Maintaining Transaction Filter Definitions

A transaction filter definition specifies filter conditions for replication, based on the values of fields in the database records. No transaction filter definitions are required. Transaction filter definitions are defined using the FILTER initialization parameter, but should be maintained using Event Replicator Administration.

Listing Transaction Filter Definitions

To use Event Replicator Administration to list the transaction filter definitions stored in the Replicator system file:

1. Select an Event Replicator Server in tree-view as described in *Selecting Event Replicator Databases*.
2. Click and expand **Replication Definitions** in tree-view under the selected database.
3. Click on **Transaction Filters** in the tree-view under **Replication Definitions**.

   A table listing the transaction filter definitions in the Replicator system file appears in detail-view.

Adding Transaction Filter Definitions

To add transaction filter definitions, complete the following steps:

Step 1. Access the Transaction Filter Definition Area of Event Replicator Administration

To access the transaction filter definition area, complete the following steps:

1. List the transaction filter definitions in Event Replicator Administration, as described in *Listing Transaction Filter Definitions*.

   The transaction filter definitions are listed in detail-view.

2. Right-click on **Transaction Filters** in the tree-view under **Replication Definitions**.

   A drop-down menu appears.

3. Click on **Create New Internal Transaction Filter** in the drop-down menu.

   A blank Internal Transaction Filter panel appears in detail-view.

Step 2. Specify a Transaction Filter Definition Name and Type

To specify a transaction filter definition name and type:

1. On the Internal Transaction Filter panel, supply values for the following fields:

Field	Description	Default
Transaction Filter Name (FILTER NAME)	Specify a unique name for the transaction filter definition. The name must use alphanumeric characters and be between one and 8 characters long.	---
Exclude or Include Records (FRECORDS)	Specify "Include" to include (replicate) the records selected by the filter definition or "Exclude" to exclude (do not replicate) records selected by the filter definition.	Include
Filter Conditions	See the next step in these instructions.	---

2. Click **OK**.

   The internal transaction filter is created. If you added filter conditions, prior to click OK, the filter conditions are included in the transaction filter definition. Filter conditions can be added and maintained using the Event Replicator Administration, as described in the next step in these instructions.

Step 3. Add Filter Conditions to the Transaction Filter Definition

For more information about rules of filter conditions, read *Rules for Writing Filter Conditions*.

To add filter conditions to the transaction filter definition:

1. Click on the checkmark in the **Filter Conditions** field on the Internal Transaction Filter panel to display a list of filter conditions.

   Note:
   You might need to click the **Modify** button on the Internal Transaction Filter screen to make the **Filter Conditions** checkmark appear.

The Filter Conditions table appears in detail-view.

2. Add, modify, and delete the filter condition definitions in the Filter Conditions table.

   - To add a new filter condition definition, click **New**. A new, blank, line appears in the Filter Conditions table. Update the fields in the filter table for the new filter definition, as described later in this step.

   - To modify a filter condition definition, edit the fields in the table as described later in this step.

     Note:
     The filter condition definition name cannot be changed.

   - To delete a filter condition definition, select the definition you want to delete by clicking the check box for the definition in the **All** column of the Filter Conditions table. More than one filter condition definition can be selected at a time. To select all filter conditions, click on the check box in the **All** column heading. Once all the filter condition definitions you want to delete are selected, click **Delete**.

In the **Value** field of the **Filter Values** table, specify a value against which the source field will be compared. Only one value can be specified in each Value field. Up to 128 Value fields are available in which you can specify values; use the scroll bar to scroll through them.

The fields in a filter condition definition are described in the following table:

Parameter Name	**Specify**	**Default**
All	A selection field you can use to select one or more filter conditions in the Filter Conditions table.	none selected.

Parameter Name	Specify	Default
Group	A group number you can use to group field filters together within a transaction filter definition. All of the field filters with the same group number are blocked and logically ANDed together when the filters are examined during subscription processing. In other words, a field in the database must meet all of the criteria of the group before it is selected. Likewise, different groups of field filters are logically ORed together. In other words, a field need only meet the criteria specified by one of the groups to be selected. Valid values are numbers ranging from "1" through "999". The equivalent processing in the Event Replicator Server startup job is available through the use of the OR keyword within a series of FFIELD parameters.	---
Source Field (FFIELD)	The two-byte Adabas field code for the field to be compared. This field must be in the format buffer specified for the file in the SFILE definition of the subscription.	---
Source PE (FSPE)	The index number (occurrence) of the periodic group (PE) to which the condition relates if the source field in this field filter is a PE field. Valid values range from 0 through 191.	0, indicating the source field is not a PE field.
Source MU (FSMU)	The index number of the multiple-value field (MU) to which the condition relates if the source field in this field filter is an MU field. Valid values range from 0 through 191.	0, indicating the source field is not an MU field.
Source Image (FSIMAGE)	Whether the source field is in the after image, before image, or the default image of the record. Valid values are "After", "Before" and "None".	"After" for adds and updates; "Before" for deletes

Parameter Name	Specify	Default
Source Begin (FSBEGIN)	The starting byte number of the partial Adabas source field to be compared. This field is only used if you want to specify a partial field for comparison and only if the field is of alphanumeric or binary format. **Note:** The format of the complete field is used for partial field comparisons. Valid comparisons of different field types are listed in *Field Type Considerations*. For fixed length fields, valid values range from "1" (the start of the field) through the maximum length of the field (the last byte of the field). For variable length fields, valid values range from "1" (the start of the field) to the maximum length allowed for that field type. Counting occurs from left to right beginning with 1 for fields defined with alphanumeric format, and from right to left beginning with 1 for fields defined with binary format.	1
Source Length (FSLENGTH)	The numeric length of the partial Adabas source field that should be compared. This field is only used if you want to specify a partial field for comparison and only if the field is of alphanumeric or binary format. **Note:** The format of the complete field is used for partial field comparisons. Valid comparisons of different field types are listed in *Field Type Considerations*. For fixed length fields, errors will occur if the sum of the values of the Source Begin and Source Length parameters exceeds the fixed length of the field. For variable length fields, the sum of the values of the Source Begin and Source Length parameters must not exceed the maximum length of the field plus 1. For example, if a variable length field has format "A" with a maximum length of 253 bytes, settings of Source Begin=1 and Source Length=253 are valid, but settings of Source Begin=2 and Source Length=254 are not.	If Source Begin is not specified, the default value is the entire field. If Source Begin is specified, the default value is the maximum length of the field minus the value of the Source Begin parameter plus 1.

Parameter Name	Specify	Default
Condition (FCOND)	A condition operator code for the filter. Valid values are "equal to", "not equal to", "less than", "less than or equal to", "greater than", or "greater than or equal to" . When "equal to" or "not equal to" are specified, multiple target values and target values using wildcards can be tested. For all other condition codes, only single target values without wildcards can be tested.	equal to
Target Field (FTARGET)	The two-byte Adabas field code for the field with which the source field will be compared. This field must be in the same record as the source field. This field is mutually exclusive with the Target Value *n* fields. If you specify the target field code, you cannot specify values in the Target Value *n* fields.	---
Target PE (FTPE)	The index number of the periodic group (PE) to which the condition relates if the target field in this field filter is a PE field. Valid values range from 0 through 191. This field is mutually exclusive with the Target Value *n* fields. If you specify the target field code, you cannot specify values in the Target Value *n* fields.	0, indicating the target field is not a PE field.
Target MU (FTMU)	The index number (occurrence) of the multiple-value field (MU) to which the condition relates if the target field in this field filter is an MU field. Valid values range from 0 through 191. This field is mutually exclusive with the Target Value *n* fields. If you specify the target field code, you cannot specify values in the Target Value *n* fields.	0, indicating the target field is not an MU field.
Target Image (FTIMAGE)	Whether the target field is in the after image, before image, or the default image of the record. Valid values are "After", "Before" and "None". This field is mutually exclusive with the Target Value *n* fields. If you specify the target field code, you cannot specify values in the Target Value *n* fields.	"After" for adds and updates; "Before""" for deletes

Parameter Name	Specify	Default
Target Begin (FTBEGIN)	The starting byte number of the partial Adabas target field at which the comparison should begin. This field should only be specified if you want to specify a partial field for comparison, if the field is of alphanumeric or binary format, and only if an Adabas target field (Target Field) is specified. **Note:** The format of the complete field is used for partial field comparisons. Valid comparisons of different field types are listed in *Field Type Considerations*. For fixed length fields, valid values range from "1" (the start of the field) through the maximum length of the field (the last byte of the field). For variable length fields, valid values range from "1" (the start of the field) to the maximum length allowed for that field type. Counting occurs from left to right beginning with 1 for fields defined with alphanumeric format, and from right to left beginning with 1 for fields defined with binary format.	1

Parameter Name	Specify	Default
Target Length (FTLENGTH)	The numeric length of the partial Adabas target field that should be used for the comparison. This field should only be specified if you want to specify a partial field for comparison, if the field is of alphanumeric or binary format, and only if an Adabas target field (Target Field) is specified. **Note:** The format of the complete field is used for partial field comparisons. Valid comparisons of different field types are listed in *Field Type Considerations*. For fixed length fields, errors will occur if the sum of the values of the Target Begin and Target Length parameters exceeds the fixed length of the field. For variable length fields, the sum of the values of the Target Begin and Target Length parameters must not exceed the maximum length of the field plus 1. For example, if a variable length field has format "A" with a maximum length of 253 bytes, settings of Target Begin=1 and Target Length=253 are valid, but settings of Target Begin=2 and Target Length=254 are not.	If Target Begin is not specified, the default value is the entire field. If Target Begin is specified, the default value is the maximum length of the field minus the value of the Target Begin parameter plus 1.
Target Value (FLIST)	See the next step in these instructions. Click on the checkmark in this field to bring up the Filter Values panel in detail-view. You can use the Filter Values panel to specify one or more values for comparison in the filter condition. This field is mutually exclusive with the Target Field, Target Image, Target MU, and Target PE fields. You cannot specify values for the Target Field, Target Image, Target MU, or Target PE fields if you have specified a value for this field.	---

3. If you elected to include filter values in a filter condition and clicked the checkmark in the **Target Value** field on the Filter Conditions screen, the Filter Values panel appears in detail-view. You can use this panel to specify one or more values for the filter condition.

- To add a new filter value, click **New**. A new, blank, line appears in the Filter Values table. Update the fields in the filter values table for the filter definition, as described later in this step.
- To modify a filter value, edit the fields in the table as described later in this step.
- To delete a filter value, select the definition you want to delete by clicking the check box for the definition in the **All** column of the Filter Conditions table. More than one filter condition definition can be selected at a time. To select all filter conditions, click on the check box in the **All** column heading. Once all the filter condition definitions you want to delete are selected, click **Delete**.

The fields in a filter condition definition are described in the following table:

Parameter Name	Specify	Default
All	A selection field you can use to select one or more filter values in the Filter Values table.	none selected.

Parameter Name	Specify	Default
Value (FLIST)	This field is mutually exclusive with the Target Field, Target Image, Target MU, and Target PE fields. You cannot specify values for the Target Field, Target Image, Target MU, or Target PE fields if you have specified a value for this field. The following information should be considered when maintaining filter condition values: ● Strings that include blanks should be enclosed in single quotes. Apostrophes in strings must be doubled (for example: ‘six o”clock’). A maximum of 254 characters can be specified for each value. ● Each value may consist of either free-format characters or a mix of elements specified using the A( ) or X( ) notation. ○ If free-format data consists entirely of numeric data (including an optional leading "+" or "-" character) it is treated as a numeric value. ○ If a value (or part of a value) is specified using A( ) notation, it will be treated as alphabetic data. ○ Hexadecimal values may be specified using X( ) notation. ● A value must be specified entirely as free-format data, or composed of one or more A( ) or X( ) subelements. If a value begins with an A( ) or X( ) subelement all remaining subelements of the value must be so specified.	---

When all filter values have been made to your satisfaction, click **OK** to save them.

The Filter Conditions panel appears in detail-view.

4. Repeat Steps 2 and 3 until all filter conditions and any necessary filter values have been specified. When all specifications have been made to your satisfaction, click **OK** to save the transaction filter conditions.

Step 4. Save the Transaction Filter Definition

To use Event Replicator Administration to save a transaction filter definition:

1. When all specifications have been made to your satisfaction, click **OK** to save the definition in the Replicator system file.

Modifying Transaction Filter Definitions

To use Event Replicator Administration to modify a transaction filter definition in the Replicator system file:

1. List the transaction filter definitions in Event Replicator Administration, as described in *Listing Transaction Filter Definitions*.

   The transaction filter definitions are listed in detail-view.

2. Locate the definition you want to modify in the table in detail-view and click on it.

   The Transaction Filter panel appears in detail-view listing the current settings for the transaction filter definition you selected.

3. Click the **Modify** button.

   The transaction filter parameters you can modify become editable in detail-view. For information on modifying these parameters, read the description of *Adding Transaction Filter Definitions*.

4. When all modifications have been made, click **OK** to save the changes or click **Cancel** to cancel the changes.

Copying Transaction Filter Definitions

To use Event Replicator Administration to copy a transaction filter definition in the Replicator system file:

1. List the transaction filter definitions in Event Replicator Administration, as described in *Listing Transaction Filter Definitions*.

   The transaction filter definitions are listed in detail-view.

2. Locate the definition you want to copy in the table in detail-view and click on it.

   The Transaction Filter panel appears in detail-view listing the current settings for the transaction filter definition you selected.

3. Click the **Copy** button.

   A copy of the transaction filter definition is created and its parameter values appear in detail-view.

4. Specify a new, unique name for the copy of the transaction filter definition in the Value column for the **Transaction Filter Name** parameter.

5. If you wish, modify any other parameters for the new copy in detail-view. For information on modifying the parameters, read the description of *Adding Transaction Filter Definitions*.

6. When all modifications have been made, click **OK** to save the changes or click **Cancel** to cancel the copy.

Deleting Transaction Filter Definitions

To use Event Replicator Administration to delete a transaction filter definition in the Replicator system file:

1. List the transaction filter definitions in Event Replicator Administration, as described in *Listing Transaction Filter Definitions*.

   The transaction filter definitions are listed in detail-view.

2. Locate the definition you want to delete in the table in detail-view and click on it.

   The Transaction Filter panel appears in detail-view listing the current settings for the transaction filter definition you selected.

3. Click the **Delete** button.

   Note:
   If you want to delete a filter condition from the transaction filter definition, read *Modifying Transaction Filter Definitions*.

   A confirmation panel appears verifying that you want to delete the definition. If you click **Yes** (indicating that you do want to delete the definition), the definition is deleted. If you click **No** (indicating that you do not want to delete the definition), the definition is not deleted.

Rules for Writing Filter Conditions

There are various things you should consider when creating filter conditions. This section describes them.

- So My Record Matches the Filter Conditions -- Now What?
- Failed or Ignored Filter Conditions
- Target (FLIST Parameter) Value Syntax
- When You Can Specify Multiple Targets
- How Multiple Filter Conditions Are Interpreted
- Specifying a Range of Values

- Field Type Considerations
- Varying Field Length Considerations
- Using Wildcards

So My Record Matches the Filter Conditions -- Now What?

Filter conditions are based on the values of fields (or partial fields) in an SFILE record definition. If a field or partial field meets all of the filter conditions specified, the record is selected. Once selected, the record will be either included or excluded from replication processing, based on what the transaction filter definition specifies. So selection of a record does not necessarily mean that it will be replicated -- merely that it passed the filter conditions specified by the transaction filter definition. If the transaction filter definition indicates that selected records should be excluded from replication, the record will not be replicated.

Transaction filter definitions indicate whether selected records are replicated or not via the FRECORDS initialization parameter in the DDKARTE statements of the Event Replicator Server startup job or via the **Exclude or Include Records** field on the Transaction Filter screen of the Adabas Event Replicator Subsystem.

Note:
Include and exclude processing function in the same way for partial fields as for complete fields in your transaction filters.

Failed or Ignored Filter Conditions

A filter condition will be ignored if it cannot be evaluated. This can occur if the image to be tested (set by the FSIMAGE or FTIMAGE parameters) is not present for replication. The effect of this on filter processing varies, based on whether the filter occurs as part of include or exclude processing and, if it is included in a group of conditions, how the other conditions in the group are matched, failed, or ignored. This is best explained in a series of examples.

Note:
The following four examples using an add command (N1) are also true for an initial-state record since an initial-state record contains only an after image. No before image is present for an initial-state record.

1. Suppose an add command (N1) adds a record containing field AB to which the following filter is applied:

   ```
   FRECORDS=INCLUDE
     FFIELD='AB',FSIMAGE=BI
       FCOND=EQ
       FLIST='1916'
   ```

   In this case, the filter cannot be evaluated because only the after image is present for an add and the filter is for the before image (FSIMAGE=BI). So this filter is ignored and no test is done on the field to see if the before image is equal to "1916". Consequently the add transaction is not included in replication.

2. Likewise, a similar exclude filter is also ignored:

```
FRECORDS=EXCLUDE
  FFIELD='AB',FSIMAGE=BI
    FCOND=EQ
    FLIST='1916'
```

In this case, the filter cannot be evaluated because only the after image is present for an add and the filter is for the before image (FSIMAGE=BI). So this filter is ignored and no test is done on the field to see if the before image is equal to "1916". However, because this is an exclude filter, the add transaction is not *excluded* from replication; in other words, it is included in replication, regardless of whether or not the before image of the AB field was equal to "1916".

3. Now consider the following transaction filter using multiple filter conditions and include processing:

```
FILTER NAME=MYINCLF
  FRECORDS=INCLUDE
    FFIELD='BA',FSIMAGE=BI,FCOND=EQ,FLIST='AAAA'
    FFIELD='BB',FSIMAGE=AI,FCOND=EQ,FLIST='VVVV'
    FFIELD='BC',FSIMAGE=AI,FCOND=EQ,FLIST='XXXX'
```

If an add command (N1) is issued for a record containing the BA, BB, and BC fields, no before image is present for these fields -- only the after image. Therefore, the filter condition for BA is ignored because the filter is for the before image; the BA filter condition is treated as if it is not even specified. The add transaction, then, is only *included* in replication if both filters for fields BB and BC are true.

4. Finally, consider the following transaction filter using multiple filter conditions, exclude processing, and OR processing:

```
FILTER NAME=MYEXCLF
  FRECORDS=EXCLUDE
    FFIELD='BA',FSIMAGE=BI,FCOND=EQ,FLIST='AAAA'
    FFIELD='BB',FSIMAGE=AI,FCOND=EQ,FLIST='VVVV'
  OR
    FFIELD='CA',FSIMAGE=AI,FCOND=EQ,FLIST='EEEE'
    FFIELD='CB',FSIMAGE=AI,FCOND=EQ,FLIST='CCCC'
  OR
    FFIELD='DA',FSIMAGE=BI,FCOND=EQ,FLIST='OOOO'
    FFIELD='DB',FSIMAGE=BI,FCOND=EQ,FLIST='CCCC'
```

If an add command (N1) is issued for a record containing these fields, no before image is present for these fields -- only the after image. Therefore, the filter conditions for BA, DA, and DB are ignored because the filters are for the before image; these filter conditions are treated as if they are not even specified.

The add transaction, then, is only *excluded* in replication if the filter for BB is satisfied OR if both the filters for field CA and field CB are satisfied. Otherwise, the add transaction is included in replication.

Target (FLIST Parameter) Value Syntax

Target (FLIST parameter) values are the values to be compared to the source field (FFIELD parameter) using the condition type specified (FCOND parameter). When multiple values are being compared for a field, they must be specified in a comma-separated list.

Each value can be expressed in one of two ways:

- You can specify values as free-format text. This text can be any set of alphanumeric set of characters. If blanks are required in the value, you should enclose the value in single quotes.

  When the data in the text is all numeric with an optional leading "+" or "-" sign, it is flagged as a numeric value and will be handled differently depending on the source field type in the Event Replicator Server definitions.

- You can specify values as a combination of `A( )` and `X( )` constructs that enable you to enter data for the same variable in alphabetic format, hexadecimal format, or both, as required. If the element value starts with the string "A(" or "X(" it is treated as an `A( )` or `X( )` value. If the value does not start with one of these strings, the value is treated as free-format text.

This section describes rules specific to these different methods of specifying target values.

- Free-Format Value Rules
- A() and X(0) Format Value Rules
- Examples

Free-Format Value Rules

The following rules apply to free-format values.

- Free-format values can be any sequence of alphanumeric data apart from the comma character itself.
- If a blank is required for the free-format value, specify the value in single quotes.
- If an apostrophe is required as part of a free-format value, double the apostrophe (for example,‘six o”clock’).
- If the value consists of all numeric characters with an optional leading "+" or "-" sign, the value will be treated as numeric.
- If the value begins with a single asterisk (*), it is interpreted as a wildcard suffix (for example, ’*xyz’).
- If the value ends with a single asterisk, it is interpreted as a wildcard prefix (for example, ’abc*’).
- If two asterisks are found together (**) in any location in the free-format value, they are interpreted as a single asterisk in the resulting data.
- If a single asterisk is found in the middle of the data, it is rejected as invalid.

Note:
The asterisk wildcard can only be used if the condition for the filter expression is EQ (equal) or NE (not equal). They cannot be used for any other types of filter expression conditions.

A() and X(0) Format Value Rules

The following rules apply to A() and X() value specifications.

- The A( ) construct is specified using the following syntax:

```
A(data)
```

  In this syntax, the *data* specified can be any alphanumeric characters, except the parentheses characters.

- The X( ) construct is specified using the following syntax:

```
X(data)
```

  In this syntax, the *data* specified must be an even number of characters in the range X’F0’ to X’F9’ (i.e. 0 to 9) and x’C1’ to X’C6’ (i.e. A to F). Each pair of characters will represent the hexadecimal value for one byte in the resultant value.

- If a value starts with an A( ) or X( ) construct, the entire value must be specified using these constructs. You cannot mix them with free-format values.
- A( ) and X( ) constructs can be specified multiple times in the same value specification. They must always have matching opening and closing parentheses, or the entire value specification is treated as invalid.
- When the A( ) construct is used, the asterisk (*) wildcard character is treated in the same manner as for free-format values.
- When the X( ) construct is used, the X’5C’ character (which represents an asterisk) is treated like any other hexadecimal character and is not interpreted as a wildcard.

Examples

In the following example, an FLIST value of "ABCDE" is specified:

```
FLIST=’ABCDE’
```

In the following example, a numeric FLIST value of "12345" is specified:

```
FLIST=’12345’
```

In the following example, a numeric FLIST value of "-678" is specified:

```
FLIST=’-678’
```

In the following example, an FLIST value of "AB123" is specified:

```
FLIST=’AB123’
```

In the following example, an FLIST value of "XyZ" is specified:

```
FLIST=’A(XyZ)’
```

In the following example, an FLIST value of "SSS" (the alphabetic equivalent of X’E2E2E2’) is specified:

```
FLIST='X(E2E2E2)'
```

In the following example, an FLIST value of "abc<<<def" is specified:

```
FLIST='A(abc)X(4C4C4C)A(def)'
```

In the following example, an FLIST value of "AX(E2E2E2)" is specified:

```
FLIST='AX(E2E2E2)'
```

In the following example, an FLIST value of "1A(BCD)" is specified:

```
FLIST='1A(BCD)'
```

In the following example, an FLIST value of "1A(BCD)" is specified:

```
FLIST='1A(BCD)'
```

In the following example, an FLIST value of "*abc*" ("abc" is the alphabetic equivalent of X’C1C2C3’) is specified. Note that this FLIST value is open-ended because wildcards are specified:

```
FLIST='A(*)X(C1C2C3)A(*)'
```

In the following example, an FLIST value of "*def**" is specified. Note that the first asterisk specifies a wildcard, but the last two asterisks specify asterisk characters (the alphabetic equivalent of X’5C5C’):

```
FLIST='A(*def)X(5C5C)'
```

The following examples are invalid because they specify a wildcard asterisk in the middle of the values:

```
FLIST='*ABC*DEF*'
FLIST='X(F5F6)*X(F7F8)'
FLIST='X(F2)A(*)X(F4)'
```

The following examples are invalid because they specify invalid hexadecimal data:

```
FLIST='X(ABACFGZZAE)'
FLIST='X(ABC)'
```

The following example is invalid because it mixes free-format and hexadecimal data:

```
FLIST='X(AB)AB'
```

The following example is invalid because it misuses commas:

```
FLIST='X(ABAC),,A(123)'
```

The following example is invalid because it misuses parentheses in the A() construct:

```
FLIST='A(12(34))'
```

When You Can Specify Multiple Targets

You can only specify multiple targets if the condition operator is EQ (equal) or NE (not equal). The LT (less than), LE (less than or equal), GT (greater than), and GE (greater than or equal) operators logically assume a comparison of the field value to a single target value, so multiple target values are not allowed for these condition operators.

Since wildcards are essentially a concise way of specifying multiple targets, you can also only use wildcards when the condition operator is EQ or NE.

If your filter checks to see if the field value is equal to a list of target values, the field value need only be equivalent to *one* of the target values for the filter condition to be true. On the other hand, if your filter checks to see if the field value is not equal to a list of target values, the field value must not be equal to *any* of the target values for the filter condition to be true.

Examples

In the following example, records for which the after image of the AA field is equal to "1", "2", "3", or "4" are selected.

```
FFIELD='AA',FSIMAGE=AI
FCOND=EQ
FLIST='1,2,3,4'
```

In the following example, records for which the after image of the AA field is greater than "5" are selected.

```
FFIELD='AA',FSIMAGE=AI
FCOND=GT
FLIST='5'
```

In the following example, records for which the first three bytes of the after image of the BB field contain the characters "abc" are selected.

```
FFIELD='BB',FSIMAGE=AI
FCOND=EQ
FLIST='abc*'
```

In the following example, records for which the last three bytes of the after image of the BB field contain the characters "xyz" are selected.

```
FFIELD='BB',FSIMAGE=AI
FCOND=EQ
FLIST='*xyz'
```

In the following example, records in which *no* bytes of the after image of the BB field contain the characters "klm" are selected.

```
FFIELD='BB',FSIMAGE=AI
FCOND=NE
FLIST='*klm*'
```

The following example is invalid because it specifies multiple FLIST target values when the condition code is not EQ or NE.

```
FFIELD='AA',FSIMAGE=AI
FCOND=LE
FLIST='1,2,3,4'
```

The following example is invalid because it specifies a wildcard in the FLIST target value when the condition code is not EQ or NE.

```
FFIELD='AA',FSIMAGE=AI
FCOND=GT
FLIST='*xyz'
```

How Multiple Filter Conditions Are Interpreted

You can specify multiple filter conditions within a single transaction filter definition. Unless otherwise grouped, all of the specified filter conditions must be true for a record to be selected. In other words, the filter conditions are logically ANDed. In the following example, cond1, cond2, cond3, and cond4 must all be true for the record to be selected as they are logically ANDed:

```
FILTER NAME=MYINCLF
FRECORDS=INCLUDE
FFIELD='field',FCOND=cond1,FTARGET or FLIST values1
FFIELD='field',FCOND=cond2,FTARGET or FLIST values2
FFIELD='field',FCOND=cond3,FTARGET or FLIST values3
FFIELD='field',FCOND=cond4,FTARGET or FLIST values4
```

If, however, you want to insert some logical ORs in this example, you can. To do this you would use the OR keyword in the DDKARTE statements of the Event Replicator Server startup job or use the **Group** field on the Filter Condition screen in the Adabas Event Replicator Subsystem. As an example of using the OR keyword, consider the following modification to the example given earlier.

```
FILTER NAME=MYINCLF
FRECORDS=INCLUDE
FFIELD='field',FCOND=cond1,FTARGET or FLIST values1
FFIELD='field',FCOND=cond2,FTARGET or FLIST values2
FFIELD='field',FCOND=cond3,FTARGET or FLIST values3
OR
FFIELD='field',FCOND=cond4,FTARGET or FLIST values4
```

In this example, condition1, condition2, and condition3 must be true OR condition 4 must true for the record to be selected.

When using the **Group** field on the Filter Condition screen of the Adabas Event Replicator Subsystem to define your transaction filter definitions, simply use the same group number for those conditions you want ANDed. Conditions with different group numbers are logically ORed.

Specifying a Range of Values

You can specify a range of values in your filter condition by creating two conditions that are logically ANDed (read *How Multiple Filter Conditions Are Interpreted*). Simply define one filter condition to test for values greater than (GT) or greater than or equal to (GE) the lowermost value. Then define the second filter condition to test for values less than (LT) or less than or equal to (LE) the uppermost value. As both conditions must be true since they are logically ANDed, your range specification is assured.

Field Type Considerations

Ideally, when a field is compared to another field, the field types will be the same. However, it is possible to compare fields of different formats. For example, you can compare a packed decimal format field with a binary format field. For a complete list of compatible Adabas field types, refer to your Adabas documentation.

This section covers the following topics related to how fields of different formats are compared

- Valid Comparison Table
- Comparison Processing by Field Type
- UES Considerations

Valid Comparison Table

An asterisk (*) in a cell in the following table indicates that a comparison of the field types is valid. A blank in a cell in the table indicates that a comparison is not supported.

Field Data Type	Alphanumeric	Unpacked	Packed	Binary	Floating Point	Wide-Character	Fixed Point
Alphanumeric	*			*		*	
Unpacked		*	*	*	*		*
Packed		*	*	*	*		*
Binary	*	*	*	*	*		*
Floating Point		*	*	*	*		*
Wide-Character	*					*	
Fixed Point		*	*	*	*		*

Comparison Processing by Field Type

When either the source or target field is of type floating point (but not both fields), the other field will be converted to floating point, and a floating point comparison will be made. SARC settings governing byte-swapping and floating point type (HFP, IEEEfloat, and VAXfloat) are honored.

Note:
The conversion of very large numbers in a numeric format other than floating point to floating may result in a loss of precision because as the numbers get bigger, the range of numbers that may be represented in the floating point format is reduced. For example, the value 99,999,999,999,999,999 will be converted to the floating point value 99,999,999,999,999,984.

In all other cases the following conversions and comparisons will apply:

Source Field Data Type	Comparison Processing Notes
Unpacked	The source and target fields are converted to packed form for comparison.
Binary	Prior to comparison, the SARC byte order setting is honored for binary source and target fields. Packed and unpacked target fields are converted to binary and then compared. An alphanumeric target field is compared as is.
Packed	Prior to comparison, target fields of type fixed, unpacked, or binary are converted to packed.
Fixed	When the target field is binary, the source is converted to binary and then compared. When the target field is packed, the source field is converted to packed and then compared. When the target field is unpacked, both the source and target fields are converted to packed and then compared. When the target field is fixed, a direct comparison is made between the source and target fields (no conversion is necessary).

When a field is compared to a list of target values, the target values are converted (if they are not the same) to the data type of the source field, once the source field type is determined. This can cause problems in the accuracy of filter condition processing if a target value in the list cannot be converted or is otherwise incompatible with the required source field type. So target values and target fields must be specified carefully to avoid such problems.

Target list values entered as alphanumeric are converted to the data type of the source field, honoring the SARC, SACODE and SWCODE parameter settings.

Target list values for alphanumeric fields may be entered as alphanumeric, hexadecimal, or a mixture of both. If it is a mixture of both -- for example, FLIST=A(ABC)X(C4C5C6)A(GHI) -- it is treated as an alphanumeric field even though some of it is specified as hexadecimal.

Target list values for binary fields may be entered in hexadecimal. The hexadecimal values are assumed to be in a form that honors the SARC parameter settings.

Target list values for floating point fields may be entered in hexadecimal. The hexadecimal values are assumed to be in a form that honors the SARC parameter settings such as floating-point format and byte order.

Target list values may not be entered in hexadecimal for zoned decimal, packed decimal and fixed point fields.

UES Considerations

When a field is compared to a target value that is entered in hexadecimal, the target value is normally accepted without any conversion. It is assumed that you have taken into account the settings of the SARC, SACODE and SWCODE parameters when constructing the hexadecimal value. It is important to remember that a given hexadecimal value may have to reflect the setting of more than one of these three parameters.

- The SARC parameter defines special data architecture for fields in the record and value buffers (see the description of record buffers in your Adabas documentation). If the byte order bit of the SARC value is set, the hexadecimal value may have to be entered with low-order bytes first. If the field is a character field, the entered hexadecimal byte values must reflect the setting of the SARC encoding family bit.

  Note:
  If you want to transfer replicated and initial-state data to a relational database using the Event Replicator Target Adapter (the Destination Class, or DCLASS parameter, is set to "SAGTARG"), set the SARC parameter to "2" -- regardless of the location of your relational database.

- The SACODE parameter assigns special encoding for alphanumeric fields during the user session. Hexadecimal bytes values must reflect the SACODE setting.

- The SWCODE assigns special encoding for wide-character fields during the user session. Hexadecimal values must reflect the SWCODE setting.

This section covers the following topics related to UES processing:

- Internal Handling of UES Settings
- Examples Honoring UES Settings

Internal Handling of UES Settings

When FILTER FLIST value parameters are being processed, the UES settings and the FCOND values are taken into account in an attempt to minimize conversion overhead at runtime.

- If the FCOND setting is either EQ or NE then the FLIST value will be stored as entered. At runtime, this FLIST value can be compared directly with the field value with no conversion required regardless of the UES settings.

- If the FCOND setting is LT, GT, LE, or GE, the FLIST value will be converted, taking into account the UES settings, so that valid comparisons can be made for the specified FCOND value. At runtime, the field value will be similarly converted to facilitate valid comparisons.

It is important to note here that FLIST values that are entered as hexadecimal values for comparison with binary fields must be entered in a form that honors the SARC settings. In other words, they must be specified in the same form as the field is stored in the record buffer. Similarly wide-character field values entered as hexadecimal must be specified in the same form as the field is stored in the record buffer.

FLIST values that are entered as EBCDIC text, or as numbers, will be converted appropriately.

Examples Honoring UES Settings

Field Data Type	Examples

Floating Point	In the following example, an FLIST value of a short HFP floating point value of 1.0 is specified in hexadecimal:
	`FLIST='X(41100000)'`
	Regardless of the setting of the SARC byte order bit the hexadecimal value for a HFP floating point value will remain the same because HFP floating point does not honor byte swapping.
	In the following example, an FLIST value of a long HFP floating point value of 50,000.0 is specified in hexadecimal:
	`FLIST='X(44C3500000000000)'`
	In the following example, an FLIST value of a short VAX floating point value of 1.0 is specified in hexadecimal:
	`FLIST='X(40800000)'`
	This value is valid when the SARC byte order bit is not set. If the SARC order bit is set, the following hexadecimal value must be entered for a value of 1.0:
	`FLIST='X(80400000)'`
	Note that the bytes are swapped in pairs.
	In the following example, an FLIST value of a long VAX floating point value of 50,000.0 is specified in hexadecimal:
	`FLIST='X(4843500000000000)'`
	This value is valid when the SARC byte order bit is not set. If the SARC order bit is set, the following hexadecimal value must be entered for a value of 50,000.0:
	`FLIST='X(4348005000000000)'`
	Note that the bytes are swapped in pairs.

Wide-character	In the following example, an FLIST value of the wide-character string `'ABB ABCDEF'` in WCODE=4095 is specified in hexadecimal:
	`FLIST='X(0041004200420040004100420043004400450046)'`
	In the following example, an FLIST value of the wide-character string `'ABB ABCDEF'` in WCODE=4095 with the SARC byte order bit turned on, is specified in hexadecimal:
	`FLIST='X(4100420042004000410042004300440045004600)'`
	Note that the bytes are swapped in pairs.

Field Data Type	Examples

Binary	In the following example, an FLIST value of the binary value of decimal 4 with the SARC byte order bit turned off, is specified in hexadecimal:
	
	`FLIST='X(000004)'`
	
	In the following example, an FLIST value of the binary value of decimal 4 with the SARC byte order bit turned on, is specified in hexadecimal:
	
	`FLIST='X(040000)'`
	
	Note that the byte order is reversed. In other words, the first byte becomes the last byte, the second byte becomes the second-to-last byte, and so on.

Varying Field Length Considerations

When the length of the source field and target field are different, the shorter value is converted to the size of the longer value. For alphanumeric data, the value is padded on the right with blanks. For numeric data, the value is padded on the left with hexadecimal zeros.

Using Wildcards

You can use an asterisk (*) as a wildcard for target values if the condition code being used is EQ (equal) or NE (not equal). You cannot use wildcard characters for any other filter conditions (GT, LT, LE, or GE).

Note:
Wildcard values are not supported for wide character fields.

- If you want to test the field for any value beginning with a specific string of characters, simply append an asterisk to the end of the value. For example, to test for a field value starting with the characters "POW", specify "POW*" as the target value.
- If you want to test the field for the occurrence of as specific string within its value, precede and supercede the string with an asterisk. For example, to test for the occurrence of the string "WER", specify "*WER*" as the target value.
- If you need to test for the occurrence of an asterisk itself in a field value, specify two asterisks in a row for the target value ("**").