



@server

iSeries

DDS concepts

*Version 5 Release 3*







@server

iSeries

DDS concepts

*Version 5 Release 3*

**Note**

Before using this information and the product it supports, be sure to read the information in Appendix G, "Notices," on page 65.

**Sixth Edition (August 2005)**

| This edition applies to version 5, release 3, modification 0 of IBM 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.

© Copyright International Business Machines Corporation 1999, 2005. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

## About DDS concepts . . . . . v

Who should read the DDS concepts information . . . . .	v
Conventions and terminology used in the DDS information. . . . .	v
Print this topic . . . . .	v
What's New for V5R3 in the DDS concepts information . . . . .	vi

## Describing data attributes using DDS . . . 1

Creating a file using data description specifications (DDS). . . . .	1
Completing the DDS form. . . . .	1
Entering the DDS source statements . . . . .	3
Creating the DDS file . . . . .	4
Rules for DDS keywords and parameter values. . . . .	4
DDS naming conventions . . . . .	6
DDS syntax coding examples . . . . .	7
DDS syntax for a physical file . . . . .	8
DDS syntax for a simple logical file. . . . .	9
DDS syntax for a join logical file . . . . .	10
DDS syntax for a display file . . . . .	11
DDS syntax for a printer file. . . . .	12
DDS syntax for an ICF file . . . . .	13

## Appendix A. Examples of DDS for Each File Type . . . . . 15

Examples of database files in DDS. . . . .	15
Example of a field reference file . . . . .	15
Example of a physical file with a new record format . . . . .	16
Example of a logical file specifying multiple formats and new keys . . . . .	17
Example of a logical file specifying a new record format . . . . .	18
Example of a join logical file. . . . .	19
Examples of device files in DDS . . . . .	20
Example of an inquiry display with two record formats in DDS . . . . .	20
Example of a subfile with SFLPAG value equal to SFLSIZ value . . . . .	23
Example of a subfile with paging by OS/400 program and high-level language program . . . . .	24

Example of a horizontal subfile displayable on two display sizes . . . . .	26
Example of a message subfile using DDS . . . . .	28
Example of a printer file using DDS . . . . .	29
Example of an ICF file using DDS. . . . .	30
Example program that uses a physical, display, and printer file. . . . .	32
Example DDS compiler listing . . . . .	34
DDS debugging template. . . . .	36

## Appendix B. When to specify REF and REFFLD keywords for DDS files. . . . . 39

## Appendix C. DDS keyword and value abbreviations . . . . . 41

## | Appendix D. General considerations for using ideographic text (DBCS) with DDS files . . . . . 45

Positional entries for files that use DBCS data . . . . .	45
Length (positions 30 through 34) . . . . .	45
Data type (position 35) . . . . .	46
Keyword entries for files that use DBCS (positions 45 through 80) . . . . .	46
DBCS character strings . . . . .	47
Entering bracketed-DBCS character strings . . . . .	47
Entering DBCS-graphic character strings. . . . .	47
DDS computer printouts with DBCS output . . . . .	48

## Appendix E. Related information . . . . . 49

## Appendix F. DDS glossary of terms . . . . . 51

## Appendix G. Notices . . . . . 65

Trademarks . . . . .	66
Terms and conditions for downloading and printing publications . . . . .	67
Code disclaimer information. . . . .	67

## Index . . . . . 69



---

## About DDS concepts

This book provides the basic concepts you need to know for coding the data description specifications (DDS) for files that can be described externally.

Use this information as you work with DDS for physical, logical, display, printer, and intersystem communications function, hereafter referred to as ICF, files. These are documented in the following books:

- DDS for physical and logical files
- DDS for display files
- DDS for printer files
- DDS for ICF files

---

## Who should read the DDS concepts information

This book is intended for programmers who use the iSeries servers.

---

## Conventions and terminology used in the DDS information

- A *keyword* is a name that identifies a function.
- A *parameter* is an argument shown between the parentheses on a keyword that identifies a value or set of values you can use to tailor the function the keyword specifies.
- A *value* is an actual value that you can use for a parameter.
- In the keyword descriptions, *this field* or *this record format* means the field or record format you are defining.
- The expression *use this file- or record-level keyword* means the keyword is valid only at the file or record level.
- *To specify a keyword* means to code the keyword in the DDS for a file. This contrasts with *to select a keyword* or *when a keyword is in effect*, which both mean that any conditioning (such as one or more option indicators) is satisfied when an application program issues an input or output operation.
- *Current source* or *source you are defining* means the DDS that together make up the description of one file.
- In sample displays, character fields are shown as Xs and numeric fields are shown as Ns.
- The 5250 Work Station Feature is a feature of the OS/2<sup>®</sup> communications manager that allows the personal computer to perform like a 5250 display station and use functions of OS/400.
- *Logical file* includes join logical files, simple logical files, and multiple-format logical files.
- *Page* means to move information up or down on the display. *Roll* means the same as page. *Paging keys* are the same as *roll keys*. The PAGEDOWN keyword is the same as the ROLLUP keyword. The PAGEUP keyword is the same as the ROLLDOWN keyword.

---

## Print this topic

To view or download the PDF version, select DDS concepts (about 1461 KB or 74 pages).

### Other information

You can view or download the following related topics:

- DDS for physical and logical files contains reference information for using DDS with physical and logical files.
- DDS for display files contains reference information for using DDS with display files.


- DDS for print files contains reference information for using DDS with printer files.
- DDS for ICF files contains reference information for using DDS with ICF files.

### **Saving PDF files**

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As...**
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

### **Downloading Adobe Acrobat Reader**

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html))  .

---

## **What's New for V5R3 in the DDS concepts information**

Appendix D, "General considerations for using ideographic text (DBCS) with DDS files," on page 45 was updated to document Unicode support.

Also, a what's new topic is available for the rest of the DDS information:

- DDS for physical and logical files
- DDS for display files
- DDS for printer files
- DDS for ICF files



---

## Describing data attributes using DDS

A traditional means of describing data attributes (such as the names and lengths of records and fields) is to specify the data attributes in the application programs themselves. However, a convenient and powerful alternative is available on iSeries servers. Data description specifications (DDS) describe data attributes in file descriptions that are external to the application program that processes the data.

This topic provides the following general information about DDS:

- Instructions for creating files that use DDS
- Rules for keywords and parameters that you should follow when you code DDS
- Naming conventions that you need to follow
- Examples of DDS syntax
- Examples of how to use DDS for each of the file types
- Information on using the REF and REFFLD keywords for DDS files
- DDS keyword and value abbreviations

### Detailed supporting reference information

This topic provides an overview of DDS on iSeries servers. The following iSeries Information Center topics provide detailed reference information for each of the listed types of files:

- DDS Reference: Physical files (DDS is optional)
- DDS Reference: Logical files (DDS is required)
- DDS Reference: Display files (DDS is optional)
- DDS Reference: Printer files (DDS is optional)
- DDS Reference: ICF files (DDS is required)

---

## Creating a file using data description specifications (DDS)

To create a file using DDS, follow these steps:

1. Complete the data description specifications (DDS) form.
2. Type the source statements into a source file. The source file can be part of the OS/400 database (in a source physical file such as the IBM-supplied QDDSSRC in library QGPL) or it can be on diskettes.
3. Create the file using the appropriate create-file command.

**Note:** Using the screen design aid (SDA) utility, you can create and test display files without coding DDS directly, using only the functions that apply to display files. See the ADTS for AS/400®: Screen

Design Aid  book on the V5R1 Supplemental Manuals Web site.

## Completing the DDS form

A sample data description specifications form is printed in reduced size in Figure 1 on page 2.

The left side of the DDS form (positions 1 through 44) is for fixed-format entries called *positional* entries. Positional entries define the most common attributes of record formats and fields, such as names and lengths of fields. For a brief description of the most important positional entries, see items **1** through **7** following the figure. For more detailed information on positional entries for each of the file types, see the following topics:

- Positional entries for physical and logical files

- Positional entries for display files
- Positional entries for printer files
- Positional entries for ICF files

The right side of the DDS form (positions 45 through 80) is for DDS *keywords*. DDS keywords define less-common and more-varied attributes of files, record formats, and fields; they follow a subset of the syntax rules for control language. For a brief description of keyword entries, see item **8** following the figure. For more detailed information on keyword entries for each of the file types, see the following topics:

- Keywords for physical and logical files
- Keywords for display files
- Keywords for printer files
- Keywords for ICF files

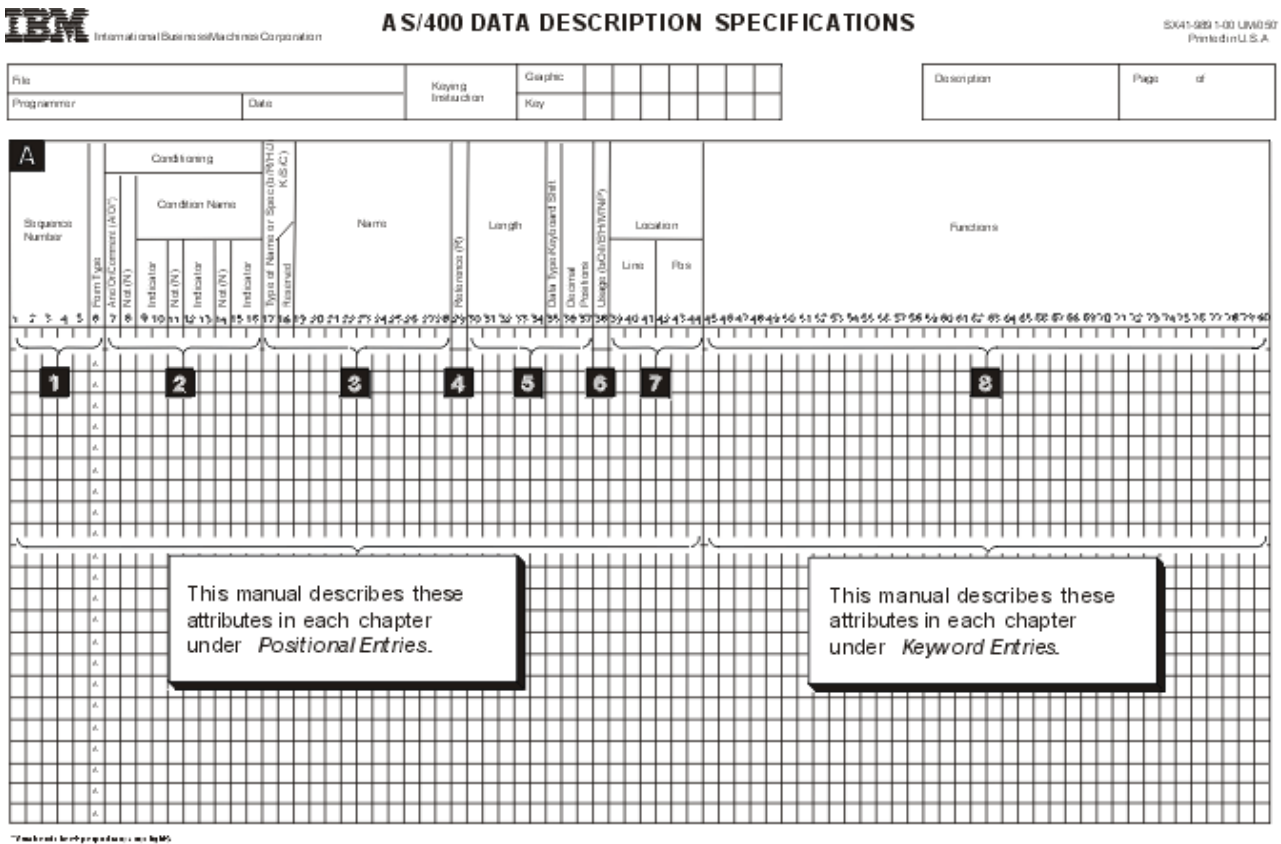


Figure 1. Overview of the Positional Entries and Keywords

- 1** **Sequence Number** and **Form Type** are optional in DDS. The form type identifies the source as DDS source. The entries are valid for all types of files.
- 2** An asterisk in position 7 makes the entire line a comment. This is valid for all types of files. When A (And), or O (Or), or a blank is in position 7, positions 8 through 16 can provide conditioning for the DDS on or immediately following the current line. Conditioning is not valid in physical or logical files.
- 3** **Type of Name or Specification** (position 17) identifies the **Name** entry (positions 19 through 28) or the specification:

Name Entry	Description	Type of File
R	Specifies a record format name	All
blank	Specifies a field name	All
K	Specifies a key field name	Physical and logical only
S	Specifies a select field name	Logical only
O	Specifies an omit field name	Logical only
J	Specifies this as a join specification	Join logical only
H	Specifies this as a help specification	Display only

- 4** R specified in position 29 indicates that the attributes of the field in **Name** (positions 19 through 28) refer to a field specified elsewhere. This is ignored for logical files.
- 5** **Length, Data Type, and Decimal Positions** specify attributes of named fields within record formats. These are valid for all types of files.
- 6** **Usage** specifies fields as input, output, output/input, neither input nor output, hidden, message, or program-to-system fields. Each type of file has its own restrictions regarding field use.
- 7** **Location** specifies the location of the field on the display or printed page. This applies for display and printer files only.
- 8** **Functions** specified through the use of keywords apply at different levels for different file types, as follows:

#### Keywords Apply at This Level

File  
Record  
Field  
Join  
Key field  
Select or omit field  
Help


#### For These Files

All types of files  
All types of files  
All types of files  
Join logical files only  
Physical and logical files  
Logical files only  
Display files only

For display and printer files, constants specified within apostrophes become the default values for displayed or printed fields.

## Entering the DDS source statements

After filling out the forms, enter the source statements into source files. You can enter the source either interactively or in batch.

**Entering Source Statements Interactively with SEU.** Use the source entry utility (SEU) of the WebSphere Development Studio. See the ADTS for AS/400: Source Entry Utility  book on the V5R1 Supplemental Manuals Web site. Type in the Start SEU (STRSEU) command to call SEU.

**Entering Source Statements in Batch Using Diskettes.** Use one of the following methods:

- Enter an input stream containing DDS source and control language (CL) commands on diskette and start a spooling reader using the Start Diskette Reader (STRDKTRDR) command.
- Enter only source statements on diskette and copy the resulting data file into a source physical file with the Copy File (CPYF) command.
- Enter only source statements on diskette and type a create-file command. Specify the name of the data file on the SRCFILE parameter and \*FILE on the SRCMBR parameter of the create-file command.

**Note:** This method does not create a source physical file.

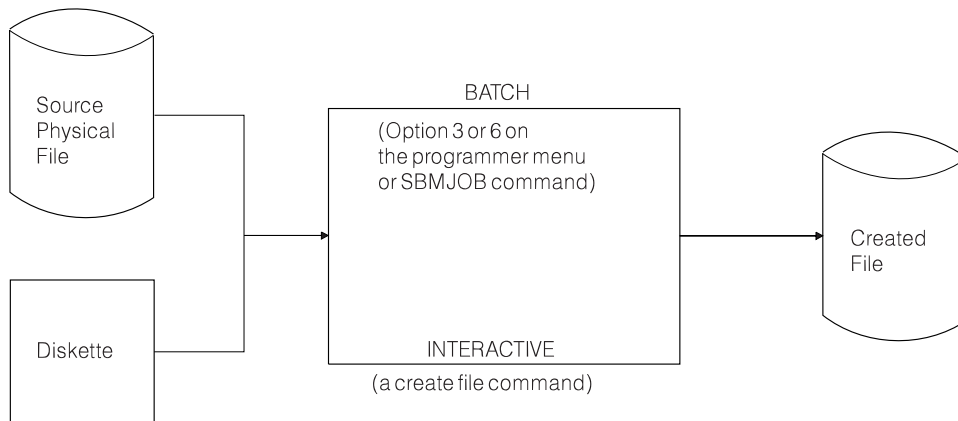
## Creating the DDS file

Create the file by issuing a create-file command. The command you use depends on the type of file you are creating. The file types and their corresponding commands are listed in the following table.

File Type	Command
Physical file	CRTPF
Logical file	CRTLFL
Display file	CRTDSPF
Printer file	CRTPRTF
ICF file	CRTICFF

When you issue a create-file command, the DDS is retrieved from the source file and validated, and a file is created as shown in the following figure. The file is created only if there are no errors in the DDS of equal or greater severity than the severity specified on the GENLVL parameter of the create-file command. Thus, you can use the GENLVL parameter to control the allowable error severity when creating the file. Depending on the options you specify on the OPTION and FLAG parameters, a DDS source (or compiler) listing may also be created. The DDS listing contains the data description and error information.

You can use the FLAG parameter to specify the minimum severity of DDS messages which will be printed. For example, you can suppress the warning messages for field overlapping.



RV2F512-0

See “Example DDS compiler listing” on page 34 for an example of a DDS listing and “DDS debugging template” on page 36 for an example of a debugging template.

---

## Rules for DDS keywords and parameter values

The DDS coding syntax for keywords and their parameter values is similar to the CL syntax. The DDS syntax rules are:

- Code all DDS entries in uppercase except for character values enclosed in apostrophes and extended names enclosed in quotation marks.
- Code keywords on the same (or subsequent) line as the entry with which they are associated.
- Separate multiple keywords with at least one blank. Parameter values for keywords must be enclosed in parentheses. The initial parenthesis must immediately follow the keyword. For example:

```
KEYWORD(VALUE)
```

This rule is slightly different from that in control language. When coding control language, the parameter values can be positional. Syntax for DDS requires that the keyword be specified, except when specifying either a constant or the parameter value for the DFT (Default) keyword.

- Separate multiple parameter values for the same keyword with at least one blank. For example:  
KEYWORD(VALUEA VALUEB)
- A parameter expression consists of a set of values surrounded by left and right parentheses. Generally, the first value within the expression is a special value. The special value begins with an asterisk and must immediately follow the left parenthesis. One or more parameter values follow the special value. Separate the special value and the parameter value(s) by at least one blank. The last parameter value must immediately precede the right parenthesis. A parameter expression represents one parameter value and must be separated from other parameter values by at least one blank. For example:  
KEYWORD(VALUEA (\*special-value VALUEB) VALUEC)
- Use apostrophes to enclose character values. Numeric values appear without apostrophes. See the coding examples for COMP, RANGE, and VALUES keywords. Character values can appear in two places in the syntax:
  - As a parameter value for some keywords. For example, TEXT (all types of files) and COLHDG (database files) require character strings as text description. Other keywords, such as CAnn and CFnn, use character strings as text descriptions for response indicators.
  - As the default value of a constant field (either with or without the DFT keyword) for display and printer files only. In display files, a character constant can also be specified for named fields. Even if you do not specify the DFT keyword, specifying a character constant implies the DFT keyword.
- To specify an apostrophe within a character string, specify two apostrophes so that one apostrophe appears in the output. For example:  
KEYWORD('Customer''s name')  
appears as  
Customer's name
- Use a plus (+) or a minus (–) sign as a continuation character when a keyword and its parameter values do not fit on a single line. The sign must be the last nonblank character in the functions field. A single statement can be continued for a maximum of 5000 character positions.
  - A **minus (–) sign** means the continuation begins in position 45 of the next line (the first position in the functions field).
  - A **plus (+) sign** means the continuation begins with the first nonblank (first significant) character in the functions field on the next line.

If you specify a continuation character within a parameter value, any blanks preceding the continuation character are included in the parameter value.

- Specify a plus (+) sign as the last nonblank character on a line to continue conditioning for keywords specified on the next line. This is helpful when a condition includes several option indicators and applies to several keywords.
- The Operating System/400 (OS/400) program continues a DDS statement until you specify one of the following:
  - A record format name (R in position 17).
  - A field specification (field name or location).
  - For physical or logical files, a key field specification (K in position 17).
  - For logical files, a select or omit specification (S or O in position 17).
  - For join logical files, a join specification (J in position 17).
  - For display files, a help specification (H in position 17).
  - For device files, an option indicator or condition name that conditions a keyword, field, or field location.

- The maximum length of a DDS statement (5000 characters). The fixed length entries (positions 1 through 44) of the first line are included in the statement, so the maximum space available for keywords is 4956.
- Keyword descriptions use the following punctuation marks to indicate the syntax for the keyword:
  - () Enclosed values are required.
  - [] Enclosed values are optional.
  - [...] Specify additional values as needed.
  - { } The upper value is the default value (see REFFLD).
  - | Specify either the value to the left or to the right (may refer to optional values).

---

## DDS naming conventions

The naming conventions used in DDS are as follows:

- Qualified names
  - Use a slash to separate the parts of a qualified name. Embedded blanks are not allowed. For example:  
KEYWORD(library/file)
  - For most keywords with a qualified name parameter value, you can code \*LIBL or \*CURLIB for the library name. If you do not specify a library name, \*LIBL is used. You cannot code \*USRLIBL for the library name. This rule differs from that in CL, which often allows \*USRLIBL.
  - Code a maximum of 10 characters for object names. If you enclose the name in quotation marks, you may specify up to 8 characters between the quotation marks. This rule is different from that in CL, which allows a basic name of up to 10 characters to be specified between the quotation marks. Refer to the Control language topic in the **Programming** category for syntax rules for object names.
- Record and field names
  - The DDS syntax rules for record and field names are:
    - Names must be 10 characters or less.
    - Names must begin with an alphabetic character (A through Z, @, \$, and #). All subsequent characters can be alphanumeric (A through Z, 0 through 9, @, \$, #, and \_ (underscore)). There can be no embedded blanks.
    - In ICF files, record names cannot start with \$\$.
  - Specify qualified field names similar to qualified names. For example:  
KEYWORD(record-name/field-name)

High-level languages can impose specific length and value restrictions on the name. Check the appropriate high-level language reference guide for the syntax requirements for your high-level language processor.

- ALIAS (alternative field) names
  - The length of an alternative field name is 1 to 30 characters. The first character must be A through Z. Subsequent characters must be A through Z, 0 through 9, or the underscore (\_).
  - Because DDS does not perform any language-specific syntax checking, you must make sure that the alternative field names you specify conform to the naming conventions of the high-level language that uses the names. The high-level language compiler checks the syntax of the names when they are brought into the program.
- Message identifiers
  - Message identifiers must be 7 characters long. The first 3 characters are the message prefix.
  - The first character of the message prefix must be an alphabetic character (A through Z, @, \$, and #). The next 2 characters of the message prefix must be alphanumeric (A through Z, @, \$, #, \_, 0 through 9).

- The last 4 characters must be a hexadecimal value (0 through 9, A through F).
- Label, document, and folder names
  - An online help information label name must be from 1 to 10 characters long and must begin with an uppercase alphabetic character (A through Z, @, #, or \$). The label name cannot contain a comma, an apostrophe, or an embedded blank.
  - A document name (and a simple folder name) must be a 1 to 8 character part. It can be followed by a period and a 1 to 3 character part called an extender. The characters used most often are A through Z, 0 through 9, @, #, \$, and \_.
  - If a folder name is concatenated, each simple folder name is separated by a forward slash (/). The total length of the folder name cannot exceed 63 characters.
  - In DDS, a document, simple folder name, or online help information label name can be enclosed in apostrophes. The enclosing apostrophes are required when the name contains an opening or closing parenthesis or an apostrophe character. When the name is enclosed in apostrophes, specify two apostrophes for each apostrophe character within the name. If a folder name is concatenated, the enclosing apostrophes, if specified, must be around the entire concatenated name.

---

## DDS syntax coding examples

This topic provides syntax coding examples. Except for HLPARA, JFILE, JFLD, and PFILE, the keywords shown in these examples are not actual keywords. They simply indicate where you should specify the keywords.

The examples show the syntax for the following types of files:

- Physical file
- Simple logical file
- Join logical file
- Display file
- Printer file
- ICF file

# DDS syntax for a physical file



## AS/400 DATA DESCRIPTION SPECIFICATIONS

SK41-989 1-00 UMD/90  
Printed in U.S.A.

File		Keying Instruction	Graphic Key	Description		Page	of
Programmer	Date						

Sequence Number	Conditioning			Name	Length	Location	Functions
	Indicator	Indicator	Indicator				
00010	*			SYNTAX FOR A PHYSICAL FILE			
00020	*						
00030	*					KEYWORD A	
00040			R	RECORD		KEYWORD B	
00050						KEYWORD C KEYWORD D	
00060				FIELD A	20	KEYWORD E ('This is a test example')	
00070						KEYWORD F (VALUE A)	
00080						KEYWORD G (VALUE B VALUE C)	
00090				FIELD B	40		
00100				FIELD C	5 2	KEYWORD H ('This text example continues with a minus sign')	
00110						KEYWORD I ('This text example continues with a plus sign')	
00120							
00130							
00140			K	FIELD A			

- 1** Comments (optional): Comments can appear on any line in DDS. They are identified by an asterisk in position 7.
- 2** File level (optional): File-level keywords appear before the record format name (RECORD on line 00040).
- 3** Record level (only one record is allowed in physical files): R in position 17 identifies RECORD as a record format name. The record level continues until the first field is named.
- 4** Field level (at least one field name is required, unless the FORMAT keyword is specified on the record): For fields in physical files, specify at least a name and length. Other attributes can be specified explicitly or by default.
- 5** Key field level (optional): K in position 17 identifies the field as a key field. A K must be specified for each key field. Specify the key field level by repeating a field name (here, FIELD A) after the field-level specifications.

**Note:** See the Positional entries for physical and logical files topic for a description of each of the columns shown in the figure.



# DDS syntax for a simple logical file



## AS/400 DATA DESCRIPTION SPECIFICATIONS

SK41-089 1-00 UMD/97  
Printed in U.S.A.

File	Keying Instruction	Graphic Key	Description	Page of
Programmer	Date			

Sequence Number	Conditioning				Name	Length	Data Type/Keyword Set	Location		Functions
	Form Type	Condition Name	Indicator	Indicator				Line	File	
00010	*									1
00020	*									2
00030	*									3
00040	*			R	RECORD1					PF1
00050	*									KEYWORDC KEYWORDD
00060	*				FIELDA					KEYWORDE('This is a text example')
00070	*									KEYWORDF(VALUEA)
00080	*									KEYWORDG(VALUEB VALUEC)
00090	*				FIELDB	40				
00100	*				FIELDC	5	2			KEYWORDH('This text example continues with a minus sign')
00110	*									KEYWORDI('This text example continues with a plus sign')
00120	*									
00130	*									
00140	*			K	FIELDA					
00150	*			S	FIELDB					KEYWORDJ

- 1** Comments (optional): Comments can appear on any line in DDS. They are identified by an asterisk in position 7.
- 2** File level (optional): File-level keywords appear before the record format name (RECORD1 on line 00040).
- 3** Record level (at least one is required): R in position 17 identifies RECORD1 as a record format name. In simple or multiple format logical files, the PFILE keyword is required for every record format. The record level continues until the first field is named.
- 4** Field level: Field names and field attributes are not required for logical files. See the topic for logical file positional entries for more information.
- 5** Key field level (optional): K in position 17 identifies the field as a key field. A K must be specified for each key field. Specify the key field level by repeating one or more field names (such as FIELDA) after the field-level specifications.
- 6** Select and omit levels (optional): S in position 17 identifies FIELDB as a select field. (O in position 17 identifies a field as an omit field.) The select and omit levels follow the key field level.

### Notes:

1. To form a multiple format logical file, specify more record formats within the file by repeating items **3** through **6**, or specify more than one file on the PFILE keyword.
2. See the Positional entries for physical and logical files topic for a description of each of the columns shown in the figure.

# DDS syntax for a join logical file



## AS/400 DATA DESCRIPTION SPECIFICATIONS

SK41-989 1-00 UVA050  
Printed in U.S.A.

File		Keying Instruction	Graphic				Description	Page of
Programmer	Date		Key					
<b>A</b>	Conditioning							
Sequence Number	Condition Name	Name	Length	Location	Functions			
1-6	7-16	17-26	27-36	37-46	47-80			
00010*	SYNTAX	FOR A JOIN LOGICAL FILE						
00020*								
00030*					KEYWORDA			
00040*		R RECORD1			JFILE(PF1 PF2)			
00050*					KEYWORDC KEYWORDD			
00060*		J			JFLD(FIELDA FLD)			
00070*		FIELDA			KEYWORDE('This is a test example')			
00080*					KEYWORDF(VALUEA)			
00090*					KEYWORDG(VALUEB VALUEC)			
00100*		FIELDB	40					
00110*		FIELDC	5	2	KEYWORDH('This test example continues with a minus sign')			
00120*								
00130*					KEYWORDI('This test example continues with a plus sign')			
00140*								
00150*		K FIELDA						
00160*		S FIELDB			KEYWORDJ			

\*An asterisk indicates a comment.

- 1** Comments (optional): Comments can appear on any line in DDS. They are identified by an asterisk in position 7.
- 2** File level (optional): File-level keywords appear before the record format name (RECORD1 on line 00040).
- 3** Record level (exactly one required): R in position 17 identifies RECORD1 as a record format name. In join logical files, the JFILE keyword is required for the record format. The record level continues until the first join specification.
- 4** Join level: J in position 17 identifies the beginning of a join specification. At the join level, specify at least one join specification. Each join specification must include at least one JFLD keyword. There must be one JOIN keyword for each join specification in a join logical file if there is more than one join specification in the file. A join specification continues until the next join specification or field name.
- 5** Field level: At least one field name with usage other than N is required for join logical files.
- 6** Key field level (optional): K in position 17 identifies the field as a key field. A K must be specified for each key field. Specify the key field level by repeating one or more field names (such as FIELDA) after the field-level specifications.
- 7** Select and omit levels (optional): S in position 17 identifies FIELDB as a select field. (O in position 17 identifies a field as an omit field.) The select and omit levels follow the key field level.

**Note:** See the Positional entries for physical and logical files topic for a description of each of the columns shown in the figure.

# DDS syntax for a display file



## AS/400 DATA DESCRIPTION SPECIFICATIONS

SK41-989 1-00 UNW/97  
Printed in U.S.A.

File		Keying	Graphic	Description		Page	
Programmer	Date	Instruction	Key			of	
00010	*						
00020	*						
00030	*				KEYWORDA		
00040	*			R	RECORDA		
00050	*				KEYWORDC KEYWORDD		
00060	*			H	HLPARA(1 1 10 80)		
00070	*				KEYWORDX		
00080	*				FIELDA	20	1 1
00090	*				FIELDB	40x	0 2 3
00100	*				FIELDC	5y	2'B 3
00110	*				FIELDA	20	1 1
00120	*				FIELDB	40x	0 2 3
00130	*				FIELDC	5y	2'B 3
00140	*						
00150	*						
00160	*						

You can specify option indicators and screen size condition names in the shaded positions.

- 1** Comments (optional): Comments can appear on any line in DDS. They are identified by an asterisk in position 7.
- 2** File level (optional): File-level keywords appear before the first record format name (RECORDA on line 00040).
- 3** Record level (at least one required): R in position 17 identifies RECORDA as a record format name. The record level continues until either the first field is specified or the first help specification.
- 4** Help level (optional): H in position 17 identifies the beginning of a help specification. A help specification continues until the next H in position 17 or until the first field. Each help specification must include at least one HLPARA keyword, and a HLPBCD or HLPDOC keyword.
- 5** Field level (optional): Display file fields that are passed between the display device and the program must be named fields and must have a length specified. Other attributes can be specified explicitly or by default. Constant (unnamed) fields require only a location and a keyword, as described in DATE, DFT, TIME, and MSGCON keyword descriptions in the keyword topic for display files. Positions 17 through 38 do not apply to constant fields.

### Notes:

- Items **3** through **5** can be repeated to specify new record formats within the display file.
- See the Positional entries for display files topic for a description of each of the columns shown in the figure.

# DDS syntax for a printer file



## AS/400 DATA DESCRIPTION SPECIFICATIONS

SK41-989 1-00 LW4050  
Printed in U.S.A.

File Programmer	Date	Keying Instruction	Graphic Key								Description	Page of
--------------------	------	-----------------------	----------------	--	--	--	--	--	--	--	-------------	------------

Sequence Number	Form Type or Availability Comment (A007)	Conditioning			Name	Length	Reference (R)	Data Type Keyword (S01)	Default Position Usage (P0405H10NF)	Location		Functions
		Indicator Not (N)	Indicator Not (N)	Indicator Not (N)						Line	File	
00010	*				SYNTAX FOR A PRINTER FILE							
00020	*											
00030	*										KEYWORDA	
00040	*				R RECORDA						KEYWORDS	
00050	*										KEYWORDC	KEYWORDD
00060	*				FIELDA	20				1	3	KEYWORDF('This is a real example')
00070	*											KEYWORDF(VALUEA)
00080	*											KEYWORDG(VALUEB VALUEC)
00090	*				FIELDB	40				2	3	
00100	*				FIELDC	5	2			3	3	KEYWORDH('This is a real example con-tinued with a minus sign')
00110	*											KEYWORDI('This is a real example + continue with a plus sign')
00120	*											KEYWORDJ('This is a real example *')
00130	*											KEYWORDK('This is a real example ^')
00140	*									4	3	'This illegal implies the DFT'
00150	*											KEYWORDL and an unnamed field +
00160	*											signifying at line 4, position 3'

You can specify option indicators in the shaded positions.

- Comments (optional): Comments can appear on any line in DDS. They are identified by an asterisk in position 7.
- File level (optional): File-level keywords appear before the first record format name (RECORDA on line 00040).
- Record level (at least one required): R in position 17 identifies RECORDA as a record format name. The record level continues until the first field is specified.
- Field level (at least one field, whether named or unnamed, is required in each record format in the file): Printer file fields that are passed from the program to the printer must be named fields and must have a length specified. Other attributes can be specified explicitly or by default. Constant (unnamed) fields require only a location and a keyword, as described in the DATE, DFT, PAGNBR, TIME, and MSGCON keyword descriptions in the keyword topic for printer files.

### Notes:

- Items **3** and **4** can be repeated to specify new record formats within the printer file.
- See the Positional entries for printer files topic for a description of each of the columns shown in the figure.

# DDS syntax for an ICF file



## AS/400 DATA DESCRIPTION SPECIFICATIONS

SS41-989 1-00 UMD/97  
Printed in U.S.A.

File	Keying Instruction	Graphic Key	Description	Page of
Programmer	Date			

Sequence Number	Form Type	7 Annotation Comment (AOC)	Conditioning				Name	Length	Location	Function
			Indicator	Indicator	Indicator	Indicator				
00010		*								
00020		*								
00030									KEYWORDA	
00040						R	RECORDA		KEYWORDB	
00050									KEYWORDC	KEYWORDD
00060							FIELDA	20	KEYWORDE('This is a test example')	
00070									KEYWORDF(VALUEA)	
00080									KEYWORDG(VALUEB VALUEC)	
00090							FIELDB	40		
00100							FIELDC	5	2	KEYWORDH('This text example continues with a minus sign')
00110										KEYWORDI('This text example continues with a plus sign')
00120										
00130										

You can specify option indicators in the shaded positions.

- 1** Comments (optional): Comments can appear on any line in DDS. They are identified by an asterisk in position 7.
- 2** File level (optional): File-level keywords appear before the first record format name (RECORDA on line 00040).
- 3** Record level (at least one required): R in position 17 identifies RECORDA as a record format name. The record level continues until the first field is specified.
- 4** Field level (optional): ICF file fields must have at least a name (as in FIELDA) and a length. Other attributes can be specified explicitly or by default.

### Notes:

- Items **3** and **4** can be repeated to specify new record formats within the ICF file.
- See the Positional entries for ICF files topic for a description of each of the columns shown in the figure.



---

## Appendix A. Examples of DDS for Each File Type

This topic provides examples of data description specifications (DDS) for each type of file discussed in this book. If you choose, you can use the examples in this section with appropriate high-level language programs. This topic also provides an example program that uses some of the file examples and illustrates the use of externally described data.

- Database files
  - Field reference file (a physical file used for reference, not data storage)
  - Physical file
  - Logical file
- Device files
  - Display file with help specifications
  - Subfile examples
  - Printer file
  - ICF file
- Example program using the physical, display, and printer file
- Example compiler listing (output from a create-file command)
- IBM Data Description Specifications Debugging Template (reduced)

---

### Examples of database files in DDS

This topic contains the following examples:

- a field reference file
- a physical file
- a logical file specifying multiple formats and new keys
- a logical file specifying a new record format
- a join logical file

### Example of a field reference file

This example defines all of the fields used in an application and refers to fields only within the field reference file itself. The following keywords are important in the example:

```
COLHDG  
EDTCDE(Z)  
REFFLD  
REFSHIFT  
TEXT
```

The following field reference file (MLGREFF) describes all fields used by any program in the application. The other files use the fields in this file.

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A** FLDREF  MLGREFP  MAILING LIST FIELD REFERENCE FILE
00020A  1 R MLGREFR          TEXT('Mailing List Field Reference')
00030A          ACTNUM      5 0      COLHDG('Account' 'Number')
00040A          EDTCDE(Z)
00050A          ACTTYP      1 0      COLHDG('Acct' 'Type')
00060A          TEXT('Acct Type 1=Bus 2=Gvt +
00070A          3=Org 4=Sch 5=Pvt 9=Oth')
00080A          NAME        18      COLHDG('Name')
00090A          REFSHIFT(X) 4
00100A          ADDR        R 2      2 REFFLD(NAME)
00110A          COLHDG('Address') 3
00120A          CITY        R 2      2 REFFLD(NAME)
00130A          COLHDG('City') 3
00140A          STATE       2      COLHDG('State')
00150A          ZIP         5 0      COLHDG('ZIP' 'Code')
00160A          EDTCDE(X)
00170A          BATNUM      6 0      COLHDG('Batch' 'Number')
00180A          EDTCDE(Z)
00190A          TRNTYP      1      COLHDG('Trans' 'Type')
A

```

Figure 2. DDS for a Field Reference File (Part 1 of 2)

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00200A          TEXT('Trans Type A=Add +
00210A          C=Change D=Delete')
00220A          XACTNM      R      REFFLD(ACTNUM)
00230A          XACTTTP     R      REFFLD(ACTTYP)
00240A          XNAME       R      REFFLD(NAME)
00250A          XADDR       R      REFFLD(ADDR)
00260A          XCITY       R      REFFLD(CITY)
00270A          XSTATE     R      REFFLD(STATE)
00280A          XZIP        R      REFFLD(ZIP)
00290A          TRNNUM      5 0      COLHDG('Transaction' 'Number')
00300A          EDTCDE(Z)
00310A          MLGLK1      3 0      COLHDG('Lock' 'Control')
00320A          TEXT('Control Number Used for +
00330A          record locking')
A

```

Figure 2. DDS for a Field Reference File (Part 2 of 2)

**Legend:**

- 1** Like all physical files, a field reference file has only one record format. The R in position 17 specifies that MLGREFR is the record format name.
- 2** The Rs in position 29 and REFFLD in positions 45 through 80 specify that the fields ADDR and CITY are to have the same attributes as NAME.
- 3** Specifying COLHDG for ADDR and CITY overrides the COLHDG attribute for NAME, which otherwise would have been in effect.
- 4** Specifying REFSHIFT for NAME will cause the keyboard shift specified (X) to be used when this field (NAME) is referred to in a display file.

**Example of a physical file with a new record format**

The REF keyword is important in the following example. This file has one record format. The names of all fields in the record format are specified. This example uses fields in a reference file (REF keyword) and uses a keyed-sequence access path.



The following physical file (called CUSMSTP for customer master physical file) describes the fields physically present in the database.

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A*
00020A* SAMPLE PHYSICAL FILE(CUSMSTP)
00030A*
00040A
00050A      2  R CUSMST
00060A          ACTNUM  R  3
00070A          NAME    R  3
00080A          ADDR    R  3
00090A          CITY    R  3
00100A          STATE   R  3
00110A          ZIP     R  3
00120A      4  SEARCH    10 0
00130A      4  CRDLMT     8 2
00140A      5  K ACTNUM
A

```

1 REF(MLGREFP)  
TEXT('Customer Master Record')

Figure 3. DDS for a Physical File

**Legend:**

- 1 At the file level, the REF keyword refers the OS/400 program to the physical file MLGREFP, which is a field reference file for this database.
- 2 At the record level, R in position 17 specifies that CUSMST is the record format name of the record in this file. (There can only be one record format in a physical file.)
- 3 At the field level, Rs in position 29 specify that the attributes of fields of the same name in the REF file are to be used as attributes of these fields.
- 4 The fields SEARCH and CRDLMT are not defined in MLGREFP; therefore, their field attributes are specified here.
- 5 At the key field level, K in position 17 specifies that ACTNUM is the key field for the file.

**Example of a logical file specifying multiple formats and new keys**

The PFILE keyword is important in the following example. The example uses new field specifications and provides two record formats. Each record format provides a different view of the associated physical file and uses a key different from the associated physical file.

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A*
00020A* SAMPLE LOGICAL FILE
00030A
00040A      R CUSMST1          1 PFILE(CUSMSTP)
00050A      ACTNUM
00060A      NAME
00070A      STATE
00080A      LASTNAME          I  3 SST(NAME 8 10)
00090A  2 K ACTNUM
00100A*
00110A      R CUSMST2          1 PFILE(CUSMSTP)
00120A      ACTNUM
00130A      NAME
00140A      ZIP
00150A      K *NONE
00160A  2 K NAME
      A

```

Figure 4. DDS for a Logical File Specifying New Keys

**Legend:**

- 1 The two record formats (CUSMST1 and CUSMST2) in this logical file are based on the same physical file (CUSMSTP).
- 2 Record format CUSMST1 has a key different from record format CUSMST2, providing the application program with a different sequence of the same records.
- 3 The LASTNAME field is a substring of the field NAME. The usage I in position 38 must be specified since this is not a join logical file.

**Example of a logical file specifying a new record format**

The UNIQUE keyword is important in the following example. The example specifies a record format different from the associated physical file.

The following logical file (called CUSMSTL2 for customer master logical file two) uses some of the fields in the physical file CUSMSTP. Another logical file could name all fields, name fields in other physical files as well, concatenate fields, change the order of fields, rename fields, or choose different key fields. In this logical file, the programmer merely omitted some fields from the physical file.

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A*
00020A* SAMPLE LOGICAL FILE (CUSMSTL2)
00030A*
00040A      R CUSREC          1 UNIQUE
00050A      PFILE(CUSMSTP)  2
00060A      TEXT('Logical File Master Record')
00070A      ACTNUM          3
00080A      NAME            3
00090A      ADDR            3
00100A  4 K ACTNUM
      A

```

Figure 5. DDS for a Logical File

**Legend:**

- 1 The UNIQUE keyword specifies that records with duplicate keys are not allowed within a member of this logical file.
- 2 The keyword PFILE (required for logical files) specifies CUSMSTP.

- 3 The field names do not have R specified in position 29 as they would in physical files or in any device file.
- 4 As in CUSMSTP, the field ACTNUM is treated as a key field.

## Example of a join logical file

The following join logical file joins three physical files (PF1, PF2, and PF3) so that an application program can get name, address, and salary information in one input operation, even though the information is stored in three different physical files. The following keywords are important in the example:

- JFILE
- JFLD
- JOIN
- JREF

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00010A      1 R JOINREC      2 JFILE(PF1 PF2 PF3)
00020A      J              JOIN(PF1 PF2)
00030A      3              JFLD(NAME NAME)
00040A      J              JOIN(PF2 PF3)
00050A      J              JFLD(NAME NAME)
00060A      4 NAME          5 JREF(1)
00070A      4 ADDR
00080A      4 PHONE
00090A      4 SALARY
A

```

Figure 6. DDS for a Join Logical File

### Legend:

- 1 R identifies the record format. There can be only one record format in a join logical file.
- 2 The JFILE keyword specifies that PF1, PF2, and PF3 are the physical files on which this join logical file is based. Because it is specified first, PF1 is the primary file. There are two secondary files: PF2 and PF3.
- 3 J identifies the join specifications. With two secondary files in this join logical file, there must be two join specifications. Each join specification defines how a pair of files is to be joined, as follows:
  - JOIN is required when more than two physical files are being joined, and it identifies which two files are being joined in this join specification. In the first join specification, PF1 and PF2 are joined. In the second join specification, PF2 and PF3 are joined.

**Note:** Secondary files can be joined to either the primary file or to another secondary file. In this example, PF2 in the second JOIN keyword could be PF1; there would be no difference in the order of records supplied to the program or in performance.

- JFLD identifies which fields are used to link together records from the physical files being joined. In the first join specification, NAME from PF1 links with NAME from PF2. In the second join specification, NAME from PF2 links with NAME from PF3.
- 4 The field names show which fields are presented to the program. At least one field name is required.
- 5 The JREF keyword identifies which physical file to search for the field name; in this example, NAME from PF1 is used. Note the use of the direct file number: JREF(1) indicates to use the first file on the JFILE keyword, which is PF1.

---

## Examples of device files in DDS

This topic contains the following examples:

- an inquiry display file
- a subfile with SFLPAG value equal to SFLSIZ value
- a subfile with paging by OS/400 program and high-level language program
- a horizontal subfile displayable on two display sizes
- a message subfile
- a printer file
- an ICF file

## Example of an inquiry display with two record formats in DDS

The following display is defined by the DDS in the example. It is displayed by output operations to the record formats PROMPT and RESPONSE.

```
CUSTOMER FILE ADD/UPDATE

Enter new or existing customer number
Enter A to ADD new Customer

Name      XXXXXXXXXXXXXXXXXXXXXXXX
Address   XXXXXXXXXXXXXXXXXXXXXXXX
City      XXXXXXXXXXXXXXXXXXXXXXXX
State     XX      Zip code NNNNN

Credit limit      $NNN,NNN.NN
```

```
F3 - End Program & Print Report  F6 - Return to prompt
```

*Name* XXXXXXXXXXXXXXXXXXXXXXXX

If the cursor is positioned in the area with Xs on the NAME field and the Help key is pressed, online help information will appear.

The following keywords are important in the example:

CAnn	HELP
CHECK	HLPARA
DSPATR(HI BL)	HLPRCD
DSPATR(UL)	HLPBDY
EDTCDE(Y)	HLPDOC
EDTCDE(2 \$)	OVERLAY
ERRMSG	PRINT

The example uses +n to specify position.

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00010A          1 PRINT
00020A          CA03(21 'End & Print')
00030A          CA06(22 'Display PROMPT')
00040A          7 HELP
00050A          8 HLPDOC(START GENERAL HELP)
00060A          2 R PROMPT
00070A          H
00080A          9 HLPDOC(LBL1 HELP#1 HELP)
00090A          9 HLPARA(2 2 2 50)
00100A          1 30'CUSTOMER FILE ADD/UPDATE'
00110A          3 2'Enter new or existing customer +
00120A          3 ACTNUM          5 0B  +1CHECK(MF)
00130A          40 4 ERRMSG('Customer number not +
00140A          4 found' 40)
00150A          4 2'Enter A to ADD new Customer'
00160A          1 I  +1
00170A          5 R RESPONSE      6 OVERLAY
00180A          H                10 HLPDCD(NAMEHELP)
00190A          H                11 HLPDCD(ADDHELP)
00200A          H                11 HLPDCD(ADDHELP)
00210A          H                11 HLPDCD(ADDHELP)
00220A          H                12 HLPBDY
00230A          H                12 HLPDCD(HELPCD1 HELPFILE)
00240A          13 HLPARA(12 18 12 40)
A
A
A

```

Figure 7. Display with Two Record Formats (Part 1 of 2)

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00250A*
00260A          6 2'Name'          14
00270A          NAME          18 B 6 10          14
00280A          7 2'Address'      14
00290A          ADDR          18 B 7 10          14
00300A          8 2'City'         14
00310A          CITY          18 B 8 10          14
00320A          9 2'State'        14
00330A          STATE         2 B 9 10          14
00340A          9 19'Zip code'    14
00350A          ZIP           5Y 0B +1          14
00360A          12 2'Credit Limit'
00370A          15 CRDLMT       8Y 2B 12 21EDTCDE(2 $) DSPATR(HI) 16
00380A          17 23 2'F3 - End Program & Print Report +
00390A          F6 - Return to prompt'
00400A*
00410A*  HELP RECORDS
00420A*
00430A          R NAMEHELP
00440A          2 2'HELP TEXT FOR NAME FIELD'
00450A          4 2'ENTER THE CUSTOMER NAME'
00460A          R ADDRHELP
00470A          4 2'HELP FOR ADDRESS,CITY,STATE,ZIP'
00480A          6 2'ENTER ADDRESS,CITY,STATE & ZIP'
A
A
A

```

Figure 7. Display with Two Record Formats (Part 2 of 2)

**Legend:**

- 1 The PRINT keyword allows the display station user to print the display at any time by pressing the Print key.
  - 2 An application program would display a prompt by issuing an output operation to the record PROMPT, displaying the constant fields 'CUSTOMER FILE ADD/UPDATE', 'ENTER EXISTING CUSTOMER NUMBER', 'ENTER A TO ADD NEW CUSTOMER', and the named fields ACTNUM and ADD.
  - 3 The CHECK(MF) keyword specifies that when the user types into one position of the field ACTNUM, he must type into all five positions before pressing the Enter key, or an error message is displayed and the keyboard is locked. The user must press the Reset key and reenter through the input field.
  - 4 If record format PROMPT is displayed and your program sets on indicator 40 when an output operation is sent to record format PROMPT, the error message 'Customer number not found' is displayed on the message line (line 24 of the 24-line display unless the MSGLOC keyword is specified). The message is highlighted, field ACTNUM is displayed with its image reversed, and the keyboard is locked until the user presses the Reset key.
  - 5 After the user presses the Enter key, the application program retrieves the desired information from the database and sends an output operation to record format RESPONSE, displaying the fields described in the next paragraphs.
  - 6 The OVERLAY keyword specifies that an output operation to this record format (RESPONSE) does not cause the entire display to be cleared, as it would be by default.
  - 7 The HELP keyword enables the Help key for this display.
  - 8 The HLPDOC keyword specified at the file level identifies the document to be displayed when no help area for the active records contains the current cursor location.
  - 9 The HLPDOC and HLPARA keywords on this H specification specify that the HELP#1 document in the HELP folder will be displayed starting at the LBL1 help label if the Help key is pressed while the cursor is in positions 2 through 50 of line 2.
  - 10 The HLPRCD and HLPARA keywords on this H specification cause the record NAMEHELP to be displayed if the Help key is pressed while the cursor is in positions 10 through 28 of line 6. The record NAMEHELP is defined in this display file; therefore, a file name does not have to be specified on the HLPRCD keyword.
- Note:** When using application help keywords, the screen is automatically cleared.
- 11 The HLPRCD and HLPARA keywords on this H specification cause the ADDRHELP record to be displayed if the Help key is pressed while the cursor is in positions 10 through 28 of lines 7, 8 or 9.
  - 12 The HLPBDY keyword limits the help records displayed when the Page key is pressed. If either NAMEHELP or ADDRHELP is displayed when the Help key is pressed, the NAMEHELP and ADDRHELP records are accessible using the Page key. If HELPRCD1 is displayed when the Help key is pressed, the help records from other H specifications are not accessible using the Page key.
  - 13 The HLPRCD and HLPARA keywords on this H specification cause the record HELPRCD1 in the file HELPFIL to be displayed if the Help key is pressed while the cursor is in positions 18 through 40 of lines 12 through 40. This record is in a separate display file called HELPFIL.
  - 14 Five constant fields ('Name', 'Address', 'City', 'State', and 'Zip Code') and five named fields (NAME, ADDRESS, CITY, STATE, ZIP) are grouped together on the display by the line/position specifications. NAME, ADDRESS, CITY, and STATE default to character-type fields (A in position 35) because no decimal positions are specified. ZIP is a numeric-only, integer field (Y in position 35; 0 in position 37), so its display length equals its specified length.
  - 15 The field CRDLMT is specified with EDTCDE (2 \$). EDTCDE(2) is used for monetary amounts and the \$ specifies the floating currency symbol.



- 1** The subfile record format SFL1 and the subfile control-record format SFLCTL1 together define one subfile. The parameter value for the SFLCTL keyword is the name of the subfile record format.
- 2** Each subfile record is made up of two fields: FLD1 and FLD2. FLD1 is 10 bytes long (11 bytes display length because it defaults to signed numeric); FLD2 is 16 bytes long. FLD1 is an input-only field; FLD2 is an output-only field. Eighteen subfile records would appear on the display, with the first one on line 3 and the last one on line 20. For each subfile record on the display, two fields (FLD1 and FLD2) would appear, with four spaces between FLD1 and FLD2.
- 3** SFLSIZ and SFLPAG (required keywords) have equal values (18). Therefore one page equals the whole subfile. For all subfiles, the value of the SFLPAG keyword is the number of subfile records displayed at any one time (unless the SFLDROP keyword or variable-length records are used).
- 4** SFLDSP (a required keyword) and SFLDSPCTL (an optional keyword) are specified with indicator 05. Therefore, when indicator 05 is set on, the subfile and subfile control records can be displayed by an output operation to the subfile control-record format SFLCTL1.
- 5** SFLCLR (an optional keyword) is specified with option indicator 05 preceded by an N. When indicator 05 is set off, the subfile can be cleared by an output operation to SFLCTL1.
- 6** ROLLUP (an optional keyword) is specified with response indicator 01, and ROLLDOWN (an optional keyword) is specified with response indicator 02. Note also that the entire subfile equals one page, which means that the whole subfile is displayed at one time. Therefore, when the display station user presses the Page Up key, control passes to the program with indicator 01 on, and when the work station user presses the Page Down key, control passes to the program with indicator 02 on. The program must handle paging, by reading, clearing, rewriting, and redisplaying the subfile. Without ROLLUP and ROLLDOWN specified, the work station user would receive an error message when pressing the Page Up or Page Down key.
- 7** Two constants ('First Field' and 'Second Field') are displayed when the subfile control-record is displayed (SFLDSPCTL in effect). As specified in this subfile, they act as column headings to the subfile records.

## **Example of a subfile with paging by OS/400 program and high-level language program**

The following example describes a combined method of paging a subfile that uses system resources efficiently. In some applications, the number of records in the subfile could be quite large. However, the application user may want to view only the first page or two of these records. In this case, it may be faster and more efficient for the application program to build the subfile a page at a time as requested by the user. The OS/400 program handles the paging of records in the subfile. It also returns a Page Up key indicator to the high-level language program when another page should be added to the end of the subfile. The application user can detect no difference between a page up request handled by the OS/400 program and one handled by the high-level language program.



NAME	DEPARTMENT	PHONE
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXX	NNNN

The following keywords are important in the example:

SFLPAG  
SFLSIZ

The SFLSIZ value is larger than the SFLPAG value. The subfile is paged by the SFLPAG value.

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R SETSFL              SFL
00020A  50          SFLNXTCHG
00030A          NAME          30  0  5  2
00040A          DEPT          10   5 40
00050A          PHONE         4  0  5 58
00060A          R SETCTL              SFLCTL(SETSFL)
00070A                                          SFLSIZ(0034) 1
00080A                                          SFLPAG(0017)
00090A  40          SFLDSP
00100A  41          SFLDSPCTL
00110A  42          SFLDLT
00120A  43          SFLCLR
00130A  49          SFLEND          2
00140A N49         ROLLUP(26)
00150A              LOCK
00160A              OVERLAY
00170A          SETRRN          4S 0H  SFLRCDNBR(CURS) 3
00180A                                          3  2'NAME'
00190A                                          3 40'DEPARTMENT'
00200A          A          3  58'PHONE'

```

Figure 9. Subfile with Paging by OS/400 Program and High-Level Language Program in DDS

Legend:

- 1** The SFLSIZ value must be greater than the SFLPAG value so that the OS/400 program will handle paging within the subfile. A maximum of 9 999 records can be stored in the subfile.
- 2** The SFLEND keyword can be specified with the ROLLUP keyword. One indicator can be used to option both keywords. The application program turns on the indicator to disable the Page Up key and omit the plus sign (+) on the last subfile page when the last page of the subfile is displayed.

**3** The SFLRCDNBR keyword should be specified so the last subfile page can be displayed after it is built by the high-level language program.

Records in the subfile with changed input fields (modified data tags) will be changed after a new page is added to the subfile by the high-level language program.

**Example of a horizontal subfile displayable on two display sizes**

The following two displays show the subfile defined in the example as it appears on the 24 x 80 and 27 x 132 display sizes, respectively:

```
COLUMN 1          COLUMN 2
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
```

```
COLUMN 1          COLUMN 2
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
____ XXXXXXXXXXXXXXXX  ____ XXXXXXXXXXXXXXXX
```

The following keywords are important in the example:

- DSPSIZ
- SFLLIN

Subfile records appear in two columns (SFLLIN keyword). The subfile can be displayed on two display sizes (DSPSIZ keyword).

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00010A* HORIZONTAL SUBFILE ON TWO DISPLAY SIZES
00020A*
00030A
00040A          1  DSPSIZ(*DS3 *DS4)
00040A          R SFL1          SFL
00050A          FLDA          10Y 0I 3 11
00060A          FLDB          16 0 3 23
A
00070A          R SFLCTL1          SFLCTL(SFL1)
00080A          SFLSIZ(50)
00090A          2  SFLPAG(16)
00100A *DS4          2  SFLPAG(40)
00110A          3  SFLLIN(5)
00120A *DS4          3  SFLLIN(5)
A
00130A 01          SFLEND
00140A 02          SFLDSP
00150A 03          SFLDSPCTL
00160A 04          SFLCLR
A
00170A          1 21' COLUMN 1'
A
00180A          1 55' COLUMN 2'
A

```

Figure 10. Horizontal Subfile on Two Display Sizes

#### Legend:

- 1** There is one keyword at the file level, the keyword DSPSIZ (optional). This keyword has two values, \*DS3 and \*DS4, which indicate that the primary display size is 24 x 80, and the secondary display size is 27 x 132.
- 2** The SFLPAG keyword (required), is specified once with a value of 16 and again with a value of 40. The first time it applies to a device with the primary display size (default of \*DS3, or 24 x 80); the second time, coded with a condition name of \*DS4, it applies to a device with the secondary display size (27 x 132).
- 3** The SFLLIN keyword causes a subfile to be displayed horizontally. The parameter value specifies the number of spaces between columns of records. In this example, five spaces separate columns of records on both the 24 x 80 display size (\*DS3) and the 27 x 132 display size (\*DS4). Because \*DS3 is the primary display size, it does not need to be specified in positions 9 through 12.

## Example of a message subfile using DDS

The following display shows the message subfile defined in the example:

```
MESSAGE SUBFILE

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

The following keywords are important in the example:

SFLMSGKEY  
SFLPGMQ  
SFLMSGRCD

Records in the subfile are messages from a message file.

```
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00010A*
00020A* MESSAGE SUBFILE
00030A*
00040A          R SFLR                      SFL
00050A          SFLMSGRCD(3) 1
00060A          MSGKEY 2          SFLMSGKEY 2
00070A          PGMQ 2          SFLPGMQ 2
          A
00080A          R STLCTLR                 SFLCTL(SFLR)
00090A          SFLSIZ(12) 3
00100A          SFLPAG(6) 3
00110A 01          SFLDSP
00120A 02          SFLDSPCTL
00130A 03          SFLCLR
00140A 04          SFLEND 3
          A
          A
00150A          1 32'MESSAGE SUBFILE'
          A
```

Figure 11. Message Subfile

### Legend:

- 1 Specifying the SFLMSGRCD keyword on the subfile record format identifies this subfile as a *message subfile*. The parameter value specified causes the subfile to appear on line 3 of the display.
- 2 The fields MSGKEY and PGMQ are user-defined names given to the two fields required for the

subfile record format for a message subfile. The only specifications allowed for them are their names and the SFLMSGKEY and SFLPGMQ keywords, in the order shown.

This subfile is built by a series of output operations to SFLR that place messages in the subfile as subfile records. Messages are truncated to fit single lines (76 characters or 128 characters, depending on display size), and second-level help is available. This subfile is displayed by an output operation to SFLCTLR with option indicator 01 set on.

- 3** This subfile is paged by the OS/400 program when the display station user presses a Page Up or a Page Down key. The SFLEND keyword allows the OS/400 program to display a plus sign whenever the subfile can be paged up.

## Example of a printer file using DDS

The following printer file example contains DDS for printing a customer master list.

The following keywords are important in the example:

EDTCDE(Y)	UNDERLINE
EDTCDE(Z)	BARCODE
PAGNBR	CHRSIZ
SKIPB	COLOR
SPACEA	

This printer file uses space and skip keywords instead of line numbers.

```

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A*
00020A* SAMPLE PRINTER FILE
00030A*
00040A
00050A      R HEADER
00060A
00070A
00080A
00090A
00100A
00110A
00120A
00130A
00140A
00150A
00160A
00170A
00180A
00190A
      A

```

**1** REF(MLGREFP)  
TEXT('TWO-LINE HEADING, UNDERLINED')

**2** SKIPB(2)

**2** 29'CUSTOMER MASTER FILE'

**2** 75DATE EDTCDE(Y)

**2** +1TIME

**2** 122'Page'

**2** +1PAGNBR EDTCDE(Z) SPACEA(2)

**2** 2'ACCOUNT CUSTOMER'

**2** SPACEA(1)

**2** 2'NUMBER NAME +

**2** ADDRESS +

**2** CITY +

**2** STATE ZIP '

**3** UNDERLINE

**3** SPACEA(2)

Figure 12. Printer File (Part 1 of 2)

	...	1	...	2	...	3	...	4	...	5	...	6	...	7	...	8
00200A				R	DETAIL											TEXT('ONE-LINE RECORD')
00210A										4						SPACEA(5)
00220A				ACTNUM		R				4						2CHRSIZ(2 2)
00230A				NAME		R				5						+4COLOR(BLU)
00240A				ADDR		R										+3
00250A				CITY		R										+3
00260A				STATE		R				6						+3BARCODE(CODE30F9 4 *NOHRI *AST)
00270A				ZIP		R										+5
																A

Figure 12. Printer File (Part 2 of 2)

**Legend:**

- 1** This printer file refers to the field reference file MLGREFP.
- 2** When SKIPB(2) is specified at the record level, the printer skips to line 2 before printing the record format (HEADER). Also, line numbers in positions 39 through 41 are not allowed.
- 3** UNDERLINE is a field-level keyword that causes the constant field preceding it to be underlined on the printout.
- 4** The CHRSIZ keyword specified here causes the ACTNBR field to print with its height and width expanded by 2.
- 5** The COLOR keyword causes the NAME field to print in blue if you use a printer that supports color (the 4224 Printer).
- 6** The BARCODE keyword specified for the STATE field causes the CODE30F9 bar code to print for the STATE field if you use an IPDS™ printer.

**Example of an ICF file using DDS**

The following example contains DDS for transmitting data between the iSeries server and a remote system or device.

The following keywords are important in the example:

CONFIRM	RECID
DETACH	SYNLVL
EVOKE	EOS
RCVDETACH	RSPCONFIRM
RCVFAIL	ALWWRT
RCVCONFIRM	FAIL
RCVTRNRND	RQSWRT

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00010A*
00020A*  SAMPLE ICF FILE
00030A*
00040A  01                               3  EOS
00050A                               2  RCVTRNRND(01 'TRNRND INDICATION')
00060A                               2  RCVDETACH(02 'DETACH RECEIVED')
00070A                               2  RCVCONFIRM(03 'CONFIRM REQUEST')
00080A                               2  RCVFAIL(04 'FAIL RECEIVED')
00090A
00100A      R CATCHALL
00110A      FLD1          132A
00120A*
00130A      R SNDEVOKE          EVOKE(&LIBNME/&PGMNME);
00140A      1  SYNLVL(*CONFIRM) SECURITY(2 PASSWRD)
00150A      1  CONFIRM
00160A      PGMNME          10A P
00170A      LIBNME          10A P
00180A      PASSWRD        8A P
00190A*
00200A      R HEADER          RECID(1 'H')
      A

```

Figure 13. ICF File (Part 1 of 2)

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00210A  09                               CONFIRM
00220A      ID          1A
00230A      4  PART#      12A
00240A      UNTCST      6S 2
00250A      QTYONORDR   9B 0
00260A      TOTAL       9B 0
00270A*
00280A      R DETAIL          RECID(1 'D') RECID(1 'E')
00290A  09                               CONFIRM
00300A      ID          1A
00310A      4  LOC       6A
00320A      QTY        9B 0
00330A*
00340A      R COMMANDS
00350A  05                               5  FALL
00360A  06                               5  ALWVRT
00370A  07                               5  DETACH
00380A  08                               5  RQSWRT
00390A  09                               5  CONFIRM
00400A  10                               5  RSPCONFIRM
      A

```

Figure 13. ICF File (Part 2 of 2)

**Legend:**

- 1** The record format SNDEVOKE causes the program specified in the field PGMNME and the library specified in the field LIBNME to be started on the remote system. It also establishes a synchronization level of \*CONFIRM for the transaction and passes the data in the field PASSWRD as security information. The CONFIRM keyword requests that the remote system confirm the start of the program.
- 2** If the remote program does any of the following:
  - Requests to end sending data
  - Requests to end the transaction
  - Requests to confirm receiving the data

- Sends a FAIL

this sets on one of the following response indicators:

- 01 (the RCVTRNRND keyword)
- 02 (the RCVDETACH keyword)
- 03 (the RCVCONFIRM keyword)
- 04 (the RCVFAIL keyword)

**3** The EOS keyword causes the session to end if indicator 01 is on and the program issues an output operation.

**4** The iSeries server sends and receives data in the form of header (record format HEADER) and detail (record format DETAIL) records. If your program is sending, option indicator 09 can be set on (enabling the CONFIRM keyword) to request that the remote system confirms receiving the data.

When receiving, the record selection processing (RECID keyword) determines which record is received. If an H is in position 1, record format HEADER is selected. If a D or E is in position 1, record format DETAIL is selected. If anything else is in position 1, (unexpected record format, application error, or indicators with no data, RCVDETACH, RCVFAIL, and so on) record format CATCHALL is selected.

**5** Record format, COMMANDS, is used to request the following communications functions:

- If indicator 05 is on, the FAIL function is performed.
- If indicator 06 is on, the ALWWRT function is performed.
- If indicator 07 is on, the DETACH function is performed.
- If indicator 08 is on, the RQSWRT function is performed.
- If indicator 09 is on, the CONFIRM function is performed.
- If indicator 10 is on, the RSPCONFIRM function is performed.

---

## Example program that uses a physical, display, and printer file

The sample program shown in this topic illustrates the use of externally described data in a program. This program uses the sample physical file, display file, and printer file given in this appendix.

If you enter the DDS for these files on your system and create them using the Create Physical File (CRTPF) command, the Create Display File (CRTDSPF) command, and the Create Printer File (CRTPRTF) commands, this program allows you to add records to the physical file, display and update the records, and print a report.

The program is written in RPG. You can enter the RPG specifications shown in the example into a source file on your system and create the program using the Create RPG Program (CRTRPGPGM) command.

For more information on RPG, refer to the RPG/400® Reference  book in the V5R1 Supplemental Manuals Web site.





value is found, indicator 30 is turned on. The program continues to prompt until the key value of an existing record is entered in field ACTNUM (indicator 30 off) or an A is entered in field ADD.

**5** This section adds a new record or updates an existing record in the database file.

If a new customer is being added (indicator 30 is on), the WRITE op code adds a new record to the physical file. Otherwise, the UPDAT op code updates an existing record. The program continues to prompt for, retrieve, add, and update records in the physical file until F3 is pressed, setting on indicator 21.

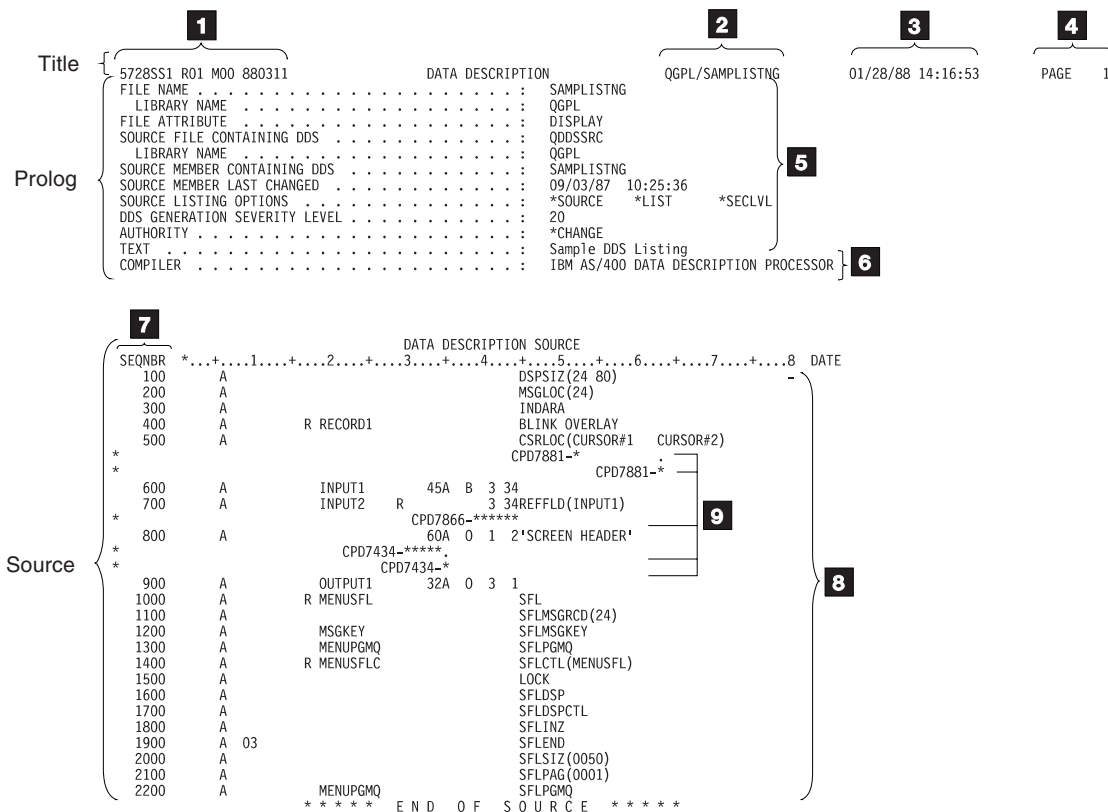
**6** This section prints the report.

A record is read from the physical file and the DETAIL record is written to the printer file until the end of the physical file is reached (indicator 45 is set on). The HEADER record is written on the first page and then written again on each new page (indicator 10 is on). When all the records have been written, the CLOSE op code closes all the files and SETON LR ends the program.

## Example DDS compiler listing

Once data description specifications are written, they must be put into a source file. Then, database or device files are created by entering the CL command that starts the data description processor. You can enter the CL command interactively or in a batch job. The data description processor retrieves the data description specifications from the source file designated on the create-file command, validates the specifications, and creates a computer printout with any errors and any referenced specifications.

An example of a data description specifications compiler computer printout follows:



RSLL913-2

Figure 15. DDS Compiler Computer Printout (Part 1 of 2)

```

Title { 5728SS1 R01 M00 880311          DATA DESCRIPTION          QGPL/SAMPLISTNG          01/28/88 14:16:53          PAGE 2
        EXPANDED SOURCE

SEQNBR *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8      FIELD      BUFFER POSITION
        *DS3                                MSGLOC(24)                                OUT      IN
100                                           DSPSIZ(24 80) INDARA
400           R RECORD1                                BLINK OVERLAY
600           INPUT1  45A B 3 34
700           INPUT2  45A O 3 34
* REFERENCED FIELD . . . . . : INPUT1
* FORMAT . . . . . : RECORD1
* FILE . . . . . : *SRC
* LIBRARY . . . . . :
800                                           2'SCREEN HEADER'
900           OUTPUT1  32A O 3 1
1000          R MENUSFL
1000          *DS3                                SFLMSGRCD(24)
1200          MSGKEY                                SFL
1300          MENUPGMQ                                SFLMSGKEY
                                           SFLPGMQ
* OPTION INDICATOR OUTPUT BUFFER POSITIONS:
* *IN03
1400          R MENUSFLC
1400          *DS3                                SFLSIZ(0050) SFLPAG(001)
1700          03                                  SFLCTL(MENUSFL) LOCK SFLDSP +
1900          MENUPGMQ                                SFLDSPCTL SFLINZ
2200          SFLSIZ(0050) SFLPAG(001)
                                           SFLEND
                                           SFLPGMQ
***** END OF EXPANDED SOURCE *****

```

Expanded Source

```

Title { 5728SS1 R01 M00 880311          DATA DESCRIPTION          QGPL/SAMPLISTNG          01/28/88 14:16:53          PAGE 3
        MESSAGES

* ID      SEVERITY  NUMBER  MESSAGE . . . . . : FIELD MUST BE BLANK FOR CONSTANT FIELD.
* CPD7434  20        2      RECOVERY . . . . . : OMIT THE VALUE IN THE INDICATED FIELD, AND THEN TRY THE REQUEST AGAIN.
* CPD7866  10        1      MESSAGE . . . . . : FIELD OVERLAPS ANOTHER FIELD WITH NO CONDITIONS SPECIFIED.
* CPD7881  20        2      MESSAGE . . . . . : THE INDICATED FIELD WILL NOT BE DISPLAYED BECAUSE IT OVERLAPS A
        CAUSE . . . . . : PREVIOUSLY DEFINED FIELD WHICH WILL ALWAYS BE DISPLAYED BECAUSE THERE
        RECOVERY . . . . . : ARE NO OPTION INDICATORS OR CONDITIONS SPECIFIED ON THE FIELD.
        MESSAGE . . . . . : FIELD SPECIFIED ON CSRLOC KEYWORD NOT FOUND.
        CAUSE . . . . . : THE INDICATED FIELD SPECIFIED ON THE CSRLOC KEYWORD MUST BE DEFINED IN
        RECOVERY . . . . . : THE RECORD FORMAT. EITHER THE FIELD WAS NOT DEFINED OR WAS IGNORED
        MESSAGE . . . . . : BECAUSE OF ERRORS.
        CAUSE . . . . . : EITHER DEFINE A FIELD WITH THE NAME SPECIFIED ON THE CSRLOC KEYWORD,
        RECOVERY . . . . . : OR CORRECT THE ERRORS LISTED ON THE PREVIOUS MESSAGES. THEN TRY THE
        MESSAGE . . . . . : REQUEST AGAIN.

```

Messages

```

Title {
TOTAL      INFORMATIONAL      MESSAGE SUMMARY
(0-9)      (10-19)      WARNING      ERROR      SEVERE
0          0          (10-19)      (20-29)      (30-99)
* CPF7302  40          1          4          0
MESSAGE . . . . . : FILE SAMPLISTNG NOT CREATED IN LIBRARY QGPL.
CAUSE . . . . . : THE FILE WAS NOT CREATED BECAUSE OF ERRORS.
RECOVERY . . . . . : SEE THE ERROR MESSAGES PREVIOUSLY LISTED. CORRECT THE ERRORS, AND THEN
        TRY THE REQUEST AGAIN.
***** END OF COMPI LATION *****

```

Message Summary

Figure 15. DDS Compiler Computer Printout (Part 2 of 2)

**Compiler listing title (appears at top of each output page):**

- 1** The program number, release modification level, and date of the OS/400 program.
- 2** The qualified name.
- 3** The date and time of this run.
- 4** The page number in the computer printout.

**Compiler listing prolog:**

- 5** The type of file and the parameter values specified (or defaults if not specified) on the create-file command.
- 6** The name of the DDS processor.

**Compiler listing source:**

- 7** The sequence numbers of lines (records) in the source. Comments are treated like any other specification line and are given sequence numbers.
- 8** The source specifications.
- 9** If an error is found during processing of the DDS and can be traced specifically to a source specification, the error message identifier and an asterisk indicating where the error is are printed immediately following the source specification line. An asterisk is also printed under the sequence number to indicate that the line contains an error message.

**Compiler listing expanded source:**

- 10** Only the valid DDS. This list is what is actually in the file description. No comments or messages are printed. Default values and referenced values are printed for the valid DDS.
- 11** The length and the buffer (input or output) position of each field.

**Compiler listing messages:**

- 12** This section contains a list of all messages (general messages and those already indicated in the source section) encountered during processing of the DDS. For each message, the message identifier, the severity, the number of times the message occurred, and the message text are listed.

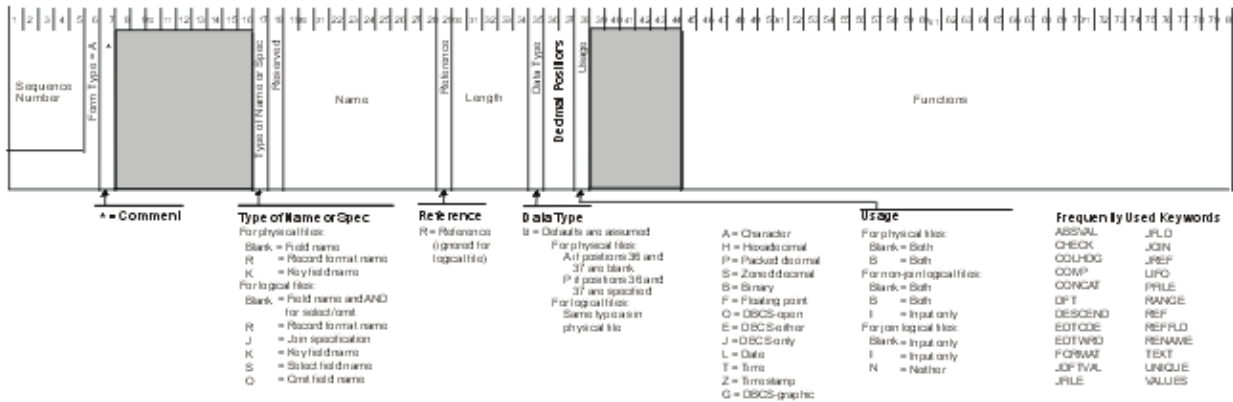
**Compiler listing message summary:**

- 13** The number of messages at each severity level.
- 14** The final completion message.

---

## **DDS debugging template**

A special template is available to help you in interpreting the fields on the DDS compiler computer printout. The following figure shows a reduced Debugging Template.



Database File Descriptions  
 Field in ECL © IBM Corp., 1991, 1992

Application System/400 Data Description Specifications Debugging Template, SX41-9890-01

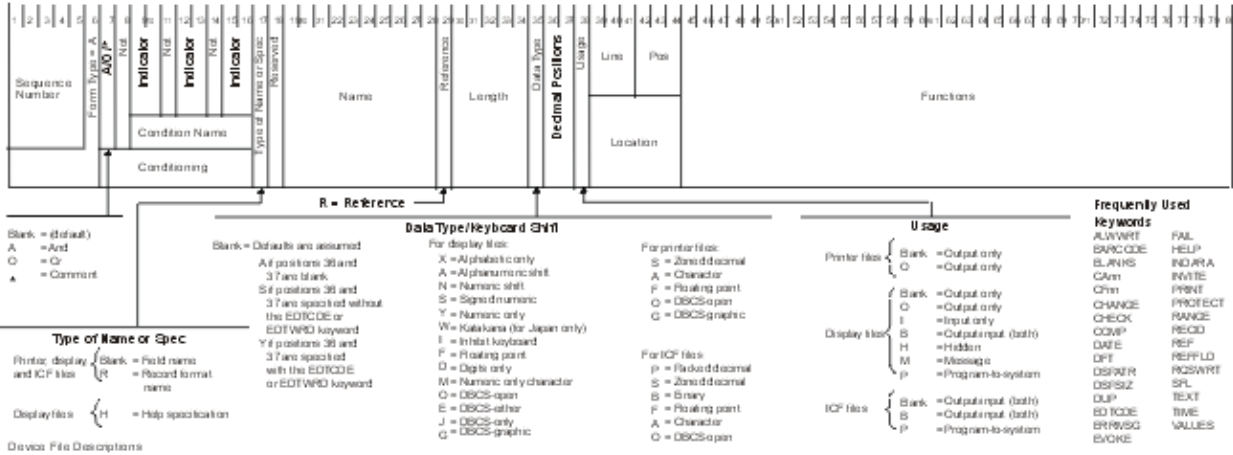


Figure 16. IBM Data Description Specifications Debugging Template (Reduced)



---

## Appendix B. When to specify REF and REFFLD keywords for DDS files

This topic provides information to help you decide if you want to specify the REF (reference) or REFFLD (referenced field) keyword or both. It also tells you how to specify parameter values for each REF or REFFLD keyword you specify.

**Note:** You must specify R in position 29 for each field that refers to another field that was previously defined.

Answer the following questions to determine which keyword to use:

- REF or REFFLD or both?

If all or most of the fields you refer to are defined REF at the file level.

Specify REFFLD for every field you reference:

- That is not in the file you specify on the REF keyword.

or

- Whose name differs from the name of the field it references. This includes fields that reference fields in the file you are defining.

- Do you need a database file name for each REFFLD keyword you specify?

The database file name specified on a REFFLD keyword overrides the database file name specified on the REF keyword.

On the REFFLD keyword, you can specify:

- \*SRC so that the OS/400 program searches the file you are defining for the referenced field. The referenced field must be defined before you define the field that references it.
- The name of the database file that the OS/400 program is to search through to find the referenced field.

If you do not specify \*SRC or a database file name on the REFFLD keyword, the default is \*SRC if the REF keyword is not specified. If the REF keyword is specified, the default is the database file name specified on the REF keyword.

- Is a library name necessary for each database file you specify?

If the job that will create the file you are defining (perhaps your interactive job) has a library list, and the database file you specified is on the library list, enter only the file name (FILE1). Otherwise, specify the file name qualified by the library name (LIB1/FILE1).

- Do you need a record format name for each REF or REFFLD keyword you specify?

If the file you reference has only one record format, do not specify a record format name.

If it has more than one record format, specify a record format name.

The following example illustrates reference function specifications. It is not a valid example of any file except an ICF file. Display and printer files must have a location specified for each field. Physical files can have only one record format. The REF and REFFLD keywords are not allowed in logical files.

		...	1	...	2	...	3	...	4	...	5	...	6	...	7	...	8
00010A													REF(FILE1)				<b>1</b>
00020A	R		RECORD1														
00030A			FIELD1	R													<b>1</b>
00040A			FIELD2	R													<b>1</b>
00050A			FIELD3	R									REFFLD(FLD3)				<b>2</b>
00060A			FIELD4	R									REFFLD(FLD4 FILE2)				<b>3</b>
00070A			FIELD5	R									REFFLD(FLD5 LIB1/FILE3)				<b>4</b>
00080A			FIELD6	R									REFFLD(RECORDB/FLD6 LIB1/FILE4)				<b>5</b>
00090A			FIELD7	R									REFFLD(FIELD6 *SRC)				<b>6</b>
00100A			FIELD8	R									REFFLD(FLD6)				<b>7</b>
00110A	R		RECORD2														
00120A			FIELD1		20												<b>8</b>
00130A																	
00140A	R		RECORD3														
00150A			FIELD1	R									REFFLD(RECORD2/FIELD1 *SRC)				<b>9</b>
00160A																	
00170A	R		RECORD4														
00180A			FIELD1	R									REFFLD(FIELD1 *SRC)				<b>10</b>
A																	

**Note:** For line 00010, you can also specify library name and record format name. See the REF keyword example.

Figure 17. Sample Reference Function Specifications

**Legend:**

- 1** FIELD1 and FIELD2 have the same attributes as FIELD1 and FIELD2 in FILE1.
- 2** FIELD3 has the same attributes as FLD3 in FILE1.
- 3** FIELD4 has the same attributes as FLD4 in FILE2.
- 4** FIELD5 has the same attributes as FLD5 in FILE3 in LIB1.
- 5** FIELD6 has the same attributes as FLD6 in record format RECORDB in FILE4 in LIB1.
- 6** FIELD7 has the same attributes as FIELD6 (on the preceding line in this file).
- 7** FIELD8 has the same attributes as FLD6 in FILE1.
- 8** FIELD1 in RECORD2 has unique field attributes. (It does not use the reference function; notice that R is not specified in position 29.)
- 9** FIELD1 in RECORD3 has the same attributes as FIELD1 in RECORD2.
- 10** FIELD1 in RECORD4 has the same attributes as FIELD1 in RECORD1.



---

## Appendix C. DDS keyword and value abbreviations

Following are all of the keyword and value abbreviations, listed in alphabetical order, used in data description specifications (DDS):

<b>Abbreviation</b>	<b>Meaning</b>
A	after
AB	allow blanks
ABS	absolute
ACC	access
ACCEL	accelerator
ALARM	alarm
ALT	alternative
ALW	allow
ARA	area
ATR	attributes
AUTO	automatic
AVAIL	available
B	before
BDY	boundary
BL	blinking field
BLANKS	blanks
BLINK	blinking cursor
BLK	blank
BOX	box
BTN	button
CA	command attention key
CCS	coded character set
CDE	code
CF	command function key
CHC	choice
CHG	change
CHK	check
CHR	character
CLR	clear
CLRL	clear line
CLS	class
CMD	command
CMP	comparison
CMT	commit
CNL	cancel
CNT	continued
CNV	conversion
COL	column
COMP	comparison
CON	constant
CONCAT	concatenate
CS	column separator
CSR	cursor
CTL	control
DAT	date
DEC	decimal
DEV	device

<b>Abbreviation</b>	<b>Meaning</b>
DFN	defined
DFR	defer
DFT	default
DLT	delete
DOC	document
DSP	display
DTA	data
DUP	duplicate
DWN	down
DYN	dynamic
EDT	edit
END	end
ENT	entry
EOS	end of session
EQ	equal
ER	end of record (same as end of field)
ERR	error
EXCLD	excluded
FAIL	fail
FCFO	first-changed first-out
FE	field exit
FIFO	first-in first-out
FIX	fixed
FLD	field
FLT	floating point
FMT	format
FNT	font
FRC	force
FULL	full
GDF	graphic data file
GE	greater than or equal to
GPH	graphics
GRD	grid
GRP	group
GT	greater than
HDG	heading
HI	high intensity
HLP	help
ID	identifier
IDX	index
IGC	double-byte character set (DBCS)
IND	indicator
INP	input
INZ	initialize
J	join
LC	lowercase
LCK	lock
LE	less than or equal to
LEN	length
LIFO	last in first out
LIN	line
LOC	location
LT	less than
LVL	level

<b>Abbreviation</b>	<b>Meaning</b>
MAP	map
MDT	modified data tag
ME	mandatory enter
MF	mandatory fill
MGT	management
MLT	multiple
MNU	menu
MOU	mouse
MSG	message
MSK	mask
M10	IBM Modulus 10
M10F	IBM Modulus 10
M11	IBM Modulus 11
M11F	IBM Modulus 11
NBR	number
ND	nondisplay
NE	not equal to
NEG	negative
NG	not greater than
NL	not less than
NULL	null
NXT	next
OF	off
OID	operator identification
OUT	output
OVR	override
P	physical
PAG	page
PC	position cursor
PCN	precision
PFILE	physical file
PGM	program
PNL	panel
POS	position
PR	protect
PRG	progression
PRP	prepare
PRT	print or printer
PSH	push
PTH	path
Q	queue
QLTY	quality
RA	record advance
RAB	right-justify with blank fill
RAZ	right-justify with zero fill
RB	right-justify with blank fill
RCD	record
RCV	receive
RECID	record identification
REF	reference
RET	retain
REV	reverse
RI	reverse image
RL	right to left

<b>Abbreviation</b>	<b>Meaning</b>
RLTB	right to left, top to bottom
RMV	remove
RNA	record not active
ROL	roll
RQS	request
RRN	relative record number
RSP	response
RST	restore
RTN	return
RTT	rotate
RZ	right-justify with zero fill
SCH	search
SCROLL	scroll
SEL	select
SEG	segment
SEP	separator
SEQ	sequence
SFL	subfile
SHELF	shelf
SIZ	size
SKIPA	skip after
SKIPB	skip before
SLNO	starting line number
SLT	select
SNG	single
SP	select by light pen
SPACEA	space after
SPACEB	space before
SST	substring
STS	status
SW	switch
SYN	sync
SYS	system
TIM	time
TITLE	title
TNS	transaction
TRNRND	turn around
TRNTBL	translation table
TXT	text
TYP	type
UL	underline
UNAVAIL	unavailable
USR	user
VAL	values
VAR	variable
VLD	valid
VN	validate name
VNE	validate name extended
WDW	window
WRD	word
WRT	write

---

## Appendix D. General considerations for using ideographic text (DBCS) with DDS files

The following sections describe the general considerations for positional entries, keyword entries, ideographic text literals, and DDS computer printouts that contain ideographic characters.

Ideographic text (also called DBCS text) can be encoded as either Unicode or EBCDIC. If you are working on a new application or enabling an existing application for ideographic text or working on an application involving Java™, ODBC, JDBC, or other web methods, then OS/400® Unicode support provides the easiest way to support not only ideographic text but also other text types. If you are working on an existing application already supporting ideographic text stored as EBCDIC, then OS/400 EBCDIC support for ideographic text is useful.

DDS uses the following terms to describe the different types of ideographic data:

- **DBCS data:** A general term to describe any form of EBCDIC encoded ideographic data.
- **DBCS field:** A general term to describe any field that can contain EBCDIC encoded ideographic data.
- **Bracketed DBCS data:** EBCDIC encoded ideographic data that begins with a shift-out character and ends with a shift-in character.
- **DBCS graphic data:** EBCDIC encoded ideographic data that contains only DBCS data and does not contain shift-out and shift-in characters.
- **Unicode data:** A general term to describe any form of Unicode encoded ideographic data.
- **Unicode field:** When the graphic data type is used with CCSID 1200 specified, then Unicode (UTF-16) is stored in the field instead of EBCDIC. When character data type is used with CCSID 1208 specified, then Unicode (UTF-8) is stored in the field instead of EBCDIC.

For specific information about using ideographic text with the different types of DDS files, see the following topics:

- Database (physical and logical) files
- Display files
- Printer files
- ICF files

---

### Positional entries for files that use DBCS data

Positional entries are adapted so that you can define data fields that contain DBCS data. The entries are adapted for the length and data type positional entries.

DBCS data is either **bracketed** or **graphic**. Bracketed-DBCS data begins with a shift-out character and ends with a shift-in character. DBCS-graphic data contains only DBCS data and does not contain shift-out and shift-in characters. The term DBCS refers to both bracketed and graphic DBCS data.

### Length (positions 30 through 34)

Consider the effects of the following when determining the length of a DBCS field.

For bracketed-DBCS fields:

- Positions occupied by double-byte characters as compared to the positions occupied by alphanumeric characters
- Shift-control characters
- Keyboard shift (if any)

For graphic-DBCS fields:

- Length is specified as the number of DBCS characters with no shift-control characters

## **Data type (position 35)**

Use the following DBCS data types to identify DBCS fields.

### **Bracketed DBCS:**

#### **J (Only)**

Fields can contain only DBCS data.

#### **E (Either)**

Fields can contain DBCS or alphanumeric data.

#### **O (Open)**

Fields can contain both DBCS and alphanumeric data.

### **Graphic DBCS:**

#### **G (Graphic)**

Fields can contain only DBCS data with no shift-control characters.

Data type O is allowed in all types of files. Data types J and E are allowed only in database and display files. Data type G is allowed in database, display, and printer files.

---

## **Keyword entries for files that use DBCS (positions 45 through 80)**

When you work with DBCS data, and specify certain DDS keywords, you can:

- Specify alternative ways to enter data through display files
- Change input- and output-capable alphanumeric data fields to DBCS data fields
- Specify the special features of the DBCS printer

Use the following keywords to perform these functions:

### **CHRSIZ (Character Size)**

Specify this keyword for printer files to be printed on the 5553 printers. CHRSIZ can expand printed characters to twice their normal size (width and height).

### **DFNLIN (Define Line)**

Specify this keyword for printer files. DFNLIN draws horizontal or vertical lines.

### **IGCALTTYP (DBCS Alternative Data Type)**

Specify this keyword for display and printer files. IGCALTTYP changes the data type of character input- and output-capable fields from A to O (open) if IGCDDTA(\*YES) is specified on the command used to create the file.

### **IGCANKCNV (Alphanumeric-to-DBCS Conversion)**

Specify this keyword for printer files (for Japanese only). IGCANKCNV converts alphanumeric characters to equivalent DBCS characters so printed alphanumeric characters have the same appearance as printed DBCS characters.

### **IGCCDEFNT (DBCS Coded Font)**

Specify this keyword for printer files. IGCCDEFNT allows you to specify the DBCS coded font for printing a named or constant field.

### **IGCCNV (DBCS Conversion)**

Specify this keyword for display files (for Japanese use only). IGCCNV allows you to use DBCS conversion, which is an alternative to directly typing in DBCS characters from a keyboard.

### IGCCHRRTT (DBCS Character Rotation)

Specify this keyword for printer files to be printed on the 5553 printers. IGCCHRRTT rotates each DBCS character 90 degrees counterclockwise before printing. By rotating characters, the printouts can be read vertically.

---

## DBCS character strings

You can use bracketed-DBCS character strings in DDS files for text-related keywords, such as TEXT and COLHDG, and both bracketed and DBCS-graphic character strings as parameters on the COMP, DFT, RANGE, and VALUES keywords.

### Considerations for using DBCS character strings:

Consider the following information when you use DBCS character strings:

- Do not specify DBCS character strings for those DDS keywords that are dependent on data type, and for which you did not specify a DBCS data type (data type O, J, E, or G).
- When the source file is defined as DBCS, DDS scans all character strings as DBCS character strings and considers all data between the shift-control characters to be part of the character string.  
Remember to include both shift-control characters. If the shift-in character indicating the end of the DBCS string is missing, the system considers the remainder of the record, including the ending apostrophe, to be part of the character string.
- When the source file is alphanumeric, DDS does not check the character string to make sure that you have included only DBCS characters between the shift-control characters.
- When the source file is alphanumeric, DDS identifies DBCS character strings as alphanumeric literals.
- You may refer to a field that contains a DBCS character string from another file, using the reference function. DDS copies the attributes of the field containing the DBCS character string (the referenced field) to the field you are defining. If the file containing the referenced field is DBCS and the file you are defining is alphanumeric, DDS does not check the character string to make sure it is a valid DBCS character string. If the file containing the referenced field is alphanumeric and the file you are defining is DBCS, DDS checks the character string to make sure it is a valid DBCS character string.

## Entering bracketed-DBCS character strings

Enter bracketed-DBCS character strings as follows:

1. Begin the character string with an apostrophe (').
2. Type a shift-out character.
3. Type the DBCS text.
4. Type a shift-in character.
5. End the character string with an apostrophe (').

For example, to type the DBCS literal ABC, enter the following, where 0<sub>E</sub> represents the shift-out character and 0<sub>F</sub> represents the shift-in character:

```
'0EABC0F'
```

## Entering DBCS-graphic character strings

Enter DBCS-graphic character strings as follows:

1. Type a G to indicate that the string contains DBCS-graphic data.
2. Begin the character string with an apostrophe (').
3. Type a shift-out character.
4. Type the DBCS text.
5. Type a shift-in character.

6. End the character string with an apostrophe (').

For example, to type the DBCS literal ABC, enter the following, where 0<sub>E</sub> represents the shift-out character and 0<sub>F</sub> represents the shift-in character:

G'0<sub>E</sub>ABC0<sub>F</sub>'

---

## DDS computer printouts with DBCS output

DDS computer printouts are printed as DBCS output in each of these instances:








- The source file is DBCS.
- DBCS character strings were added to the source file as a result of a reference operation.



---

## Appendix E. Related information

The following topics provide information related to the use of DDS on OS/400:

- Backup and Recovery  describes how to make copies of the system and database files for safe keeping.
- *Advanced Function Printing™: Data Stream Reference*, S544-3202, provides information on the Advanced Function Printing data stream.
- ADTS for AS/400: Screen Design Aid  book on the V5R1 Supplemental Manuals Web site provides the application programmer, system programmer, or data processing manager with information about using the WebSphere Development Studio screen design aid (SDA) to design, create, and maintain display formats and menus.
- ADTS for AS/400: Source Entry Utility  book on the V5R1 Supplemental Manuals Web site provides the application programmer or system programmer with information about using the WebSphere Development Studio source entry utility (SEU) to create and edit source members.
- Application Display Programming  describes the use of display devices by application programs.
- Printer Device Programming  provides information to help the programmer manage key aspects of the system. For example, it describes how to use printer files.
- IBM® *Personal Computer Enhanced 5250 Emulation Program Version 2.12 Technical Reference*, GS57-0222-02, provides the information on the 5250 Emulation Program.
- *Intelligent Printer Data Stream™ Reference*, S544-3417, provides information on bar codes and valid check digits.
- RPG/400 Reference  book on the V5R1 Supplemental Manuals Web site describes how to design, code, enter, compile, test, and run RPG III programs.
- CL Programming  provides a wide-range discussion of programming topics.
- The Control language topic in the **Programming** category of the iSeries Information Center describes control language syntax, commands, and command parameters.
- The Work Management topic in the **Systems Management** category of the iSeries Information Center provides information about how to create and change a work management environment.
- Getting started with iSeries topic in the **System Overview, Planning, and Installation** category of the iSeries Information Center describes how to operate work stations and how to use the function keys to enter commands.
- *Experience RPG IV Tutorial* is an interactive self-study program explaining the differences between RPG III and RPG IV and how to work within the new ILE environment. An accompanying workbook provides additional exercises and doubles as a reference upon completion of the tutorial. RPG code examples are shipped with the tutorial and run directly on OS/400. Dial 1-800-IBM-CALL to order the tutorial.



---

## Appendix F. DDS glossary of terms

This glossary includes terms and definitions from:

- The *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Committee (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

**A/N/K.** Pertaining to alphabetic, numeric, or Katakana characters.

**absolute value.** The magnitude of a number.

**access path.** The order in which records in a database file are organized for processing by a program. See *arrival sequence access path* and *keyed sequence access path*.

**active record.** An active subfile record or any record format that is currently shown on a display. See also *active subfile record*. Contrast with *inactive record*.

**active subfile.** A subfile in which a write operation was issued to the subfile record format or to the subfile control record format with the DDS keyword SFLINZ in effect.

**active subfile record.** A record that was added to the subfile by a write operation, or a record that was initialized by the DDS keyword SFLINZ. Contrast with *inactive subfile record*.

**Advanced Function Printing (AFP™).** Pertaining to the ability of programs to use the all-points-addressable concept to print text and images on a printer.

**Advanced Function Printing data stream (AFPDS).** In AFP support, the printer data stream used for printing Advanced Function Printing data. The AFPDS includes composed text, page segments, electronic overlays, form definitions, and fonts that are downloaded from the system to the printer.

**AFP.** See *Advanced Function Printing (AFP)*.

**AFP resources.** The form definitions, page definitions, fonts, overlays (electronic forms), and page segments (graphic images). With the PrintManager™ program, resources can either exist in a system library, or be placed inline with a print job as the job is written to the spool.

**AFPDS.** See *Advanced Function Printing data stream (AFPDS)*.

**alphabetic character.** In DDS and IDDU, any one of the uppercase letters A through Z or one of the characters #, \$, or @.

**alphanumeric.** Pertaining to the letters A through Z or a through z; numbers 0-9; and special symbols \$, #, @, ., or \_.

**alternative collating sequence.** A user-defined collating sequence that replaces the standard EBCDIC collating sequence. See *collating sequence*.

**arrival sequence access path.** An access path to a database file that is arranged according to the order in which records are stored in the physical file. See also *keyed sequence access path* and *access path*.

**ascending sequence.** The arrangement of data in order from the lowest value to the highest value, according to the rules for comparing data. Contrast with *descending sequence*.

**asterisk fill.** A type of numeric editing that puts asterisks to the left of a number to fill unused positions. Example:  
\*\*\*\*476.12

**attribute.** A characteristic or trait of one or more items.

**attribute character.** A character associated with a field in a display file record format that defines how the field is displayed.

**bar code.** A pattern of bars of various widths containing data to be interpreted by a scanning device.

**beginning attribute character.** For a display file, the character that precedes the first position in a field and that defines how the data in the field is displayed.

**both field.** A field that can be used for either input data or output data.

**bracketed DBCS.** A character string in which each character is represented by 2 bytes. The character string starts with a shift-out (SO) character and ends with a shift-in (SI) character. Contrast with *DBCS-graphic*.

**byte.** A group of 8 adjacent bits. In the EBCDIC coding system, 1 byte can represent a character. In the double-byte coding system, 2 bytes represent a character.

**changed subfile record.** A subfile record into which the work station user has entered data, or a subfile record for which a write or change operation was issued with the DDS keyword SFLNXTCHG or DSPATR(MDT) in effect.

**character.** Any letter, number, or other symbol in the data character set that is part of the organization, control, or representation of data.

**character constant.** The actual character value (a symbol, quantity, or constant) in a source program that is itself data, instead of reference to a field that contains the data. Contrast with *numeric constant*.

**character field.** An area that is reserved for information that can contain any of the characters in the character set. Contrast with *numeric field*.

**character set.** A group of characters used for a specific reason; for example, the set of characters the display station can display, the set of characters a printer can print, or a particular set of graphic characters in a code page; for example, the 256 EBCDIC characters.

**character string.** A sequence of consecutive characters that are used as a value.

**characters per inch (cpi).** The number of characters printed horizontally within an inch across a page.

**check digit.** The far right number of a self-check field used to verify the accuracy of the field.

**close.** The function that ends the connection between a file and a program, and ends the processing. Contrast with *open*.

**code page.** (1) A particular assignment of hexadecimal identifiers to graphic characters. (2) In AFP support, a font file that associates code points and graphic character identifiers.

**code point.** (1) One of the bit patterns assigned to a character in a character set. On iSeries servers, a code point is represented by a hexadecimal number. For example, in code page 256 (EBCDIC), the letter "e" is assigned a code point of hex 85. (2) In AFP support, an 8-bit binary number representing one of 256 potential characters.

**code-page ID.** A 5-digit registered identifier used to specify a particular assignment of code points to graphic characters. The code-page ID is the second part of the QCHRID system value or the CHRID parameter value. See also *graphic character-set ID*.

**coded character set identifier (CCSID).** A 16-bit number identifying a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other relevant information that uniquely identifies the coded graphic character representation used.

**coded font.** In AFP support, a font file that associates a code page and a font character set. For double-byte fonts, a coded font associates multiple pairs of code pages and font character sets.

**coded graphic character-set ID.** A 10-digit identifier (two 5-digit identifiers separated by a space) that is the combination of a graphic character-set ID and a code-page ID. See also *graphic character-set ID* and *code-page ID*.

**collating sequence.** The order in which characters are arranged within the computer for sorting, combining, or comparing.

**column separator.** A symbol on each side of a position of a field on a display. This symbol does not occupy a position on the display.

**command.** A statement used to request a function of the system. A command consists of the command name abbreviation, which identifies the requested function, and its parameters.

**command attention (CA) key.** In DDS, a keyboard key that can be specified with the CA keyword to request the function specified by the keyword. Data is not returned to the system. Contrast with *command function (CF) key*.

**command function (CF) key.** In DDS, a keyboard key that can be specified with the CF keyword to request the function specified by the keyword. Data is returned to the system. Contrast with *command attention (CA) key*.

**compile.** To translate a program written in a high-level programming language into a machine-language program.

**compile time.** The time during which a source program is translated by a compiler into a machine-language program.

**compiler listing.** A printout that is produced by compiling a program or creating a file and that optionally includes, for example, a line-by-line list of the high-level language source, a cross-reference list, diagnostic information; and for programs, the description of the externally described files. See also *source listing*.

**composite key.** A key for a file or record format that is composed of more than one key field.

**concatenated field.** Two or more fields that are combined to make one field in a logical file.

**condition name.** For display files, a name used to control the selection of DDS keywords and display locations based on the model of the display station.

**conditioning.** A feature that helps users to develop and create display files, printer files, and database files.

**constant field.** In an externally described display or printer file, an unnamed field that contains actual data that is passed to the display or printer but is unknown to the program passing it.

**continuation line.** An additional line (or lines) required to continue the coding of a CL command or a DDS keyword and its value.

**country or region ID.** See *country or region identifier*.

**country or region identifier.** The 2-character representation for the country or region associated with an object. For example, documents and user profiles can have a country or region associated with them.

**creation date.** The system date when an object is created. See also *job date*, and *system date*.

**currency symbol.** A character such as the dollar sign (\$) used to identify monetary values.

**cursor.** A movable symbol, often a blinking or solid block of light, that tells the display station user where to type, or identifies a choice to select.

**data description specifications (DDS).** A description of the user's database or device files that is entered into the system in a fixed form. The description is then used to create files.

**data file.** A group of related data records organized in a specific order. A data file can be created by the specification of FILETYPE(\*DATA) on the create commands. Contrast with *source file*.

**data file utility (DFU).** The part of the WebSphere Development Studio that is used to enter, maintain, and display records in a database file.

**data stream.** All information (data and control commands) sent over a data link usually in a single read or write operation.

**data type.** A characteristic used for defining data as numeric or character.

**database.** All the data files stored in the system.

**database file.** One of several types of the system object type \*FILE kept in the system that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. See also *physical file* and *logical file*.

**DBCS.** See *double-byte character set (DBCS)*.

**DBCS conversion.** A function of the operating system that allows a display station user to enter alphanumeric data and request that the alphanumeric data be converted to double-byte data.

**DBCS-either.** Pertaining to a character string that is either SBCS or bracketed DBCS, but not both. Contrast with *DBCS-graphic*, *DBCS-only*, and *DBCS-open*.

**DBCS-graphic.** Pertaining to a character string in which each character is represented by 2 bytes. The character string does not contain shift-out (SO) and shift-in (SI) characters. Contrast with *DBCS-either*, *DBCS-only*, and *DBCS-open*.

**DBCS-only.** Pertaining to a character string that is only bracketed DBCS. Contrast with *DBCS-either*, *DBCS-graphic*, and *DBCS-open*.

**DBCS-open.** Pertaining to a character string that can be a mixture of SBCS and bracketed DBCS. Contrast with *DBCS-either*, *DBCS-graphic*, and *DBCS-only*.

**DDS.** See *data description specifications (DDS)*.

**decimal position.** (1) The location of the decimal point in a series of numbers. (2) Numbers to the right of the decimal point. For example, 4.009 has three decimal positions.

**default.** (1) A value that is automatically supplied or assumed by the system or program when no value is specified by the user. (2) In DDS, the value specified by the user with the DFT or DFTVAL keyword in DDS.

**descending sequence.** The arrangement of data in order from the highest value to the lowest value, according to the rules for comparing data. Contrast with *ascending sequence*.

**DEVVD.** See *device description*.

**device description.** An object that contains information describing a particular device or logical unit (LU) that is attached to the system. A device description is a description of the logical connection between two LUs (local and remote locations). The system-recognized identifier for the object type is \*DEVVD.

**device file.** One of several types of the system object type \*FILE. A device file contains a description of how data is to be presented to a program from a device or how data is to be presented to the device from the program. Devices can be display stations, printers, diskette units, tape units, or remote systems.

**DFU.** See *data file utility (DFU)*.

**digit.** Any of the numerals from 0 through 9.

**display file.** A device file to support a display station.

**display screen.** The part of the display device, which is similar to a television (TV) picture tube, used to display information entered or received at a display station.

**display station.** A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or the information received from the system.

**double precision.** The specification that causes a floating-point value to be stored (internally) in the long format (two computer words). Double precision is known as *long precision* in BASIC. Contrast with *single precision*.

**double-byte character.** An entity that requires two character bytes.

**double-byte character set (DBCS).** A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, displaying, and printing of DBCS characters requires hardware and programs that support DBCS. Four double-byte character sets are supported by the system: Japanese, Korean, Simplified Chinese, and Traditional Chinese. Contrast with *single-byte character set (SBCS)*.

**double-byte coded font.** In AFP support, a font in which the characters are defined by 2 bytes: the first defining a coded font section, the second defining a code point. Synonymous with double-byte font. Contrast with *single-byte coded font*.

**draft.** A printed copy of a document that is not yet completed.

**duplicate key value.** The occurrence of the same value in a key field or in a composite key in more than one record in a file.

**dynamic select/omit.** Selection and omission of logical file records performed during processing, instead of when the access path (if any) is maintained. Dynamic select/omit may also be used when no keyed access path exists.

**edit.** To change a numeric field for output by suppressing zeros and inserting commas, periods, currency symbols, the sign status, or other constant information.

**edit code.** A letter or number indicating that editing should be done according to a defined pattern before a field is displayed or printed. Contrast with *edit word*.

**edit description.** A description of a user-defined edit code. The system-recognized identifier is \*EDTD.

**edit word.** A user-defined word with a specific format that indicates how editing should be done. Contrast with *edit code*.

**electronic overlay.** An AFP resource object that is a collection of predefined data, such as lines, shading, text, boxes, or logos, that can be merged with variable data on a page while printing. The system-recognized identifier for the object type is \*OVL.

**embedded blank.** A space between characters within a unit of data.

**ending attribute character.** For a display file, the character following the last position in a field.

**exponent.** (1) A number, indicating to which power another number (the base) is to be raised. (2) In floating-point format, an integer constant specifying the power of ten by which the base of the decimal floating-point number is to be multiplied.

**expression.** In DDS, a pair of values that represents a single parameter value.

**externally described data.** Data contained in a file for which the fields and the records are described outside of the program (such as with files created by DDS, IDDU, or the DB2 UDB for iSeries licensed program) that processes the file. Contrast with *program-described data*.

**externally described file.** A file in which the records and fields are described to the system when the file is created, and used by the program when the file is processed. Contrast with *program-described file*.

**field.** A group of related bytes (such as name or amount) that is treated as a unit in a record.

**field level specifications.** In DDS, specifications coded on the same line as a field name or on lines immediately following a field name. See also *file level specifications, record level specifications, help level specifications, join level specifications, key field level specifications, and select/omit level specification*.

**field reference file.** A physical file that contains no data, only descriptions of fields.

**field selection.** A function that uses the state of the option indicators to display or print data when a record format is written.

**file.** A generic term for the object type that refers to a database file, a device file, or a save file. The system-recognized identifier for the object type is \*FILE.

**file level specifications.** In DDS, specifications coded on the lines before the first record format name. See also *field level specifications, key field level specifications, record level specifications, join level specifications, select/omit level specifications, and help level specifications.*

**fixed currency symbol.** A currency symbol that appears in the far left position of an edited field. Contrast with *floating currency symbol.*

**fixed-length.** Pertaining to a characteristic of a file in which all of the records are the same length, or pertaining to a characteristic of a field on a display that is of a defined length.

**floating currency symbol.** A currency symbol that appears immediately to the left of the far left position in an edited field. Contrast with *fixed currency symbol.*

**floating-point.** A method of encoding real numbers within the limits of finite precision available on computers.

**fold.** To continue data on the next line. Contrast with *truncate.*

**folder.** A directory for documents. A folder is used to group related documents and to find documents by name. The system-recognized identifier for the object type is \*FLR. See also *document library object.* Compare with *library.*

**font.** An assortment of characters of a given size and type style.

**font character set.** In AFP support, a font file that contains the raster patterns, identifiers, and descriptions of characters.

**font ID.** A number that identifies the character style and size for certain printers.

**function key.** A keyboard key that allows the user to select keyboard functions or programmer functions. Contrast with *character key.*

**GDF file.** See *graphics data format (GDF) file.*

**graphic character set.** A set of graphic characters in a code page.

**graphic character-set ID.** A 5-digit registered identifier used to specify a graphic character set. The graphic character-set ID is the first part of the QCHRID system value or the CHRID parameter value. See also *code-page ID.*

**graphics data format (GDF) file.** A picture definition in a coded order format used internally by the GDDM<sup>®</sup> function and, optionally, providing the user with a lower-level programming interface than the GDDM application programming interface.

**help level specifications.** In a display file, data description specifications coded between the record and field level that define areas on the screen and associate help information with those areas. See also *file level specifications, field level specifications, join level specifications, key field level specifications, record level specifications, and select/omit level specifications.*

**hexadecimal.** Pertaining to a numbering system with a base of 16.

**hidden field.** A field in a display file that is passed to and from the program but is not sent to the display.

**high-level language (HLL).** A programming language, such as RPG, BASIC, PL/I, Pascal, COBOL, and C used to write computer programs.

**Hiragana.** A graphic character set that is used to write Japanese words phonetically. This set of characters is used as word endings when writing in Kanji. Contrast with *Katakana.*

**human readable interpretation (HRI).** In IBM Advanced Function Printing Utilities for iSeries, the characters printed above or below a bar code. These characters are read by people, not by scanners.

**ICF.** See *intersystem communications function (ICF).*

**ICF file.** A device file that allows a program on one system to communicate with a program on another system. There can be one or more sessions with the same or different communications devices at the same time.

**IGC.** Abbreviation used in commands and keywords to represent double-byte character set functions.



**ILE.** See *Integrated Language Environment*<sup>®</sup> (ILE).

**implicit.** Capable of being understood from something else, though unexpressed.

**inactive record.** An inactive subfile record or any record format that is not currently shown on a display. See also *inactive subfile record*. Contrast with *active record*.

**inactive subfile record.** A subfile record that either was not added to a subfile by a write operation or was described as inactive by the data description specification (DDS) keywords SFLINZ and SFLRNA. Contrast with *active subfile record*.

**indicator.** (1) A 1-character or 2-character code that is used by a program to test a field or record or to tell when certain operations are to be performed. (2) An internal switch used by a program to remember when a certain event occurs and what to do when that event occurs.

**input field.** A field specified in a display file or database file that is reserved for information supplied by a user. Contrast with *output field*.

**integer.** A positive or negative whole number.

**Integrated Language Environment (ILE).** Pertaining to a set of constructs and interfaces that provides a common run-time environment and run-time bindable application program interfaces (APIs) for all ILE-conforming high-level languages.

**Intelligent Printer Data Stream (IPDS).** Pertaining to an all-points-addressable data stream that allows users to position text, images, and graphics at any defined point on a printed page.

**intersystem communications function (ICF).** A function of the operating system that allows a program to communicate interactively with another program or system.

**IPDS.** See *Intelligent Printer Data Stream (IPDS)*.

**job date.** The date associated with a job. The job date usually assumes the system date, but it can be changed by the user. See also *creation date* and *system date*.

**join.** An operation that combines data from two or more files using specified fields.

**join field.** A comparison field that identifies records from two files to be combined into one record.

**join level specifications.** For a join logical file, data description specifications coded between the record and field level that define how to join two physical files. See also *file level specifications*, *field level specifications*, *key field level specifications*, *help level specifications*, *record level specifications*, and *select/omit level specifications*.

**join logical file.** A logical file that combines (in one record format) fields from two or more physical files. See also *logical file*.

**justify.** To adjust text so that line endings are even. See *left-justify* and *right-justify*.

**Kanji.** Characters originating from the Chinese characters used in the Japanese written language.

**Katakana.** A graphic Japanese character set that is used to write non-Japanese words phonetically in Japanese. Contrast with *Hiragana*.

**key field.** A field used to arrange the records of a particular type within a file member.

**key field level specifications.** Data description specifications coded on the lines following the last field specification. Key field level specifications are permitted only for physical files or logical files. See also *field level specifications*, *file level specifications*, *record level specifications*, *help level specifications*, *join level specifications*, and *select/omit level specifications*.

**keyboard shift.** In DDS, a characteristic that can be specified for a field in a display file that automatically shifts the display station keyboard to control what the display station user can enter into the field. In IDDU and DDS, the keyboard shift can also be specified in database files, but only applies when these fields are referred to in a display file.

**keyed sequence access path.** An access path to a database file that is arranged according to the contents of key fields contained in the individual records. See also *arrival sequence access path* and *access path*.

**keyword.** In DDS, a name that identifies a function.

**keyword functions.** The result of processing DDS keywords in a record format specified on an operation. See also *operation*.

**library.** A system object that serves as a directory to other objects. A library groups related objects, and allows the user to find objects by name. The system-recognized identifier for the object type is \*LIB. Compare with *folder* and *document library*.

**library list.** A list that indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is \*LIBL.

**library name.** A user-defined word that names a library.

**lines per inch (lpi).** The number of characters that can be printed vertically within an inch.

**local.** Pertaining to a device or system that is connected directly to your system or a file that is read directly from your system, without the use of a communications line. Contrast with *remote*.

**local work station.** A work station that is connected directly to the system without a need for data transmission functions. Contrast with *remote work station*.

**locked keyboard.** A keyboard condition where the display station accepts no input.

**logical file.** A description of how data is to be presented to or received from a program. This type of database file contains no data, but it defines record formats for one or more physical files. See also *join logical file* and *database file*. Contrast with *physical file*.

**logical file member.** A named logical grouping of data records from one or more physical file members. See also *member*.

**lpi.** See *lines per inch (lpi)*.

**magnetic stripe reader.** A device, attached to a display station, that reads data from a magnetic stripe on a badge before allowing an operator to sign on.

**mandatory entry field.** A field in which an operator must enter at least one character.

**mandatory fill field.** A field that an operator must leave blank, or must fill in completely.

**MDT.** See *modified data tag (MDT)*.

**member.** Different sets of data, each with the same format, within one database file. See also *source member*.

**message file.** An object that contains message descriptions. The system-recognized identifier for the object type is \*MSGF.

**message identifier.** A seven-character code that identifies a predefined message, and is used to get the message description from a message file. See *predefined message*.

**message line.** An area on the display where messages are displayed.

**message reference key.** A key assigned to every message on a message waiting line. This key is used to remove a message from a message waiting line, to receive a message, and to reply to a message.

**message subfile.** A subfile where the records are messages from a program message queue.

**modified data tag (MDT).** An indicator, associated with each input or output field in a displayed record, that is automatically set on when data is typed into the field. The modified data tag is maintained by the display file and can be used by the program using the file.

**modulus.** In BASIC and communications, a number, such as a positive integer, in a relationship that divides the difference between two related numbers without leaving a remainder. For example, 9 and 4 have a modulus of 5 ( $9 - 4 = 5$ ;  $4 - 9 = -5$ ; and 5 divides both 5 and -5 without leaving a remainder).

**modulus 10 checking/modulus 11 checking.** (1) A method for verifying data. (2) Formulas used to calculate the check digit for a self-check field.

**negative response.** In data communications, a reply indicating that data was not received correctly or that a command was incorrect or unacceptable. Contrast with *positive response*. See also *exception response*.

**numeric character.** Any one of the digits 0 through 9.

**numeric constant.** The actual numeric value to be used in processing, instead of the name of a field containing the data. A numeric constant can contain any of the numeric digits 0 through 9, a sign (plus or minus), and a decimal point. See also *floating-point constant*. Contrast with *character constant*.

**numeric field.** An area that is reserved for a particular unit of information and that can contain only the digits 0 through 9. Contrast with *character field*.

**object.** A named storage space that consists of a set of characteristics that describe itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed. Some examples of objects are programs, files, libraries, and folders.

**object name.** The name of an object. Contrast with *qualified name*.

**online information.** Information on the display screen that explains displays, messages, and programs. See also *textual data*.

**open.** The function that connects an object of type \*FILE to a program for processing. Contrast with *close*.

**operation.** The result of processing statements in a high-level language. See also *keyword functions*.

**option indicator.** A 1-character field that is passed with an output data record from a program to the system that is used to control the output function, such as controlling which fields in the record are displayed.

**OS/400.** See *IBM Operating System/400® Version 2 (OS/400)*.

**output.** Information or data received from a computer that is shown on a display, printed on the printer, or stored on disk, diskette, or tape.

**output field.** A field specified in a display file, database file, printer file, or ICF file that is reserved for the information processed by a program. Contrast with *input field*.

**output/input field.** A field specified in a database, display, or ICF file that can be used for both the information supplied to the program and the information received from the program during processing. See also *input field* and *output field*.

**overflow.** The condition that occurs when the last line specified as the overflow line to be printed on a page has been passed.

**overlapping fields.** Fields in the same display or printer record that occupy the same positions on the display or page. Option indicators can be used to select which of the overlapping fields is to be displayed or printed.

**overlay.** For AFP support, see *electronic overlay*.

**page.** (1) Each group of records in a subfile that are displayed at the same time. (2) To move information up or down on the display.

**page down.** To move to the information below the information currently shown on the display. Contrast with *page up*.

**page up.** To move to the information above the information currently shown on the display. Contrast with *page down*.

**parameter.** A value supplied to a command or program that is used either as input or to control the actions of the command or program.

**physical file.** A description of how data is to be presented to or received from a program and how data is actually stored in the database. A physical file contains one record format and one or more members. See also *database file*. Contrast with *logical file*.

**physical file member.** A named subset of the data records in a physical file. See also *member*.

**PIP.** See *problem isolation procedure (PIP)* or *program initialization parameters (PIP)*.

**positive response.** In SNA, a response indicating that a request arrived and was successfully received and processed. Contrast with *negative response*. See also *definite response*.

**primary file.** In the DDS for a join logical file, the first physical file specified on the JFILE keyword. Contrast with *secondary file*.

**printer file.** A device file that determines what attributes printed output will have. A particular printer may or may not support all of the attributes specified in a printer file.

**program initialization parameters (PIP).** The initial parameter value(s) passed to a target program as input or used to set up the process environment.

**program message queue.** An object used to hold messages that are sent between program calls of a routing step. The program message queue is part of the job message queue.

**program-described data.** Data contained in a file for which the fields in the records are described in the program that processes the file. Contrast with *externally described data*.

**program-described file.** A file for which the fields in the records are described only in the programs that process the file. To the operating system, the record appears as a character string. Contrast with *externally described file*.

**protected field.** A field on a display in which a user cannot add, change, or delete data.

**qualified name.** The name of the library containing the object and the name of the object. Contrast with *object name*.

**read operation.** An input operation that obtains data from a file or device and passes it to a program.

**read-from-invited-program-devices operation.** An input operation that waits for input from any one of the invited program devices for a user-specified time. Contrast with *read-from-one-program-device operation*.

**read-from-one-program-device operation.** An input operation that will not complete until the specified device has responded with input. Contrast with *read-from-invited-program-devices operation*.

**record.** A group of related data, words, or fields treated as a unit, such as one name, address, and telephone number.

**record format.** A named part of a file that identifies records of a specified record format description.

**record level specifications.** Data description specifications coded on the same line as a record format name or on lines immediately following a record format name (until the first field is specified). See also *field level specifications*, *file level specifications*, *key field level specifications*, *help level specifications*, *join level specifications*, and *select/omit level specifications*.

**relational operator.** The reserved words or symbols used to express a relational condition or a relational expression.

**relative file number.** In the DDS for a join logical file, a sequential number assigned to a physical file based on the position of that file on the JFILE keyword specification.

**relative record number.** A number that specifies the relationship between the location of a record and the beginning of a database file, member, or subfile. For example, the first record in a database file, member, or subfile has a relative record number of 1.

**remote.** Pertaining to a device, system, or file that is connected to another device, system, or file through a communications line. Contrast with *local*.

**remote work station.** A work station that is connected to the system by data communications. Contrast with *local work station*.

**response indicator.** A 1-character field passed with an input record from the system to a program to provide information about the data record or actions taken by the work station user.

**reverse image.** Text that appears on the display in the opposite color (for example, black on green instead of green on black).

**screen design aid (SDA).** A function of the WebSphere Development Studio that helps the user design, create, and maintain displays and menus.

**SCS.** See *SNA character string (SCS)*.

**SDA.** See *screen design aid (SDA)*.

**secondary file.** In the DDS for a join logical file, any physical file, other than the first physical file, that is specified on the JFILE keyword. Contrast with *primary file*.

**select/omit field.** A field in a logical file record format whose value is tested by the system to determine if records including that field are to be used. The test is a comparison with a constant, the contents of another field, a range of values, or a list of values; and the record is either selected or omitted as a result of the test. See also *dynamic select/omit*.

**select/omit level specifications.** Data description specifications coded on the lines following the last key-field specification. These specifications are permitted only in a logical file. See also *field level specifications, file level specifications, key field level specifications, record level specifications, help level specifications, and join level specifications*.

**self-check digit.** The far right digit of a self-check field.

**self-check field.** A field, such as an account number, consisting of a base number and a self-check digit. For data entry applications, the operator-entered self-check number is compared with the self-check number calculated by the system.

**sequence number.** The number of a record that identifies the record within the source member.

**session.** (1) The length of time that starts when a user signs on at a display station and ends when the user signs off. (2) In communications, the logical connection by which a program or device can communicate with a program or device at a remote location. See also *conversation* and *transaction*.

**SEU.** See *source entry utility (SEU)*.

**shared file.** A file whose open data path can be shared between two or more programs processing in the same job. See *open data path (ODP)*.

**shift control character.** See *shift-in character* and *shift-out character*.

**shift-in character.** A control character (hex 0F) that indicates the end of a string of double-byte characters. Contrast with *shift-out character*.

**shift-out character.** A control character (hex 0E) that indicates the start of a string of double-byte characters. Contrast with *shift-in character*.

**significant.** In binary floating-point format, the part of a number that contains the whole number and fraction.

**significant digit.** Any number of a series of numbers that follows the farthest left number, that is not a zero, and that is within the accuracy allowed.

**single precision.** The specification that causes the floating-point value to be stored (internally) in the short format. Contrast with *double precision*.

**single-byte character set (SBCS).** A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set (DBCS)*.

**single-byte coded font.** In AFP support, a font in which the characters are defined by a 1-byte code point. A single-byte coded font has only one coded font section. Synonymous with single-byte font. Contrast with *double-byte coded font*.

**SNA.** See *Systems Network Architecture (SNA)*.

**SNA character string (SCS).** In SNA, a data stream composed of EBCDIC controls, optionally intermixed with end-user data, that is carried within a request/response unit.

**source entry utility (SEU).** A function of the WebSphere Development Studio that is used to create and change source members.

**source file.** A file of programming code that is not compiled into machine language. A source file can be created by the specification of FILETYPE(\*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications. Contrast with *data file*.

**source member.** A member of a database source file that contains source statements. See also *member*.

**source statement.** A statement written in symbols of a programming language.

**special character.** A character other than a digit, a letter, or \$, #, @, ., or \_. For example, \*, +, and % are special characters.

**subfile.** A group of records of the same record format that can be displayed at the same time at a display station. The system sends the entire group of records to the display in a single operation and receives the group from the display in another operation.

**subfile control record format.** One of two record formats required to define a subfile in DDS. The subfile control record format describes the size of the subfile and the size of the subfile page, and is used by the program to write the subfile to and read the subfile from the display. See also *subfile record format*.

**subfile record format.** One of two record formats required to define a subfile in DDS. The subfile record format defines the fields in a subfile record and is used by the program to perform input, output, and update operations to the subfile. See also *subfile control record format*.

**substring.** A part of a character string.

**syntax checking.** A function of the system, a compiler, the BASIC interpreter, the CoOperative Development Environment/400 syntax checker, or SEU that checks individual statements for errors in their structure.

**system date.** The date assigned in the system values when the system is started. See also *creation date* and *job date*.

**system value.** Control information for the operation of certain parts of the system. A user can change the system value to define his working environment. System date and library list are examples of system values. Contrast with *network attribute*.

**Systems Application Architecture® (SAA®).** Pertaining to an architecture defining a set of rules for designing a common user interface, programming interface, application programs, and communications support for strategic operating systems such as the OS/2, OS/400, VM, and MVS™ operating systems.

**Systems Network Architecture (SNA).** In IBM networks, the description of the layered logical structure, formats, protocols, and operational sequences that are used for transmitting information units through networks, as well as controlling the configuration and operation of networks.

**time.** A three-part value or data type that designates a time of day in hours, minutes, and seconds.

**time stamp.** (1) To apply the current system time. (2) The value of an object that indicates the system time at some critical point in the object's history.

**timestamp.** In database support, a seven-part value or data type that consists of a date and time, expressed in years, months, days, hours, minutes, seconds, and microseconds.

**transaction.** In communications, an exchange between a program on a local system and a program on a remote system that accomplishes a particular action or result. See also *conversation* and *session*.

**translation table.** An object that contains a set of hexadecimal characters used to translate one or more characters of data. The table can be used for translation of data being moved between the system and a device. For example, data stored in one national language character set may need to be displayed or entered on display devices that support a different national language character set. The table can also be used to specify an alternative collating sequence or field translation functions. The system-recognized identifier for the object type is \*TBL. See also *table*.

**truncate.** (1) To cut off data that cannot be printed or displayed in the line width specified or available. Contrast with *fold*. (2) To cut off data that does not fit in the specified field length in a field definition.

| **Unicode.** A universal encoding scheme for written characters and text that enables the exchange of data  
| internationally. A Unicode field can contain all types of characters used on an iSeries™ server, including ideographic  
| (DBCS). Unicode data is composed of code units, which represent the minimal byte combination that can represent a  
| unit of text.

**unprotected field.** A displayed field for which operators can enter, change, or delete data.

**update operation.** An I/O process that changes the data in a record.

**user profile.** An object with a unique name that contains the user's password, the list of special authorities assigned to a user, and the objects the user owns. The system-recognized identifier for the object type is \*USRPRF.

**user-defined edit code.** A number (5 through 9) indicating that editing should be done on a numeric output field according to a pattern predefined to the system program. User-defined edit codes can take the place of edit words, so that repetitive coding of the same edit word is not necessary.

**validity checking.** To verify the contents of a field.

**variable.** A name used to represent data whose value can be changed while the program is running by referring to the name of the variable.

**variable-length.** Pertaining to a characteristic of a file in which the records can be of varying length, or pertaining to a characteristic of a field on a display that can be of varying length. Contrast with *fixed-length*.

**work station.** A device used to transmit information to or receive information from a computer, for example, a display station or printer.

**write operation.** An output operation that sends a processed record to an output device or output file.

**zero suppression.** The substitution of blanks for leading zeros in a number. For example, 00057 becomes 57 with zero suppression.





---

## Appendix G. 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:

- | IBM Corporation

| Software Interoperability Coordinator, Department 49XA  
| 3605 Highway 52 N  
| Rochester, MN 55901  
| U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced Function Printing  
AFP  
AS/400  
GDDM  
IBM  
Integrated Language Environment  
Intelligent Printer Data Stream  
IPDS  
iSeries  
MVS  
Operating System/400  
OS/2  
OS/400  
PrintManager  
RPG/400  
SAA  
Systems Application Architecture

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

---

## 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.

**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.

---

## Code disclaimer information

This document contains programming examples.

IBM 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.



---

# Index

## A

abbreviations of values and keywords 41

## B

batch source statements 3

## C

commands used to create files 4  
compiler example 34  
completing the DDS form 1  
creating files 4  
creating files with DDS 1

## D

DBCS  
    considerations 45  
    keywords 46  
DBCS considerations  
    character strings 47  
    entering bracketed character strings 47  
    printouts 48  
DDS form  
    completing 1  
DDS naming conventions 6  
debugging template 36  
describing data attributes 1  
display files  
    creating 1  
    syntax coding examples 11  
double-byte character set  
    considerations 45  
    keywords 46

## E

entering source statements 3  
example of compiler listing 34  
examples of DDS 15  
    database files 15  
    device files 20  
    field reference file 15  
    horizontal subfile 26  
    ICF file 30  
    inquiry display 20  
    join logical file 19  
    logical file with a new record  
        format 18  
    logical file with multiple formats and new keys 17  
    message subfile 28  
    physical file with new record  
        format 16  
    printer file 29

examples of DDS (*continued*)  
    program using physical, display, and printer files 32  
    subfile 23, 24  
examples of syntax coding 7

## F

FLAG parameter 4

## G

GENLVL (Severity Level) parameter 4

## I

ICF files  
    creating 1  
    syntax coding examples 13  
ideographic text (DBCS)  
    considerations 45  
interactive source statements 3

## K

keyword  
    abbreviations 41

## L

library list 39  
logical files  
    creating 1  
    syntax coding examples 9

## N

naming conventions 6  
numeric-only input-only field 20

## O

OPTION parameter 4  
overview of DDS 1

## P

PFILE keyword 17  
physical files  
    creating 1  
    syntax coding examples 8  
printer files  
    creating 1  
    syntax coding examples 12

## R

REF keyword 16, 39  
REFFLD keyword 39  
rules for keywords and parameter values 4

## S

Severity Level (GENLVL) parameter 4  
source statements 3  
specifying REF and REFFLD keywords 39  
syntax  
    keywords and parameter values 4  
    rules 4  
syntax coding examples 7  
    display files 11  
    ICF files 13  
    join logical files 10  
    logical files 9  
    physical files 8  
    printer files 12

## U

UNIQUE keyword 18

## V

value  
    abbreviations 41

## W

when to specify REF and REFFLD keywords 39







Printed in USA