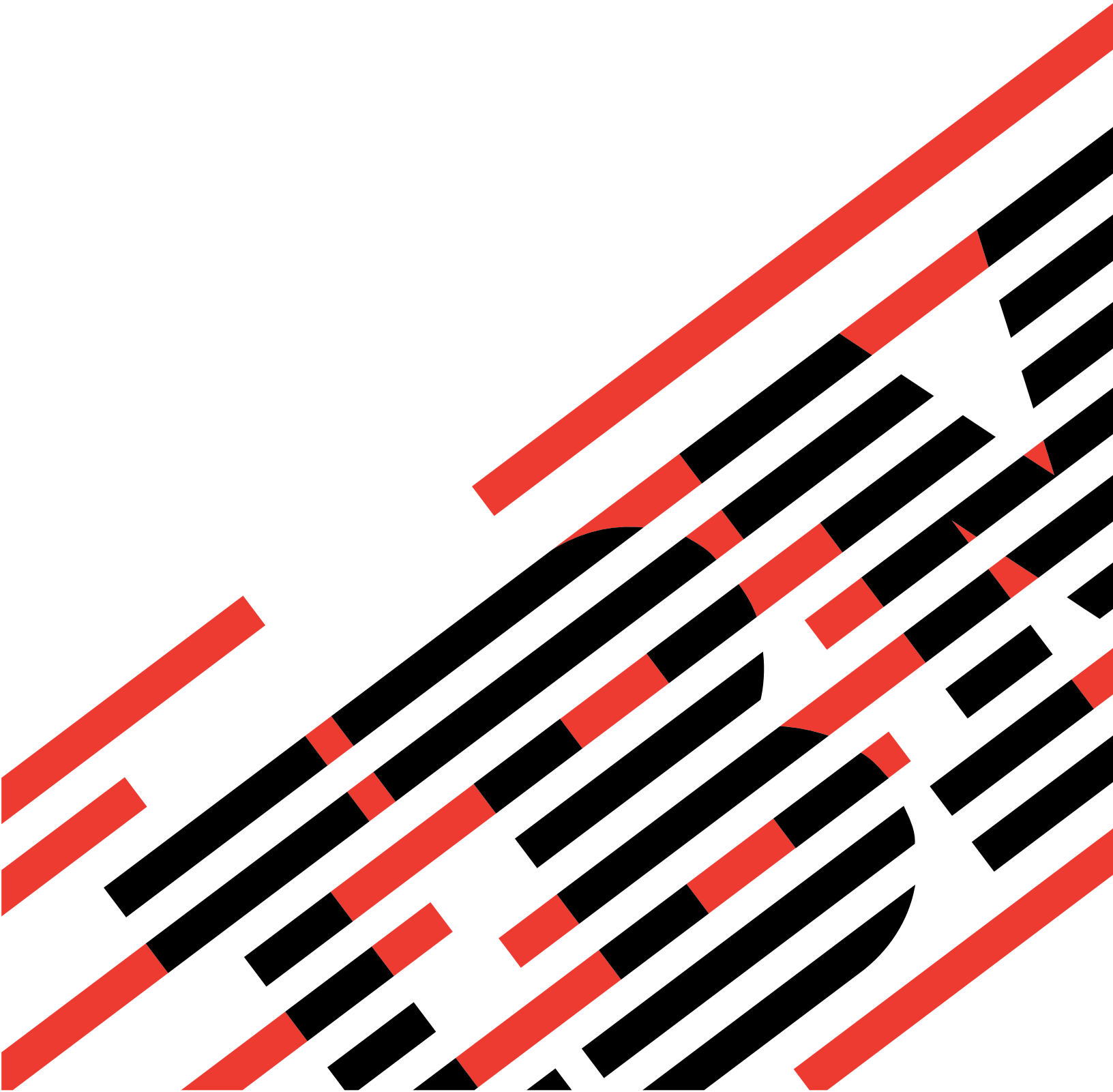
IBM

eserver

iSeries

DDS Reference: ICF Files

Version 5 Release 3



IBM

@server

iSeries

DDS Reference: ICF Files

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 43.

Fourth 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.

Contents

About DDS Reference: ICF Files

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

Who should read the DDS Reference: ICF Files book

This information is intended for programmers who use iSeries servers.

Conventions and terminology used in this book

- 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® communications manager that allows the personal computer to perform like a 5250 display station and use functions of the iSeries servers.
- *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 Reference: ICF files (about 369 KB).

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)  .

Chapter 1. Defining an Intersystem Communications Function file using DDS

This topic gives rules and examples for filling in positions 1 through 44 of the data description specifications (DDS) form. Chapter 2, "DDS keyword entries for ICF files (positions 45 through 80)" gives rules and examples for specifying DDS keywords.

For more information about keywords for ICF files, see the ICF Programming book.

Specify the entries in the following order to define an ICF file:

1. File-level entries (optional)
2. Record-level entries
3. Field-level entries (optional)

Repeat the record-level entries and field-level entries for each record format in the file.

Specify at least one record format in the file.

The maximum number of record formats in an ICF file is 1024. The maximum number of fields in any one record format is 32 767.

Note: Specify the file name with the Create Intersystem Communications Function File (CRTICFF) command, not DDS.

You can find an explanation of file level, record level, and field level as well as the syntax rules for specifying DDS keywords in the DDS Reference: Concepts information. You can also find a complete ICF example in the DDS Reference: Concepts information.

The following figure shows an ICF file coding example.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A* ICF FILE CODING EXAMPLE
00020A*
00030A          R RCD1                        RCVENDGRP(14)
00040A            FLDA           5
00050A            FLDB           5  0
00060A            FLDC          10  2
00070A
00080A          R RCD2
00090A  72 73
00110AON74                                    FAIL
00120A            FLDD          12
     A            FLDC      R                 REFFLD(A LIB1/FILEA)
     A
```

Figure 1. ICF File Coding Example

Positional entries for ICF files (positions 1 through 44)

This section describes how to specify the first 44 positions of the data description specifications (DDS) form for ICF files. To code the remaining part of the form, see Chapter 2, "DDS keyword entries for ICF files (positions 45 through 80)."

The following positional entries are described:

- Sequence number (positions 1 through 5)
- Form type (position 6)
- Comment (position 7)
- Conditioning (positions 7 through 16)
- Type of name or specification (position 17)
- Reserved (position 18)
- Name (positions 19 through 28)
- Reference (position 29)
- Length (positions 30 through 34)
- Data type (position 35)
- Decimal positions (positions 36 and 37)
- Usage (position 38)
- Location (positions 39 through 44)

Figure 1 on page 1 shows some positional entries for ICF files.

Sequence number for ICF files (positions 1 through 5)

Use these positions to specify a sequence number for each line on the form. The sequence number is optional and is for documentation purposes only.

Form type for ICF files (position 6)

Specify an A in this position to identify this as a DDS form. The form type is optional and is for documentation purposes only.

Comment for ICF files (position 7)

Specify an asterisk (*) in this position to identify this line as a comment. Comment lines can appear anywhere in DDS and are kept only in the source file. They appear on the source computer printout but do not appear on the expanded source computer printout.

Use positions 8 through 80 for comment text. A blank line (no characters specified in positions 7 through 80) is handled as a comment.

Conditioning for ICF files (positions 7 through 16)

Positions 7 through 16 are a multiple field area in which you can specify option indicators. Option indicators are 2-digit numbers from 01 to 99. Your program can set option indicators on (hex F1) or off (hex F0) to select a keyword for output operations. In ICF files, option indicators are valid only for record- and file-level keywords.

A condition is an ANDed grouping of two through nine indicators that must all be in effect (set off if the letter N is specified; set on if N is not specified) before the keyword is selected. You can specify a maximum of nine indicators for each condition and nine conditions for each keyword. Therefore, a maximum of 81 indicators can be specified for each keyword, when nine indicators are used with nine conditions. An AND condition occurs when you specify a condition requiring that more than one indicator must be on or off before the condition is satisfied. The first indicator you specify, AND the

second, AND the third, and so on, must all be in effect before the condition is satisfied and the keyword is selected. You must specify the keyword on the same line as the last (or only) set of indicators specified.

You can also specify several conditions for a keyword so that if any one of them is satisfied, the keyword is selected. This is called an OR relationship. In an OR relationship, if the first condition is satisfied, OR the second condition, OR the third condition, and so on, the keyword is selected. Note that conditions within the OR relationship can consist of just one indicator or can consist of several indicators ANDed together. Indicators can be ANDed to form a condition. Conditions can be ORed to give your program several ways to select the keyword.

Specify the conditions by entering the following values:

Position 7 (AND)
If you need more than three indicators to form an ANDed condition, specify the indicators on the next line or lines. You can specify an A in position 7 on the second or following lines to continue the ANDed condition, or you can leave it blank because A is the default.

Position 7 (OR)
If you specify several conditions that are to be ORed together, each condition must start on a new line and each condition, except the first, must have an O in position 7. An O specified for the first condition produces a warning message, and that position is assumed to be blank.

Positions 8, 11, 14 (NOT)
If you want an indicator to be off instead of on to satisfy a condition, specify an N in the position just preceding the indicator (position 8, 11, or 14).

Conditioning more than one keyword for ICF files:

If you want to condition one or more keywords, the last (or only) indicator must appear on the same line as the keywords. If the conditioning applies to keywords on more than one line, you must use keyword continuation for the indicators to apply to all keywords. See the syntax rules in the DDS Concept information for details.

Type of name or specification for ICF files (position 17)

Enter a value in this position to identify the type of name specified in positions 19 through 28. The valid entries for ICF files are:

Entry	Meaning
R	Record format name
Blank	Field name

Figure 1 on page 1 shows how to code the name type.

For more information on types of names, see “Name for ICF files (positions 19 through 28).”

Reserved for ICF files (position 18)

This position does not apply to any file type. Leave this position blank unless you use it for comment text.

Name for ICF files (positions 19 through 28)

Use these positions to specify record format names and field names.

Refer to the DDS Concepts information for rules to use when specifying record or field names in DDS.

Names must begin in position 19.

Record format name for ICF files

When you specify an R in position 17, the name specified in positions 19 through 28 is a record format name. You can specify more than one record format for an ICF file, but each record format name must be unique within that file.

Field name for ICF files

When you specify a blank in position 17, the name specified in positions 19 through 28 is a field name. Field names must be unique within the record format. For ICF files, the order in which field names are specified in the DDS is the order the fields take in the input and output buffers.

The keywords CANCEL, EOS, FAIL, and RQSWRT must have option indicators when they apply to a record with fields. Fields are ignored at run time (not sent across the line) when any of these keywords are in effect. At creation time, if any of these keywords have no option indicator and apply to a record with fields, a severe error is issued and the file will not be created.

Reference for ICF files (position 29)

Specify R in this position to use the reference function of the OS/400 program to copy the attributes of a previously defined named field (called the referenced field) to the field you are defining. The referenced field can be previously defined in the ICF file you are defining. The referenced field can also be defined in a previously created database file (the database file to be referenced is specified with the REF or REFFLD keyword). The field attributes referenced are the length, data type, and decimal positions of the field, as well as the ALIAS, FLTPCN, and TEXT keywords.

If you do not specify R, you cannot use the reference function for this field and you must specify field attributes for this field.

Position 29 must be blank at the file and record levels.

The name of the referenced field can be either the same as the field you are defining or different from the field you are defining. If the name of the referenced field is the same as the field you are defining, you need only specify the letter R in position 29 (in addition to specifying the name of the field you are defining in positions 19 through 28). If the name of the field you are defining is different, you must specify the name of the referenced field with the REFFLD (Referenced Field) keyword.

You can specify the name of the file defining the referenced field as a parameter value with the REF or REFFLD keyword. See REF and REFFLD keyword descriptions for ICF files, and the topic "When to specify REF and REFFLD keywords for DDS files" in the DDS Concepts information, for explanations of how the OS/400 program finds the referenced field.

You do not need to copy all attributes from the previously described field to the field you are defining. To override specific attributes of the referenced field, specify those attributes for the field you are defining. For example, if you specify a length for the field you are defining, the length is not copied from the referenced field.

When you override the data type to character (by specifying A in position 35), the decimal positions value is not copied from the referenced field.

Note: Once the ICF file is created, you can delete or change the referenced file without affecting the field descriptions in the ICF file. Delete and re-create the ICF file to incorporate changes made in the referenced file.

Length for ICF files (positions 30 through 34)

Specify the field length for each field (unless you copy the field's attributes from a referenced field). Specify the number of digits for a numeric field, or the number of characters for a character field. The length specification must be right-justified; leading zeros are optional. Valid length specifications for ICF

files are as follows:

Data Type	Valid Length
Character	1 through 32 767
Binary	1 through 9
Zoned decimal	1 through 31
Packed decimal	1 through 31
Floating-point single precision	1 through 9
Floating-point double precision	1 through 17

You can specify a maximum of 9 digits for single precision and 17 digits for double precision. However, the OS/400 program supports a floating-point accuracy of 7 digits for single precision and 15 digits for double precision.

The sum of the number of bytes occupied by all fields in a record must not exceed 32 767 for ICF files. The system determines the number of bytes actually occupied as follows:

Data Type	Bytes Occupied in Storage
Character	Number of characters
Binary	
1-4 digits	2 bytes
5-9 digits	4 bytes
Zoned decimal	Number of digits
Packed decimal	(Number of digits/2) + 1 (truncated if fractional)
Floating-point (single precision)	4 bytes
Floating-point (double precision)	8 bytes

If you are using a referenced field, you can override the length of the field by specifying a new value or by specifying the increase or decrease in length. To increase the length, specify +n where n is the increase. To decrease the length, specify –n, where n is the decrease. For example, an entry of +4 for a numeric field indicates that it is to be 4 digits longer than the referenced field.

Note: High-level languages can impose specific length and value restrictions on the field length. Observe these restrictions for files used by those languages.

Data type for ICF files (position 35)

Use this position to specify the data type of the field within the file. The valid data type entries for ICF files are as follows:

Entry	Meaning
P	Packed decimal
S	Zoned decimal
B	Binary
F	Floating-point
A	Character

Note: The data type O (DBCS-capable) supports DDS ICF files that use DBCS.

If a data type is not specified for the field and is not duplicated from a referenced field, DDS assigns a default depending on the value in the decimal positions (positions 36 and 37). If the decimal positions are blank, a default of character (A) is assigned. If the decimal positions contain a number in the range 0 through 31, a default of zoned decimal (S) is assigned.

Note: Specifying F in position 35 results in a single precision floating-point field. Use the FLTPCN keyword to specify double precision or to change the precision of an already specified floating-point field.

Decimal positions for ICF files (positions 36 and 37)

Use these positions to specify the decimal placement within a packed decimal, a zoned decimal, a floating point, or a binary field. Specify a decimal number from 0 through 31 to indicate the number of decimal positions to the right of the decimal point. (This number must not be greater than the number of digits specified in the field length.)

You can override or change these positions if you are using a referenced field. To override the positions, specify the new value. To change the positions, specify the amount by which you want the field increased or decreased and precede it with either a + or −, respectively. For example, an entry of +4 indicates there are to be four more digits to the right of the decimal point than were in the referenced field. If the resulting number of decimal positions is greater than the maximum allowed, you will receive an error message.

Note: High-level languages can impose specific length and value restrictions on the decimal positions. Observe these restrictions for files used by those languages.

Usage for ICF files (position 38)

The valid entries for this position are:

Entry Meaning

B or blank
Both input/output field

P Program-to-system field

A program-to-system field is used to communicate between an application program and the sending system (which is local to the application program). It is not sent as part of the data record across the line to the receiving system.

The following rules apply to program-to-system fields:

- The field must be a named, numeric, or alphanumeric output-only field.
- In the record format, the program-to-system fields must be defined after all the data fields (those with a use of B or blank).
- A field cannot be defined as both a data field and a program-to-system field; the field names must be unique.
- A program-to-system field can be named on an EVOKE, SECURITY, TIMER, or VARLEN keyword.
- The only valid keywords for a program-to-system field are ALIAS, FLTPCN, REFFLD and TEXT.

Location for ICF files (positions 39 through 44)

These positions do not apply to ICF files. Leave these positions blank unless you use them for comment text.

Chapter 2. DDS keyword entries for ICF files (positions 45 through 80)

This section contains the keyword entries that you can specify when you define ICF files. They are typed in positions 45 through 80 (functions). See the DDS Reference: Concepts information for a discussion of the general rules for specifying keywords.

The following keywords are valid for ICF files:

ALIAS
ALWWRT
CANCEL
CNLINVITE
CONFIRM
CTLDTA
DETACH
DFREVOKE
ENDGRP
EOS
EVOKE
FAIL
FLTPCN
FMH
FMTNAME
FRCDTA
INDARA
INDTXT
INVITE
NEGRSP
PRPCMT
RCVCANCEL
RCVCONFIRM
RCVCTLDTA
RCVDETACH
RCVENDGRP
RCVFAIL
RCVFMH
RCVNEGRSP
RCVROLLB
RCVTKCMT
RCVTRNRND
RECID
REF
REFFLD
RQSWRT
RSPCONFIRM
SECURITY
SUBDEV
SYNLVL
TEXT
TIMER
TNSSYNLVL
VARBUFMGT
VARLEN

ALIAS (Alternative Name) keyword for ICF files

Use this field-level keyword to specify an alternative name for a field. When the program is compiled, the alternative name is brought into the program instead of the DDS field name. The high-level language compiler in use determines whether the ALIAS name is used. Refer to the appropriate high-level language reference manual for information about ALIAS support for that language.

The format of the keyword is:

```
ALIAS(alternative-name)
```

Refer to the DDS Reference: Concepts information for ALIAS naming conventions.

The alternative-name must be different from all other alternative names and from all DDS field names in the record format. If a duplicate name is found, an error is issued on the field name or alternative name.

An alternative name cannot be used within DDS or any other OS/400 function (for example, as a key field name, as the field name specified for the REFFLD keyword, or as a field name used in the Copy File (CPYF) command).

When you refer to a field that has the ALIAS keyword, the ALIAS keyword is copied in unless the ALIAS keyword is explicitly specified on the referencing field.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the ALIAS keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00070A            FIELDA        25A         ALIAS(CUSTOMERNAME)
     A
```

ALWWRT (Allow Write) keyword for ICF files

Use this file- or record-level keyword so that your program can indicate when it has finished sending.

This keyword has no parameters.

ALWWRT is ignored at run time when DETACH, EOS, RSPCONFIRM, or RQSWRT are in effect. These keywords must have option indicators if they apply to a record for which ALWWRT applies. If a DETACH, EOS, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which ALWWRT applies, an error message is issued and the ALWWRT keyword is ignored at creation time.

You cannot specify ALWWRT with the TIMER keyword.

The ALWWRT keyword can be specified once at the file-level or once for each record format.

Option indicators are valid with this keyword. When you specify this keyword at the file-level, you should specify an option indicator.

Example:

The following example shows how to specify the ALWWRT keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
01000A  21                                ALWWRT
02000A          R CUSMST
     A
```

CANCEL (Cancel) keyword for ICF files

Use this file- or record-level keyword to cancel the current chain of data (group of records) that is being sent to the remote program.

This keyword has no parameters.

The CANCEL keyword must have an option indicator when it applies to a record for which any of the following keywords apply:

CNLINVITE
EVOKE
RQSWRT
RSPCONFIRM
VARBUFMGT
VARLEN

Data fields and these keywords are ignored at run time when the CANCEL keyword is in effect. If a CANCEL keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time. If a CANCEL keyword with no option indicator applies to a record with data fields, a severe error is issued and the file is not created.

The CANCEL keyword is ignored at run time when EOS, FAIL, or NEGRSP is in effect. These keywords must have option indicators if they apply to a record for which CANCEL applies. If a EOS, FAIL, or NEGRSP keyword with no option indicator applies to a record for which CANCEL applies, an error message is issued and the CANCEL keyword is ignored at creation time.

You cannot specify CANCEL with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the CANCEL keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A  02                                CANCEL
     A          R RCD1
     A
```

CNLINVITE (Cancel Invite) keyword for ICF files

Use this file- or record-level keyword to cancel any valid invite operation for which no input has yet been received.

This keyword has no parameters.

The CNLINVITE keyword must have an option indicator when it applies to a record for which a RQSWRT, RSPCONFIRM, or EVOKE keyword applies. At run time, these keywords are ignored when CNLINVITE is in effect. If a CNLINVITE keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time.

The CNLINVITE keyword is ignored at run time when CANCEL, EOS, FAIL, or NEGRSP is in effect. These keywords must have option indicators when they apply to a record for which the CNLINVITE keyword applies. If a CANCEL, EOS, FAIL, or NEGRSP keyword with no option indicator applies to a record for which CNLINVITE applies, an error message is issued and the CNLINVITE keyword is ignored at creation time.

You cannot specify CNLINVITE with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the CNLINVITE keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R RCD1                    CNLINVITE
     A
```

CONFIRM (Confirm) keyword for ICF files

Use this file- or record-level keyword to request that the remote program confirms receiving the data.

This keyword has no parameters.

The CONFIRM keyword is valid only if the transaction was established with a synchronization level of confirm (SYNLVL(*CONFIRM) keyword). If the transaction was established with a synchronization level of none (SYNLVL(*NONE) keyword), the CONFIRM keyword is rejected with an OS/400 error message.

The CONFIRM keyword is ignored at run time when EOS, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the CONFIRM keyword applies. If an EOS, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which CONFIRM applies, an error message is issued and the CONFIRM keyword is ignored at creation time.

You cannot specify CONFIRM with the TIMER keyword.

The CONFIRM keyword can be specified once at the file-level or once for every record format.

Option indicators are valid with this keyword.

Example:

The following example shows how to specify the CONFIRM keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R RCD
00020A  01                                CONFIRM
     A
```

If option indicator 01 is on, the remote program will confirm receiving the data by sending either a positive or negative response.

CTLDTA (Control Data) keyword for ICF files

Use this file- or record-level keyword to inform the remote program that control data is being sent.

This keyword has no parameters.

The CTLDTA keyword is ignored at run time when the EOS, RSPCONFIRM, or RQSWRT keywords are in effect. These keywords must have option indicators if they apply to a record for which CTLDTA applies. If an EOS, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which CTLDTA applies, an error message is issued and the CTLDTA keyword is ignored at creation time.

You cannot specify CTLDTA with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the CTLDTA keyword at the record level.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R SNDCTLD
     A                                CTLDTA
     A            USRSCTLD     100A
     A
```

DETACH (Detach) keyword for ICF files

Use this file- or record-level keyword to explicitly inform the remote program that your program is done sending data and wants to end the transaction.

This keyword has no parameters.

The DETACH keyword must have an option indicator when it applies to a record for which any of the following keywords apply:

ALWWRT
ENDGRP
FMH
FRCDTA
INVITE
SUBDEV

At run time, these keywords will be ignored when the DETACH keyword is in effect. If a DETACH keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time.

The DETACH keyword is ignored at run time when EOS, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the DETACH keyword applies. If an EOS, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which DETACH applies, an error message is issued and the DETACH keyword is ignored at creation time.

You cannot specify DETACH with the TIMER keyword.

At most, the DETACH keyword can be specified once at the file-level or once per record format.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the DETACH keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R RCD
00020A  01                                DETACH
     A
```

If option indicator 01 is on, the transaction between your program and the remote program will be ended.

DFREVOKE (Defer Evoke) keyword for ICF files

Use this file- or record-level keyword with the EVOKE keyword to delay an evoke request until either the send buffer is full of data or a FRCDTA keyword is received. The DFREVOKE keyword is useful only for specialized applications that must have the data sent at the same time as the EVOKE.

This keyword has no parameters.

You cannot specify DFREVOKE with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the DFREVOKE keyword at the record level.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R PGMSTART
     A                                    EVOKE(&LIB/&PGMID);
     A                                    DFREVOKE
     A
     A
```

ENDGRP (End of Group) keyword for ICF files

Use this file- or record-level keyword so that your program can indicate the end of a user-defined group of records.

This keyword has no parameters.

The ENDGRP keyword is ignored at run time when DETACH, EOS, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the ENDGRP keyword applies. If a DETACH, EOS, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which ENDGRP applies, an error message is issued and the ENDGRP keyword is ignored at creation time.

You cannot specify ENDGRP with the TIMER keyword.

Option indicators are valid for this keyword. (When you specify this keyword at the file-level, you should specify an option indicator.)

Example:

The following example shows how to specify the ENDGRP keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00030A          R RECORD1                   ENDGRP
     A
```

EOS (End of Session) keyword for ICF files

Use this file- or record-level keyword to specify an end of session function. To end a session, your program issues a write operation with the EOS keyword in effect.

This keyword has no parameters.

The EOS keyword must have an option indicator when it applies to a record for which any of the following keywords apply:

ALWWRT	EVOKE	NEGRSP
CANCEL	FAIL	RQSWRT
CNLINVITE	FMH	RSPCONFIRM
CONFIRM	FMTNAME	SUBDEV
DETACH	FRCDTA	VARBUFMGT
ENDGRP	INVITE	VARLEN

At run time, data fields and these keywords are ignored when the EOS keyword is in effect. If an EOS keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time. If an EOS keyword with no option indicator applies to a record with data fields, a severe error is issued and the file is not created.

You cannot specify EOS with the TIMER keyword.

Option indicators are valid for this keyword. When you specify this keyword at the file level, you should specify an option indicator.

Example:

The following example shows how to specify the EOS keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A  01                              EOS
     A          R RCD
     A
```

If indicator 01 is on and the program does an output operation, the session will be ended.

EVOKE (Evoke) keyword for ICF files

Use this file- or record-level keyword to start a program on the remote system.

The format of the keyword is:

```
EVOKE([library-/e1e/]program-name [parameter-1...[parameter-255]])
```

The program-name can be any one of the following:

program-name
This is the name of the program to be started on the remote system. The name is syntax-checked at creation time for a valid object name.

'character-string-1'
This is the name of the program to be started on the remote system. The name you specify must be in a format acceptable to the remote system because the character string will not be syntax-checked.

&field-name-1
The specified field contains the name of the program to be started on the remote system. The field name must be a valid field you have specified in the record format and must be a character field (data type of A). The name you specify must be in a format acceptable to the remote system.

The optional library-name can be any one of the following:

library-name/
This is the name of the library that contains the program to be started on the remote system. The name is syntax-checked at creation time for a valid object name. For this keyword, *CURLIB and *LIBL are not valid names. If either one needs to be specified, a quoted character string should be used.

'character-string-2'/
This is the name of the library that contains the program to be started on the remote system. The name you specify must be in a format acceptable to the remote system because the character string will not be syntax-checked.

&field-name-2/
The specified field contains the name of the library that contains the program to be started on the remote system. The field name must be a valid field you have specified in the record format and must be a character field (data type of A). The name you specify must be in a format acceptable to the remote system.

Note: If the remote system is an iSeries server and no library was specified, the library list will be used to search for the program.

Parameter-1...parameter-255 can be any of the following:

'character-string-3'
This is a character string that is passed to the program on the remote system. The character string must be in a format acceptable to the remote system because it will not be syntax-checked.

[&]field-name-3
This is the name of the field that contains the data you want passed to the program on the remote system. The field name must be a valid field you have specified in the record format.

numeric-value-3
This is a numeric value that is passed to the program on the remote system. The numeric value can be a negative or positive value (signed or unsigned). A decimal point of , or . is optional. No decimal alignment will be performed. Leading zeros will not be suppressed. The data is sent as a zoned decimal value. The following are all valid numeric values:

999.6
–999,6
01587

Special considerations when using the EVOKE keyword with ICF files

The following are special considerations when using the EVOKE keyword.

- When the EVOKE keyword is specified at the file-level, you cannot specify a field name as a parameter value.
- The maximum length allowed for the combined program name and library name is 64. The slash between the program name and the library name is counted as part of the 64 bytes. APPC does not send the slash unless it is specified within a literal (for example, LIBRARY/PROGRAM).
- The total length of parameter-1 through parameter-255 cannot be more than 32 767 bytes.

  Note: In calculating the maximum length of PIP data for APPC, the following must be considered:

  Four bytes must be added to the length of each of these parameters. An additional 4 bytes must be added if any parameters are specified. These bytes are required by the system.

  Use the following formula to determine the total length of the parameters:

  4 + (length of 1st parameter + 4) + (length of
  2nd parameter + 4)
  + ... (length of nth parameter + 4)

  Following is an example using this formula:

  EVOKE(LIBRARY1/PROGRAM1 'THIS IS AN EXAMPLE OF
  A
  CHARACTER STRING' &FIELD1 35)

  Assume that &FIELD1 has a length of 10.
  4 + (40 + 4) + (10 + 4) + (2 + 4) = 68
- The length of each parameter (parameter-1 through parameter-255) should be the same as the length of the corresponding parameter in the remote program.
- If a field name with a usage of P is specified as a parameter of the EVOKE keyword, this field is not sent as part of the data record.
- A program evoked on an iSeries server will receive any parameters sent by the remote program just as if they had been passed by the CL CALL command.

  Note: If the job on the iSeries server is a prestart job, the program must use the RTVDTAARA command to receive the parameters.

This keyword is required when either the SECURITY or SYNLVL keywords are specified. At run time, the SECURITY and SYNLVL keywords are used only when EVOKE is also in effect.

The EVOKE keyword is ignored at run time when CANCEL, CNLINVITE, EOS, FAIL, NEGRSP, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the EVOKE keyword applies. If a CANCEL, CNLINVITE, EOS, FAIL, NEGRSP, or RQSWRT keyword with no option indicator applies to a record for which EVOKE applies, an error message is issued and the EVOKE keyword is ignored at creation time.

You cannot specify EVOKE with the TIMER keyword.

Option indicators are valid for this keyword and are required if this keyword is specified more than once for each record format or file.

Example:

The following example shows how to specify the EVOKE keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R RCD
00020A  01        :                       EVOKE(LIBRARY1/PROGRAM1)    1
00030A  02        :                       EVOKE(LIBRARY2/PROGRAM2)    1
     A            :                        :
     A            :                        :
00090A          R RCD2                    EVOKE(&FIELD2/&FIELD1 'ABC' 10.1 +
00100A                                    FIELD3)             2
00110A            FIELD1        10A  P
00120A            FIELD2        10A  P
00130A            FIELD3         5B  P
     A
```

1 If indicator 01 is on, PROGRAM1 in LIBRARY1 will be started. If indicator 02 is on, PROGRAM2 in LIBRARY2 will be started.

2 &FIELD1 contains the name of the program to be started. &FIELD2 contains the name of the library. The character string ABC, numeric value 10.1, and the value in FIELD3 will be passed to the program on the remote system.

FAIL (Fail) keyword for ICF files

Use this file- or record-level keyword to inform the remote program that the data sent or received was invalid.

This keyword has no parameters.

The FAIL keyword must have an option indicator when it applies to a record that has data fields (use of B or blank) or for which any of the following keywords apply:

CANCEL
CNLINVITE
EVOKE
NEGRSP
RQSWRT
RSPCONFIRM
VARBUFMGT
VARLEN

At run time, data fields and these keywords will be ignored when the FAIL keyword is in effect. If a FAIL keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time. If a FAIL keyword with no option indicator applies to a record with data fields, a severe error is issued and the file will not be created.

The FAIL keyword is ignored at run time when the EOS keyword is in effect. EOS must have an option indicator when it applies to a record for which the FAIL keyword applies. If an EOS keyword with no option indicator applies to a record for which FAIL applies, an error message is issued and the FAIL keyword is ignored at creation time.

FAIL cannot be specified with the TIMER keyword.

Option indicators are valid for this keyword. When you specify this keyword at the file level, you should specify an option indicator.

Example:

The following example shows how to specify the FAIL keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R INQ
00020A  99                                FAIL
     A
```

FLTPCN (Floating-Point Precision) keyword for ICF files

Use this field-level keyword to specify the precision of a floating-point field.

The format of the keyword is:

```
FLTPCN(*SINGLE | *DOUBLE)
```

where the *SINGLE parameter is single precision and the *DOUBLE parameter is double precision.

This keyword is valid for floating-point fields only (data type F).

A single-precision field can be up to 9 digits. A double-precision field can be up to 17 digits. If you specify a field length greater than 9 (single precision) or 17 (double precision), an error message is issued and the file is not created. ICF supports a floating-point accuracy of 7 digits for single precision and 15 digits for double precision.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the FLTPCN keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00090A            FIELDA        17F 4       FLTPCN(*DOUBLE)
     A
```

FMH (Function Management Header) keyword for ICF files

Use this file- or record-level keyword to inform the remote program that a function management header (FMH) is being sent.

This keyword has no parameters.

The FMH keyword is ignored at run time when EOS, DETACH, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the FMH keyword applies. If an EOS, DETACH, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which FMH applies, an error message is issued and the FMH keyword is ignored at creation time.

You cannot specify FMH with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the FMH keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD                      FMH
     A            FLD1          10A  B
     A
```

FMTNAME (Format Name) keyword for ICF files

Use this file- or record-level keyword to specify that the record format name is to be sent to the remote program when your program issues an output operation.

This keyword has no parameters.

The FMTNAME keyword is ignored at run time when EOS, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the FMTNAME keyword applies. If an EOS, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which FMTNAME applies, an error message is issued and the FMTNAME keyword is ignored at creation time.

You cannot specify FMTNAME with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the FMTNAME keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1
     A  01                                FMTNAME
     A             FIELD1        10A  B
     A
```

If indicator 01 is on and the program does a write operation, the record format name will be sent as an APPC map name to the remote system.

FRCDTA (Force Data) keyword for ICF files

Use this record-level keyword to clear the buffer when there is no more data to send, without waiting for the buffer to become full.

Note: If the keyword is specified after each write statement, performance problems may occur.

There is no wait for confirmation. (The CONFIRM keyword provides similar function but additionally provides confirmation of data sent. Your program must wait for the response from the other end before continuing to the next program statement.)

This keyword has no parameters.

The FRCDTA keyword is ignored at run time when any of the following keywords is in effect:

DETACH
EOS
RQSWRT
RSPCONFIRM

These keywords must have option indicators when they apply to a record specifying FRCDTA. If a keyword from this list has no option indicator and applies to a record with FRCDTA, an error message is issued and the FRCDTA keyword is ignored at creation time.

You cannot specify FRCDTA with the TIMER keyword.

The FRCDTA keyword can be specified at most once per record format.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the FRCDTA keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A        R REC1
00020A  10                                  FRCDTA
00030A          FLD1            10
00040A          FLD2             5
     A
```

When option indicator 10 is on and the program does a write operation, the FRCDTA keyword sends communications data currently held in the buffer.

INDARA (Indicator Area) keyword for ICF files

Use this file-level keyword to remove option and response indicators from the buffer, or record area, and place them in a 99-byte separate indicator area. Specifying the INDARA keyword provides the following advantages:

- Simplifies COBOL/400* programming when both option and response indicators are used. If the same indicator is used as a response indicator and as an option indicator, both indicators always have the same value, regardless of the order in which they are specified in the DDS.
- Assists the RPG/400* programmer using program-described WORKSTN files.

This keyword has no parameters.

If you specify the INDARA keyword, some high-level languages require that you specify in your program that a separate indicator area is to be used. See the appropriate high-level language manual.

If you specify the INDARA keyword, you can add, change, or delete option and response indicators in the DDS and recompile the file without recompiling the high-level language program. This is allowed because the field locations in the buffer have not changed and, therefore, the level check data has not changed. However, if the program is to take advantage of the new indicators, the program would still need to be changed and recompiled.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the INDARA keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A                                      INDARA
00020A  41                                  FAIL
00030A                                      RCVTRNRND(14 'Turn around')
00040A          R RCD
00050A            ACTNBR        10
     A
```

With the INDARA keyword specified, option indicator 41 and response indicator 14 are removed from the buffer for RCD and placed in the separate indicator area. Only ACTNBR, a named field, remains in the buffer for record format RCD.

INDTXT (Indicator Text) keyword for ICF files

Use this file- or record-level keyword to associate descriptive text (indicating intent or use) with a specific response or option indicator.

The format of the keyword is:

```
INDTXT(response-or-option-indicator 'indicator-text')
```

You can specify this keyword once for each response and option indicator.

Indicator-text is a required parameter value, and must be a character string enclosed in apostrophes. If the length of the string is greater than 50 positions, only the first 50 characters are used by the high-level language compiler. The text is used during compilation to help program documentation.

The INDTXT keyword does not cause the specified indicator to appear in either the input or output record area. It provides text to be associated with the indicator. Once an indicator has been given a textual assignment (either by this keyword or by the response indicator text), no other textual assignment is made. A message is issued and the keyword is ignored. This differs from other keywords that can have indicators specified as parameter values; for other keywords, only the text is ignored.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the INDTXT keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A                                     INDTXT(02 'Alternate month')
00020A          R MASTER
00030A            MTH                   2 10
00040A  02        ALTMTH                2 10
     A
```

The INDTXT keyword describes the use of option indicator 02. In a compiler computer printout for a high-level language, 'Alternate month' would be printed as a comment with the description of indicator 02.

INVITE (Invite) keyword for ICF files

Use this file- or record-level keyword to invite the program device for a later read. To send an invite to the program device, your program issues a write operation to that program device with the INVITE keyword in effect.

The INVITE keyword provides some performance improvement if your application program is doing interactive processing with the program device. Normally, a read request is sent to a device when your program issues an input operation. However, the INVITE keyword allows you to request the read when you issue the output operation. After the output operation is complete, your program can do other processing while the invited program device is sending data and the OS/400 program processes the received data. This might improve the performance of your program. When your application program is ready to process the data, it issues an input operation.

This keyword has no parameters.

The INVITE keyword is ignored at run time when EOS, RSPCONFIRM, or DETACH is in effect. These keywords must have option indicators when they apply to a record for which the INVITE keyword applies. If an EOS, RSPCONFIRM, or DETACH keyword with no option indicator applies to a record for which INVITE applies, an error message is issued and the INVITE keyword is ignored at creation time.

You cannot specify INVITE with the TIMER keyword.

Option indicators are valid for this keyword.

The INVITE keyword cannot be specified at both the file- and record-level.

Example:

The following example shows how to specify the INVITE keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A  01                                  INVITE
00020A          R RCD1
00030A            FLD1          10
00040A            FLD2           5
     A
```

The INVITE keyword is in effect only when option indicator 01 is set on.

NEGRSP (Negative Response) keyword for ICF files

This file- or record-level keyword sends a negative response to the remote program to indicate that your program detected an error in the data received.

The format of the keyword is:

```
NEGRSP[(&field-name)]
```

The optional parameter, &field-name,; specifies the name of a field that contains sense data to be sent to the remote program with the negative response. The specified field name must exist in the record format, and the field must be a character field with length at least 8, data type A, and usage B or blank.

The NEGRSP keyword must have an option indicator when it applies to a record for which any of the following keywords apply:

CANCEL
CNLINVITE
EVOKE
RQSWRT
RSPCONFIRM
VARBUFMGT
VARLEN

At run time, these keywords are ignored when the NEGRSP keyword is in effect. If a NEGRSP keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time.

NEGRSP is ignored at run time when EOS or FAIL is in effect. These keywords must have option indicators if they apply to a record for which NEGRSP applies. If an EOS or FAIL keyword with no option indicator applies to a record for which NEGRSP applies, an error message is issued and the NEGRSP keyword is ignored at creation time.

When you specify NEGRSP at the file-level, you cannot specify the field name parameter.

You cannot specify NEGRSP with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the NEGRSP keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1
     A  01                                NEGRSP(&FIELDB);
     A            FIELDA        25A  B
     A            FIELDB        80A  B
     A
```

If indicator 01 is on, a write operation to RCD1 will send a negative response and send the first 8 bytes of FIELDB to the remote program. Note that no data from RCD1 other than the sense data will be sent with the negative response.

PRPCMT (Prepare for Commit) keyword for ICF files

Use this record-level keyword to request that the remote program prepare for a synchronization point. An output operation with the PRPCMT keyword specified, forces any data in the output buffer to be sent.

This keyword has no parameters.

When this operation does not complete, your program does not continue until a response is received. The remote program must perform a commit or rollback operation or issue a FAIL or EOS to indicate whether it is prepared to commit its protected resources.

PRPCMT is only valid with a synchronization level of *COMMIT specified on the EVOKE.

The only keywords that can be specified with PRPCMT are VARBUFMGT and VARLEN.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the PRPCMT keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1
     A                                    PRPCMT
     A
```

RCVCANCEL (Receive Cancel) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator to inform your program that the remote program has sent a CANCEL.

The format of the keyword is:

```
RCVCANCEL(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. The text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVCANCEL with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVCANCEL keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1                      RCVCANCEL(34 'Received - +
     A                                      Cancel')
     A
```

Indicator 34 is set on when a CANCEL is received on an input operation from RCD1.

RCVCONFIRM (Receive Confirm) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator if the data received by your program contains a confirmation request from the remote program.

The format of the keyword is:

```
RCVCONFIRM(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. The text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVCONFIRM with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVCONFIRM keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A                                  RCVCONFIRM(44 'Waiting for a +
00020A                                  response')
00030A          R RCD
     A
```

Response indicator 44 is set on to indicate receipt of the confirmation request from the remote program.

RCVCTLDTA (Receive Control Data) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator to inform your program that control data has been received.

The format of this keyword is:

```
RCVCTLDTA(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. The text has no function in the file or the program except as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, only the first 50 characters are printed.

You cannot specify RCVCTLDTA with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVCTLDTA keyword at the record level.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCVCTLD
     A                                    RCVCTLDTA(66 'received control +
     A                                    data')
     A            USRRCTLD     100A
     A
```

RCVDETACH (Receive Detach) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator if the remote program is ending the transaction.

The format of the keyword is:

```
RCVDETACH(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. The text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVDETACH with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVDETACH keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A                                    RCVDETACH(44 'Transaction is  +
00020A                                    finished')
00030A          R RCD
     A
```

Response indicator 44 is set on when the remote program ends the transaction.

RCVENDGRP (Receive End of Group) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator to inform your program of the end of a user-defined group of records.

The format of the keyword is:

```
RCVENDGRP(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. This text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVENDGRP with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVENDGRP keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00100A          R CUSMST
00200A                                  RCVENDGRP(68 'End of group received-
     A                                  ')
     A
```

Response indicator 66 is set on when the remote program indicates that this is the end of a user-defined group of records.

RCVFAIL (Receive Fail) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator when the local program determines that the remote program has sent a FAIL. If this condition occurs when the RCVFAIL keyword was not specified, an OS/400 message notifies the local program that the remote program has sent a FAIL.

The format of the keyword is:

```
RCVFAIL(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. The text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVFAIL keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A                                  RCVFAIL(10 'Fail received')
00020A
00030A          R RCD
     A
```

Indicator 10 is set on when the remote program sends a fail indication.

RCVFMH (Receive Function Management Header) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator to inform your program that a function management header has been received.

The format of the keyword is:

```
RCVFMH(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. This text has no function in the file or the program other than as a

comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVFMH with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVFMH keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                     RCVFMH(24 'Received FMH')
     A          R RCD1
     A
```

Indicator 24 is set on when a function management header is received.

RCVNEGRSP (Receive Negative Response) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator to inform your program that the remote program has sent a negative response.

The format of the keyword is:

```
RCVNEGRSP(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. This text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVNEGRSP with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVNEGRSP keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                     RCVNEGRSP(67 'Negative re-
     A                                     sponse')
     A          R RCD1
     A
```

Indicator 67 is set on when a negative response is received.

RCVROLLB (Receive Rollback Response Indicator) keyword for ICF files

Use this file- or record-level keyword to indicate if a rollback operation has been received.

The format of the keyword is:

```
RCVROLLB(response-indicator {'text'})
```

The response indicator parameter is a required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. This text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

The TIMER keyword is not allowed with the RCVROLLB keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVROLLB keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                     RCVROLLB(67 'Receive RB')
     A          R REC1
     A
```

Indicator 67 is set on if a receive rollback has been received.

RCVTKCMT (Receive Take Commit Response Indicator) keyword for ICF files

Use this file- or record-level keyword to indicate if a take_commit request has been received.

The format of the keyword is:

```
RCVTKCMT(response-indicator {'text'})
```

The response indicator parameter is a required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. This text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

The TIMER keyword is not allowed with the RCVTKCMT keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVTKCMT keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
     A                                     RCVTKCMT(67 'Take Commit')
     A          R REC1
     A
```

The example shows that indicator 67 is set on if a take_commit request has been received.

RCVTRNRND (Receive Turnaround) keyword for ICF files

Use this file- or record-level keyword to set on a response indicator to inform your program that the sending program has stopped sending and has given the local program the right to send.

The format of the keyword is:

```
RCVTRNRND(response-indicator ['text'])
```

The response-indicator parameter is required.

The optional text is included on the computer printout created at program compilation time to explain the intended use of the indicator. This text has no function in the file or the program other than as a comment. The apostrophes are required. If you specify more than 50 characters between the apostrophes, the text is truncated to 50 characters on the program computer printout.

You cannot specify RCVTRNRND with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the RCVTRNRND keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A                                  RCVTRNRND(44 'Host has stopped +
00020A                                  sending')
00030A          R CUSMST
     A
```

RECID (Record Identification) keyword for ICF files

Use this record-level keyword to allow your program to identify a record-by-record format when it issues a read-from-invited-devices operation using the name of the file. When you use an input operation, the OS/400 program compares data in the record received with the selection value specified in the parameter values. The selection value is that data beginning at the specified starting position must equal the specified compare value. Your program can then determine the record format of the data just read.

The format of the keyword is:

```
RECID(starting-position compare-value)
```

The starting-position parameter specifies a position relative to the start of the data in the buffer (disregarding indicators) to test for the record's ID. If the INDARA keyword is used, the start of data and buffer positions are the same. For a description of the buffer, see example 3 below. The position parameter can be either

```
nnnnn
```

or

```
*POSnnnnn
```

where nnnnn is a number that is one to five digits long. For example, the following are equivalent pairs:

```
1 and *POS1
34 and *POS34
12025 and *POS12025
```

The compare-value parameter can be one of the following:

Value Meaning

***ZERO**
The value to be tested for is zero (hex F0). Equivalent to 0.

***BLANK**
The value to be tested for is blank (hex 40). Equivalent to blank.

'character-string'
The value to be tested for is the specified character string. The length of the string is limited to

the length from the RECID position parameter specified to the end of the shortest nonzero record format in the file (not including that record format's indicators or program fields).

A record format specifying the RECID keyword must contain at least one data field (usage of B).

You can specify the RECID keyword more than once in a record format. If you do so, data in the record is compared with each RECID keyword in the order specified until a match is found. The first record format whose selection value is satisfied by the data is the record format selected. If no match is found or no user data is received, the RECID default record format is used. The RECID default record format will be the first record format in the file that does not have the RECID keyword specified for it. However, if every record format in the file has the RECID keyword specified for it, the default record format will be the first record format in the file.

A message is issued to your program when data is received and no match is found and the RECID default record format has the RECID keyword specified for it.

When comparing the data received with the RECID keyword, if the position to be compared is beyond the last byte of data received, the data will be assumed to be blanks (hex 40).

This keyword is ignored at program run time unless the FMTSLT(*RECID) parameter is specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

You cannot specify RECID on the same record format as the VARBUFMGT keyword.

Option indicators are not valid for this keyword.

Example 1:

Record format DFTFMT will be the RECID default record format.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R DFTFMT
00020A            ID             3A
00030A            FLD1          20A
00040A            FLD2           5B 0
     A
00050A          R RCD1                      RECID(1 'ABC')
00060A            ID             3A
00070A            FLD1          10S 0
00080A            FLD2           5B 0
     A
00090A          R RCD2                      RECID(1 'DEF')
00100A            ID             3A
00110A            FLD1          10S 0
00120A            FLD2           5A
00130A            FLD3           2B 0
     A
```

Example 2:

Record format RCD1 will be the RECID default record format. If no match is found, an escape message will be issued to your program because the RECID default record format has the RECID keyword specified for it. If no data is received, record format RCD1 will be used.

An application program reads header and detail records from an ICF file. The program issues input operations to the file name (not to individual record names) and receives the records (headers and details) in the order the sending application sends them. In this example, the sending and receiving applications provide for an explicit code (an H for header records and a D for detail records) to identify

which type of record is being sent and received. The RECID keyword identifies where in the input buffer (disregarding indicators) the H or D appears and specifies the value (starting in the position specified) that identifies the type of record.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R RCD1                      RECID(1 'H')
00020A            ID             1A
00030A            FLD1          10A
00040A            FLD2          10A
00050A            FLD3           6S 2
     A
00060A          R RCD2                      RECID(1 'D')
00070A            ID             1A
00080A            FLD1           8S 2
00090A            FLD2          10A
00100A            FLD3           5B 0
     A
00110A          R RCD3                      RECID(1 'L')
00120A            ID             1A
00130A            FLD1          50A
     A
```

Example 3:

In this example, three record formats are defined in ICF file. The application program issues input operations using the file name, for instance, RPTFILE.

Assume that the records received on nine successive input operations are one header, then three details, then one header, then four details. The sending application must identify the headers by placing an H in field CODE and the details by placing a D in field CODE. For each input operation, the OS/400 program compares the value in position 1 in the buffer with the value specified on the RECID keyword. (Position 1 is the location of field CODE in the buffer.) If the value in a record is H, the OS/400 program selects record format name HEADER; if the value in a record is D, the OS/400 program selects record format name DETAIL.

Record format CATCH (the RECID default record format) is the record format name selected if a record is received that does not contain either H or D in the first position of the data portion of the buffer.

Following is the buffer for record format HEADER:

Response indicator 10 (1 byte)
CODE (1 byte)
TITLE (30 bytes)
ACTNBR (6 bytes)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R HEADER                    RECID(1 'H')
00020A                                      RCVTRNRND(10 'Host stopped sending')
00030A            CODE           1
00040A            TITLE         30
00050A            ACTNBR         6  0
00060A          R DETAIL                    RECID(1 'D')
00070A            CODE           1
00080A            ITMNBR         8  0
00090A            DESCRP        20
00100A          R CATCH
00110A            FIELD         37
     A
```

Example 4:

Three record formats need to be distinguished from one another; the first character in the value parameter is the same. Specifying the most specific (longest) value parameter first in the DDS enables the OS/400 program to distinguish the first record format from the others. The reason is that if the first 10 positions of the buffer contain ABCDEFGHIJ, and RCD3 were specified first, RCD3 would be identified even though RCD1 is desired. RCD1 and RCD2 could not be identified because the OS/400 program does not test after one successful match.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1                      RECID(1 'ABC')
     A            FLD1          10
     A          R RCD2                      RECID(1 'AB')
     A            FLD1          10
     A          R RCD3                      RECID(1 'A')
     A            FLD1          10
     A          R CATCH
     A            FIELD         10
     A
```

REF (Reference) keyword for ICF files

Use this file-level keyword to specify the name of a file from which field descriptions are to be retrieved. Use it when you want to duplicate descriptive information from several fields in a previously defined record format. This keyword allows you to code the file name once rather than on each field that refers to the file as is required by the REFFLD keyword. To refer to more than one file, use the REFFLD keyword. The REF keyword can be specified only once.

The format of the keyword is:

```
REF([library-name/]database-file-name [record-format-name])
```

If there is more than one record format in the reference file, specify a record format name as a parameter value for this keyword to tell the OS/400 program which one to use unless the record formats should be searched sequentially.

The database-file-name is a required parameter value for this keyword. The library-name and the record-format-name are optional.

If you do not specify the library-name, the current library list (*LIBL) at file creation time is used. If the record-format-name is not specified, each format is searched in order (as they are specified). The first occurrence of the field is used. For more information, see the topic "When to specify REF and REFFLD keywords for DDS files" in the DDS Reference: Concepts information.

You can specify a Distributed Data Management (DDM) file on this keyword. When using a DDM file, the database-file-name and the library-name are the DDM file name and library name on the source system. The record-format-name is the record format name in the remote file on the target system.

Note: IDDU files cannot be used as reference files.

Option indicators are not valid for this keyword.

Example:

The following examples show how to specify the REF keyword.

In the first example, FLD1 has the same attributes as the first (or only) FLD1 in the file, FILE1:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                      REF(FILE1)
     A          R RECORD
     A            FLD1      R
     A
```

In the second example, FLD1 has the same attributes as FLD1 in RECORD2 in FILE1 in LIB1:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                  REF(LIB1/FILE1 RECORD2)
     A          R RECORD
     A            FLD1      R
     A
```

REFFLD (Referenced Field) keyword for ICF files

Use this field-level keyword when referring to a field under one of these three conditions:

- The name of the referenced field is different from the name in positions 19 through 28.
- The name of the referenced field is the same as the name in positions 19 through 28, but the record format, file, or library of the referenced field is different from that specified with the REF keyword.
- The referenced field occurs in the same DDS source file as the referencing field.

The format of the keyword is:

```
REFFLD([record-format-name/]referenced-field-name [  {*SRC |
[library-name/]database-file-name}])
```

The referenced-field-name is required even if it is the same as the referencing field. Use the record-format-name when the referenced file contains more than one record format. Use *SRC (rather than the database-file-name) when the referenced-field-name is in the same DDS source file as the referencing field. *SRC is the default value when the database-file-name, the library-name, and the REF keyword are not specified.

Note: When you refer to a field in the same DDS source file, the field you are referring to must precede the field you are defining.

Specify the database-file-name (qualified by its library-name if necessary) when you want to search a particular database file.

If, in the same DDS source file, you specify REF at the file-level and REFFLD at the field level, the particular search sequence depends on both the REF and REFFLD keywords. For more information, see the topic "When to specify REF and REFFLD keywords for DDS files" in the DDS Reference: Concepts information.

The letter R must be specified in position 29. In some cases, if you specify a value for length, some keywords specified with the field in the database file are not included in the ICF file. For more information, see "Reference for ICF files (position 29)" on page 4.

You can specify a Distributed Data Management (DDM) file on this keyword.

When using a DDM file, the database-file-name and the library-name are the DDM file and library name on the source system. The referenced-field-name and the record-format-name are the field name and the record format name in the remote file on the target system.

Note: IDDU files cannot be used as reference files.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the REFFLD keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R FMAT1
00020A            ITEM           5
```

```
00030A            ITEM1     R               REFFLD(ITEM)
00040A            ITEM2     R               REFFLD(FMAT1/ITEM)
00050A            ITEM3     R               REFFLD(ITEM FILEX)
00060A            ITEM4     R               REFFLD(ITEM LIBY/FILEX)
00070A            ITEM5     R               REFFLD(FMAT1/ITEM LIBY/FILEX)
00080A            ITEM6     R               REFFLD(ITEM *SCR)
     A
```

Because the REF keyword is not specified, the default for lines 00030 and 00040 is to search the DDS source file in which they are specified. In line 00080, the parameter value *SRC explicitly specifies the source file.

RQSWRT (Request Write) keyword for ICF files

Use this file- or record-level keyword to request permission for your program to send data.

This keyword has no parameters.

The RQSWRT keyword must have an option indicator when it applies to a record for which any of the following keywords apply:

ALWWRT
CONFIRM
DETACH
ENDGRP
EVOKE
FMH
FMTNAME
FRCDTA
SUBDEV
VARBUFMGT
VARLEN

At run time, these keywords are ignored when the RQSWRT keyword is in effect. If a RQSWRT keyword with no option indicator applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time.

The RQSWRT keyword is ignored at run time when CANCEL, CNLINVITE, EOS, FAIL, RSPCONFIRM, or NEGRSP is in effect. These keywords must have option indicators when they apply to a record for which the RQSWRT keyword applies. If a CANCEL, CNLINVITE, EOS, FAIL, or NEGRSP keyword with no option indicator applies to a record for which RQSWRT applies, an error message is issued and the RQSWRT keyword is ignored at creation time.

You cannot specify RQSWRT with the TIMER keyword.

Option indicators are valid for this keyword. When you specify this keyword at the file level, you should specify an option indicator.

Example:

The following example shows how to specify the RQSWRT keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R CUSMST
00020A  14                                RQSWRT
     A
```

RSPCONFIRM (Respond Confirm) keyword for ICF files

Use this file- or record-level keyword to send a positive response to a received confirm request.

This keyword has no parameters.

The RSPCONFIRM keyword must have an option indicator when it applies to a record for which any of the following keywords apply:

ALWWRT	FMH	RQSWRT
CONFIRM	FMTNAME	SUBDEV
DETACH	FRCDTA	VARBUFMGT
ENDGRP	INVITE	VARLEN
EVOKE		

At run time, data fields and these keywords are ignored when the RSPCONFIRM keyword is in effect. If an unoptioned RSPCONFIRM keyword applies to a record for which any of these keywords apply, error messages are issued and these keywords are ignored at creation time. If an unoptioned RSPCONFIRM applies to a record with data fields, a severe error is issued and the file is not created.

The RSPCONFIRM keyword is ignored at run time when EOS, FAIL, NEGRSP, CANCEL, or CNLINVITE is in effect. These keywords must have option indicators when they apply to a record for which RSPCONFIRM applies. If an unoptioned EOS, FAIL, NEGRSP, CANCEL, or CNLINVITE applies to a record for which RSPCONFIRM applies, an error message is issued and the RSPCONFIRM keyword is ignored at creation time.

Option indicators are valid for this keyword. When you specify this keyword at the file level, you should specify an option indicator.

You cannot specify RSPCONFIRM with the TIMER keyword.

Example:

The following example shows how to specify the RSPCONFIRM keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD
     A
     A  20                                RSPCONFIRM
     A
```

If option indicator 20 is on, an output operation to RCD will send a positive response to the confirm request received from the remote program.

SECURITY (Security) keyword for ICF files

Use this file- and record-level keyword to include security information when your program starts a program on a remote system (see the EVOKE keyword). Any record format that has the SECURITY keyword specified for it or implied for it by being specified at the file level must have the EVOKE keyword specified on that record format or implied for that record format by being specified at the file level. If you do not specify the EVOKE keyword, a severe error occurs and the file is not created.

The format of the keyword is:

```
SECURITY(security-subfield subfield-definition[.3.])
```

The security-subfield parameter identifies the subfield being defined. This parameter is required. The value specified must be one of the following:

Value	Meaning
1	(Profile ID)
2	(Password)
3	(User ID)

The subfield-definition parameter must be one of the following. If you enter the password as literal (character string), the characters are interpreted by the CCSID of the ICF file; otherwise, characters are interpreted by the CCSID of the current job.

***USER**
Indicates that the user profile name of the user should be used as the value of the security subfield. For example, if *USER is specified for the password subfield, the user profile name is used as the password.

***NONE**
Indicates that a null security value should be used.

'character-string'
You can specify up to 128 single-byte characters for a password.

field-name
The specified field contains the security information.

The length of the field can range from 1 to 10 bytes, or it may be 512 bytes. The number of characters, as interpreted by the CCSID of the current job, cannot exceed 128. Values greater than 128 should only be used if multi-byte characters are specified for the password. The default length of the field is 10 bytes.

This parameter is not valid if you specify the SECURITY keyword at the file level.

&field-name
The specified field contains the security information.

The length of the field can range from 1 to 10 bytes, or it may be 512 bytes. The number of characters, as interpreted by the CCSID of the current job, cannot exceed 128. Values greater than 128 should only be used if multi-byte characters are specified for the password. The default length of the field is 10 bytes.

This parameter is not valid if you specify the SECURITY keyword at the file level.

You cannot specify SECURITY with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the SECURITY keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A  01                                SECURITY(2 'JONES' 3 'WHITE')
00020A
00030A
00040A
00050A
00060A          R RCD1
00070A  03                                SECURITY(2 'JONES' 3 *USER)
00080A                                    EVOKE(LIB2/PGM2)
00090A
00100A          R RCD2
00110A                                    EVOKE(LIB3/PGM3)
00120A
00130A            FIELD1         5A
00140A
00150A          R RCD3
00160A  60                                SECURITY(2 &CLVAR1 3 &CLVAR2);
00170A                                    EVOKE(LIB4/PGM4)
00180A            CLVAR1        10A
00190A            CLVAR2        10A
     A
```

SECURITY specified at the file-level applies to all formats and if selected (indicator 01 is on), the password of JONES and user ID of WHITE are sent to the remote system.

For RCD1, if indicator 03 is set on, the user profile name of the current user is used as the user ID and is sent with the password JONES as security information to the remote system.

For RCD2, no security information is sent to the remote system.

For RCD3, if indicator 60 is set on, the value contained in CLVAR1 is used as the password, the value in CLVAR2 is used as the user ID, and both are sent as security information to the remote system.

SUBDEV (Subdevice) keyword for ICF files

Use this file- or record-level keyword to allow your program to request a specific subdevice (for example, a printer) to which transmitted data should be directed.

The format of the keyword is:

```
SUBDEV(*DC1 | *DC2 | *DC3 | *DC4)
```

The SUBDEV keyword is ignored at run time when EOS, DETACH, RSPCONFIRM, or RQSWRT is in effect. These keywords must have option indicators when they apply to a record for which the SUBDEV keyword applies. If an EOS, DETACH, RSPCONFIRM, or RQSWRT keyword with no option indicator applies to a record for which SUBDEV applies, an error message is issued and the SUBDEV keyword is ignored at creation time.

You can specify only one parameter value for each SUBDEV keyword.

You can specify this keyword more than once in the file; however, you cannot specify the same parameter value at the file-level and again at the record-level. This is true even if you specify option indicators each time. For example, if you specify SUBDEV(*DC1) at the file-level, you cannot specify SUBDEV(*DC1) anywhere else in the file.

If you specify the SUBDEV keyword at both the file-level and the record level, and your program selects the one at the file-level, the record-level keyword(s) have no effect even if also selected.

You can specify the SUBDEV keyword a maximum of four times for each record format. If you specify the SUBDEV keyword more than once, you must specify option indicators each time, and you can specify each keyword value only once.

The OS/400 program sends a device selection character as follows. The meaning of the device selection character is set by the remote system or device.

Parameter Value	**Character Sent**
***DC1**	Hex 11
***DC2**	Hex 12
***DC3**	Hex 13
***DC4**	Hex 5D

You cannot specify SUBDEV with the TIMER keyword.

Option indicators are valid for this keyword.

Example:

The following example shows how to specify the SUBDEV keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A  01                                SUBDEV(*DC1)
00020A  02                                SUBDEV(*DC4)
     A          R RECORD
     A
```

If indicator 01 is on, the OS/400 program sends the component selection character hex 11 on an output operation (no matter how indicator 02 is set).

If indicator 02 is on and indicator 01 is off, the OS/400 program sends component selection character hex 5D.

SYNLVL (Synchronization Level) keyword for ICF files

This file- and record-level keyword indicates the level of synchronization your program requires. The SYNLVL keyword is valid only when the EVOKE keyword is in effect.

The format of the keyword is:

```
SYNLVL[(*NONE | *CONFIRM | *COMMIT)]
```

Specify *NONE when neither your program nor the remote program will use the CONFIRM keyword. Specify *CONFIRM if either your program or the remote program will use the CONFIRM keyword.

Specify *COMMIT to indicate that the local program may use the local system's commitment control support using the PRPCMT keyword or the commit and rollback operations. The CONFIRM keyword is allowed for the *COMMIT level conversations.

If the SYNLVL(*NONE) keyword is specified when the program is evoked, the CONFIRM keyword cannot be specified.

An EVOKE keyword must apply for any record for which this keyword applies.

You cannot specify SYNLVL with the TIMER keyword.

Option indicators are valid for this keyword and are required if this keyword is specified on more than one record format in the file.

Example:

The following example shows how to specify the SYNLVL keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R RCD
00020A                                    EVOKE(LIBRARY1/PROGRAM1)
00030A                                    SYNLVL(*CONFIRM)
     A
```

The EVOKE keyword will start PROGRAM1 in LIBRARY1 on the remote system. The SYNLVL keyword will set up a synchronization level that will confirm whether the data was received. When you request a confirmation (specify the CONFIRM keyword), the remote program must acknowledge whether it received the data by sending a positive or negative response.

TEXT (Text) keyword for ICF files

Use this record- or field-level keyword to supply a text description (or comment) for the record format or field that is used for program documentation.

The format of the keyword is:

```
TEXT('description')
```

The text must be enclosed in apostrophes. If the length of the text is greater than 50 positions, only the first 50 characters are used by the high-level language compiler.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the TEXT keyword at the record and field levels.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R CUSMST                    TEXT('Customer Master Record')
00020A            FLD1           3  0       TEXT('ORDER NUMBER FIELD')
     A
```

TIMER (Timer) keyword for ICF files

Use this record-level keyword to specify an interval of time for your program to wait before performing some specified function. To set the timer, your program issues an output operation with the TIMER keyword in effect.

The format of the keyword is:

```
TIMER(HHMMSS | &field-name);
```

HHMMSS
The time interval is a 6-digit value where HH is the number of hours (00 through 99), MM is the number of minutes (00 through 59), and SS is the number of seconds (00 through 59).

&field-name
The time interval parameter is the name of a field that contains the timer value in the form HHMMSS described above. The specified field name must exist in the record format, and the field must be a zoned field with length 6, data type S, usage P, and zero decimal positions.

The following keywords cannot be specified with TIMER:

ALWWRT	ENDGRP	RCVCONFIRM	RCVTRNRND
CANCEL	EVOKE	RCVCANCEL	RECID
CNLINVITE	FAIL	RCVCTLDTA	RQSWRT
CONFIRM	FMH	RCVDETACH	SECURITY
CTLDTA	FMTNAME	RCVENDGRP	SUBDEV
DETACH	FRCDTA	RCVFAIL	SYNLVL
DFREVOKE	INVITE	RCVFMH	VARBUFMGT
EOS	NEGRSP	RCVNEGRSP	VARLEN

TIMER overrides the WAITRCD parameter on the Create ICF File (CRTICFF), Change ICF File (CHGICFF), and Override ICF File (OVRICFF) commands. The WAITRCD parameter value is ignored during the interval that the timer function is in effect.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the TIMER keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1                      TIMER(002512)
     A          R RCD2                      TIMER(&FIELD1);
     A            FIELD1         6S  P
     A
     A
```

On an output operation to RCD1, the timer will be set to 0 hours, 25 minutes, and 12 seconds. On an output operation to RCD2, the timer will be set to the value that has been set in FIELD1.

TNSSYNLVL (Transaction Synchronization Level) keyword for ICF files

Use this file- or record-level keyword to specify the transaction synchronization level (specified on the SYNLVL keyword) that is performed while issuing a write operation when a DETACH or ALWWRT keyword is specified.

This keyword has no parameters.

The DETACH or ALWWRT keywords must be specified at either the file-level or on the same record as the TNSSYNLVL keyword.

The TIMER keyword is not allowed with the TNSSYNLVL keyword.

Option indicators are not valid for this keyword.

Examples:

The following examples show how to specify the TNSSYNLVL keyword.

The following example shows a write operation is issues for RCD2. The transaction between your program and the remote program will not be ended until the remote program confirms that the detach was received.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1
     A                                      EVOKE(LIBRARY1/PROGRAM1)
     A                                      SYNLVL(*CONFIRM)
     A          R RCD2
     A                                      DETACH
     A                                      TNSSYNLVL
     A
```

The following example shows a write operation is issues for RCD2. The conversation between your program and the remote program is put into a defer receive state. The conversation will be in receive state when a CONFIRM or COMMIT operation is completed.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1
     A                                      EVOKE(LIBRARY1/PROGRAM1)
     A                                      SYNLVL(*CONFIRM)
     A          R RCD2
     A                                      ALWWRT
     A                                      TNSSYNLVL
     A
```

VARBUFMGT (Variable Buffer Management) keyword for ICF files

Use this record-level keyword to send or receive multiple or partial records using one record format per output operation. On a send operation, you must specify the length of data to be sent using the VARLEN keyword. Otherwise, the length of the record format is used. On a receive operation, the length of data received is the length of the record format.

This keyword has no parameters.

VARBUFMGT is ignored at run time when a CANCEL, EOS, FAIL, NEGRSP, RSPCONFIRM, or RQSWRT keyword is in effect. These keywords must have option indicators when they apply to a record that has the VARBUFMGT keyword specified. If a CANCEL, EOS, FAIL, NEGRSP, or RQSWRT keyword with no option indicator applies to a record for which VARBUFMGT applies, an error message results and the VARBUFMGT keyword is ignored at create time.

Specify at least one data field (usage B or blank) for the user data in the record format.

You cannot specify the VARBUFMGT keyword:

- When the TIMER keyword is used
- On the same record format as the RECID keyword
- On the RECID or INVITE default record formats

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the VARBUFMGT keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A          R MULTFMT1
00020A                                      VARBUFMGT
00030A            DATAFLD       32A
00040A          R MULTFMT2
00050A                                      VARLEN(&LENFLD);
00060A                                      VARBUFMGT
00070A            DATAFLD       32A
00080A            LENFLD         5S P
```

Suppose `42THIS RECORD WILL NOT FIT INTO ONE BUFFER` was the data to be sent or received. The VARBUFMGT keyword on the first record format sends or receives the first 32 bytes of data. The second record format sends 10 bytes of data. The data length of 10 is set in LENFLD.

VARLEN (Variable-Length User Data) keyword for ICF files

Use this record-level keyword to indicate that the length of the record sent across the line is variable. The length is specified at run time in the field parameter.

The format of the keyword is:

```
VARLEN(&field-name);
```

The &field-name parameter is required and specifies the name of the field that contains the length of the user data to be sent. The field name must exist in the record format and the field must be defined as a zoned field of length 5, data type S, usage P, and zero decimal positions.

The length value set in the parameter field is the length of the user data and does not include indicators. The length value is specified in decimal and is checked at run time. The value should not be greater than the length of the DDS record format. The maximum value depends on the communication type you are using.

VARLEN is valid only on output operations.

VARLEN is ignored at run time when an CANCEL, EOS, FAIL, NEGRSP, RSPCONFIRM, or RQSWRT keyword is in effect. These keywords must have option indicators when they apply to a record that has the VARLEN keyword specified. If a CANCEL, EOS, FAIL, NEGRSP, or RQSWRT keyword with no option indicator applies to a record for which VARLEN applies, an error message is issued and the VARLEN keyword is ignored at creation time.

At least one data field (usage B or blank) for the user data must be specified in the record format.

You cannot specify VARLEN with the TIMER keyword.

Option indicators are not valid for this keyword.

Example:

The following example shows how to specify the VARLEN keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD                       VARLEN(&LENFLD);
     A            DATAFLD    32760A
     A            LENFLD         5S  P
     A
```

On an output operation to RCD, the length of the data in DATAFLD that is sent across the line will be the length set in LENFLD.

Chapter 3. DBCS considerations for ICF files

This section describes the bracketed-DBCS considerations for positional entries for ICF files, along with general considerations.

See the DDS Reference: Concepts information for additional general information relating to the use of the double-byte character set (DBCS) with DDS.

The functions described in this appendix are supported on both DBCS and non-DBCS systems.

Positional entry considerations for ICF files that use DBCS

The following topics describe DBCS considerations for describing the length and data type positions for ICF files. Those positions not mentioned have no special considerations for DBCS.

Length (positions 30 through 34)

The length of a field containing bracketed-DBCS data can range from 4 through 32 767 bytes.

Consider the following when determining the length of a DBCS field:

- Each DBCS character is 2 bytes long.
- Include both shift-control characters in the length of the field. Together, these characters are 2 bytes long.

  For example, a field that contains up to 3 DBCS characters, 1 shift-in character, and 1 shift-out character, has 8 bytes of data:

  ```
  (3 characters x 2 bytes) + (shift-out + shift-in) = 8
  ```

Data type (position 35)

The data type O (DBCS capable) makes a field DBCS. Both bracketed-DBCS and alphanumeric data can be used in a DBCS-capable field. Use shift-control characters to distinguish DBCS data from alphanumeric data.

Additional considerations for describing ICF files that contain DBCS data

Consider the following when describing an ICF file that contains DBCS data:

- Use ICF files only to send data from the iSeries server to another system or to a remote work station.
- Send shift-control characters with DBCS data. The system is not aware of DBCS data in an ICF file, and treats DBCS data the same as alphanumeric data.
- When using the reference function in an ICF file, if you refer to a field in a database file that has data type J, O, or E, DDS will assign data type O for the field in the ICF file.
- DBCS-graphic fields cannot be referenced from a database file because fields with data type G are not allowed in an ICF file. If a field with a data type of G is referenced from a database file, an error message is issued.

Appendix. 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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.

Programming Interface Information

This DDS Reference: ICF Files information documents intended Programming Interfaces that allow the customer to write programs to obtain the services of OS/400.

Trademarks

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

IBM
iSeries
Operating System/400
OS/2
OS/400

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

U

V

Printed in USA