



# Logic Text Types

## SAFR Workbench 4.15.000





# Logic Text Types

## SAFR Workbench 4.15.000

**Note**

Before using this information and the product it supports, read the information in “Appendix: Notices” on page 49.

**Fourth Edition**

This edition applies to version 4, release 15, modification 000 of SAFR Workbench (part of SAFR product number 6949-17P) and to all subsequent releases and modifications until otherwise indicated in new editions.

---

## Contents

### Chapter 1. Logic text 1: Extract Record

<b>Filter</b> . . . . .	<b>1</b>
Examples: SELECTIF statements . . . . .	3
Examples: SKIPIF statements . . . . .	5
Examples: IF with SELECT . . . . .	8
Examples: IF with SKIP . . . . .	12
Examples: WRITE statements . . . . .	15

### Chapter 2. Logic text 2: Extract Column

<b>Assignment</b> . . . . .	<b>17</b>
Examples: COLUMN and COL.nnn statements ..	19
Examples: IF with COLUMN and COL.nnn statements . . . . .	19
Examples: WRITE statements . . . . .	22

### Chapter 3. Logic text 3: Format Column

<b>Calculations</b> . . . . .	<b>23</b>
Examples: COLUMN statements . . . . .	24
Examples: IF with COLUMN statements. . . . .	25

### Chapter 4. Logic text 4: Format Record

<b>Filter</b> . . . . .	<b>27</b>
Examples: SELECTIF statements . . . . .	28
Examples: SKIPIF statements . . . . .	29
Examples: IF with SELECT . . . . .	30
Examples: IF with SKIP . . . . .	32

### Chapter 5. Rules for all logic text . . . . .

### Chapter 6. Cross reference of topics

<b>and PDF files</b> . . . . .	<b>37</b>
--------------------------------	-----------

### Appendix: Notices. . . . .

Trademarks . . . . .	51
----------------------	----

### Index . . . . .



---

# Chapter 1. Logic text 1: Extract Record Filter

## 01 Summary of this topic

The sections in this topic are as follows:





- "10 Introduction"
- "20 How do I create logic text for Extract Record Filter?"
- "30 What do I use this for?" on page 2
- "40 Statements for selecting input files" on page 2
- "50 Statements for writing input records" on page 2
- "90 Examples" on page 3

## 10 Introduction

See topic "**Logic text 1: Extract Record Filter overview**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".

## 20 How do I create logic text for Extract Record Filter?

This logic text is part of a view and is associated with a view source file. To create this logic text in an existing view, do the following:

1. Ensure you are on the "**Edit View**" screen for the relevant view. For help with displaying this screen, see topic "**Modifying Views**" in the General User Guide.
2. The view must already have at least one view source defined. If there is no view source defined, see topic "**Edit View (View Editor tab) screen help**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".
3. Ensure you are on the "**View Editor**" tab. If the "**View Properties**" tab is displayed, click the **Show Grid / Properties** button.  or press F9 or select **Edit -> Show Grid/Properties**.
4. In the column immediately below "View Editor", if there is a plus + sign to the left of "**View Source**", double click the plus sign to expand the list of view sources.
5. Click in the cell with a relevant view source name under "**View Sources**".
6. Double click on the cell to the right of "**Record Filter**" and click .
7. Type your logic text in the window "**Create New Extract Record Filter**" or "**Edit Extract Record Filter**".
8. Click  to check if the logic text is valid.
9. Fix any errors shown in the "**Logic Text Validation Errors**" window.
10. When the new logic text is complete and valid,
  - **EITHER** click  (the save icon),
  - **OR** select **File, Save**,
  - **OR** press **Ctrl+S**.

11. You may wish to close some of the open windows, such as "**Logic Text Helper**" and "**Create New Extract Record Filter**" or "**Edit Extract Record Filter**".

## 30 What do I use this for?

Extract Record Filter logic text is used for the following purposes:

- **Select input files from an input logical file** for processing in the extract phase. For example, you may select input records in a certain date range. This is the primary purpose of this logic text. If this logic text is empty, then all input records are selected automatically. More details are below.
- **Write input records to other logical files.** You may write all input records, or you may write only the selected input records. Such copies of input records may be useful as a trace for debugging purposes, or as an audit trail. This purpose is optional and less common than the previous purpose. More details are below.

## 40 Statements for selecting input files

The overall idea is that your logic text describes how to select or skip, but not both.

Choice for logic text	Notes
One SELECTIF statement	A SELECTIF statement gives a condition for selecting input records. Skip any records that do not meet the condition.
One SKIPIF statement	A SKIPIF statement gives a condition for skipping input records. Select any records that do not meet the condition.
One IF statement using only SELECT statements	One IF statement can contain a SELECT statement for the THEN or ELSE cases. Alternatively, the THEN or ELSE cases might contain other IF statements that also has SELECT statements or IF statements, and the whole thing still counts as one IF statement. Skip any records that do not qualify for a SELECT somewhere in the IF statement. No actual SKIP statements are allowed.
One IF statement using only SKIP statements	One IF statement can contain a SKIP statement for the THEN or ELSE cases. Alternatively, the THEN or ELSE cases might contain other IF statements that also has SKIP statements or IF statements, and the whole thing still counts as one IF statement. Select any records that do not qualify for a SKIP somewhere in the IF statement. No actual SELECT statements are allowed.

## 50 Statements for writing input records

The only valid statement is a WRITE statement. The **position** of the WRITE statement decides which record are written, as follows:

- WRITE statements **before** any of the select or skip logic text result in write of **all** input records.
- WRITE statements **after** any of the select or skip logic text result in write of only the **selected** input records.

For syntax of WRITE statements, see topic "**Syntax: WRITE statements in Extract Record Filter**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".



## 90 Examples

See the topics below.

---

### Examples: SELECTIF statements

#### Examples: SELECTIF in Extract Record Filter

Example logic text	Meaning
<pre>SELECTIF(CURRENT({field1}) &lt;&gt;           PRIOR({field1}))</pre>	Select only records with unique values for field1. This assumes the input file is sorted into field1 order.
<pre>SELECTIF({field2} &gt; 1000)</pre>	Select for output only those records with field2 greater than 1000. Skip all other records. The code at left is a shorthand for: <pre>IF {field2} &gt; 1000   THEN SELECT ENDIF</pre>
<pre>SELECTIF({field3} = "ABC")</pre>	Select for output only those records with field3 equal to "ABC". Skip all other records.
<pre>SELECTIF(NOT {field3} = "ABC")</pre>	Select those output records with field3 not equal to "ABC". Skip all other records. This example gives the same result as: <pre>SKIPIF({field3} = "ABC")</pre>
<pre>SELECTIF({field3} = "A" OR           {field3} = "D")</pre>	Select for output only those records with field3 equal to "A" or "D". Skip all other records.
<pre>SELECTIF({field3} = "A" AND           {field4} &gt; 10)</pre>	Select for output only those records with field3 equal to "A" and field4 greater than 10. Skip all other records.
<pre>SELECTIF({field4} + {field5}           &gt; {field6})</pre>	Select for output only those records with field4 plus field5 is greater than field6. Skip all other records.
<pre>SELECTIF(NOT {field7} = ALL("-"))</pre>	Select for output those records with field7 is not equal to all dashes. Skip all other records. This example gives the same result as: <pre>SKIPIF({field7} = ALL("-"))</pre>
<pre>SELECTIF(NOT {field7} =           REPEAT("-", 13))</pre>	Select for output those records with field7 is not equal to 13 dashes. Skip all other records. This example gives the same result as: <pre>SKIPIF({field7} = REPEAT("-", 13))</pre>

**Example logic text**

```
SELECTIF(NOT {field7} = "\xFF")
```

**Meaning**

Select for output those records with field7 is not equal to hexadecimal FF. Skip all other records. This example gives the same result as:

```
SKIPIF({field7} = "\xFF")
```

```
SELECTIF(ISFOUND({Lookup1}))
```

Select all input records where the lookup path Lookup1 successfully finds a target record, and skip all other records.

```
SELECTIF(ISFOUND({Lookup1,field7}))
```

Select all input records where the lookup path Lookup1 successfully finds a target record using effective date of field7, and skip all other records.

```
SELECTIF(ISFOUND({Lookup1;$SYM="A"}))
```

Select all input records where the lookup path Lookup1 successfully finds a target record using symbol SYM set to "A", and skip all other records.

```
SELECTIF(ISFOUND({Lookup1,  
field7;$SYM1=3,$SYM2=0}))
```

Select all input records where the lookup path Lookup1 successfully finds a target record using effective date of field7 and symbol SYM1 set to 3 and symbol SYM2 set to zero. Skip all other records.

```
SELECTIF(ISNOTNULL({field1}))
```

Select all input records where field1 does not contain null values, and skip all other records.

```
SELECTIF(ISNUMERIC({field2}))
```

Select all input records where field2 is numeric, and skip all other records.

```
SELECTIF(ISNOTSPACES({field3}))
```

Select all input records where field3 is not spaces, and skip all other records.

```
SELECTIF(DAYSBETWEEN({field1},{field2})  
> 7)
```

Select only records where there are more than 7 days between field1 and field2, and skip all other records.

```
SELECTIF({field1} BEGINS_WITH "BBB")
```

Select input records where field1 begins with characters "BBB", and skip all other records.

```
SELECTIF({field2} CONTAINS "CCC")
```

Select input records where field2 contains characters "CCC", and skip all other records.

```
SELECTIF({field3} ENDS_WITH "EEE")
```

Select input records where field3 ends with characters "EEE", and skip all other records.

```
SELECTIF({field4} MATCHES "...")
```

Select input records where field4 matches characters "...", and skip all other records.

**Example logic text**

```
SELECTIF({field5} LIKE "MA...")
```

**Meaning**

Select input records where field5 is exactly 5 characters starting with "MA", and skip all other records.

```
SELECTIF({field6} LIKE "..VA..")
```

Select input records where field6 is exactly 6 characters with characters 3 and 4 as "VA", and skip all other records.

```
SELECTIF({field7} LIKE ".....NA")
```

Select input records where field7 is exactly 6 characters ending in "NA", and skip all other records.

```
SELECTIF({field8} LIKE "^BBB*")
```

Select input records where field8 begins with characters "BBB", and skip all other records. This example has the same effect as:

```
SELECTIF({field8} BEGINS_WITH "BBB")
```

It is better to use BEGINS\_WITH because the logic text executes faster.

```
SELECTIF({field9} LIKE "*CCC*")
```

Select input records where field9 contains characters "CCC", and skip all other records. This example has the same effect as:

```
SELECTIF({field9} CONTAINS "CCC")
```

It is better to use CONTAINS because the logic text executes faster.

```
SELECTIF({field1} LIKE "*EEE$")
```

Select input records where field1 ends with characters "EEE", and skip all other records. This example has the same effect as:

```
SELECTIF({field1} ENDS_WITH "EEE")
```

It is better to use ENDS\_WITH because the logic text executes faster.

```
SELECTIF({field2} LIKE "^B*C*E$")
```

Select input records where field2 begins with "B", contains "C" and ends with "E", and skip all other records.

---

## Examples: SKIPIF statements

**Examples: SKIPIF in Extract Record Filter****Example logic text**

```
SKIPIF(CURRENT({field4}) =  
      PRIOR({field4}))
```

**Meaning**

Skip records where the new field4 value is the same as the previous field4. This assumes the input file is sorted into field4 order. This selects only the input records where field4 is a new value (compared to the previous record).

### Example logic text

SKIPIF({field1} > 1000)

SKIPIF({field2} = "ABC")

SKIPIF(NOT {field2} = "ABC")

SKIPIF({field3} = "A" OR {field3} = "D")

SKIPIF({field4} = "A" AND {field5} > 10)

SKIPIF({field6} \* 2 > {field8} + 5)

SKIPIF({field6} = ALL("-"))

SKIPIF({field6} = REPEAT("-", 13))

SKIPIF({field6} = "\xFF")

SKIPIF(ISNOTFOUND({Lookup2}))

SKIPIF(ISNOTFOUND({Lookup2, field7}))

### Meaning

Skip for output those records with field1 greater than 1000. Select all other records. The code at left is a shorthand for:

```
IF ({field1} > 1000)
  THEN SKIP
ENDIF
```

Skip for output those records with field2 equal to "ABC". Select all other records.

Skip those output records with field2 not equal to "ABC". Select all other records. This example gives the same result as:

SELECTIF({field2} = "ABC")

Skip for output those records with field3 equal to "A" or "D". Select all other records.

Skip for output those records with field4 equal to "A" and field5 greater than 10. Select all other records.

Skip for output those records with field6 times 2 is greater than field8 plus 5. Select all other records.

Skip for output those records with field6 is equal to all dashes. Select all other records.

Skip for output those records with field6 is equal to 13 dashes. Select all other records.

Skip for output those records with field6 is equal to hexadecimal FF. Select all other records.

Skip all input records where the lookup path Lookup2 does not successfully find a target record, and select all other records.

Skip all input records where the lookup path Lookup2 does not successfully find a target record using effective date of field7, and select all other records.

**Example logic text**

SKIPIF(ISNOTFOUND({Lookup2;\$SYM="A"}))

**Meaning**

Skip all input records where the lookup path Lookup2 using symbol SYM set to "A" does not successfully finds a target record, and select all other records.

SKIPIF(ISNOTFOUND({Lookup2,  
field8;\$SYM1=3,\$SYM2=0}))

Skip all input records where the lookup path Lookup2 using effective date of field8 and symbol SYM1 set to 3 and symbol SYM2 set to zero does not successfully finds a target record. Select all other records.

SKIPIF(ISNULL({field1}))

Skip all input records where field1 contains null values, and select all other records.

SKIPIF(ISNOTNUMERIC({field2}))

Skip all input records where field2 is not numeric, and select all other records.

SKIPIF(ISSPACES({field3}))

Skip all input records where field3 is spaces, and select all other records.

SKIPIF(DAYSBETWEEN({field4},{field5})  
> 7)

Skip only records where there are more than 7 days between field4 and field5, and select all other records.

SKIPIF({field1} BEGINS\_WITH "BBB")

Skip input records where field1 begins with characters "BBB", and select all other records.

SKIPIF({field2} CONTAINS "CCC")

Skip input records where field2 contains characters "CCC", and select all other records.

SKIPIF({field3} ENDS\_WITH "EEE")

Skip input records where field3 ends with characters "EEE", and select all other records.

SKIPIF({field4} MATCHES "...")

Skip input records where field4 matches characters "...", and select all other records.

SKIPIF({field5} LIKE "MA...")

Skip input records where field5 is exactly 5 characters starting with "MA", and select all other records.

SKIPIF({field6} LIKE "..VA..")

Skip input records where field6 is exactly 6 characters with characters 3 and 4 as "VA", and select all other records.

SKIPIF({field7} LIKE ".....NA")

Skip input records where field7 is exactly 6 characters ending in "NA", and select all other records.

**Example logic text**

```
SKIPIF({field8} LIKE "^BBB*")
```

**Meaning**

Skip input records where field8 begins with characters "BBB", and select all other records. This example has the same effect as:

```
SKIPIF({field8} BEGINS_WITH "BBB")
```

It is better to use BEGINS\_WITH because the logic text executes faster.

```
SKIPIF({field9} LIKE "*CCC*")
```

Skip input records where field9 contains characters "CCC", and select all other records. This example has the same effect as:

```
SKIPIF({field9} CONTAINS "CCC")
```

It is better to use CONTAINS because the logic text executes faster.

```
SKIPIF({field1} LIKE "*EEE$")
```

Skip input records where field1 ends with characters "EEE", and select all other records. This example has the same effect as:

```
SKIPIF({field1} ENDS_WITH "EEE")
```

It is better to use ENDS\_WITH because the logic text executes faster.

```
SKIPIF({field2} LIKE "^B*C*E$")
```

Skip input records where field2 begins with "B", contains "C" and ends with "E", and select all other records.

---

## Examples: IF with SELECT

### Examples: IF with SELECT in Extract Record Filter

**Example logic text**

```
IF (CURRENT({field1}) <>
    PRIOR({field1}))
  THEN SELECT
ENDIF
```

**Meaning**

Select only records with unique values for field1. This assumes the input file is sorted into field1 order. This example can also be written:

```
SELECTIF(CURRENT({field1}) <>
    PRIOR({field1}))
```

```
IF ({field3} > 1000)
  THEN SELECT
ENDIF
```

Select for output only those records with field3 greater than 1000. Skip all other records. The code at left can also be written as:

```
SELECTIF({field3} > 1000)
```

```
IF ({field2} = "ABC")
  THEN SELECT
ENDIF
```

Select for output only those records with field2 equal to "ABC". Skip all other records.

### Example logic text

```
IF NOT ({field2} = "ABC")
  THEN SELECT
ENDIF
```

```
IF ({field2} = "ABC") OR
  ({field2} = "DEF")
  THEN SELECT
ENDIF
```

```
IF ({field2} = "ABC") AND
  ({field3} > 1000)
  THEN SELECT
ENDIF
```

```
IF ({field3} + {field4} >
  {field5} * 100)
  THEN SELECT
ENDIF
```

```
IF NOT ({field6} = ALL("-"))
  THEN SELECT
ENDIF
```

```
IF ({field6} = REPEAT("-", 13))
  THEN SELECT
ENDIF
```

```
IF ({field6} = "\xFF")
  THEN SELECT
ENDIF
```

```
IF ({field2} = "ABC") AND
  ({field3} > 10)
  THEN SELECT
ELSE IF ({field2} = "DEF")
  THEN SELECT
  ELSE IF ({field2} = "GHI")
    THEN SELECT
  ENDIF
ENDIF
ENDIF
```

### Meaning

Select those output records with field2 not equal to "ABC". Skip all other records. The code at left gives the same result as:

```
SKIPIF({field2} = "ABC")
```

Select for output only those records with field2 equal to "ABC" or "DEF". Skip all other records.

Select for output only those records with field2 equal to "ABC" and field3 greater than 1000. Skip all other records.

Select for output only those records with field3 plus field4 is greater than field5 times 100. Skip all other records.

Select for output those records with field6 is not equal to all dashes. Skip all other records. This example gives the same result as:

```
SKIPIF({field6} = ALL("-"))
```

Select for output those records with field6 is equal to 13 dashes. Skip all other records.

Select for output those records with field6 is equal to hexadecimal FF. Skip all other records.

Select for output those records with field2 equal to "ABC" and field3 greater than 10.

In addition, select for output those records with field2 equal to "DEF".

In addition, select for output those records with field2 equal to "GHI".

Skip all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF ({field2} = "ABC" AND
  {field3} > 10) OR
  ({field2} = "DEF") OR
  ({field2} = "GHI")
  THEN SELECT
ENDIF
```

### Example logic text

```
IF ({field2} = "ABC") AND
  ({field3} > 10)
THEN IF ({field4} + {field5}
  > {field6})
  THEN SELECT
  ELSE IF ({field7} = 0)
    THEN SELECT
    ENDIF
  ENDIF
ENDIF
```

### Meaning

Consider those records with field2 equal to "ABC" and field3 greater than 10.

Of the considered records, select for output those records with field4 plus field5 is greater than field6.

Of the considered records not yet selected, select also for output those records with field7 equal to zero.

Skip all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF ({field2} = "ABC" AND
    {field3} > 10)
    AND
    (({field4} + {field5}
      > {field6}) OR
    ({field7} = 0))
  THEN SELECT
ENDIF
```

```
IF ISFOUND({Lookup3})
  THEN SELECT
ENDIF
```

Select all input records where the lookup path Lookup3 successfully finds a target record, and skip all other records. This example is the same as:

```
SELECTIF(ISFOUND({Lookup3}))
```

```
IF ISFOUND({Lookup3,field7})
  THEN SELECT
ENDIF
```

Select all input records where the lookup path Lookup3 successfully finds a target record using effective date of field7, and skip all other records. This example is the same as:

```
SELECTIF(ISFOUND({Lookup3,field7}))
```

```
IF ISFOUND({Lookup3;$SYM="A"})
  THEN SELECT
ENDIF
```

Select all input records where the lookup path Lookup3 successfully finds a target record using symbol SYM set to "A", and skip all other records. This example is the same as:

```
SELECTIF(ISFOUND({Lookup3;$SYM="A"}))
```

```
IF ISFOUND({Lookup3,
  field7;$SYM1=3,$SYM2=0})
  THEN SELECT
ENDIF
```

Select all input records where the lookup path Lookup3 successfully finds a target record using effective date of field7 and symbol SYM1 set to 3 and symbol SYM2 set to zero. Skip all other records. This example is the same as:

```
SELECTIF(ISFOUND({Lookup3,
  field7;$SYM1=3,$SYM2=0}))
```



**Example logic text**

```
IF DAYSBEWEEN({field1},{field2})
    > 7
    THEN SELECT
ENDIF
```

```
IF ({field1} BEGINS_WITH "BBB")
    THEN SELECT
ENDIF
```

```
IF ({field2} CONTAINS "CCC")
    THEN SELECT
ENDIF
```

```
IF ({field3} ENDS_WITH "EEE")
    THEN SELECT
ENDIF
```

```
IF ({field4} MATCHES "...")
    THEN SELECT
ENDIF
```

```
IF ({field1} LIKE "^BBB*")
    THEN SELECT
ENDIF
```

```
IF ({field1} LIKE "*CCC*")
    THEN SELECT
ENDIF
```

**Meaning**

Select only records where there are more than 7 days between field1 and field2, and skip all other records. This example can also be written:

```
SELECTIF(DAYSBEWEEN({field1},{field2})
    > 7)
```

Select input records where field1 begins with characters "BBB", and skip all other records. This example can be written:

```
SELECTIF({field1} BEGINS_WITH "BBB")
```

It is better to use BEGINS\_WITH because the logic text executes faster.

Select input records where field2 contains characters "CCC", and skip all other records. This example can be written:

```
SELECTIF({field2} CONTAINS "CCC")
```

It is better to use CONTAINS because the logic text executes faster.

Select input records where field3 ends with characters "EEE", and skip all other records. This example can be written:

```
SELECTIF({field3} ENDS_WITH "EEE")
```

It is better to use ENDS\_WITH because the logic text executes faster.

Select input records where field4 matches characters "...", and skip all other records. This example can be written:

```
SELECTIF({field4} MATCHES "...")
```

Select input records where field1 begins with characters "BBB", and skip all other records. This example has the same effect as:

```
SELECTIF({field1} BEGINS_WITH "BBB")
```

It is better to use BEGINS\_WITH because the logic text executes faster.

Select input records where field1 contains characters "CCC", and skip all other records. This example has the same effect as:

```
SELECTIF({field1} CONTAINS "CCC")
```

It is better to use CONTAINS because the logic text executes faster.

**Example logic text**

```
IF ({field1} LIKE "*EEE$")
  THEN SELECT
ENDIF
```

**Meaning**

Select input records where field1 ends with characters "EEE", and skip all other records. This example has the same effect as:

```
SELECTIF({field1} ENDS_WITH "EEE")
```

It is better to use ENDS\_WITH because the logic text executes faster.

```
IF ({field1} LIKE "^B*C*E$")
  THEN SELECT
ENDIF
```

Select input records where field1 begins with "B", contains "C" and ends with "E", and skip all other records.

---

## Examples: IF with SKIP

### Examples: IF with SKIP in Extract Record Filter

**Example logic text**

```
IF (CURRENT({field1}) =
  PRIOR({field1}))
  THEN SKIP
ENDIF
```

**Meaning**

Skip records where field1 is the same as the previous record. This assumes the input file is sorted into field1 order. This example can also be written:

```
SKIPIF(CURRENT({field1}) =
  PRIOR({field1}))
```

```
IF ({field3} > 1000)
  THEN SKIP
ENDIF
```

Skip for output those records with field3 greater than 1000. Select all other records. The code at left can also be written as:

```
SKIPIF({field3} > 1000)
```

```
IF ({field2} = "ABC")
  THEN SKIP
ENDIF
```

Skip for output those records with field2 equal to "ABC". Select all other records.

```
IF NOT ({field2} = "ABC")
  THEN SKIP
ENDIF
```

Skip those output records with field2 not equal to "ABC". Select all other records. The code at left gives the same result as:

```
SELECTIF({field2} = "ABC")
```

```
IF ({field2} = "A") OR
  ({field2} = "D")
  THEN SKIP
ENDIF
```

Skip for output those records with field2 equal to "A" or "D". Select all other records.

```
IF ({field2} = "A") AND
  ({field3} > 10)
  THEN SKIP
ENDIF
```

Skip for output those records with field2 equal to "A" and field3 greater than 10. Select all other records.

```
IF ({field3} + {field4}
  > {field5})
  THEN SKIP
ENDIF
```

Skip for output those records with field3 plus field4 is greater than field5. Select all other records.

### Example logic text

```
IF ({field6} = ALL("-"))
  THEN SKIP
ENDIF
```

```
IF ({field6} = REPEAT("-", 13))
  THEN SKIP
ENDIF
```

```
IF ({field6} = "\xFF")
  THEN SKIP
ENDIF
```

```
IF ({field2} = "A") AND
  ({field3} > 10)
  THEN SKIP
ELSE IF ({field2} = "D")
  THEN SKIP
  ELSE IF ({field2} = "G")
    THEN SKIP
  ENDIF
ENDIF
ENDIF
```

```
IF ISNOTFOUND({Lookup4})
  THEN SKIP
ENDIF
```

```
IF ISNOTFOUND({Lookup4,field7})
  THEN SKIP
ENDIF
```

```
IF ISNOTFOUND({Lookup4;$SYM="A"})
  THEN SKIP
ENDIF
```

### Meaning

Skip for output those records with field6 is equal to all dashes. Select all other records. This example gives the same result as:

```
SKIPIF({field6} = ALL("-"))
```

Skip for output those records with field6 is equal to 13 dashes. Select all other records. This example gives the same result as:

```
SKIPIF({field6} = REPEAT("-", 13))
```

Skip for output those records with field6 is equal to hexadecimal FF. Select all other records. This example gives the same result as:

```
SKIPIF({field6} = "\xFF")
```

Skip for output those records with field2 equal to "A" and field3 greater than 10.

In addition, skip for output those records with field2 equal to "D".

In addition, skip for output those records with field2 equal to "G".

Select all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF ({field2} = "A" AND
    {field3} > 10) OR
  ({field2} = "D") OR
  ({field2} = "G")
  THEN SKIP
ENDIF
```

Skip all input records where the lookup path Lookup4 does not successfully find a target record, and select all other records. This example is the same as:

```
SKIPIF(ISNOTFOUND({Lookup4}))
```

Skip all input records where the lookup path Lookup4 does not successfully find a target record using effective date of field7, and select all other records. This example is the same as:

```
SKIPIF(ISNOTFOUND({Lookup4,field7}))
```

Skip all input records where the lookup path Lookup4 does not successfully find a target record using symbol SYM set to "A", and select all other records. This example is the same as:

```
SKIPIF(ISNOTFOUND({Lookup4;$SYM="A"}))
```

**Example logic text**

```
IF ISNOTFOUND
  ({Lookup4,field7;$SYM1=3,$SYM2=0})
  THEN SKIP
ENDIF
```

**Meaning**

Skip all input records where the lookup path Lookup4 does not successfully find a target record using effective date of field7 and symbol SYM1 set to 3 and symbol SYM2 set to zero. Select all other records. This example is the same as:

```
SKIPIF(ISNOTFOUND({Lookup4,
                    field7;$SYM1=3,$SYM2=0}))
```

```
IF DAYSBEWEEN({field1},{field2})
  > 7
  THEN SKIP
ENDIF
```

Skip records where there are more than 7 days between field1 and field2, and select all other records. This example can also be written:

```
SKIPIF(DAYSBEWEEN({field1},{field2})
        > 7)
```

```
IF ({field1} BEGINS_WITH "BBB")
  THEN SKIP
ENDIF
```

Skip input records where field1 begins with characters "BBB", and select all other records. This example can be written:

```
SKIPIF({field1} BEGINS_WITH "BBB")
```

It is better to use BEGINS\_WITH because the logic text executes faster.

```
IF ({field2} CONTAINS "CCC")
  THEN SKIP
ENDIF
```

Skip input records where field2 contains characters "CCC", and select all other records. This example can be written:

```
SKIPIF({field2} CONTAINS "CCC")
```

It is better to use CONTAINS because the logic text executes faster.

```
IF ({field3} ENDS_WITH "EEE")
  THEN SKIP
ENDIF
```

Skip input records where field3 ends with characters "EEE", and select all other records. This example can be written:

```
SKIPIF({field3} ENDS_WITH "EEE")
```

It is better to use ENDS\_WITH because the logic text executes faster.

```
IF ({field4} MATCHES "...")
  THEN SKIP
ENDIF
```

Skip input records where field4 matches characters "...", and select all other records. This example can be written:

```
SKIPIF({field4} MATCHES "...")
```

```
IF ({field1} LIKE "^BBB*")
  THEN SKIP
ENDIF
```

Skip input records where field1 begins with characters "BBB", and select all other records. This example has the same effect as:

```
SKIPIF({field1} BEGINS_WITH "BBB")
```

It is better to use BEGINS\_WITH because the logic text executes faster.

**Example logic text**

```
IF ({field1} LIKE "*CCC*")
  THEN SKIP
ENDIF
```

**Meaning**

Skip input records where field1 contains characters "CCC", and select all other records. This example has the same effect as:

```
SKIPIF({field1} CONTAINS "CCC")
```

It is better to use CONTAINS because the logic text executes faster.

```
IF ({field1} LIKE "*EEE$")
  THEN SKIP
ENDIF
```

Skip input records where field1 ends with characters "EEE", and select all other records. This example has the same effect as:

```
SKIPIF({field1} ENDS_WITH "EEE")
```

It is better to use ENDS\_WITH because the logic text executes faster.

```
IF ({field1} LIKE "^B*C*E$")
  THEN SKIP
ENDIF
```

Skip input records where field1 begins with "B", contains "C" and ends with "E", and select all other records.

---

## Examples: WRITE statements

**Examples: WRITE in Extract Record Filter****Example logic text**

```
' This WRITE statement is before
' any SELECT or SKIP.
WRITE (SOURCE = INPUT,
      USEREXIT = {Backup} )
```

**Meaning**

This WRITE sends all input records to the UserExit Routine called Backup.

```
' This WRITE statement is before
' any SELECT or SKIP.
WRITE (SOURCE = INPUT,
      USEREXIT = {Decryption} )
```

This WRITE decrypts all input records and puts the results "in place" so that the input records are processed later in the extract phase.

```
SELECTIF ( {product_code}
          = "ABC" )
WRITE (SOURCE = INPUT,
      DEST = FILE = {All_ABC} )
```

The WRITE sends only input records with product\_code = "ABC" to the logical file "All\_ABC".

```
IF ( {product_code} = "ABC" )
  THEN SELECT
ENDIF
WRITE (SOURCE = INPUT,
      DEST = FILE = {All_ABC} )
```

This example has the same effect as the previous example. Ensure the IF statement is purely for SELECT statements only.



---

## Chapter 2. Logic text 2: Extract Column Assignment

### 01 Summary of this topic

The sections in this topic are as follows:





- "10 Introduction"
- "20 How do I create logic text for Extract Column Assignment?"
- "30 What do I use this for?" on page 18
- "40 Statements to define column values" on page 18
- "50 Statements for writing records" on page 18
- "90 Examples" on page 18

### 10 Introduction

See topic "**Logic text 2: Extract Column Assignment overview**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".

### 20 How do I create logic text for Extract Column Assignment?

This logic text is part of a view and is associated with a column and a view source file. To create this logic text in an existing view, do the following:

1. Ensure you are on the "**Edit View**" screen for the relevant view. For help with displaying this screen, see topic "**Modifying Views**" in the General User Guide.
2. The view must already have at least one view source defined. If there is no view source defined, see topic "**Edit View (View Editor tab) screen help**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".
3. Ensure you are on the "**View Editor**" tab. If the "**View Properties**" tab is displayed, click the **Show Grid / Properties** button.  or press F9 or select **Edit -> Show Grid/Properties**.
4. In the column immediately below "View Editor", if there is a plus + sign to the left of "**View Source**", double click the plus sign to expand the list of view sources.
5. Click on a cell in the relevant column that is also in the row for the relevant view source file.
6. The "**Column Source Properties**" window appears. Double click "**Column Source Type**" and select "**Formula**" from the drop down box.
7. Double click "**Column Source Value**" and click .
8. Type your logic text in the window "**Create New Extract Column Assignment**" or "**Edit Extract Column Assignment**".
9. Click  to check if the logic text is valid.
10. Fix any errors shown in the "**Logic Text Validation Errors**" window.
11. When the new logic text is complete and valid,
  - **EITHER** click  (the save icon),
  - **OR** select **File, Save**,

- OR press **Ctrl+S**.
12. You may wish to close some of the open windows, such as "**Logic Text Helper**" and "**Create New Extract Column Assignment**" or "**Edit Extract Column Assignment**".

### 30 What do I use this for?

Extract Column Assignment logic text can be placed in each column in a view. The logic text for each column is executed in order of the column, so column 1 logic text is executed first, and so on. The Extract Column Assignment logic text can be used for the following purposes:

- **Define column values.** Define the value of the current column in the view, or any other column. All the fields in the selected input records are available for access. A column can be defined by any input field, on lookups of values in other logical files or by arithmetic and logic. The logic text for Extract Column Assignment has the most functions and the most flexibility of all the logic text, so most of the processing should be done here. More details are below.
- **Write records to other logical files.** You may write the entire input record, or all columns up to a given column. You can write to view output files. You can write records to extract work files (EXT files) that are passed to the format phase for more processing. More details are below.

### 40 Statements to define column values

The overall idea is that your logic text describes how to select or skip, but not both.

Choice for logic text	Notes
COLUMN =	Define the value of the current column.
COL.nnn =	Set the value of any other column in the view other than the current one. The "nnn" is the number of the particular column, for example COL.3 for column 3.
IF statements	There can be any number of IF statements, and one IF can have other IF statements nested inside. COLUMN and COL.nnn statements can be inside any of these IF statements.

### 50 Statements for writing records

For syntax of WRITE statements, see topic "**Syntax: WRITE statements in Extract Column Assignment**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".

### 90 Examples

See the topics below.



---

## Examples: COLUMN and COL.nnn statements

### Examples: COLUMN and COL.nnn in Extract Column Assignment

In all the following examples, **COLUMN** can be replaced by **COL.nnn**, for example COL.3. You can set the value of any COL.nnn from any other column. You can create multiple COL.nnn statements in Extract Column Assignment logic text.

#### Example logic text

```
COLUMN = ({field2}/{field1}) * 100  
COL.27 = {field1} * {field26}  
COL.28 = {field14} + {field1}  
COL.29 = 0  
COL.30 = "ABC"
```

```
COLUMN = ALL("-")  
COLUMN = REPEAT("-", 13)  
COLUMN = "\xFF"
```

```
COLUMN = {Lookup1.Field3}
```

```
COLUMN = {Lookup1.Field3,field7}
```

```
COLUMN = {Lookup1.Field3,;$SYM="A"}
```

```
COLUMN = {Lookup1.Field3,field7;$SYM1=3,$SYM2=0}
```

```
COLUMN = DAYSBEWEEN({BUY_DATE},{SHIP_DATE})
```

#### Meaning

Set the current column to field2 divided by field1 all multiplied by 100. Set column 27 to field1 times field26. Set column 28 to field14 plus field1. Set column 29 to zero. Set column 30 to "ABC".

Set the current column to all dashes.

Set the current column to 13 dashes.

Set the current column to hexadecimal FF.

Set the current column to Field3 found by lookup path Lookup1

Set the current column to Field3 found by lookup path Lookup1 using effective date of field7.

Set the current column to Field3 found by lookup path Lookup1 using symbol SYM set to "A".

Set the current column to Field3 found by lookup path Lookup1 using effective date of field7 and symbols SYM1 set to 3 and SYM2 set to zero.

Set the current column to the days between the transaction date and the shipping date.

---

## Examples: IF with COLUMN and COL.nnn statements

### Examples: IF with COLUMN and COL.nnn in Extract Column Assignment

In all the following examples, **COLUMN** can be replaced by **COL.nnn**, for example COL.3. You can set the value of any COL.nnn from any other column. You can create multiple IF statements in Extract Column Assignment logic text. However, you cannot inquire on COL.nnn (for example, IF COL.4 = 0 is not allowed).

#### Example logic text

#### Meaning

### Example logic text

```
IF ({field1} > 0) THEN
  COLUMN = ({field2}/{field1}) * 100
  COL.27 = {field1} * {field26}
  COL.28 = {field14} + {field1}
ELSE
  COLUMN = 0
  COL.27 = 0
  COL.28 = 0
ENDIF
```

```
IF (CURRENT({field5}) <> PRIOR({field5}))
  THEN COLUMN = "PRODUCT: "
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field5} = "Total")
  THEN COLUMN = ALL("-")
ENDIF
```

```
IF {field6} = ALL("-")
  THEN COLUMN = {field2} + {field3}
ENDIF
```

```
IF ({field5} = "Total")
  THEN COLUMN = REPEAT("-", 13)
ENDIF
```

```
IF ({field6} = REPEAT("-", 13))
  THEN COLUMN = {field2} + {field3}
ENDIF
```

```
IF ({field5} = "Total")
  THEN COLUMN = "\xFF"
ENDIF
```

```
IF ({field6} = "\xFF")
  THEN COLUMN = {field2} + {field3}
ENDIF
```

```
IF ISNOTSPACES({field1})
  THEN COLUMN = {field1}
  ELSE COLUMN = "DEFAULT"
ENDIF
```

```
IF ISFOUND({Lookup1})
  THEN COLUMN = {Lookup1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ISFOUND({Lookup2;$SYM="A"})
  THEN COLUMN = {Lookup2;$SYM="A"}
  ELSE COLUMN = 0
ENDIF
```

### Meaning

If field1 is greater than zero then set the current column to field2 divided by field1 all multiplied by 100, set column 27 to field1 times field26 and set column 28 to field 14 plus field1. If field1 is not greater than zero then set the current column and columns 27 and 28 to zero.

If the current record has a different value of field5 from the previous record, set the current column to "PRODUCT: " otherwise set the current column to blank. This assumes the input file is sorted into field5 order.

If field5 is "Total" then set the current column to all dashes.

If field6 is all dashes, then set the current column to a total of fields 2 and 3.

If field5 is "Total" then set the current column to 13 dashes.

If field6 is 13 dashes, then set the current column to a total of fields 2 and 3.

If field5 is "Total" then set the current column to hexadecimal FF.

If field6 is hexadecimal FF, then set the current column to a total of fields 2 and 3.

If field1 is not spaces then set the current column to field1, otherwise set the current column to "DEFAULT".

If the lookup path Lookup1 uses the current record to successfully find a target record, then set the current column to the lookup path field found, otherwise set the current column to blank.

If the lookup path Lookup2 using symbol SYM set to "A", then set the current column to that lookup field, otherwise set the current column to zero.

### Example logic text

```
IF ISNULL({field4}
  THEN COLUMN = "EMPTY"
  ELSE COLUMN = {field4}
ENDIF
```

```
IF ISNUMERIC({field4}
  THEN COLUMN = {field4} * 100
  ELSE COLUMN = 0
ENDIF
```

```
IF (DAYSBEWEEN({BUY_DATE},{SHIP_DATE})
  > 10)
  THEN COLUMN = {SHIP_DATE}
  ELSE COLUMN = {BUY_DATE}
ENDIF
```

```
IF ({field1} BEGINS_WITH "BBB")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field2} CONTAINS "CCC")
  THEN COLUMN = {field2}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field3} ENDS_WITH "EEE")
  THEN COLUMN = {field3}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field4} MATCHES "...")
  THEN COLUMN = {field4}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field1} LIKE "MA...")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field1} LIKE "..VA..")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field1} LIKE ".....NA")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field1} LIKE "^BBB*")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

### Meaning

If field4 for the current record contains null values, then set the current column to "EMPTY", otherwise set the current column to field4.

If field4 for the current record is numeric, then set the current column to field4 times 100, otherwise set the current column to zero.

If there are more than 10 days between the transaction date and the shipping date, then set the current column to the shipping date, otherwise set the current column to the transaction date.

If field1 begins with characters "BBB" then set the current column to field1, otherwise set the current column to blank.

If field2 contains characters "CCC" then set the current column to field2, otherwise set the current column to blank.

If field3 ends with characters "EEE" then set the current column to field3, otherwise set the current column to blank.

If field4 matches characters "..." then set the current column to field4, otherwise set the current column to blank.

Select input records where field1 is exactly 5 characters starting with "MA", and skip all other records.

Select input records where field1 is exactly 6 characters with characters 3 and 4 as "VA", and skip all other records.

Select input records where field1 is exactly 6 characters ending in "NA", and skip all other records.

If field1 begins with characters "BBB" then set the current column to field1, otherwise set the current column to blank.

### Example logic text

```
IF ({field1} LIKE "*CCC*")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field1} LIKE "EEE$")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

```
IF ({field1} LIKE "^B*C*$")
  THEN COLUMN = {field1}
  ELSE COLUMN = " "
ENDIF
```

### Meaning

If field1 contains characters "CCC" then set the current column to field1, otherwise set the current column to blank.

If field1 ends with characters "EEE" then set the current column to field1, otherwise set the current column to blank.

If field1 begins with "B", contains "C" and ends with "E" then set the current column to field1, otherwise set the current column to blank.

---

## Examples: WRITE statements

### Examples: IF with WRITE in Extract Column Assignment

#### Example logic text

```
IF (ISNUMERIC({field4}) AND
    ({field5} > {field6} * 10) AND
    (ISNOTSPACES{field7})
  THEN WRITE (SOURCE=DATA,
              USEREXIT={DB2_Update})
ENDIF
```

```
IF (ISNOTNULL({field3}) AND
    ({field2} = {field1} + {field5})
  THEN WRITE (SOURCE=INPUT,
              DEST=FILE=
                {LogicalFile3})
ENDIF
```

```
IF (DAYSBETWEEN({field12},{field15})
    > 10) AND
    (ISFOUND({Lookup3;$SYM="A"}))
  THEN WRITE (SOURCE=VIEW,
              DEST=EXT=03)
ENDIF
```

#### Meaning

If field4 is numeric and field5 is greater than field6 times 10 and field7 is not spaces, then call the user-exit routine DB2\_Update for the columns up to the current point. This effectively does a writes to a DB2 table the columns in that record up to the current point.

If field3 is not nulls and field2 equals field1 plus field 5 then write the entire input record to LogicalFile3. All columns in the input record are included, no matter what column contains this logic text.

If field12 and field15 are more than 10 days apart and the lookup path Lookup3 works with symbol SYM set to "A", then write the columns up to the current point to extract work file (EXT) 3. This assumes that the control record for this environment has a Maximum Extract File Number that is at least 3, or any overwrite of the VDP has also set this parameter to at least 3.

---

## Chapter 3. Logic text 3: Format Column Calculations

### 01 Summary of this topic

The sections in this topic are as follows:

- "10 Introduction"
- "20 How do I create logic text for Format Column Calculations?"
- "30 What do I use this for?" on page 24
- "40 Statements" on page 24
- "90 Examples" on page 24





### 10 Introduction

The logic text "Format Column Calculations" can also be called "Format-Phase Calculations".

See topic "**Logic text 3: Format Column Calculations overview**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".

### 20 How do I create logic text for Format Column Calculations?

This logic text is part of a view and is associated with a column during the format phase. To create this logic text in an existing view, do the following:

1. Ensure you are on the "**Edit View**" screen for the relevant view. For help with displaying this screen, see topic "**Modifying Views**" in the General User Guide.
2. The view must have a format phase. If there is no format phase or you are not sure, see topic "**Edit View (View Properties, General tab) screen help**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".
3. Ensure you are on the "**View Editor**" tab. If the "**View Properties**" tab is displayed, click the **Show Grid / Properties** button.  or press F9 or select **Edit -> Show Grid/Properties**.
4. Select a numeric column that is not part of the sort key. (A numeric column is any Data Type that is not Alphanumeric.) In that column double click on the cell for row "**Format-Phase Calculation**". That cell may be grey, which means this logic text cannot be applied. This may be because the column is part of the sort key, or is not numeric. This may also be because there is no format phase.
5. Click .
6. Type your logic text in the window "**Create New Format-Phase Calculation**" or "**Edit Format-Phase Calculation**".
7. Click  to check if the logic text is valid.
8. Fix any errors shown in the "**Logic Text Validation Errors**" window.
9. When the new logic text is complete and valid,
  - **EITHER** click  (the save icon),
  - **OR** select **File, Save**,

- OR press Ctrl+S.
10. You may wish to close some of the open windows, such as "Logic Text Helper" and "Create New Format-Phase Calculation" or "Edit Format-Phase Calculation".

### 30 What do I use this for?

Format Column Calculations logic text can be placed in some columns in a view, if a format phase exists for that view. The column with this logic text must be numeric and must not be part of the sort key.

The logic text for columns is executed in order of the columns, so column 1 logic text is executed first, and so on. The Format Column Calculations logic text is used for updating values of columns during the format phase.

You can only update the value of the current column (that contains that logic text). The logic text can read only the columns to the left of the column that contains the logic text. This means, for example, that in column 1 the values of all other columns are not available to the logic text.

Less functions and other syntax is available, as compared to the Extract Column Assignment logic text. Format Column Calculations is mainly intended for creating subtotals and totals of numeric fields.

### 40 Statements

The overall idea is that your logic text describes how to select or skip, but not both.

Choice for logic text	Notes
COLUMN =	Define the value of the current column.
IF statements	There can be any number of IF statements, and one IF can have other IF statements nested inside. COLUMN statements can be inside any of these IF statements.

### 90 Examples

See the topics below.

---

## Examples: COLUMN statements

### Examples: COLUMN statement in Format Column Calculation

You must set a value of COLUMN.

Example logic text	Meaning
COLUMN = Col.3 * Col.4	Set the current column to column 3 times column 4. The current column must be to the right of Column 4.
COLUMN = "TOTAL"	Set the current column to "TOTAL".

Example logic text	Meaning
<code>COLUMN = ALL("-")</code>	Set the current column to all dashes.
<code>COLUMN = REPEAT("-", 13)</code>	Set the current column to 13 dashes.
<code>COLUMN = "\xFF"</code>	Set the current column to hexadecimal FF.

---

## Examples: IF with COLUMN statements

### Examples: IF with COLUMN in Format Column Calculation

You must set a value of COLUMN.

Example logic text	Meaning
<pre>IF (Col.7 = 999)   THEN COLUMN = "TOTAL" ENDIF</pre>	If column 7 is 999 then set the current column to "TOTAL". The current column must be to the right of Column 7.
<pre>IF (Col.7 = 999)   THEN COLUMN = Col.3 * Col.4 ENDIF</pre>	If column 7 is 999 then set the current column to column 3 times column 4. The current column must be to the right of Column 7.
<pre>IF (Col.2 = Col.3)   THEN COLUMN = ALL("-") ENDIF</pre>	If column 2 equals column 3 then set the current column to all dashes. The current column must be to the right of Column 3.
<pre>IF (Col.2 = Col.3)   THEN COLUMN =     REPEAT("-", 13) ENDIF</pre>	If column 2 equals column 3 then set the current column to 13 dashes. The current column must be to the right of Column 3.
<pre>IF (Col.4 = "14733")   THEN COLUMN = "\xFF" ENDIF</pre>	If column 4 is "14733" then set the current column to hexadecimal FF. The current column must be to the right of Column 4.





---

## Chapter 4. Logic text 4: Format Record Filter

### 01 Summary of this topic

The sections in this topic are as follows:

- "10 Introduction"
- "20 How do I create logic text for Format Record Filter?"
- "30 What do I use this for?" on page 28
- "40 Statements" on page 28
- "90 Examples" on page 28




### 10 Introduction

The logic text "Format Record Filter" can also be called "Format-Phase Record Filter".

See topic "**Logic text 4: Format Record Filter overview**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".

### 20 How do I create logic text for Format Record Filter?

This logic text is part of a view and is associated with the format phase. To create this logic text in an existing view, do the following:

1. Ensure you are on the "**Edit View**" screen for the relevant view. For help with displaying this screen, see topic "**Modifying Views**" in the General User Guide.
2. The view must have a format phase. If there is no format phase or you are not sure, see topic "**Edit View (View Properties, General tab) screen help**". To find that topic in a PDF, see chapter "**Cross reference of topics and PDF files**".
3. Ensure you are on the "**View Properties**" tab. If the "**View Editor**" tab is displayed, click the **Show Grid / Properties** button.  or press F9 or select **Edit -> Show Grid/Properties**.
4. Go to the "**View Properties, Format Phase**" tab, and click on the "**Create**" or "**Edit**" button under heading "**Format-Phase Record Filter**".
5. Type your logic text in the window "**Create New Format-Phase Record Filter**" or "**Edit Format-Phase Record Filter**".
6. Click  to check if the logic text is valid.
7. Fix any errors shown in the "**Logic Text Validation Errors**" window.
8. When the new logic text is complete and valid,
  - **EITHER** click  (the save icon),
  - **OR** select **File, Save**,
  - **OR** press **Ctrl+S**.
9. You may wish to close some of the open windows, such as "**Logic Text Helper**" and "**Create New Format-Phase Record Filter**" or "**Edit Format-Phase Record Filter**".

## 30 What do I use this for?

Format Record Filter logic text is used for selecting records in the view for final output. More details are below.

## 40 Statements

The overall idea is that your logic text describes how to select or skip, but not both.

Choice for logic text	Notes
One SELECTIF statement	A SELECTIF statement gives a condition for selecting view records. Skip any records that do not meet the condition.
One SKIPIF statement	A SKIPIF statement gives a condition for skipping view records. Select any records that do not meet the condition.
One IF statement using only SELECT statements	One IF statement can contain a SELECT statement for the THEN or ELSE cases. Alternatively, the THEN or ELSE cases might contain other IF statements that also has SELECT statements or IF statements, and the whole thing still counts as one IF statement. Skip any records that do not qualify for a SELECT somewhere in the IF statement. No actual SKIP statements are allowed.
One IF statement using only SKIP statements	One IF statement can contain a SKIP statement for the THEN or ELSE cases. Alternatively, the THEN or ELSE cases might contain other IF statements that also has SKIP statements or IF statements, and the whole thing still counts as one IF statement. Select any records that do not qualify for a SKIP somewhere in the IF statement. No actual SELECT statements are allowed.

## 90 Examples

See the topics below.

---

### Examples: SELECTIF statements

#### Examples: SELECTIF in Format Record Filter

Example logic text	Meaning
SELECTIF(COL.3 > 1000)	Select for output only those records with column 3 greater than 1000. Skip all other records. The code at left is a shorthand for: IF (COL.3 > 1000) THEN SELECT ENDIF
SELECTIF(COL.2 = "ABC")	Select for output only those records with column 2 equal to "ABC". Skip all other records.
SELECTIF(NOT COL.2 = "ABC")	Select those output records with field column 2 not equal to "ABC". Skip all other records. This example gives the same result as: SKIPIF(COL.2 = "ABC")

Example logic text	Meaning
SELECTIF(COL.2 = "A" OR COL.2 = "D")	Select for output only those records with column 2 equal to "A" or "D". Skip all other records.
SELECTIF(COL.2 = "A" AND COL.3 > 10)	Select for output only those records with column 2 equal to "A" and column 3 greater than 10. Skip all other records.
SELECTIF(COL.3 + COL.4 > COL.5)	Select for output only those records with column 3 plus column 4 is greater than column 5. Skip all other records.
SELECTIF(NOT COL.6 = ALL("-"))	Select for output those records with column 6 is not equal to all dashes. Skip all other records. This example gives the same result as: SKIPIF(COL.6 = ALL("-"))
SELECTIF(NOT COL.6 = REPEAT("-", 13))	Select for output those records with column 6 is not equal to 13 dashes. Skip all other records. This example gives the same result as: SKIPIF(COL.6 = REPEAT("-", 13))
SELECTIF(NOT COL.6 = "\xFF")	Select for output those records with column 6 is not equal to hexadecimal FF. Skip all other records. This example gives the same result as: SKIPIF(COL.6 = "\xFF")

---

## Examples: SKIPIF statements

### Examples: SKIPIF in Format Record Filter

Example logic text	Meaning
SKIPIF(COL.3 > 1000)	Skip for output those records with column 3 greater than 1000. Select all other records. The code at left is a shorthand for: IF (COL.3 > 1000) THEN SKIP ENDIF
SKIPIF(COL.2 = "ABC")	Skip for output those records with column 2 equal to "ABC". Select all other records.
SKIPIF(NOT COL.2 = "ABC")	Skip those output records with field column 2 not equal to "ABC". Select all other records. This example gives the same result as: SELECTIF(COL.2 = "ABC")
SKIPIF(COL.2 = "A" OR COL.2 = "D")	Skip for output those records with column 2 equal to "A" or "D". Select all other records.
SKIPIF(COL.2 = "A" AND COL.3 > 10)	Skip for output those records with column 2 equal to "A" and column 3 greater than 10. Select all other records.

Example logic text	Meaning
SKIPIF(COL.3 + Col.4 > Col.5)	Skip for output those records with column 3 plus column 4 is greater than column 5. Select all other records.
SKIPIF(COL.6 = ALL("-"))	Skip for output those records with column 6 is equal to all dashes. Select all other records.
SKIPIF(COL.6 = REPEAT("-", 13))	Skip for output those records with column 6 is equal to 13 dashes. Select all other records.
SKIPIF(COL.6 = "\xFF")	Skip for output those records with column 6 is equal to hexadecimal FF. Select all other records.

---

## Examples: IF with SELECT

### Examples: IF with SELECT in Format Record Filter

Example logic text	Meaning
IF (COL.3 > 1000) THEN SELECT ENDIF	Select for output only those records with column 3 greater than 1000. Skip all other records. The code at left can also be written as: SELECTIF(COL.3 > 1000)
IF (COL.2 = "ABC") THEN SELECT ENDIF	Select for output only those records with column 2 equal to "ABC". Skip all other records.
IF NOT (COL.2 = "ABC") THEN SELECT ENDIF	Select those output records with field column 2 not equal to "ABC". Skip all other records. The code at left gives the same result as: SKIPIF(COL.2 = "ABC")
IF (COL.2 = "A") OR (COL.2 = "D") THEN SELECT ENDIF	Select for output only those records with column 2 equal to "A" or "D". Skip all other records.
IF (COL.2 = "A") AND (COL.3 > 10) THEN SELECT ENDIF	Select for output only those records with column 2 equal to "A" and column 3 greater than 10. Skip all other records.
IF (COL.3 + Col.4 > Col.5 * 100) THEN SELECT ENDIF	Select for output only those records with column 3 plus column 4 is greater than column 5 times 100. Skip all other records.
IF NOT (COL.6 = ALL("-")) THEN SELECT ENDIF	Select for output those records with column 6 is not equal to all dashes. Skip all other records. This example gives the same result as: SKIPIF(COL.6 = ALL("-"))
IF (COL.6 = REPEAT("-", 13)) THEN SELECT ENDIF	Select for output those records with column 6 is equal to 13 dashes. Skip all other records.

### Example logic text

```
IF (COL.6 = "\xFF")
  THEN SELECT
ENDIF
```

```
IF (COL.2 = "A") AND
  (COL.3 > 10)
  THEN SELECT
ELSE IF (COL.2 = "D")
  THEN SELECT
  ELSE IF (COL.2 = "G")
    THEN SELECT
  ENDIF
ENDIF
ENDIF
```

```
IF (COL.2 = "A") AND
  (COL.3 > 10)
  THEN IF (COL.4 + COL.5
    > COL.6)
    THEN SELECT
  ELSE IF (COL.7 = 0)
    THEN SELECT
  ENDIF
ENDIF
ENDIF
```

### Meaning

Select for output those records with column 6 is equal to hexadecimal FF. Skip all other records.

Select for output those records with column 2 equal to "A" and column 3 greater than 10.

Also select for output those records with column 2 equal to "D".

Also select for output those records with column 2 equal to "G".

Skip all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF (COL.2 = "A" AND COL.3 > 10) OR
  (COL.2 = "D") OR
  (COL.2 = "G")
  THEN SELECT
ENDIF
```

Consider those records with column 2 equal to "A" and column 3 greater than 10.

Of the considered records, select for output those records with column 4 plus column 5 is greater than column 6.

Of the considered records not yet selected, select also for output those records with column 7 equal to zero.

Skip all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF (COL.2 = "A" AND COL.3 > 10) AND
  ((COL.4 + COL.5 > COL.6) OR
  (COL.7 = 0))
  THEN SELECT
ENDIF
```

---

## Examples: IF with SKIP

### Examples: IF with SKIP in Format Record Filter

Example logic text	Meaning
IF (COL.3 > 1000) THEN SKIP ENDIF	Skip for output those records with column 3 greater than 1000. Select all other records. The code at left can also be written as: SKIPIF(COL.3 > 1000)
IF (COL.2 = "ABC") THEN SKIP ENDIF	Skip for output those records with column 2 equal to "ABC". Select all other records.
IF NOT (COL.2 = "ABC") THEN SKIP ENDIF	Skip those output records with field column 2 not equal to "ABC". Select all other records. The code at left gives the same result as: SELECTIF(COL.2 = "ABC")
IF (COL.2 = "A") OR (COL.2 = "D") THEN SKIP ENDIF	Skip for output those records with column 2 equal to "A" or "D". Select all other records.
IF (COL.2 = "A") AND (COL.3 > 10) THEN SKIP ENDIF	Skip for output those records with column 2 equal to "A" and column 3 greater than 10. Select all other records.
IF (COL.3 + COL.4 > COL.5 * 100) THEN SKIP ENDIF	Skip for output those records with column 3 plus column 4 is greater than column 5 times 100. Select all other records.
IF (COL.6 = ALL("-")) THEN SKIP ENDIF	Skip for output those records with column 6 is equal to all dashes. Select all other records. This example gives the same result as: SKIPIF(COL.6 = ALL("-"))
IF (COL.6 = REPEAT("-", 13)) THEN SKIP ENDIF	Skip for output those records with column 6 is equal to 13 dashes. Select all other records. This example gives the same result as: SKIPIF(COL.6 = REPEAT("-", 13))
IF (COL.6 = "\xFF") THEN SKIP ENDIF	Skip for output those records with column 6 is equal to hexadecimal FF. Select all other records. This example gives the same result as: SKIPIF(COL.6 = "\xFF")

### Example logic text

```
IF (COL.2 = "A") AND
  (COL.3 > 10)
THEN SKIP
ELSE IF (COL.2 = "D")
  THEN SKIP
  ELSE IF (COL.2 = "G")
    THEN SKIP
    ENDIF
  ENDIF
ENDIF
```

```
IF (COL.2 = "A") AND
  (COL.3 > 10)
THEN IF (COL.4 + COL.5
  > COL.6)
  THEN SKIP
  ELSE IF (COL.7 = 0)
    THEN SKIP
    ENDIF
  ENDIF
ENDIF
```

### Meaning

Skip for output those records with column 2 equal to "A" and column 3 greater than 10.

In addition, skip for output those records with column 2 equal to "D".

In addition, skip for output those records with column 2 equal to "G".

Select all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF (COL.2 = "A" AND COL.3 > 10) OR
  (COL.2 = "D") OR
  (COL.2 = "G")
  THEN SKIP
ENDIF
```

Consider those records with column 2 equal to "A" and column 3 greater than 10.

Of the considered records, skip for output those records with column 4 plus column 5 is greater than column 6.

Of the considered records not yet skipped, skip also for output those records with column 7 equal to zero.

Select all other records.

Notice that the logic text at left counts as only one IF statement, because the extra IF statements are nested inside the first.

The code at left can also be written as follows (note the use of brackets to control the evaluation of the conditions):

```
IF (COL.2 = "A" AND COL.3 > 10) AND
  ((COL.4 + COL.5 > COL.6) OR
  (COL.7 = 0))
  THEN SKIP
ENDIF
```





---

## Chapter 5. Rules for all logic text

Rules	Notes
Extra blanks between keywords and expressions have no effect.	<p>For example, these IF statements are all the same:</p> <pre>IF ({field1}={field2}) THEN COLUMN= {field1} ENDIF  IF ({field1} = {field2}) THEN   COLUMN = {field1} ENDIF  IF   ({field1}     =       {field2})     THEN       COLUMN       =       {field1}     ENDIF  IF ({field1} = {field2})THEN   COLUMN = {field1} ENDIF  IF ({field1} = {field2})   THEN COLUMN = {field1} ENDIF</pre> <p><b>WARNING: Extra blanks change text strings</b>, for example "ABC" and "A B C" are different strings.</p>
Logic text can continue on the next line. A backslash (\) is optional at line end.	<p>In previous versions of SAFR, a backslash (\) was required in order to continue a line of logic text on the next line. This backslash is no longer required. The backslash is still allowed for backwards compatibility. This means the following statements are the same:</p> <pre>IF ({FIELD1} &gt;= 2)\   THEN COLUMN = {FIELD1} ENDIF  IF ({FIELD1} &gt;= 2)   THEN COLUMN = {FIELD1} ENDIF</pre>
The case of keywords has no effect.	<p>For example, these IF statements are all the same:</p> <pre>IF ({FIELD1} &gt;= COL.2)   THEN COLUMN = {FIELD1}   ELSE COLUMN = COL.2 ENDIF  if ({field1} &gt;= col.2)   then column = {field1}   else column = col.2 endif  If ({Field1} &gt;= Col.2)   Then Column = {Field1}   Else Column = Col.2 Endif</pre> <p><b>WARNING: Case changes text strings</b>, for example "ABC" and "abc" are different strings.</p>

## Rules

### After a single quote

everything on that line is a comment

## Notes

Examples are:

```
' Make col the larger of field1 & col.2
IF ({FIELD1} >= COL.2)
  THEN COLUMN = {FIELD1} ENDIF
COLUMN = {field3} ' Rest is comment (OK).
```

**WARNING: comments to the left of keywords result in hiding the keywords**, for example:

```
IF ({field1} = {field2}) THEN
  COLUMN = {field1}
' This comment hides keyword:      ENDIF
```

Use **curly brackets { }** to enclose **input fields** or names of metadata from the SAFR Workbench (such as a lookup path, logical file, physical file).

Examples are:

```
{shipping_date}
{product_category}
{CA_Sales_2009_Logical_File}
{Products_Logical_File}
{Products_USA_File}
{Lookup_Sales_to_Product_Category}
```

## Keyboard shortcuts for all logic text screens

See topic "What are the keyboard shortcuts?"

To find that topic in a PDF, see chapter "Cross reference of topics and PDF files".

---

## Chapter 6. Cross reference of topics and PDF files

### How to download a PDF

Go to SAFR Information Center, select **About this Information Center** and select **PDF**. Follow the instructions on that page.

### Alphabetical list of topics

Note the following:

- "InfoCtr4150" means the PDF called "SAFR Information Center 4.15.00" which contains all help topics.
- "Top 3" means the PDF called "Top 3 - Admin Guide, General Users Guide and Overviews".

Find the required topic in the first column below. The columns to the right show the PDFs that contain that topic.

Topic	PDF	PDF	PDF	PDF
Add View Source errors	Troubleshooting			InfoCtr4150
Add View Source screen help	Screens			InfoCtr4150
Admin Guide	Admin Guide		Top 3	InfoCtr4150
Administrators START HERE	Admin Guide		Top 3	InfoCtr4150
Basics of using the SAFR Workbench	Admin Guide	General Users Guide	Top 3	InfoCtr4150
Batch activate lookup paths	Admin Guide		Top 3	InfoCtr4150
Batch Activate Lookup Paths errors	Troubleshooting			InfoCtr4150
Batch Activate Lookup Paths screen help	Screens			InfoCtr4150
Batch activate views	Admin Guide		Top 3	InfoCtr4150
Batch Activate Views errors	Troubleshooting			InfoCtr4150
Batch Activate Views screen help	Screens			InfoCtr4150
Change Log Path screen help	Screens			InfoCtr4150
Changing Log Path	Admin Guide	General Users Guide	Top 3	InfoCtr4150
Checking metadata dependencies	Admin Guide		Top 3	InfoCtr4150
Clear environment	Admin Guide		Top 3	InfoCtr4150
Clear environment messages	Screens			InfoCtr4150
Column Source Properties errors	Troubleshooting			InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Column Source Properties screen help	Screens			InfoCtr4150
Common Key Buffers overview	Overviews		Top 3	InfoCtr4150
Control records overview	Overviews		Top 3	InfoCtr4150
Copying metadata	Admin Guide		Top 3	InfoCtr4150
Create New Extract Column Assignment errors	Troubleshooting			InfoCtr4150
Create New Extract Column Assignment screen help	Screens			InfoCtr4150
Create New Extract Record Filter errors	Troubleshooting			InfoCtr4150
Create New Extract Record Filter screen help	Screens			InfoCtr4150
Create New Format-Phase Calculation errors	Troubleshooting			InfoCtr4150
Create New Format-Phase Calculation screen help	Screens			InfoCtr4150
Create New Format-Phase Record Filter errors	Troubleshooting			InfoCtr4150
Create New Format-Phase Record Filter screen help	Screens			InfoCtr4150
Creating control records	Admin Guide		Top 3	InfoCtr4150
Creating environments	Admin Guide		Top 3	InfoCtr4150
Creating global fields	Admin Guide		Top 3	InfoCtr4150
Creating groups	Admin Guide		Top 3	InfoCtr4150
Creating logical files	Admin Guide		Top 3	InfoCtr4150
Creating logical records	Admin Guide		Top 3	InfoCtr4150
Creating lookup paths	General Users Guide		Top 3	InfoCtr4150
Creating physical files	Admin Guide		Top 3	InfoCtr4150
Creating user-exit routines	Admin Guide		Top 3	InfoCtr4150
Creating users	Admin Guide		Top 3	InfoCtr4150
Creating view folders	Admin Guide		Top 3	InfoCtr4150
Creating views	General Users Guide		Top 3	InfoCtr4150
Deleting metadata	Admin Guide		Top 3	InfoCtr4150
Deleting metadata messages	Troubleshooting			InfoCtr4150
Dependency Checker errors	Troubleshooting			InfoCtr4150
Dependency Checker screen help	Screens			InfoCtr4150
Do SAFR batch processes run sequentially or in parallel?				InfoCtr4150
Edit Control Record errors	Troubleshooting			InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Edit Control Record screen help	Screens			InfoCtr4150
Edit Environment errors	Troubleshooting			InfoCtr4150
Edit Environment screen help	Screens			InfoCtr4150
Edit Extract Column Assignment errors	Troubleshooting			InfoCtr4150
Edit Extract Column Assignment screen help	Screens			InfoCtr4150
Edit Extract Record Filter errors	Troubleshooting			InfoCtr4150
Edit Extract Record Filter screen help	Screens			InfoCtr4150
Edit Format-Phase Calculation errors	Troubleshooting			InfoCtr4150
Edit Format-Phase Calculation screen help	Screens			InfoCtr4150
Edit Format-Phase Record Filter errors	Troubleshooting			InfoCtr4150
Edit Format-Phase Record Filter screen help	Screens			InfoCtr4150
Edit Global Field errors	Troubleshooting			InfoCtr4150
Edit Global Field screen help	Screens			InfoCtr4150
Edit Group errors	Troubleshooting			InfoCtr4150
Edit Group screen help	Screens			InfoCtr4150
Edit Logical File errors	Troubleshooting			InfoCtr4150
Edit Logical File screen help	Screens			InfoCtr4150
Edit Logical Record errors	Troubleshooting			InfoCtr4150
Edit Logical Record (Assoc. Log. Files tab) screen help	Screens			InfoCtr4150
Edit Logical Record (LR Fields tab) screen help	Screens			InfoCtr4150
Edit Logical Record (LR Properties tab) screen help	Screens			InfoCtr4150
Edit Lookup Path errors	Troubleshooting			InfoCtr4150
Edit Lookup Path (General tab) screen help	Screens			InfoCtr4150
Edit Lookup Path (Lookup Path Definition tab) screen help	Screens			InfoCtr4150
Edit Physical File errors	Troubleshooting			InfoCtr4150
Edit Physical File screen help	Screens			InfoCtr4150
Edit User errors	Troubleshooting			InfoCtr4150
Edit User screen help	Screens			InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Edit User-Exit Routine errors	Troubleshooting			InfoCtr4150
Edit User-Exit Routine screen help	Screens			InfoCtr4150
Edit View (View Editor tab) screen help	Screens			InfoCtr4150
Edit View (View Properties, Extract Phase tab) screen help	Screens			InfoCtr4150
Edit View (View Properties, Format Phase tab) screen help	Screens			InfoCtr4150
Edit View (View Properties, General tab) screen help	Screens			InfoCtr4150
Edit View (View Properties, Header/ Footer tab) screen help	Screens			InfoCtr4150
Edit View errors	Troubleshooting			InfoCtr4150
Edit View Folder errors	Troubleshooting			InfoCtr4150
Edit View Folder screen help	Screens			InfoCtr4150
Empty "Deleted Views" folder	Admin Guide		Top 3	InfoCtr4150
Environment Checker errors	Troubleshooting			InfoCtr4150
Environment Checker screen help	Screens			InfoCtr4150
Environments - advanced overview	Overviews		Top 3	InfoCtr4150
Environments overview	Overviews		Top 3	InfoCtr4150
Examples: COLUMN and COL.nnn statements	Logic text types			InfoCtr4150
Examples: COLUMN statements	Logic text types			InfoCtr4150
Examples: IF with COLUMN and COL.nnn statements	Logic text types			InfoCtr4150
Examples: IF with COLUMN statements	Logic text types			InfoCtr4150
Examples: IF with SELECT	Logic text types			InfoCtr4150
Examples: IF with SKIP	Logic text types			InfoCtr4150
Examples: SELECTIF statements	Logic text types			InfoCtr4150
Examples: SKIPIF statements	Logic text types			InfoCtr4150
Examples: WRITE statements	Logic text types			InfoCtr4150
Export metadata overview	Overviews		Top 3	InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Export Utility errors	Troubleshooting			InfoCtr4150
Export Utility screen help	Screens			InfoCtr4150
Exporting metadata	Admin Guide		Top 3	InfoCtr4150
FAQ				InfoCtr4150
Finding and replacing logic text	Admin Guide		Top 3	InfoCtr4150
Find/Replace Logic Text errors	Troubleshooting			InfoCtr4150
Find/Replace Logic Text screen help	Screens			InfoCtr4150
Finding all environments for a particular a metadata item name	Admin Guide	General Users Guide	Top 3	InfoCtr4150
Finding screen help	Admin Guide	General Users Guide	Top 3	InfoCtr4150
General users guide	General Users Guide		Top 3	InfoCtr4150
General users START HERE	General Users Guide		Top 3	InfoCtr4150
Generating reports on metadata	General Users Guide		Top 3	InfoCtr4150
Global fields overview	Overviews		Top 3	InfoCtr4150
Glossary				InfoCtr4150
Group Membership errors	Troubleshooting			InfoCtr4150
Group Membership screen help	Screens			InfoCtr4150
Group Permissions By Environment (Component Security section) screen help	Screens			InfoCtr4150
Group Permissions By Environment (Environment, Assoc. Groups sections) screen help	Screens			InfoCtr4150
Group Permission By Environment messages	Troubleshooting			InfoCtr4150
Group Permissions (Component Security section) screen help	Screens			InfoCtr4150
Group Permissions (Groups, Assoc. Environs sections) screen help	Screens			InfoCtr4150
Group Permission messages	Troubleshooting			InfoCtr4150
Groups - advanced overview	Overviews		Top 3	InfoCtr4150
Groups overview	Overviews		Top 3	InfoCtr4150
How do I generate a Dependency Checker Report?				InfoCtr4150

Topic	PDF	PDF	PDF	PDF
How do I generate an Environment Checker Report?				InfoCtr4150
How do I generate an Environment Security Report?				InfoCtr4150
How do I generate a Logical Record Report?				InfoCtr4150
How do I generate a Lookup Path Report?				InfoCtr4150
How do I generate a View Column PIC Report?				InfoCtr4150
How do I generate a View Column Report?				InfoCtr4150
How do I generate a View Properties Report?				InfoCtr4150
How many address spaces does SAFR use?				InfoCtr4150
How many processes per address space?				InfoCtr4150
How many SAFR passes will there be?				InfoCtr4150
How scalable is SAFR in z/OS?				InfoCtr4150
I want to see more logic text for all rows in the Find/Replace Logic Text screen				InfoCtr4150
Import metadata overview	Overviews		Top 3	InfoCtr4150
Import Utility screen help	Screens			InfoCtr4150
Import Utility errors	Troubleshooting			InfoCtr4150
Importing metadata	Admin Guide		Top 3	InfoCtr4150
Logging into the SAFR Workbench	General Users Guide		Top 3	InfoCtr4150
Logic text 1: Extract Record Filter	Logic text types			InfoCtr4150
Logic text 1: Extract Record Filter overview	Overviews		Top 3	InfoCtr4150
Logic text 2: Extract Column Assignment	Logic text types			InfoCtr4150
Logic text 2: Extract Column Assignment overview	Overviews		Top 3	InfoCtr4150
Logic text 3: Format Column Calculations	Logic text types			InfoCtr4150
Logic text 3: Format Column Calculations overview	Overviews		Top 3	InfoCtr4150
Logic text 4: Format Record Filter	Logic text types			InfoCtr4150



Topic	PDF	PDF	PDF	PDF
Logic text 4: Format Record Filter overview	Overviews		Top 3	InfoCtr4150
Logic text diagrams 1: Extract Record Filter	Logic text diagrams			InfoCtr4150
Logic text diagrams 2: Extract Column Assignment	Logic text diagrams			InfoCtr4150
Logic text diagrams 3: Format Column Calculation	Logic text diagrams			InfoCtr4150
Logic text diagrams 4: Format Record Filter	Logic text diagrams			InfoCtr4150
Logic Text Helper errors	Troubleshooting			InfoCtr4150
Logic Text Helper screen help	Screens			InfoCtr4150
Logic text overview	Overviews		Top 3	InfoCtr4150
Logic Text Validation Errors message help	Troubleshooting			InfoCtr4150
Logic Text Validation Errors screen help	Screens			InfoCtr4150
Logic text: summary diagrams	Logic text diagrams			InfoCtr4150
Logic text: syntax	Logic text syntax			InfoCtr4150
Logical files overview	Overviews		Top 3	InfoCtr4150
Logical records overview	Overviews		Top 3	InfoCtr4150
Lookup paths overview	Overviews		Top 3	InfoCtr4150
Metadata overview	Overviews		Top 3	InfoCtr4150
Metadata - advanced overview	Overviews		Top 3	InfoCtr4150
Migrate metadata overview	Overviews		Top 3	InfoCtr4150
Migrating metadata	Admin Guide		Top 3	InfoCtr4150
Migration Utility errors	Troubleshooting			InfoCtr4150
Migration Utility screen help	Screens			InfoCtr4150
Modifying control records	Admin Guide		Top 3	InfoCtr4150
Modifying Environments	Admin Guide		Top 3	InfoCtr4150
Modifying global fields	Admin Guide		Top 3	InfoCtr4150
Modifying group membership	Admin Guide		Top 3	InfoCtr4150
Modifying group permissions by environment	Admin Guide		Top 3	InfoCtr4150
Modifying group permissions by group	Admin Guide		Top 3	InfoCtr4150
Modifying groups	Admin Guide		Top 3	InfoCtr4150
Modifying logical files	Admin Guide		Top 3	InfoCtr4150
Modifying logical records	Admin Guide		Top 3	InfoCtr4150
Modifying lookup paths	General Users Guide		Top 3	InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Modifying own user account	General Users Guide		Top 3	InfoCtr4150
Modifying physical files	Admin Guide		Top 3	InfoCtr4150
Modifying user-exit routines	Admin Guide		Top 3	InfoCtr4150
Modifying users	Admin Guide		Top 3	InfoCtr4150
Modifying view folders	Admin Guide		Top 3	InfoCtr4150
Modifying views	General Users Guide		Top 3	InfoCtr4150
New Control Record errors	Troubleshooting			InfoCtr4150
New Control Record screen help	Screens			InfoCtr4150
New Environment errors	Troubleshooting			InfoCtr4150
New Environment screen help	Screens			InfoCtr4150
New Global Field errors	Troubleshooting			InfoCtr4150
New Global Field screen help	Screens			InfoCtr4150
New Group errors	Troubleshooting			InfoCtr4150
New Group screen help	Screens			InfoCtr4150
New Logical File errors	Troubleshooting			InfoCtr4150
New Logical File screen help	Screens			InfoCtr4150
New Logical Record errors	Troubleshooting			InfoCtr4150
New Logical Record (Assoc. Log. Files tab) screen help	Screens			InfoCtr4150
New Logical Record (LR Fields tab) screen help	Screens			InfoCtr4150
New Logical Record (LR Properties tab) screen help	Screens			InfoCtr4150
New Lookup Path errors	Troubleshooting			InfoCtr4150
New Lookup Path (General tab) screen help	Screens			InfoCtr4150
New Lookup Path (Lookup Path Definition tab) screen help	Screens			InfoCtr4150
New Physical File errors	Troubleshooting			InfoCtr4150
New Physical File screen help	Screens			InfoCtr4150
New User errors	Troubleshooting			InfoCtr4150
New User screen help	Screens			InfoCtr4150
New User-Exit Routine errors	Troubleshooting			InfoCtr4150
New User-Exit Routine screen help	Screens			InfoCtr4150
New View (View Editor tab) screen help	Screens			InfoCtr4150

Topic	PDF	PDF	PDF	PDF
New View (View Properties, Extract Phase tab) screen help	Screens			InfoCtr4150
New View (View Properties, Format Phase tab) screen help	Screens			InfoCtr4150
New View (View Properties, General tab) screen help	Screens			InfoCtr4150
New View (View Properties, Header/ Footer tab) screen help	Screens			InfoCtr4150
New View errors	Troubleshooting			InfoCtr4150
New View Folder errors	Troubleshooting			InfoCtr4150
New View Folder screen help	Screens			InfoCtr4150
Opening a log file	General Users Guide		Top 3	InfoCtr4150
Overviews	Overviews		Top 3	InfoCtr4150
Parallelism overview	Overviews		Top 3	InfoCtr4150
Performance Engine (PE) overview	Overviews		Top 3	InfoCtr4150
Physical files overview	Overviews		Top 3	InfoCtr4150
Pipes overview	Overviews		Top 3	InfoCtr4150
Return to login	General Users Guide		Top 3	InfoCtr4150
Rules for all logic text	Logic text diagrams	Logic text syntax	Logic text types	InfoCtr4150
SAFR Connection Manager errors	Troubleshooting			InfoCtr4150
SAFR Connection Manager screen help	Screens			InfoCtr4150
SAFR Login errors	Troubleshooting			InfoCtr4150
SAFR Login screen help	Screens			InfoCtr4150
SAFR optimization overview	Overviews		Top 3	InfoCtr4150
SAFR overview - START HERE	Overviews		Top 3	InfoCtr4150
SAFR phases overview	Overviews		Top 3	InfoCtr4150
Searching lists of metadata	Admin Guide	General Users Guide	Top 3	InfoCtr4150
Sort Key Properties errors	Troubleshooting			InfoCtr4150
Sort Key Properties screen help	Screens			InfoCtr4150
Sort Key Titles errors	Troubleshooting			InfoCtr4150
Sort Key Titles screen help	Screens			InfoCtr4150
Syntax: BEGINS_WITH	Logic text syntax			InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Syntax: COL.nnn in Format Record Filter	Logic text syntax			InfoCtr4150
Syntax: COLUMN and COL.nnn in Extract Column Assignment	Logic text syntax			InfoCtr4150
Syntax: COLUMN and COL.nnn in Format Column Calculations	Logic text syntax			InfoCtr4150
Syntax: COLUMN and COL.nnn statements	Logic text syntax			InfoCtr4150
Syntax: CONTAINS	Logic text syntax			InfoCtr4150
Syntax: ENDS_WITH	Logic text syntax			InfoCtr4150
Syntax: function ALL	Logic text syntax			InfoCtr4150
Syntax: function BATCHDATE	Logic text syntax			InfoCtr4150
Syntax: function CURRENT	Logic text syntax			InfoCtr4150
Syntax: function DATE	Logic text syntax			InfoCtr4150
Syntax: function DAYSBEWEEN	Logic text syntax			InfoCtr4150
Syntax: function FISCALDAY	Logic text syntax			InfoCtr4150
Syntax: function FISCALMONTH	Logic text syntax			InfoCtr4150
Syntax: function FISCALPERIOD	Logic text syntax			InfoCtr4150
Syntax: function FISCALQUARTER	Logic text syntax			InfoCtr4150
Syntax: function FISCALYEAR	Logic text syntax			InfoCtr4150
Syntax: function ISFOUND	Logic text syntax			InfoCtr4150
Syntax: function ISNOTFOUND	Logic text syntax			InfoCtr4150
Syntax: function ISNOTNULL	Logic text syntax			InfoCtr4150
Syntax: function ISNOTNUMERIC	Logic text syntax			InfoCtr4150
Syntax: function ISNOTSPACES	Logic text syntax			InfoCtr4150
Syntax: function ISNULL	Logic text syntax			InfoCtr4150
Syntax: function ISNUMERIC	Logic text syntax			InfoCtr4150
Syntax: function ISSPACES	Logic text syntax			InfoCtr4150
Syntax: function MONTHSBETWEEN	Logic text syntax			InfoCtr4150
Syntax: function PRIOR	Logic text syntax			InfoCtr4150
Syntax: function REPEAT	Logic text syntax			InfoCtr4150
Syntax: function RUNDAY	Logic text syntax			InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Syntax: function RUNMONTH	Logic text syntax			InfoCtr4150
Syntax: function RUNPERIOD	Logic text syntax			InfoCtr4150
Syntax: function RUNQUARTER	Logic text syntax			InfoCtr4150
Syntax: function RUNYEAR	Logic text syntax			InfoCtr4150
Syntax: function YEARSBETWEEN	Logic text syntax			InfoCtr4150
Syntax: functions	Logic text syntax			InfoCtr4150
Syntax: functions Q1, Q2, Q3 and Q4	Logic text syntax			InfoCtr4150
Syntax: IF statements	Logic text syntax			InfoCtr4150
Syntax: IF statements in Extract Column Assignment	Logic text syntax			InfoCtr4150
Syntax: IF statements in Extract Record Filter	Logic text syntax			InfoCtr4150
Syntax: IF statements in Format Column Calculations	Logic text syntax			InfoCtr4150
Syntax: IF statements in Format Record Filter	Logic text syntax			InfoCtr4150
Syntax: LIKE	Logic text syntax			InfoCtr4150
Syntax: lookup paths	Logic text syntax			InfoCtr4150
Syntax: MATCHES	Logic text syntax			InfoCtr4150
Syntax: SELECT and SELECTIF statements in Extract Record Filter	Logic text syntax			InfoCtr4150
Syntax: SELECT and SELECTIF statements in Format Record Filter	Logic text syntax			InfoCtr4150
Syntax: SELECT, SELECTIF, SKIP and SKIPIF statements	Logic text syntax			InfoCtr4150
Syntax: SKIP and SKIPIF statements in Extract Record Filter	Logic text syntax			InfoCtr4150
Syntax: SKIP and SKIPIF statements in Format Record Filter	Logic text syntax			InfoCtr4150
Syntax: string comparison	Logic text syntax			InfoCtr4150
Syntax: WRITE statements	Logic text syntax			InfoCtr4150
Syntax: WRITE statements in Extract Column Assignment	Logic text syntax			InfoCtr4150
Syntax: WRITE statements in Extract Record Filter	Logic text syntax			InfoCtr4150
Tokens overview	Overviews		Top 3	InfoCtr4150
User-exit routines overview	Overviews		Top 3	InfoCtr4150

Topic	PDF	PDF	PDF	PDF
Users overview	Overviews		Top 3	InfoCtr4150
View Activation Errors message help	Troubleshooting			InfoCtr4150
View Activation Errors screen help	Screens			InfoCtr4150
View folders overview	Overviews		Top 3	InfoCtr4150
View Source Properties errors	Troubleshooting			InfoCtr4150
View Source Properties screen help	Screens			InfoCtr4150
Views - advanced overview	Overviews		Top 3	InfoCtr4150
Views overview	Overviews		Top 3	InfoCtr4150
WE log file overview	Overviews		Top 3	InfoCtr4150
WE Security overview	Overviews		Top 3	InfoCtr4150
What are the keyboard shortcuts?	Admin Guide	General Users Guide	Top 3	InfoCtr4150
What does SAFR stand for?				InfoCtr4150
What does WE stand for?				InfoCtr4150
What metadata do I want to see?	General Users Guide		Top 3	InfoCtr4150
What's new in this Information Center				InfoCtr4150
Where is the WE log file?	Admin Guide	General Users Guide	Top 3	InfoCtr4150
Why use SAFR?				InfoCtr4150
Workbench overview	Overviews		Top 3	InfoCtr4150
XML structure for metadata overview	Overviews		Top 3	InfoCtr4150

---

## Appendix: Notices

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

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

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
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:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 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 document 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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.



Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

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

---

## Trademarks

### Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



---

# Index

## A

Appendix: Notices 49

## B

Backslash for line continuation in logic  
text in WE 35

## I

Info Center

Appendix: Notices 49

Trademarks 51

## L

Logic text 1: Extract Record Filter

2 Main details 1

4 Rules for all logic text 35

Backslash for line continuation 35

Logic text 2: Extract Column Assignment

2 Main details 17

4 Rules for all logic text 35

Backslash for line continuation 35

Logic text 3: Format Column Calculation

2 Main details 23

4 Rules for all logic text 35

Backslash for line continuation 35

Logic text 4: Format Record Filter

2 Main details 27

4 Rules for all logic text 35

Backslash for line continuation 35

## R

Rules for all logic text 35

## S

SAFR Info Center

Appendix: Notices 49

Trademarks 51

## T

Trademarks 51







Printed in USA