRC	PF	SQL C source file.
	QSYS	QSYSPRT	PRTF	SQL C printer file.[1]
CRTSQLCI	QGPL	QCSRC	PF	SQL C source file.
	QSYS	QSYSPRT	PRTF	SQL C printer file.[1]
CRTSQLCBL	QGPL	QLBLSRC	PF	SQL COBOL source file.
	QSYS	QSYSPRT	PRTF	SQL COBOL printer file.[1]
CRTSQLCBLI	QGPL	QCBLLESRC	PF	SQL COBOL source file.
	QSYS	QSYSPRT	PRTF	SQL COBOL printer file.[1]
CRTSQLFTN	QGPL	QFTNSRC	PF	SQL FORTRAN source file.
	QSYS	QSYSPRT	PRTF	SQL FORTRAN printer file.[1]
CRTSQLPLI	QGPL	QPLISRC	PF	SQL PLI source file.
	QSYS	QSYSPRT	PRTF	SQL PLI printer file.[1]
CRTSQLRPG	QGPL	QRPGSRC	PF	SQL RPG source file.
	QSYS	QSYSPRT	PRTF	SQL RPG printer file.[1]
CRTSQLRPGI	QGPL	QRPGLESRC	PF	SQL RPG source file.
	QSYS	QSYSPRT	PRTF	SQL RPG printer file.[1]
CRTSRVPGM	QSYS	QSYSPRT	PRTF	Service program source listing printer file.[1]
CRTS36CBL	#LIBRARY	QS36SRC	PF	S/36-compatible COBOL source default input file.
	QSYS	QSYSPRT	PRTF	S/36-compatible COBOL source listing printer file.[1]
CRTS36DSPF	QGPL	QDDSSRC	PF	DDS source created for display file by $SFGR.
	QGPL	QS36DDSSRC	PF	DDS source created for display file by $SFGR.
	QSSP	QPUTSFGR	PRTF	$SFGR source listing printer file.
CRTS36MNU	QGPL	QDDSSRC	PF	DDS source created for menu by $BMENU.

Command Name	File Library	File Name	File Type	File Usage
	QGPL	QS36DDSSRC	PF	DDS source created for menu by $BMENU.
	QSSP	QPUTMENU	PRTF	$BMENU source listing printer file.
CRTS36RPG	#LIBRARY	QS36SRC	PF	System/36™ RPG II source default input file.
	QSYS	QSYSPRT	PRTF	System/36™ RPG II source listing printer file.[1]
CRTS36RPGR	#LIBRARY	QS36SRC	PF	System/36™ RPG II source default input file.
CRTS36RPT	#LIBRARY	QS36SRC	PF	System/36™ RPG II Auto Report source default input file.
	QSYS	QSYSPRT	PRTF	System/36™ RPG II Auto Report source listing printer file.[1]
CRTTAPCGY	QUSRSYS	QATACGY	PF	Library device database file.
	QUSRSYS	QLTACGY	LF	Library device logical database file.
CRTTBL	QGPL	QTBLSRC	PF	Table source default input file.
CVTPFRDTA	QPFR	QACPxxxx	PF	QACPxxxx performance data collection files, where xxxx = CNFG, GPHF, PROF, and RESP.
	QPFR	QAITMON	PF	Performance data collection file: job and Licensed Internal Code task data.
	QSYS	QAPMyyyy	PF	QAPMyyyy performance data collection files, where yyyy = CIOP, CONF, DIOP, DISK, JOBS, LIOP, POOL, RESP, RWS, and SYS.[2]
	QSYS	QAPMzzzz	PF	QAPMzzzz performance data collection files, where zzzz = ASYN, BSC, BUS, DMPT, ECL, ETH, HDLC, MIOP, and X25.[2]
	QPFR	QAPTAPGP	PF	Performance data collection file for functional area data.
CVTRPGSRC	user-lib	QRPGSRC	PF	RPG/400® source file (from file).
	user-lib	QRPGLESRC	PF	ILE RPG/400® source file (to file).
	QSYS	QSYSPRT	PRTF	ILE RPG/400® listing printer file.
	user-lib	QRNCVTLG	PF	ILE RPG/400® conversion log file.
DLTALR	QUSRSYS	QAALERT	PF	Alert database file.
DLTPFRDTA	QPFR	QAPGSUMD	PF	Performance data collection file for graphics data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = ASYN, BSC, BUS, CIOP, CONF, DIOP, DISK, DMPT, ECL, ETH, HDLC, IOBS, JOBS, LIOP, MIOP, POOL, RESP, RWS, SBSD, SYS, TSK, and X25.[2]
	QPFR	QAPTLCKD	PF	Performance data collection file: lock and seizure conflict data.
	QPFR	QTRxxxx	PF	QTRxxxx performance data collection files, where xxxxx = IDX, JOBT, JSUM, SLWT, and TSUM.
DLTPRB Command: All 8 of the QASXxxxx files for the DLTPRB command are the same subset of files that are shown in the QUSRSYS library for the DSPPRB command. For the descriptions of these files, see the DSPPRB command.				
DLTPRB	QUSRSYS	QASXxxxx	PF	See the DSPPRB command.
DLTQST	QSYS	QPQAPRT	PRTF	Q & A printer file.

Command Name	File Library	File Name	File Type	File Usage
DLTQSTDB	QSYS	QPQAPRT	PRTF	Q & A printer file.
DLTTAPCGY	QUSRSYS	QATACGY	PF	Library device database file.
	QUSRSYS	QLTACGY	LF	Library device logical database file.
DMPCLPGM	QSYS	QPPGMDMP	PRTF	CL program dump printer file.
DMPJOB	QSYS	QPSRVDMP	PRTF	Service dump printer file.
DMPOBJ	QSYS	QPSRVDMP	PRTF	Service dump printer file.
DMPSYSOBJ	QSYS	QPSRVDMP	PRTF	Service dump printer file.
DMPTAP	QSYS	QPTAPDMP	PRTF	Tape dump printer file.
DMPTRC	QSYS	QAPMDMPT	PF	Performance trace file.
	QSYS	QSYSPRT	PRTF	SDA source printer file.
DSPACC	QSYS	QSYSPRT	PRTF	Display access codes printer file.[1]
DSPACCAUT	QSYS	QSYSPRT	PRTF	Display access code authority printer file.[1]
DSPACCGRP	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QAPTPAGD	PF	Performance data collection file: data about files and programs in a process access group (PAG).
	QPFR	QPPTPAG	PRTF	Display process access group printer file.[1]
	QPFR	QPPTPAGD	PRTF	Performance printer file containing files and programs in a process access group (PAG).
DSPACTPJ	QSYS	QSYSPRT	PRTF	Display active prestart jobs printer file.[1]
DSPAPPNINF	QUSRSYS	QALSxxx	PF	Set of 4 QALSxxx model database files that contain the record formats used for storing APPN information, where xxx = DIR, END, INM, and TDB.[2]
	QSYS	QSYSPRT	PRTF	Display APPN information printer file.[1]
DSPAUTHLR	QSYS	QADSHLR	PF	Model database file that contains the record format for the authority holder object entries.[2]
	QSYS	QPSYDSHL	PRTF	Display authority holders printer file.[1]
DSPAUTL	QSYS	QAOBJAUT	PF	Model database file that contains the record format for the authorization list entries.[2]
	QSYS	QPOBJAUT	PRTF	Authorization list entries printer file.[1]
DSPAUTLDLO	QSYS	QSYSPRT	PRTF	Authorization list printer file.[1]
DSPAUTLOBJ	QSYS	QADALO	PF	Model database file that contains the record format for the authorization list object entries.[2]
	QSYS	QPSYDALO	PRTF	Display authorization list objects printer file.[1]
DSPAUTUSR	QSYS	QPAUTUSR	PRTF	Authorized users printer file.[1]
DSPBCKSTS	QSYS	QSYSPRT	PRTF	Display backup status printer file.[1]
DSPBCKUP	QSYS	QSYSPRT	PRTF	Display backup options printer file.[1]
DSPBCKUPL	QSYS	QSYSPRT	PRTF	Display backup list printer file.[1]
DSPBKP	QSYS	QPDBGDSP	PRTF	Breakpoint (debug mode) printer file.[1]
DSPBNDDIR	QSYS	QABNDBND	PF	Model database file that contains the record format for the binding directory entries.[2]
	QSYS	QSYSPRT	PRTF	Displays the contents of the binding directory printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
DSPCFGL	QSYS	QPDCCFGL	PRTF	Configuration list printer file.[1]
DSPCHT	QSYS	QPGDDM	PRTF	BGU-defined chart output printer file.[1]
DSPCLS	QSYS	QPDSPCLS	PRTF	Class printer file.[1]
DSPCMD	QSYS	QPCMD	PRTF	Command values printer file.[1]
DSPCNNL	QSYS	QPDCCNNL	PRTF	Connection list printer file.[1]
DSPCNNSTS	QSYS	QSYSPRT	PRTF	Connection status printer file.[1]
DSPCOSD	QSYS	QPDCCOS	PRTF	Class-of-service description printer file.[1]
DSPCSI	QSYS	QSYSPRT	PRTF	Communications side information printer file.[1]
DSPCTLD	QSYS	QPDCCTL	PRTF	Controller description printer file.[1]
DSPDBG	QSYS	QPDBGDSP	PRTF	Debug display (debug mode) printer file.[1]
DSPDBR	QSYS	QADSPDBR	PF	Model database file that defines the record format of the file created to store information about database file relationships.[2]
	QSYS	QPDSPDBR	PRTF	Printer file that contains information about database file relationships.[1]
DSPDDMF	QSYS	QPDSPDDM	PRTF	Distributed data management (DDM) file listing printer file.[1]
DSPDEVD	QSYS	QPDCDEV	PRTF	Device description printer file.[1]
	QSYS	QAOSDIRO	PF	Display directory output file: OUTFILFMT(*TYPE1)
	QSYS	QAOSDIRB	PF	Display directory output file: OUTFILFMT(*TYPE2) DETAIL(*BASIC)
	QSYS	QAOSDIRF	PF	Display directory output file: OUTFILFMT(*TYPE2) DETAIL(*FULL)
	QSYS	QAOSDIRX	PF	Display directory output file: OUTFILFMT(*TYPE3) DETAIL(*FULL)
	QSYS	QPDSPDDL	PRTF	Printer file for *full* details of displayed directory entries.[1]
	QSYS	QPDSPDSM	PRTF	Printer file for *basic* details of displayed directory entries.[1]
DSPDLOAUT	QSYS	QSYSPRT	PRTF	Display document library object authority printer file.[1]
DSPDLONAM	QSYS	QSYSPRT	PRTF	Display document library object printer file.[1]
DSPDSTL	QSYS	QAOSDSTO	PF	Distribution list output file.
	QSYS	QPDSPLDL	PRTF	Distribution list *details* printer file.[1]
	QSYS	QPDSPLSM	PRTF	Distribution list *summary* printer file.[1]
DSPDSTCLGE	QSVMSS	QACQFVOF	PF	Output file model for MSS/400 Display Distribution Catalog Entries command.[2]
DSPDSTLOG	QSYS	QPDSTDLG	PRTF	Display distribution log printer file.[1]
DSPDSTSRV	QSYS	QPDSTSRV	PRTF	Distribution services printer file.[1]
	QUSRSYS	QASNADSA	PF	SNADS secondary node ID table.
	QUSRSYS	QASNADSQ	PF	SNADS destination queues table.
	QUSRSYS	QASNADSR	PF	Routing table database file.
DSPDTA	QIDU	QDTAPRT	PRTF	DFU audit control printer file.

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QPDZDTALOG	PRTF	DFU run-time audit log.
	QSYS	QPDZDTAPRT	PRTF	DFU run-time printer data file.
DSPDTAARA	QSYS	QPDSPDTA	PRTF	Data area printer file.[1]
DSPDTADCT	QSYS	QPDSPFFD	PRTF	Data dictionary printer file.[1]
DSPEDTD	QSYS	QPDCEDSP	PRTF	Edit description printer file.[1]

Files Used by CL Commands (Part 2)

Command Name	File Library	File Name	File Type	File Usage
DSPFD Command: For the DSPFD command, all of the following entries having a file type of PF are physical files (model files that are not actual output files) that define the record formats of created files used to store a specific type of information about a type (or group) of files. (See the DSPFD command for descriptions of the TYPE and FILEATR parameters. These parameters identify all the values that cause these files to be used.) Therefore, the common part of each model file's description *begins* here and the unique part of each description *continues* under the **File Usage** column: **Model database file that defines the record format of the file created to store...**				
DSPFD	QSYS	QAFDACCP	PF	...access path file information.[2]
	QSYS	QAFDBASI	PF	...basic file information common to all files.[2]
	QSYS	QAFDBSC	PF	...BSC file and mixed file device attribute information.[2]
	QSYS	QAFDCMN	PF	...communications file and mixed file device attribute information.[2]
	QSYS	QAFDCSEQ	PF	... collating sequence information.[2]
	QSYS	QAFDCST	PF	... constraint relationship information.[2]
	QSYS	QAFDDDM	PF	...distributed data management (DDM) file attribute information.[2]
	QSYS	QAFDDSP	PF	...display file and mixed file display device attribute information.[2]
	QSYS	QAFDICF	PF	...ICF file attribute information.[2]
	QSYS	QAFDJOIN	PF	...join logical file information.[2]
	QSYS	QAFDLGL	PF	...logical file attribute information.[2]

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QAFDMBR	PF	...database member information.[2]
	QSYS	QAFDMBRL	PF	...database member list information.[2]
	QSYS	QAFDNGP	PF	... node group information.[2]
	QSYS	QAFDPHY	PF	...physical file attribute information.[2]
	QSYS	QAFDPRT	PF	...printer file attribute information.[2]
	QSYS	QAFDRFMT	PF	...record format information.[2]
	QSYS	QAFDSAV	PF	...save file information.[2]
	QSYS	QAFDSELO	PF	...select/omit information.[2]
	QSYS	QAFDSPOL	PF	...device file spooled information.[2]
	QSYS	QAFDTAP	PF	...tape file attribute information.[2]
	QSYS	QAFDTRG	PF	... trigger information.[2]
	QSYS	QPDSPFD	PRTF	File description printer file.[1]
DSPFFD	QSYS	QADSPFFD	PF	Model database file that defines the record format of the file created to store file field descriptions.[2]
	QSYS	QPDSPFFD	PRTF	File field description printer file.[1]
DSPFLR	QSYS	QADSPDOC	PF	Document list output database file.
	QSYS	QADSPFLR	PF	Folder list output database file.
	QSYS	QPDSPFLR	PRTF	Display folder printer file.[1]
DSPFNTRSCA	QSYS	QPDSPFNT	PRTF	Font resource attributes printer file.[1]
DSPGDF	QSYS	QPGDDM	PRTF	BGU-defined graphics data printer file.[1]
DSPGNDDIR	QSYS	QABNDBND	PF	System-supplied output file.[1]

Command Name	File Library	File Name	File Type	File Usage
DSPHDWRSC Command: For the DSPHDWRSC command, all of the following entries having a file type of PF are physical files (model files that are not actual output files) that define the record formats of created files used to store a specific type of hardware resource information. Therefore, the common part of each model file's description *begins* here and the unique part of each description *continues* under the **File Usage** column: **Model database file that defines the record format of the file created to store information about...**				
DSPHDWRSC	QSYS	QARZDCMN	PF	...communications resources.[2]
	QSYS	QARZDLWS	PF	...local work station resources.[2]
	QSYS	QARZDPRC	PF	...processor resources.[2]
	QSYS	QARZDTRA	PF	...token-ring local area network (TRLAN) adapter resources.[2]
	QSYS	QARZDSTG	PF	...storage device resources.[2]
	QSYS	QSYSPRT	PRTF	Hardware resources printer file.[1]
DSPHFS	QSYS	QSYSPRT	PRTF	Display hierarchical file systems printer file.[1]
DSPHSTGPH	QPFR	QPPGGPH	PRTF	Display historical graph printer file.[1]
DSPIDXSTS	QUSRSYS	QABBADMTB	PF	OfficeVision(R) text search services administration table.
DSPIGCDCT	QSYS	QPDSPDCT	PRTF	DBCS printer file.[1]
DSPJOB	QSYS	QPDSPJOB	PRTF	Display job printer file.[1]
DSPJOBD	QSYS	QPRTJOBD	PRTF	Job description printer file.[1]
DSPJOBLOG	QSYS	QPJOBLOG	PRTF	Job log printer file.[1]
	QSYS	QPJOBLOGO	PRTF	Job log printer file for jobs prior to Version 2 Release 3 of the AS/400(R).[1]
	QSYS	QAMHJLPR	PF	Primary job log model file.[2]
	QSYS	QAMHJLSC	PF	Secondary job log model file.[2]

Command Name	File Library	File Name	File Type	File Usage
DSPJRN Command: All of the following files for the DSPJRN command having a file type of PF are physical files (model files, not actual output files) that define the record formats of files created to store a group of journal entries retrieved and converted from journal receivers. The retrieved group of entries can be a specific type of information or all types of information that was journaled. Each created file stores the retrieved journal entries after they are converted into one of four basic formats (*TYPE1, *TYPE2, *TYPE3 or *TYPE4) or into the format defined for the specific type of data being retrieved. • *TYPE1: the basic file format described in the Journal Management topic. • *TYPE2: all of *TYPE1 plus the user profile field. • *TYPE3: all of *TYPE2 plus the null value indicators. • *TYPE4: all of *TYPE3 plus the JID, referential integrity, and trigger information. • » *TYPE5: all of *TYPE4 plus more information. « • Type-dependent format - the format associated with the specific type (as described below for the fourth and following files) of information being retrieved. For example, the model file QASY**AF**JE has a unique format for storing all retrieved journal entries related to authority failures (**AF**) on the system. For the DSPJRN PF files listed below, the common part of all the model files' descriptions begins here and the unique part of each file's description continues under the **File Usage** column: **Model database file that defines the record format of the file created to contain retrieved and converted journal entries related to...**				
DSPJRN	QSYS	QADSPJRN	PF	...a specific type (or all types) of information; stored in the *TYPE1 format.[2]
	QSYS	QADSPJR2	PF	...a specific type (or all types) of information; stored in the *TYPE2 format.[2]
	QSYS	QADSPJR3	PF	...a specific type (or all types) of information; stored in the *TYPE3 format.[2]
	QSYS	QADSPJR4	PF	...a specific type (or all types) of information; stored in the *TYPE4 format.[2]
»	QSYS	QADSPJR5	PF	...a specific type (or all types) of information; stored in the *TYPE5 format.[2] «
	QSYS	QADXERLG	PF	...DSNX logged errors.[2]
»	QSYS	QADXERL4	PF	...DSNX logged errors; stored in the *TYPE4 format. [2] «
	QSYS	QADXJRNL	PF	...DSNX logged data.[2]
»	QSYS	QADXJRN4	PF	...DSNX logged data; stored in the *TYPE4 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QAJBACG	PF	...job accounting.[2]
»	QSYS	QAJBACG4	PF	...job accounting; stored in the *TYPE4 format.[2] «
	QSYS	QALZALK	PF	...invalid license keys.[2]
»	QSYS	QALZALK4	PF	...invalid license keys; stored in the *TYPE4 format.[2] «
	QSYS	QALZALL	PF	...usage limit increases.[2]
»	QSYS	QALZALL4	PF	...usage limit increases; stored in the *TYPE4 format.[2] «
	QSYS	QALZALU	PF	...licensed users that exceed usage limits.[2]
»	QSYS	QALZALU4	PF	...licensed users that exceed usage limits; stored in the *TYPE4 format.[2] «
	QSYS	QAPTACG	PF	...print job accounting.[2]
»	QSYS	QAPTACG4	PF	...print job accounting; stored in the *TYPE4 format.[2] «
»	QSYS	QAPTACG5	PF	...print job accounting; stored in the *TYPE5 format.[2] «
	QSYS	QASYADJE	PF	...changes to auditing attributes.[2]
»	QSYS	QASYADJ4	PF	...changes to auditing attributes; stored in the *TYPE4 format.[2] «
»	QSYS	QASYADJ5	PF	...changes to auditing attributes; stored in the *TYPE5 format.[2] «
	QSYS	QASYAFJE	PF	...authority failures.[2]
»	QSYS	QASYAFJ4	PF	...authority failures; stored in the *TYPE4 format.[2] «
»	QSYS	QASYAFJ5	PF	...authority failures; stored in the *TYPE5 format.[2] «
	QSYS	QASYAPJE	PF	...use of adopted authority.[2]

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYAPJ4	PF	...use of adopted authority; stored in the *TYPE4 format.[2] «
»	QSYS	QASYAPJ5	PF	...use of adopted authority; stored in the *TYPE5 format.[2] «
»	QSYS	QASYAUJ5	PF	...security attribute changed; stored in the *TYPE5 format.[2] «
	QSYS	QASYCAJE	PF	...changes to object authority (authorization list or object).[2]
»	QSYS	QASYCAJ4	PF	...changes to object authority (authorization list or object); stored in the *TYPE4 format.[2] «
»	QSYS	QASYCAJ5	PF	...changes to object authority (authorization list or object); stored in the *TYPE5 format.[2] «
	QSYS	QASYCDJE	PF	...command strings.[2]
»	QSYS	QASYCDJ4	PF	...command strings; stored in the *TYPE4 format.[2] «
»	QSYS	QASYCDJ5	PF	...command strings; stored in the *TYPE5 format.[2] «
	QSYS	QASYCOJE	PF	...objects created on the system.[2]
»	QSYS	QASYCOJ4	PF	...objects created on the system; stored in the *TYPE4 format.[2] «
»	QSYS	QASYCOJ5	PF	...objects created on the system; stored in the *TYPE5 format.[2] «
	QSYS	QASYCPJE	PF	...create, change, and restore user profile operations.[2]
»	QSYS	QASYCPJ4	PF	...create, change, and restore user profile operations; stored in the *TYPE4 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYCPJ5	PF	...create, change, and restore user profile operations; stored in the *TYPE5 format.[2] «
	QSYS	QASYCQJE	PF	...changes to *CRQD object.[2]
»	QSYS	QASYCQJ4	PF	...changes to *CRQD object; stored in the *TYPE4 format.[2] «
»	QSYS	QASYCQJ5	PF	...changes to *CRQD object; stored in the *TYPE5 format.[2] «
»	QSYS	QASYCUJ4	PF	...cluster operation; stored in the *TYPE4 format.[2] «
»	QSYS	QASYCUJ5	PF	...cluster operation; stored in the *TYPE5 format.[2] «
»	QSYS	QASYCVJ4	PF	...connection verification; stored in the *TYPE4 format.[2] «
»	QSYS	QASYCVJ5	PF	...connection verification; stored in the *TYPE5 format.[2] «
»	QSYS	QASYCYJ4	PF	...Cryptographic configuration; stored in the *TYPE4 format.[2] «
»	QSYS	QASYCYJ5	PF	...Cryptographic configuration; stored in the *TYPE5 format.[2] «
»	QSYS	QASYDIJ4	PF	...Directory services; stored in the *TYPE4 format.[2] «
»	QSYS	QASYDIJ5	PF	...Directory services; stored in the *TYPE5 format.[2] «
	QSYS	QASYDOJE	PF	...objects deleted from the system.[2]
»	QSYS	QASYDOJ4	PF	...objects deleted from the system; stored in the *TYPE4 format.[2] «
»	QSYS	QASYDOJ5	PF	...objects deleted from the system; stored in the *TYPE5 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QASYDSJE	PF	...resetting the DST security officer password.[2]
»	QSYS	QASYDSJ4	PF	...resetting the DST security officer password; stored in the *TYPE4 format.[2] «
»	QSYS	QASYDSJ5	PF	...resetting the DST security officer password; stored in the *TYPE5 format.[2] «
»	QSYS	QASYEVJ4	PF	...Environment variable; stored in the *TYPE4 format.[2] «
»	QSYS	QASYEVJ5	PF	...Environment variable; stored in the *TYPE5 format.[2] «
»	QSYS	QASYGRJ4	PF	...General purpose audit record; stored in the *TYPE4 format.[2] «
»	QSYS	QASYGRJ5	PF	...General purpose audit record; stored in the *TYPE5 format.[2] «
	QSYS	QASYGSJE	PF	...gives of descriptors.[2]
»	QSYS	QASYGSJ4	PF	...gives of descriptors; stored in the *TYPE4 format.[2] «
»	QSYS	QASYGSJ5	PF	...gives of descriptors; stored in the *TYPE5 format.[2] «
	QSYS	QASYIPJE	PF	...interprocess communications.[2]
»	QSYS	QASYIPJ4	PF	...interprocess communications; stored in the *TYPE4 format.[2] «
»	QSYS	QASYIPJ5	PF	...interprocess communications; stored in the *TYPE5 format.[2] «
»	QSYS	QASYIRJ4	PF	...IP rules actions; stored in the *TYPE4 format.[2] «
»	QSYS	QASYIRJ5	PF	...IP rules actions; stored in the *TYPE5 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYISJ5	PF	...Internet security management; stored in the *TYPE5 format.[2] «
	QSYS	QASYJDJE	PF	...changes to the USER parameter of job descriptions.[2]
»	QSYS	QASYJDJ4	PF	...changes to the USER parameter of job descriptions; stored in the *TYPE4 format.[2] «
»	QSYS	QASYJDJ5	PF	...changes to the USER parameter of job descriptions; stored in the *TYPE5 format.[2] «
	QSYS	QASYJSJE	PF	...job changes.[2]
»	QSYS	QASYJSJ4	PF	...job changes; stored in the *TYPE4 format.[2] «
»	QSYS	QASYJSJ5	PF	...job changes; stored in the *TYPE5 format.[2] «
»	QSYS	QASYKFJ4	PF	...Key ring file; stored in the *TYPE4 format.[2] «
»	QSYS	QASYKFJ5	PF	...Key ring file; stored in the *TYPE5 format.[2] «
	QSYS	QASYLDJE	PF	...link/unlink/lookup directory.[2]
»	QSYS	QASYLDJ4	PF	...link/unlink/lookup directory; stored in the *TYPE4 format.[2] «
»	QSYS	QASYLDJ5	PF	...link/unlink/lookup directory; stored in the *TYPE5 format.[2] «
	QSYS	QASYMLJE	PF	...mail actions.[2]
»	QSYS	QASYMLJ4	PF	...mail actions; stored in the *TYPE4 format.[2] «
»	QSYS	QASYMLJ5	PF	...mail actions; stored in the *TYPE5 format.[2] «
	QSYS	QASYNAJE	PF	...changes to network attributes.[2]

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYNAJ4	PF	...changes to network attributes; stored in the *TYPE4 format.[2] «
»	QSYS	QASYNAJ5	PF	...changes to network attributes; stored in the *TYPE5 format.[2] «
	QSYS	QASYNDJE	PF	...directory search violations.[2]
»	QSYS	QASYNDJ4	PF	...directory search violations; stored in the *TYPE4 format.[2] «
»	QSYS	QASYNDJ5	PF	...directory search violations; stored in the *TYPE5 format.[2] «
	QSYS	QASYNEJE	PF	...end point violations.[2]
»	QSYS	QASYNEJ4	PF	...end point violations; stored in the *TYPE4 format.[2] «
»	QSYS	QASYNEJ5	PF	...end point violations; stored in the *TYPE5 format.[2] «
	QSYS	QASYOMJE	PF	...object move and rename operations.[2]
»	QSYS	QASYOMJ4	PF	...object move and rename operations; stored in the *TYPE4 format.[2] «
»	QSYS	QASYOMJ5	PF	...object move and rename operations; stored in the *TYPE5 format.[2] «
	QSYS	QASYORJE	PF	...object restore operations.[2]
»	QSYS	QASYORJ4	PF	...object restore operations; stored in the *TYPE4 format.[2] «
»	QSYS	QASYORJ5	PF	...object restore operations; stored in the *TYPE5 format.[2] «
	QSYS	QASYOWJE	PF	...changes to object ownership.[2]

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYOWJ4	PF	...changes to object ownership; stored in the *TYPE4 format.[2] «
»	QSYS	QASYOWJ5	PF	...changes to object ownership; stored in the *TYPE5 format.[2] «
	QSYS	QASYO1JE	PF	...single optical object accesses.[2]
»	QSYS	QASYO1J4	PF	...single optical object accesses; stored in the *TYPE4 format.[2] «
»	QSYS	QASYO1J5	PF	...single optical object accesses; stored in the *TYPE5 format.[2] «
	QSYS	QASYO2JE	PF	...dual optical object accesses.[2]
»	QSYS	QASYO2J4	PF	...dual optical object accesses; stored in the *TYPE4 format.[2] «
»	QSYS	QASYO2J5	PF	...dual optical object accesses; stored in the *TYPE5 format.[2] «
	QSYS	QASYO3JE	PF	...optical volume accesses.[2]
»	QSYS	QASYO3J4	PF	...optical volume accesses; stored in the *TYPE4 format.[2] «
»	QSYS	QASYO3J5	PF	...optical volume accesses; stored in the *TYPE5 format.[2] «
	QSYS	QASYPAJE	PF	...changes to programs (CHGPGM) that will now adopt the owner's authority.[2]
»	QSYS	QASYPAJ4	PF	...changes to programs (CHGPGM) that will now adopt the owner's authority; stored in the *TYPE4 format.[2] «
»	QSYS	QASYPAJ5	PF	...changes to programs (CHGPGM) that will now adopt the owner's authority; stored in the *TYPE5 format.[2] «
	QSYS	QASYPGJE	PF	...changes to object primary group.[2]

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QASYPGJ4	PF	...changes to object primary group; stored in the *TYPE4 format.[2]
	QSYS	QASYPGJ5	PF	...changes to object primary group; stored in the *TYPE5 format.[2]
	QSYS	QASYPOJE	PF	...printer output actions.[2]
	QSYS	QASYPOJ4	PF	...printer output actions; stored in the *TYPE4 format.[2]
	QSYS	QASYPOJ5	PF	...printer output actions; stored in the *TYPE5 format.[2]
	QSYS	QASYPSJE	PF	...profile swapping.[2]
	QSYS	QASYPSJ4	PF	...profile swapping; stored in the *TYPE4 format.[2]
	QSYS	QASYPSJ5	PF	...profile swapping; stored in the *TYPE5 format.[2]
	QSYS	QASYPWJE	PF	...attempted usage of invalid passwords or user profile names.[2]
	QSYS	QASYPWJ4	PF	...attempted usage of invalid passwords or user profile names; stored in the *TYPE4 format.[2]
	QSYS	QASYPWJ5	PF	...attempted usage of invalid passwords or user profile names; stored in the *TYPE5 format.[2]
	QSYS	QASYRAJE	PF	...restore of objects when authority changes.[2]
	QSYS	QASYRAJ4	PF	...restore of objects when authority changes; stored in the *TYPE4 format.[2]
	QSYS	QASYRAJ5	PF	...restore of objects when authority changes; stored in the *TYPE5 format.[2]
	QSYS	QASYRJJE	PF	...restore of job descriptions that contain user profile names.[2]

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYRJJ4	PF	...restore of job descriptions that contain user profile names; stored in the *TYPE4 format.[2] «
»	QSYS	QASYRJJ5	PF	...restore of job descriptions that contain user profile names; stored in the *TYPE5 format.[2] «
	QSYS	QASYROJE	PF	...restore of objects when ownership was changed to QDFTOWN.[2]
»	QSYS	QASYROJ4	PF	...restore of objects when ownership was changed to QDFTOWN; stored in the *TYPE4 format.[2] «
»	QSYS	QASYROJ5	PF	...restore of objects when ownership was changed to QDFTOWN; stored in the *TYPE5 format.[2] «
	QSYS	QASYRPJE	PF	...restore of programs that adopt their owner's authority.[2]
»	QSYS	QASYRPJ4	PF	...restore of programs that adopt their owner's authority; stored in the *TYPE4 format.[2] «
»	QSYS	QASYRPJ5	PF	...restore of programs that adopt their owner's authority; stored in the *TYPE5 format.[2] «
	QSYS	QASYRQJE	PF	...restores of *CRQD object.[2]
»	QSYS	QASYRQJ4	PF	...restores of *CRQD object; stored in the *TYPE4 format.[2] «
»	QSYS	QASYRQJ5	PF	...restores of *CRQD object; stored in the *TYPE5 format.[2] «
	QSYS	QASYRUJE	PF	...operations restoring authority for user profiles, using the RSTAUT command.[2]

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYRUJ4	PF	...operations restoring authority for user profiles, using the RSTAUT command; stored in the *TYPE4 format.[2] «
»	QSYS	QASYRUJ5	PF	...operations restoring authority for user profiles, using the RSTAUT command; stored in the *TYPE5 format.[2] «
	QSYS	QASYRZJE	PF	...changes on primary group restores.[2]
»	QSYS	QASYRZJ4	PF	...changes on primary group restores; stored in the *TYPE4 format.[2] «
»	QSYS	QASYRZJ5	PF	...changes on primary group restores; stored in the *TYPE5 format.[2] «
	QSYS	QASYSDJE	PF	...changes to the system distribution directory.[2]
»	QSYS	QASYSDJ4	PF	...changes to the system distribution directory; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSDJ5	PF	...changes to the system distribution directory; stored in the *TYPE5 format.[2] «
	QSYS	QASYSEJE	PF	...changes to subsystem routings.[2]
»	QSYS	QASYSEJ4	PF	...changes to subsystem routings; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSEJ5	PF	...changes to subsystem routings; stored in the *TYPE5 format.[2] «
	QSYS	QASYSFJE	PF	...actions with spooled files.[2]
»	QSYS	QASYSFJ4	PF	...actions with spooled files; stored in the *TYPE4 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYSFJ5	PF	...actions with spooled files; stored in the *TYPE5 format.[2] «
»	QSYS	QASYSGJ4	PF	...Asynchronous signals; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSGJ5	PF	...Asynchronous signals; stored in the *TYPE5 format.[2] «
»	QSYS	QASYSKJ4	PF	...secure sockets connections; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSKJ5	PF	...secure sockets connections; stored in the *TYPE5 format.[2] «
	QSYS	QASYSMJE	PF	...system management changes.[2]
»	QSYS	QASYSMJ4	PF	...system management changes; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSMJ5	PF	...system management changes; stored in the *TYPE5 format.[2] «
	QSYS	QASYSOJE	PF	...server security changes.[2]
»	QSYS	QASYSOJ4	PF	...server security changes; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSOJ5	PF	...server security changes; stored in the *TYPE5 format.[2] «
	QSYS	QASYSTJE	PF	...use of system service tools.[2]
»	QSYS	QASYSTJ4	PF	...use of system service tools; stored in the *TYPE4 format.[2] «
»	QSYS	QASYSTJ5	PF	...use of system service tools; stored in the *TYPE5 format.[2] «
	QSYS	QASYSVJE	PF	...changes to system values.[2]
»	QSYS	QASYSVJ4	PF	...changes to system values; stored in the *TYPE4 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYSVJ5	PF	...changes to system values; stored in the *TYPE5 format.[2] «
	QSYS	QASYVAJE	PF	...changes to access control list.[2]
»	QSYS	QASYVAJ4	PF	...changes to access control list; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVAJ5	PF	...changes to access control list; stored in the *TYPE5 format.[2] «
	QSYS	QASYVCJE	PF	...connection starts and ends.[2]
»	QSYS	QASYVCJ4	PF	...connection starts and ends; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVCJ5	PF	...connection starts and ends; stored in the *TYPE5 format.[2] «
	QSYS	QASYVFJE	PF	...closes of server files.[2]
»	QSYS	QASYVFJ4	PF	...closes of server files; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVFJ5	PF	...closes of server files; stored in the *TYPE5 format.[2] «
	QSYS	QASYVLJE	PF	...exceeding account limit.[2]
»	QSYS	QASYVLJ4	PF	...exceeding account limit; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVLJ5	PF	...exceeding account limit; stored in the *TYPE5 format.[2] «
	QSYS	QASYVNJE	PF	...network logons and logoffs.[2]
»	QSYS	QASYVNJ4	PF	...network logons and logoffs; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVNJ5	PF	...network logons and logoffs; stored in the *TYPE5 format.[2] «
»	QSYS	QASYVOJ4	PF	...actions on validation lists; stored in the *TYPE4 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYVOJ5	PF	...actions on validation lists; stored in the *TYPE5 format.[2] «
	QSYS	QASYVPJE	PF	...network password errors.[2]
»	QSYS	QASYVPJ4	PF	...network password errors; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVPJ5	PF	...network password errors; stored in the *TYPE5 format.[2] «
	QSYS	QASYVRJE	PF	...accesses to network resources.[2]
»	QSYS	QASYVRJ4	PF	...accesses to network resources; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVRJ5	PF	...accesses to network resources; stored in the *TYPE5 format.[2] «
	QSYS	QASYVSJE	PF	...server session starts and ends.[2]
»	QSYS	QASYVSJ4	PF	...server session starts and ends; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVSJ5	PF	...server session starts and ends; stored in the *TYPE5 format.[2] «
	QSYS	QASYVUJE	PF	...changes to network profiles.[2]
»	QSYS	QASYVUJ4	PF	...changes to network profiles; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVUJ5	PF	...changes to network profiles; stored in the *TYPE5 format.[2] «
	QSYS	QASYVVJE	PF	...changes to service status.[2]
»	QSYS	QASYVVJ4	PF	...changes to service status; stored in the *TYPE4 format.[2] «
»	QSYS	QASYVVJ5	PF	...changes to service status; stored in the *TYPE5 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QASYXOJ4	PF	...network authentication; stored in the *TYPE4 format.[2] «
»	QSYS	QASYXOJ5	PF	...network authentication; stored in the *TYPE5 format.[2] «
	QSYS	QASYYCJE	PF	...changes to document library objects.[2]
»	QSYS	QASYYCJ4	PF	...changes to document library objects; stored in the *TYPE4 format.[2] «
»	QSYS	QASYYCJ5	PF	...changes to document library objects; stored in the *TYPE5 format.[2] «
	QSYS	QASYYRJE	PF	...read operations of document library objects.[2]
»	QSYS	QASYYRJ4	PF	...read operations of document library objects; stored in the *TYPE4 format.[2] «
»	QSYS	QASYYRJ5	PF	...read operations of document library objects; stored in the *TYPE5 format.[2] «
	QSYS	QASYZCJE	PF	...changes to objects.[2]
»	QSYS	QASYZCJ4	PF	...changes to objects; stored in the *TYPE4 format.[2] «
»	QSYS	QASYZCJ5	PF	...changes to objects; stored in the *TYPE5 format.[2] «
	QSYS	QASYZMJE	PF	...object method access.[2]
»	QSYS	QASYZMJ4	PF	...object method access; stored in the *TYPE4 format.[2] «
	QSYS	QASYZRJE	PF	...read operations of objects.[2]
»	QSYS	QASYZRJ4	PF	...read operations of objects; stored in the *TYPE4 format.[2] «
»	QSYS	QASYZRJ5	PF	...read operations of objects; stored in the *TYPE5 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
»	QSYS	QATOFIPF	PF	...IP filter rule actions.[2] «
»	QSYS	QATOFNAT	PF	...IP NAT rule actions.[2] «
»	QSYS	QATOQQOS	PF	...modification of QoS policies.[2] «
	QSYS	QATOSLOG	PF	...SNMP log entries.[2]
»	QSYS	QATOSLOG	PF	...SNMP log entries; stored in the *TYPE4 format.[2] «
»	QSYS	QATOVSOF	PF	...VPN information.[2] «
	QSYS	QAWCTPJE	PF	...performance tuning.[2]
»	QSYS	QAWCTPJ4	PF	...performance tuning; stored in the *TYPE4 format.[2] «
»	QSYS	QAZDALLG	PF	...SNADS alert logging.[2] «
	QSYS	QAZDCFLG	PF	...configuration changes to SNADS distribution queues table.[2]
»	QSYS	QAZDCFL4	PF	...configuration changes to SNADS distribution queues table; stored in the *TYPE4 format.[2] «
	QSYS	QAZDERLG	PF	...SNADS logged errors.[2]
»	QSYS	QAZDERL4	PF	...SNADS logged errors; stored in the *TYPE4 format.[2] «
	QSYS	QAZDJRNL	PF	...SNADS logged data.[2]
»	QSYS	QAZDJRN4	PF	...SNADS logged data; stored in the *TYPE4 format.[2] «
	QSYS	QAZDRTLG	PF	...changes to SNADS routing and secondary system name tables.[2]
»	QSYS	QAZDRTL4	PF	...changes to SNADS routing and secondary system name tables; stored in the *TYPE4 format.[2] «

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QAZDSYLG	PF	...SNADS miscellaneous logged system-level occurrences.[2]
»	QSYS	QAZDSYL4	PF	...SNADS miscellaneous logged system-level occurrences; stored in the *TYPE4 format.[2] «
	QSYS	QAZMFCF	PF	...Mail server framework (MSF) configuration changes logging.[2]
»	QSYS	QAZMFCF4	PF	...Mail server framework (MSF) configuration changes logging; stored in the *TYPE4 format.[2] «
	QSYS	QAZMFER	PF	...Mail server framework (MSF) error logging.[2]
»	QSYS	QAZMFER4	PF	...Mail server framework (MSF) error logging; stored in the *TYPE4 format.[2] «
	QSYS	QAZMFLG	PF	...Mail server framework (MSF) data logging.[2]
»	QSYS	QAZMFLG4	PF	...Mail server framework (MSF) data logging; stored in the *TYPE4 format.[2] «
	QSYS	QAZMFSY	PF	...Mail server framework (MSF) system information logging.[2]
»	QSYS	QAZMFSY4	PF	...Mail server framework (MSF) system information logging; stored in the *TYPE4 format.[2] «
	QSYS	QPDSPJRN	PRTF	Display journal printer file.[1]

Files Used by CL Commands (Part 3)

Command Name	File Library	File Name	File Type	File Usage
DSPJRNRCVA	QSYS	QPDSPRCV	PRTF	Journal receiver attributes printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
DSPLIB	QSYS	QPDSPLIB	PRTF	Library printer file.[1]
DSPLIBD	QSYS	QPRTLIBD	PRTF	Library description printer file.[1]
DSPLIBL	QSYS	QPRTLIBL	PRTF	Library list printer file.[1]
DSPLIND	QSYS	QPDCLINE	PRTF	Line description printer file.[1]
DSPLOG	QSYS	QPDSPLOG	PRTF	Log display printer file.[1]
DSPMNUA	QSYS	QPDSPMNU	PRTF	Menu attributes printer file.[1]
DSPMOD Command: For the DSPMOD command, all of the following entries having a file type of files used to store a specific type of information about a type (or group) of files. (See the DSPMOD command for a description of the DETAIL parameter. This parameter identifies all the values that cause these files to be used.) Therefore, the common part of each model file's description *begins* here and the unique part of each description *continues* under the **File Usage** column: **Model database file that defines the record format of the file created to store...**				
DSPMOD	QSVMSS	QACQSRC	PRTF	...contains sources of example security exit programs.[1]
	QSYS	QABNDMBA	PF	...basic information and compatibility sections.[1]
	QSYS	QABNDMSI	PF	...decompressed size and size limits.[1]
	QSYS	QABNDMEX	PF	...symbols defined in this module and exported to others.[1]
	QSYS	QABNDMIM	PF	...defined symbols that are external to this module.[1]
	QSYS	QABNDMPR	PF	...a list of procedure names and types.[1]
	QSYS	QABNDMRE	PF	...a list of system objects referred to by the module at the time the module is bound into a program or service program.[1]
	QSYS	QABNDMCO	PF	...module copyright information.
	QSYS	QSYSPRT	PRTF	...module printer file.[1]
DSPMODD	QSYS	QPDCMOD	PRTF	Mode description printer file.[1]
DSPMODSTS	QSYS	QPDCMOD	PRTF	Mode status printer file.[1]
DSPMSG	QSYS	QPDSPMSG	PRTF	Message display printer file.[1]
DSPMSGD	QSYS	QPMSGD	PRTF	Message description printer file.[1]
DSPNETA	QSYS	QANFDNTF	PF	Model database file used to define the record format for network file entries.[2]
	QSYS	QPDSPNET	PRTF	Display network attributes printer file.[1]
	QSYS	QPNFNJE	PRTF	Display network job entries printer file.[1]
DSPNWID	QSYS	QPDCNWID	PRTF	Network interface description printer file.[1]
DSPOBJAUT	QSYS	QAOBJAUT	PF	Model database file that defines the record format for the object authority entries.[2]
	QSYS	QPOBJAUT	PRTF	Object authority printer file.[1]
DSPOBJD	QSYS	QADSPOBJ	PF	Model database file that defines the record format for the object description entries.[2]
	QSYS	QPRTOBJD	PRTF	Object description printer file.[1]
DSPOVR	QSYS	QPDSPOVR	PRTF	Display overrides printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
DSPPDGPRF	QGPL	QPCJPDGPRF	PRTF	Printer file for print descriptor group profile.[1]
DSPPFRGPH	QPFRDATA	QAPGGPHF	PF	Performance database file: graph format data.
	QPFRDATA	QAPGPKGF	PF	Performance database file: graph package data.
	QPFR	QPPGGPH	PRTF	Performance graphs printer file.[1]
DSPPGM	QSYS	QPDPGM	PRTF	Display program printer file.[1]
DSPPGMADP	QSYS	QADPGMAD	PF	Model database file that defines the record format of the file created to store the names of programs that adopt the specified profile.[2]
	QSYS	QPPGMADP	PRTF	Printer file that lists the programs that adopt the specified profile.[1]
DSPPGMREF	QSYS	QADSPPGM	PF	Model database file that defines the record format of the file created to store program references.[2]
	QSYS	QPDSPPGM	PRTF	Printer file that contains program references.[1]
DSPPGMVAR	QSYS	QPDBGDSP	PRTF	Program variable (debug mode) printer file.[1]
All 8 of the QASXxxxx files shown below in the QUSRSYS library are also used by the DLTPRB and WRKPRB commands. The other files below (in QSYS) are *not* used by those commands.				
DSPPRB	QSYS	QASXxxxx	PF	Set of 5 QASXxxxx model database files that contain the layouts for problem output files, where xxxx = CAOF, FXOF, PBOF, SDOF, and TXOF.[2]
	QSYS	QSXPRTD	PRTF	Problem log *detail* printer file.[1]
	QSYS	QSXPRTL	PRTF	Problem log *summary* printer file.[1]
[3] The following 8 files are also used by the DLTPRB and WRKPRB commands.				
	QUSRSYS	QASXCALL[3]	PF	Problem log call override file.
	QUSRSYS	QASXDTA[3]	PF	Problem log data identifier file.
	QUSRSYS	QASXEVT[3]	PF	Problem log event file.
	QUSRSYS	QASXFRU[3]	PF	Problem log possible cause file.
	QUSRSYS	QASXNOTE[3]	PF	Problem log user notes file.
	QUSRSYS	QASXPROB[3]	PF	Problem log problem file.
	QUSRSYS	QASXPTF[3]	PF	Problem log PTF file.
	QUSRSYS	QASXSYMP[3]	PF	Problem log symptom string file.
DSPPTF	QSYS	QADSPPTF	PF	Model database file that defines the record format of the file created to store program temporary fix (PTF) information.[2]
	QSYS	QSYSPRT	PRTF	Display programming temporary fix (PTF) printer file.[1]
DSPPWRSCD	QSYS	QSYSPRT	PRTF	Display power schedule printer file.[1]
DSPRCDLCK	QSYS	QPDSPRLK	PRTF	Record locks display printer file.[1]
» DSPRCYAP	QSYS	QSYSPRT	PRTF	Display recovery for access paths printer file. «
DSPRDBDIRE	QSYS	QSYSPRT	PRTF	Distributed relational database directory printer file.[1]
	QSYS	QADSPDE	PF	Model database file that defines the record format for the RDB directory entries.

Command Name	File Library	File Name	File Type	File Usage
DSPRJECFG	QRJE	QPRTCFGC	PRTF	RJE configuration printer file.[1]
DSPSAVF	QSYS	QPSRODSP	PRTF	Printer file for save file save/restore information.[1]
DSPSBSD	QSYS	QPRTSBSD	PRTF	Subsystem description printer file.[1]
DSPSFWRSC	QSYS	QARZLCOF	PF	Model database file of the file created to store information about IBM licensed programs and SystemView^R packaged applications.[2]
	QSYS	QSYSPRT	PRTF	Software resources printer file.[1]
DSPSOCSTS	QSYS	QSYSPRT	PRTF	Sphere of control status printer file.[1]
	QUSRSYS	QAALSOC	PF	Sphere of control database file.
DSPSRVPGM	QSYS	QSYSPRT	PRTF	Service program printer file.[1]
DSPSYSSTS	QSYS	QPDSPSTS	PRTF	Display system status printer file.[1]
DSPSYSVAL	QSYS	QPDSPSVL	PRTF	System value printer file.[1]
DSPTAP	QSYS	QPTAPDSP	PRTF	Printer file for tape output.[1]
	QSYS	QPSRODSP	PRTF	Printer file for tapes in save/restore format.[1]
	QSYS	QSYSTAP	TAPF	Tape device file for input.
DSPTAPCGY	QSYS	QTAPCGY	PRTF	Printer file for tape categories 1.
	QSYS	QATACOF	PF	Model output file for tape categories 2.
	QUSRSYS	QATACGY	PF	Library device database file.
	QUSRSYS	QLTACGY	LF	Library device logical database file.
DSPTAPCTG	QSYS	QPTACTG	PRTF	Printer file for tape cartridge identifiers 1.
	QSYS	QATAVOF	PF	Model output file for tape cartridge identifiers 2.
	QUSRSYS	QATAMID	PF	Library device database file.
	QUSRSYS	QATAMID	LF	Library device logical database file.
DSPTAPSTS	QSYS	QPTAPSTS	PRTF	Printer file for tape library
	QSYS	QATAIOF	PF	Model output file for tape
	QSYS	QSYSTAP	TAPF	Tape device file or input
	QSYS	QPTYSWTD	PRTF	Printer file for printing a telephony switch entry.[1]
DSPTRAPRF	QSYS	QSYSPRT	PRTF	Printer file containing a token-ring network adapter profile.[1]
DSPTRC	QSYS	QPDBGDSP	PRTF	Trace (debug mode) printer file.[1]
DSPTRCDTA	QSYS	QPDBGDSP	PRTF	Trace data (debug mode) printer file.[1]
DSPTRNSTS	QSYS	QSYSPRT	PRTF	Token-ring network status printer file.[1]
DSPUSRPMN	QSYS	QSYSPRT	PRTF	Display document authority printer file.[1]
DSPUSRPRF	QSYS	QADSPUPA	PF	Model database file that defines the record format of user profiles for TYPE(*OBJAUT).[2]
	QSYS	QADSPUPB	PF	Model database file that defines the record format of user profiles for TYPE(*BASIC).[2]
	QSYS	QADSPUPO	PF	Model database file that defines the record format of user profiles for TYPE(*OBJOWN).[2]
	QSYS	QPUSRPRF	PRTF	User profile printer file.[1]
DSPWSUSR	QSYS	QSYSPRT	PRTF	Work station user printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
DUPTAP	QSYS	QSYSTAP	TAPF	Tape device file used for input and output.
EDTIGCDCT	QSYS	QPDSPDCT	PRTF	DBCS printer file.
EDTQST	QSYS	QPQAPRT	PRTF	Q & A printer file.
EJTEMLOUT	QSYS	QPEMPRTF	PRTF	Emulation printer file.
ENDDSKCOL	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QAPTDSKD	PF	Performance data collection file: disk activity data.
ENDIDXMON	QUSRSYS	QABBADMTB	PF	OfficeVision^R text search services administration table.
ENDJOBTRC	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QAPTTRCJ	PF	Performance data collection file: job trace data.
	QPFR	QPPTTRCD	PRTF	Performance printer file containing job trace analysis *detail* data.
	QPFR	QPPTTRC1	PRTF	Performance printer file containing job trace analysis *summary* data of physical disk activity.
	QPFR	QPPTTRC2	PRTF	Performance printer file containing job trace analysis I/O *summary* data.
ENDPRTEML	QSYS	QPEMPRTF	PRTF	Emulation printer file.
	QPFR	QAPTSAMH	PF	Performance data collection file: high-level sampled address monitor (SAM) data.
	QPFR	QAPTSAMV	PF	Performance data collection file: low-level sampled address monitor (SAM) data.
FMTDTA	QSYS	QSYSPRT	PRTF	Data format printer file.
	QGPL	QFMTSRC	PF	Sort source default input file.
FNDSTRPART	QPDA	QPUOPRTF	PRTF	PDM printer output file for user's find string requests.
FNDSTRPDM	QPDA	QPUOPRTF	PRTF	PDM printer output file for user's find string requests.
HLDDSTQ	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
INZTAP	QSYS	QSYSTAP	TAPF	Tape device file used for output.
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
LODQSTDB	QQALIB	QAQAxxxx00	PF	Q & A supplied model database files.[2]
	QSYS	QAQA00xxxx	LF	Q & A database model files.[2]
	QSYS	QAQA00xxxx	PF	Q & A database model files.[2]
	QSYS	QPQAPRT	PRTF	Q & A printer file.
MRGFORMD	QPDA	QPAPFPRT	PRTF	Merge form description printer file.
MRGTCPHT	QUSRSYS	QATOCHOST	PF	TCP/IP hosts file.
PRTACTRPT	user-lib	QAITMON	PF	Performance data collection file: job and Licensed Internal Code task data.
	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPITACTR	PRTF	Performance printer file containing job and Licensed Internal Code task data.
PRTAFPDTA	QSYS	QSYSPRT	PRTF	Advanced Function Printing^{TM} data printer file.

Command Name	File Library	File Name	File Type	File Usage
PRTCMDUSG	QSYS	QSYSPRT	PRTF	Command usage printer file.
PRTCMNTRC	QSYS	QASCCMNT	PF	Model database file that defines the record format of the file created to store communications trace records.[2]
	QSYS	QPCSMPRT	PRTF	Communications trace printer file (concurrent service monitor).[1]
PRTCPTRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTCPTR	PRTF	Performance printer file containing component-level activity data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = ASYN, BSC, CIOP, CONF, DIOP, DISK, ECL, ETH, HDLC, JOBS, LIOP, MIOP, POOL, RESP, RWS, and SYS.[2]
PRTDEVADR	QSYS	QPDDEVA	PRTF	Print device address printer file.
PRTDOC	QSYS	QAOPOUFL	PF	Document output database file.[1]
	QSYS	QSYSPRT	PRTF	Print document printer file.[1]
PRTDSKINF	QSYS	QAEZDISK	PF	Model outfile for disk space information.
	QUSRSYS	QAEZDISK	PF	Database input file of disk space information.
	QSYS	QPEZDISK	PRTF	Disk space report printer file.
	QSYS	QSYSPRT	PRTF	Disk space report printer file. This file must be specified if using OVRPRTF.
PRTDSKRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QAPTDSKD	PF	Performance database input file of disk activity collection data.
	QPFR	QPPTDSK	PRTF	Printer file of disk unit I/O activity data.
PRTERRLOG	QSYS	QAPRTELG	PF	Model database file that defines the record format of the file created to store error log records.[2]
	QSYS	QAVOLSTA	PF	Model database file that defines the record format of the file created to store volume statistics.[2]
	QSYS	QPCSMPRT	PRTF	Error log (for concurrent service monitor) printer file.[1]
PRTINTDTA	QSYS	QPCSMPRT	PRTF	Internal data (for concurrent service monitor) printer file.
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
PRTJOBRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTITVJ	PRTF	Performance printer file containing job interval collection data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = CONF and JOBS.[2]
PRTJOBTRC	QPFR	QAJOBTRC	PF	Performance data collection file: job trace data.
	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QAPTTRCJ	PF	Performance data collection file: job trace data.
	QPFR	QPPTTRCD	PRTF	Performance printer file containing job trace analysis *detail* data.

Command Name	File Library	File Name	File Type	File Usage
	QPFR	QPPTTRC1	PRTF	Performance printer file containing job trace analysis *summary* data of physical disk activity.
	QPFR	QPPTTRC2	PRTF	Performance printer file containing job trace analysis I/O *summary* data.
PRTLCKRPT	user-lib	QAPTLCKD	PF	Performance data collection *output* file containing lock and seizure conflict data.
	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTLCK	PRTF	Performance data collection *printer* file containing lock and seizure conflict data.
	QSYS	QAPMDMPT	PF	Performance data collection *input* file containing system lock and seizure conflict trace data.[2]
PRTPOLRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTITVP	PRTF	Performance printer file containing subsystem and pool activity interval data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = CONF, JOBS, and POOL.[2]
PRTRSCRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTITVR	PRTF	Performance printer file containing disk and communications line activity interval data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = ASYN, BSC, CIOP, CONF, DIOP, DISK, ECL, ETH, HDLC, LIOP, MIOP, RESP, RWS, and SYS.[2]
	QPFR	QAPTSAMH	PF	Performance data collection file: high-level sampled address monitor (SAM) data.
	QPFR	QAPTSAMV	PF	Performance data collection file: low-level sampled address monitor (SAM) data.
	QPFR	QPPTSAM	PRTF	Printer file of SAM performance data.
PRTSYSRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPPTSYSR	PRTF	Performance printer file containing system workload and resource utilization data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = ASYN, BSC, CONF, DISK, ECL, ETH, HDLC, JOBS, POOL, SYS, and X25.[2]
PRTTNSRPT	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QPFR	QPSPDJS	PRTF	Performance printer file containing job *summary* data.
	QPFR	QPSPDTD	PRTF	Performance printer file containing job state *transition* data.
	QPFR	QPSPDTS	PRTF	Performance printer file containing job *transaction* data.
	QSYS	QAPMxxxx	PF	QAPMxxxx performance data collection files, where xxxx = DMPT and JOBS.[2]
PRTTRC	QSYS	QPSRVTRC	PRTF	Job trace printer output file.[1]
PRTTRCRPT	user-lib	QTRTJOBT	PF	Performance data collection *input* file containing batch job trace data.

Command Name	File Library	File Name	File Type	File Usage
	QPFR	QAPTDDS	PF	Performance data DDS source file.
	QSYS	QAPMDMPT	PF	Performance data collection input file containing system trace data.[2]
QRYDOCLIB	QSYS	QAOSIQDL	PF	Query document library output file.
QRYDST	QSYS	QAOSILIN	PF	Incoming distribution output file.
	QSYS	QAOSILOT	PF	Outgoing distribution output file.
RCLSTG	QSYS	QPRCLDMP	PRTF	Reclaim dump output listing.
RCVDST	QSYS	QAOSIRCV	PF	Receive incoming mail distribution model database file.[2]
RCVTIEF	QSYS	QPTIRCV	PRTF	Received files summary printer file.[1]
RLSDSTQ	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
RMVDSTQ	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
	QUSRSYS	QASNADSR	PF	SNADS routing table.
RMVDSTRTE	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
	QUSRSYS	QASNADSR	PF	SNADS routing table.
RMVDSTSYSN	QUSRSYS	QASNADSA	PF	SNADS secondary node ID table.
» RMVJRNCHG	QSYS	QAJRNCHG	PF	Model output file for remove journaled changes. «
RMVNETJOBE	QUSRSYS	QANFNJE	PF	Network job entry database file.
RMVSOCE	QUSRSYS	QAALSOC	PF	Sphere of control file.
RMVTAPCTG	QUSRSYS	QATAMID	PF	Cartridge ID database file.
	QUSRSYS	QLTAMID	LF	Cartridge ID logical database file.
	QUSRSYS	QATACGY	PF	Category database file.
	QUSRSYS	QLTACGY	LF	Category logical file.
RMVTCPHTE	QUSRSYS	QATOCHOST	PF	TCP/IP host file.
RMVTCPIFC	QUSRSYS	QATOCIFC	PF	TCP/IP interface file.
RMVTCPPORT	QUSRSYS	QATOCPORT	PF	TCP/IP port restrictions file.
RMVTCPRSI	QUSRSYS	QATOCRSI	PF	TCP/IP remote system information file.
RMVTCPRTE	QUSRSYS	QATOCRTE	PF	TCP/IP route file.
RMVTXTIDXE	QUSRSYS	QABBADMTB	PF	OfficeVision^R text search services administration table.
RNMTCPHTE	QUSRSYS	QATOCHOST	PF	TCP/IP host file.
» RST	QSYS	QSYSTAP	TAPF	Tape device file used for input. «
RSTCFG	QSYS	QASRRSTO	PF	Model output file for configuration.[2]
	QSYS	QPSRLDSP	PRTF	Restored objects status printer file.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
RSTDLO	QSYS	QAOJRSTO	PF	Model output file for restored document library objects.[2]
	QSYS	QPRSTDLO	PRTF	Printer file for restored documents and folders.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
RSTLIB	QSYS	QASRRSTO	PF	Model output file for libraries.[2]

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QPSRLDSP	PRTF	Restored objects status printer file.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
RSTLICPGM	QSYS	QPSRLDSP	PRTF	Restored objects status printer file.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
RSTOBJ	QSYS	QASRRSTO	PF	Model output file for restored objects.[2]
	QSYS	QPSRLDSP	PRTF	Restored objects status printer file.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
RSTUSRPRF	QSYS	QASRRSTO	PF	Model output file for user profiles.[2]
	QSYS	QSYSTAP	TAPF	Tape device file used for input.
RTVDOC	QSYS	QAOSIRTV	PF	Retrieve document from document library output file.
RTVDSKINF	QSYS	QAEZDISK	PF	Model file for disk information.
	QUSRSYS	QAEZDISK	PF	Database output file for disk information.
RUNQRY	QSYS	QPQUPRFIL	PRTF	Printer file used for query output.[1]
» SAV	QSYS	QSYSTAP	TAPF	Tape device file used for output. «
SAVCFG	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVCHGOBJ	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVDLO	QSYS	QAOJSAVO	PF	Model output file for saved documents and folders.[2]
	QSYS	QPSAVDLO	PRTF	Printer file for saved documents and folders.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVLIB	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
»	QUSRSYS	QSRPNTWK	PRTF	Printer file for saved database file networks. «
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVOBJ	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVSAVFDTA	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVSECDTA	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.
SAVSYS	QSYS	QASAVOBJ	PF	Model output file for saved objects.[2]
	QSYS	QPSAVOBJ	PRTF	Printer file for saved objects.[1]
	QSYS	QSYSTAP	TAPF	Tape device file used for output.

Command Name	File Library	File Name	File Type	File Usage
SBMFNCJOB	QSYS	QDFNDATA	DSPF	Non-ICF finance display file.
	QUSRSYS	QFNDEVTBL	PF	File containing data for device tables.
	QUSRSYS	QFNPGMTBL	PF	File containing data for program tables.
	QUSRSYS	QFNUSRTBL	PF	File containing data for user tables.
SETTAPCGY	QUSRSYS	QATACGY	PF	Category database file.
	QUSRSYS	QLTACGY	LF	Category logical file.
SNDDSTQ	QUSRSYS	QASNADSQ	PF	SNADS distribution queues table.
SNDFNCIMG	QSYS	QCRFDWNL	ICFF	ICF file used for communication with 4701 controller.
SNDPTFORD	QGPL	Qnnnnnnn	SAVF	PTF save file, where nnnnnnn is the PTF number.
	QSYS	QESPRTF	PRTF	Printer file for PTF cover letters.
SNDSRVRQS	QGPL	Qnnnnnnn	SAVF	PTF save file, where nnnnnnn is the PTF number.
	QSYS	QESPRTF	PRTF	Printer file for PTF cover letters.
	QUSRSYS	QAEDCDBPF	PF	File containing service contact data.
STRCODE	user-lib	EVFCICFF	ICFF	ICF file used for communication with workstation.
STRCPYSCN	QSYS	QASCCPY	PF	Pattern for copy screen output file.
STRDFU	QSYS	QDFUPRT	PRTF	DFU printer file.
	QSYS	QPDZDTALOG	PRTF	DFU run-time audit log.
	QSYS	QPDZDTAPRT	PRTF	DFU run-time printer data file.
STRIDXMON Command: The STRIDXMON and STRRGZIDX commands use 9 of the 10 files in OfficeVision(R) text search services that are also used by the STRUPDIDX command. **For those 9 files (all named as QABBxxxxx), see the list under the STRUPDIDX command;** the 10th and last file in that list (QABBLADN) is *not* used by STRIDXMON or STRRGZIDX.				
STRIDXMON	QUSRSYS	QABBxxxxx	PF	See Note E.
STRPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.

Files Used by CL Commands (Part 4)

Command Name	File Library	File Name	File Type	File Usage
CRTPFRDTA Command: All of the following files for the CRTPFRDTA command are physical files (PF) or logical files (LF) that are used as model files (not actual output files) to define the record formats of files created to store performance data collected by this command. All of these model files are in the QSYS library, and the files they create are in a library determined by the user (defaults to the same library specified for the *MGTCOL object - usually QPFRDATA). Each created file stores the specific type of performance data in the format defined for the specific type of data being collected. For the files listed below, the common part of each model file's description begins here and the unique part of each description continues under the **File Usage** column: **Model performance monitor database file that defines the record format of the data collection file created to store...**				
CRTPFRDTA	QSYS	QAPMAPPN	PF	...APPN data.[2]

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QAPMASYN	PF	...asynchronous data.[2]
	QSYS	QAPMBSC	PF	...binary synchronous data.[2]
	QSYS	QAPMJOBMI	PF	...job data from MI.
	QSYS	QAPMJOBOS	PF	...job data from OS/400.
	QSYS	QAPMJOBWT	PF	...Job wait bucket data.[2]
	QSYS	QAPMJOBWTD	PF	...Job wait bucket descriptions.[2]
	QSYS	QAPMJSUM	PF	...job data summarized.
	QSYS	QAPMBUS	PF	...bus counter data.[2]
	QSYS	QAPMCIOP	PF	...communications controller data.[2]
	QSYS	QAPMCONF	PF	...system configuration data.[2]
	QSYS	QAPMDDI	PF	...DDI distributed interface data.[2]
	QSYS	QAPMDIOP	PF	...storage device controller data.[2]
	QSYS	QAPMDISK	PF	...disk storage (DASD) data.[2]
	QSYS	QAPMDOMINO	PF	...Domino™ data.[2]
	QSYS	QAPMECL	PF	...ECL or token-ring LAN data.[2]
	QSYS	QAPMETH	PF	...Ethernet statistics data.[2]
	QSYS	QAPMFRLY	PF	...frame relay data.[2]
	QSYS	QAPMHDLC	PF	...HDLC and SDLC control data.[2]
	QSYS	QAPMHTTPB	PF	...HTTP server (powered by Apache) base data.[2]
	QSYS	QAPMHTTPD	PF	...HTTP server (powered by Apache) detail data.[2]
	QSYS	QAPMIDLC	PF	...ISDN data link control file entries data.[2]
	QSYS	QAPMJOBL	LF	...logical view of job data from QAPMJOBMI and QAPMJOBOS.[2]
	QSYS	QAPMLAPD	PF	...ISDN LAPD file entries data.[2]

Command Name	File Library	File Name	File Type	File Usage
	QSYS	QAPMLIOP	PF	...local work station controller (WSC) data.[2]
	QSYS	QAPMMIOP	PF	...multifunction controller data.[2]
	QSYS	QAPMPOOLB	PF	...main storage pool data.[2]
	QSYS	QAPMPOOLT	PF	...pool tuning data.[2]
	QSYS	QAPMPOOLL	LF	...logical view as pool data.[2]
	QSYS	QAPMPPP	PF	...PPP protocol data.[2]
	QSYS	QAPMRESP	PF	...local work station response data.[2]
	QSYS	QAPMSAP	PF	...token-ring LAN, Ethernet, Distributed Data Interface, and frame relay service access point.[2]
	QSYS	QAPMSBSD	PF	...subsystem description data.[2]
	QSYS	QAPMSNA	PF	...SNA data.[2]
	QSYS	QAPMSNADS	PF	...SNADS data.[2]
	QSYS	QAPMSTND	PF	...DDI station data.[2]
	QSYS	QAPMSTNE	PF	...Ethernet station data.[2]
	QSYS	QAPMSTNL	PF	...token-ring LAN station data.[2]
	QSYS	QAPMSTNY	PF	...frame relay station data.[2]
	QSYS	QAPMSYSCPU	PF	...system CPU data.[2]
	QSYS	QAPMSYSL	LF	...log view CPU sys data.[2]
	QSYS	QAPMSYSTEM	PF	...system performance data.[2]
	QSYS	QAPMTCP	PF	...TCP system data.[2]
	QSYS	QAPMTCPIFC	PF	...TCP interface data.[2]
	QSYS	QAPMTSK	PF	...system internal data.[2]
	QSYS	QAPMUSRTNS	PF	...User defined transaction data .[2]
	QSYS	QAPMX25	PF	...X.25 communications data.[2]
STRPRTEML	QSYS	QPEMPRTF	PRTF	Emulation printer file.
STRPRTWTR	QSYS	QPSPLPRT	PRTF	Printer device file used for all printer writing.

Command Name	File Library	File Name	File Type	File Usage
STRQMQRY	QSYS	QPQXPRTF	PRTF	Printer file used by Query CPI.[1]
STRQST	QQALIB	QAQAxxxx00	PF	Q & A supplied model database files.[2]
	QSYS	QAQA00xxxx	LF	Q & A database model files.[2]
	QSYS	QAQA00xxxx	PF	Q & A database model files.[2]
	QSYS	QPQAPRT	PRTF	Q & A printer file.
STRRGZIDX Command: The STRRGZIDX and STRIDXMON commands use 9 of the 10 files in OfficeVision text search services that are also used by the STRUPDIDX command. **For those 9 files (all named as QABBxxxxx), see the list under the STRUPDIDX command;** the 10th and last file in that list (QABBLADN) is *not* used by STRRGZIDX or STRIDXMON.				
STRRGZIDX	QUSRSYS	QABBxxxxx	PF	See Note F.
STRSDA	QGPL	QDDSSRC	PF	DDS source default input file.
STRSEU	QGPL	QTXTSRC	PF	SEU source default input file.
	QPDA	QPSUPRTF	PRTF	SEU source member printer file.
STRSST	QSYS	QPCSMPRT	PRTF	Service tools printer file.[1]
	QTY	QATYALMF	PF	Telephony database model file of alarm record formats.
	QUSRSYS	QATYSWTE	PF	Telephony database file of user-created switch entries.
	QTY	QATYCDRF	PF	Telephony database model file of alarm record formats.
	QUSRSYS	QATYSWTE	PF	Telephony database file of user-created switch entries.
STRUPDIDX Command: The STRUPDIDX command uses the following 10 files associated with OfficeVision^R text search services. The first 9 of these files (all but QABBLADN) are also used by the STRIDXMON and STRRGZIDX commands. All of the following files have a common beginning description. Therefore, the common part of each file's description *begins* here and the unique part of each description *continues* under the **File Usage** column: **OfficeVision^R text search services...**				
STRUPDIDX	QUSRSYS	QABBADMTB	PF	...administration table.
	QUSRSYS	QABBCAN	PF	...candidates file.
	QUSRSYS	QABBCOX	PF	...context index.

Command Name	File Library	File Name	File Type	File Usage
	QUSRSYS	QABBDEX	PF	...external document index identifiers.
	QUSRSYS	QABBDIC	PF	...dictionary.
	QUSRSYS	QABBDIX	PF	...internal document index identifiers.
	QUSRSYS	QABBDOX	PF	...document index table.
	QUSRSYS	QABBFIX	PF	...fragment index.
	QUSRSYS	QABBIQTB	PF	...scheduling queue.
	QUSRSYS	QABBLADN[4]	PF	...list of indexed LADNs (library-assigned document names).
TRCCNN	QSYS	QSYSPRT	PRTF	Connection trace printer file.
[4] This QABBLADN file is *not* used by the STRRGZIDX and STRIDXMON commands.				
TRCCPIC	QSYS	QACM0TRC	PF	Trace CPI-Communications database model output file.[2]
	QSYS	QSYSPRT	PRTF	Trace CPI-Communications printer file.[1]
TRCICF	QSYS	QAIFTRCF	PF	Trace ICF database output file.
	QSYS	QPIFTRCF	PRTF	Trace ICF printer file.[1]
TRCINT	QSYS	QPCSMPRT	PRTF	Internal trace (for concurrent service monitor) printer file.
	QSYS	QSYSTAP	TAPF	Tape device file for output.
TRCJOB	QSYS	QATRCJOB	PF	Database file that defines the record format of the file created to store trace records.[2]
	QSYS	QPSRVTRC	PRTF	Job trace printer output file.[1]
UPDDTA	QSYS	QPDZDTALOG	PRTF	DFU run-time audit log.
	QSYS	QPDZDTAPRT	PRTF	DFU run-time printer data file.
VFYLNKLPDA^R	QSYS	QSYSPRT	PRTF	Verify link supporting LPDA^R -2 printer file.[1]
WRKACTJOB	QSYS	QPDSPAJB	PRTF	Active jobs display printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
WRKALR	QUSRSYS	QAALERT	PF	Alert database file.
	QSYS	QSYSPRT	PRTF	Alert printer file.[1]
WRKCFGSTS	QSYS	QSYSPRT	PRTF	Configuration status printer file.[1]
WRKCNTINF	QUSRSYS	QAEDCDBPF	PF	Database file containing contact data.
WRKDDMF	QSYS	QPWRKDDM	PRTF	Distributed data management (DDM) file attributes printer file.[1]
WRKDEVTBL	QUSRSYS	QFNDEVTBL	PF	File containing data for device tables.
WRKDIR	QSYS	QPDSPDDL	PRTF	Directory entry *details* printer file.
	QSYS	QPDSPDSM	PRTF	Directory entry *summary* printer file.
WRKDOCCVN	QUSRSYS	QAO1CRL	LF	Document conversion *logical* file for input or output.
	QUSRSYS	QAO1CVNP	PF	Document conversion *physical* file for input or output.
	QUSRSYS	QAO1DCVN	PRTF	Document conversion printer file.
WRKDPCQ	QSYS	QPDXWRKD	PRTF	Printer file for DSNX/PC queued distribution requests.[1]
WRKDSKSTS	QSYS	QPWCDSKS	PRTF	Disk status printer file.[1]
WRKDSTL	QSYS	QPDSPLDL	PRTF	Distribution list *details* printer file.
	QSYS	QPDSPLSM	PRTF	Distribution list *summary* printer file.
WRKDSTQ	QSYS	QPDSTSTS	PRTF	Distribution status printer file.[1]
	QUSRSYS	QASNADSQ	PF	SNADS destination queues table.
WRKFCT	QRJE	QPDSPFCT	PRTF	Forms control table printer file.[1]
WRKGRPPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.[1]
WRKHDWRSC	QSYS	QASUPTEL	PF	Hardware resources locking database file.
WRKHTTPCFG	QUSRSYS	QATMHTTPC	PF	TCP/IP HTTP file.
WRKJOB	QSYS	QPDSPJOB	PRTF	Job display printer file.[1]

Command Name	File Library	File Name	File Type	File Usage
WRKJOBQ	QSYS	QPRTSPLQ	PRTF	Job queue printer file (spooling queue).[1]
WRKJOBSCDE	QSYS	QSYSPRT	PRTF	Job schedule entries printer file.[1]
WRKJRNA	QSYS	QPDSPJNA	PRTF	Journal attributes printer file.[1]
WRKLIBPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.[1]
WRKMBRPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.[1]
WRKMSG	QSYS	QPDSPMSG	PRTF	Printer file for message queue messages.[1]
WRKMSGD	QSYS	QPMSGD	PRTF	Message description printer file.
WRKNAMSMTP	QSYS	QATMSMTP	PF	TCP/IP SMTP personal alias table.
	QSYS	QATMSMTPA	PF	TCP/IP SMTP system alias table.
WRKNETF	QSYS	QANFDNTF	PF	Database file for display network files.[1]
	QSYS	QPNFDNTF	PRTF	Printer file for display network files.[1]
WRKNETJOBE	QUSRSYS	QANFNJE	PF	Database file for network job entries.[1]
	QSYS	QPNFNJE	PRTF	Printer file for network job entries.[1]
WRKOBJLCK	QSYS	QPDSPOLK	PRTF	Object locks display printer file.[1]
WRKOBJPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.[1]
WRKOUTQ	QSYS	QPRTSPLQ	PRTF	Output spooling queue printer file.[1]
WRKOUTQD	QSYS	QPDSPSQD	PRTF	Output queue description.[1]
WRKPARTPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.
WRKPGMTBL	QUSRSYS	QFNPGMTBL	PF	File containing data for program tables.
WRKPRB Command: All 8 of the QASXxxxx files shown in the QUSRSYS library for the WRKPRB command are the same subset of files that are shown in the QUSRSYS library for the DSPPRB command. For the descriptions of these files, see the DSPPRB command.				

Command Name	File Library	File Name	File Type	File Usage
WRKPRB	QSYS	QSXPRTD	PRTF	Problem log *detail* printer file.
	QSYS	QSXPRTL	PRTF	Problem log *summary* printer file.
	QUSRSYS	QASXxxxx	PF	See the DSPPRB command for these 8 files in the QUSRSYS library.
WRKPRJPDM	QPDA	QPUOPRTF	PRTF	Printer file for displayed lists in PDM.[1]
WRKQRY	QSYS	QPQUPRFIL	PRTF	Printer file used for query output.
WRKQST	QSYS	QPQAPRT	PRTF	Q & A printer file.
WRKRDBDIRE	QSYS	QSYSPRT	PRTF	Distributed relational database directory printer file.[1]
WRKRDR	QSYS	QPRTRDWT	PRTF	Reader display printer file.[1]
WRKRJESSN	QRJE	QPRJESTS	PRTF	Active status printer file for RJE session.[1]
WRKRPYLE	QSYS	QPRTRPYL	PRTF	System reply list printer file.[1]
WRKSBMJOB	QSYS	QPDSPSBJ	PRTF	Submitted jobs printer file.[1]
WRKSBS	QSYS	QPDSPSBS	PRTF	Subsystem display printer file.[1]
WRKSBSJOB	QSYS	QPDSPSBJ	PRTF	Subsystem jobs display printer file.[1]
WRKSHRPOOL	QSYS	QSYSPRT	PRTF	Shared storage pools printer file.[1]
WRKSOC	QUSRSYS	QAALSOC	PF	Sphere of control database file.
WRKSPLF	QSYS	QPRTSPLF	PRTF	Spooled file printer file.[1]
WRKSPLFA	QSYS	QPDSPSFA	PRTF	Spooled file attributes printer file.[1]
WRKSPTPRD	QSYS	QSYSPRT	PRTF	Supported products printer file.[1]
WRKSRVPVD	QUSRSYS	QAEDSPI	PF	Service provider information file.
WRKSRVRQS	QUSRSYS	QANSSRI	PF	Service requester file.
WRKSSND	QRJE	QPRTSSND	PRTF	Session description printer file.[1]
WRKSYSACT	user-lib	QAITMON	PF	Performance data collection file: job and Licensed Internal Code task data.[1]

Command Name	File Library	File Name	File Type	File Usage
	QPFR	QAPTDDS	PF	Performance data DDS source file.
WRKSYSSTS	QSYS	QPDSPSTS	PRTF	System status printer file.[1]
WRKSYSVAL	QSYS	QSYSPRT	PRTF	System values printer file.[1]
WRKTAPCTG	QUSRSYS	QATAMID	PF	Cartridge ID database file.
	QUSRSYS	QATAMID	LF	Cartridge ID logical database file.
	QUSRSYS	QATACGY	PF	Category database file.
	QUSRSYS	QLTACGY	LF	Category logical file.
	QSYS	QSYSTAP	TAPF	Tape device file for input.
WRKTCPPTP	QUSRSYS	QATOCPTP	PF	TCP/IP point-to-point profile configuration.
	QUSRSYS	QATOCMODEM	PF	TCP/IP point-to-point modem configuration.
WRKTIE	QSYS	QPTIRCV	PRTF	Printer file summary of files received.
WRKTRA	QSYS	QSYSPRT	PRTF	Printer file containing list of token-ring network adapters.[1]
WRKTXTIDX	QUSRSYS	QABBADMTB	PF	OfficeVision^{R} text search services administration table.
WRKUSRJOB	QSYS	QPDSPSBJ	PRTF	User jobs display printer file.[1]
WRKUSRTBL	QUSRSYS	QFNUSRTBL	PF	File containing data for user tables.
WRKWTR	QSYS	QPRTRDWT	PRTF	Writer display printer file.[1]

Printing command descriptions on the server

To print the parameter and value descriptions for a command on your iSeries$^{(TM)}$ server, follow these instructions:

- To print help for an entire command, do either of the following:
  - From any command line, type the command name (for example, CRTUSRPRF) and press F1. The display shows general help for the command and help for each command parameter. Press F14 to print the command help.
  - On a prompt display for a given command, move the cursor to the top line and press F1. Then press F14.
- To print the help for one CL command keyword parameter, do the following:

– From a command line, type the CL command name and press F4 to display the command prompt display. Position the cursor anywhere on the line of the keyword parameter for which you want help. Press F1 to display the help for the keyword parameter. Press F14 to print the help.

For information about how to print PDF versions of CL information, see Print these topics.

Appendix A. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY  10594-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Trademarks

Appendix B. Terms and conditions for downloading and printing publications

Permissions for the use of the publications you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM$^{(R)}$.

Commercial Use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

All material copyrighted by IBM Corporation.

By downloading or printing a publication from this site, you have indicated your agreement with these terms and conditions.

Appendix C. Code disclaimer information

This document contains programming examples.

IBM(R) grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

All sample code is provided by IBM for illustrative purposes only. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

All programs contained herein are provided to you ″AS IS″ without any warranties of any kind. The implied warranties of non-infringement, merchantability and fitness for a particular purpose are expressly disclaimed.

Printed in USA