How Actuate Reports Process Adhoc Parameter Values and Expressions

By Chris Geiss
chris_geiss@yahoo.com

How Actuate Reports Process Adhoc Parameter Values and Expressions

By Chris Geiss (chris_geiss@yahoo.com)
Revised 2/25/2002

Table of Contents

About the Author

Chris Geiss is an independent consultant and software developer with over 16 years of software development experience. He has been working with the Actuate reporting products for 4 years and provides consulting, development, training and mentoring services for the Actuate e.Reporting Suite. Chris can be contacted at chris_geiss@yahoo.com.

Introduction

Adhoc report parameters are one of the many useful and powerful features of Actuate e.Reporting Suite 5.

Unfortunately, users occasionally run into problems when using reports that have adhoc parameters because they are not aware of all of the syntax rules regarding the entry of adhoc parameter values and expressions. The syntax for numeric and date parameters is fairly straightforward, but the syntax for string type parameters can be a little tricky and confusing to users because the Actuate suite attempts to make the user's life easier by providing automatic wildcard handling (pattern matching) for string type parameters. As a result, the user sometimes does not get the data that he or she expected.

This document summarizes the adhoc parameter value and expression syntax rules that I have discovered by experimenting with the e.Reporting Suite 5. Specifically, the adhoc parameter behavior discussed in this document is based on my work with the e.Report Designer Professional Version 5.0 SP1.

Tables of Adhoc Parameter Value and Expression Examples

The sections that follow provide tables of examples for each of the major adhoc parameter types: string, numeric and date. Each table provides examples of parameter values and expressions, the SQL Where clause syntax that is generated by the AcSQLQuerySource class for the parameter value or expression, and any applicable comments. The table for string type parameters has an additional column, which clearly defines the data filter that results from the specified parameter value or expression. This is done to help clarify the wildcard handling that Actuate implements for string type parameters. This document focuses on the AcSQLQuerySource class that is associated with the Graphical Query Editor. It is assumed that the same adhoc parameter behavior applies to the AcTextQuerySource class associated with the Textual Query Editor, but I have not yet tested this assumption.

The experiments I conducted were performed by capturing the SQL syntax generated by the ObtainSelectStatement method of the AcSQLQuerySource class. It would have been more effective to trace through the Actuate Foundation Class (AFC) code to see how adhoc parameters are processed. I tried that, but came to a roadblock of sorts because the ConditionExpression method of the AcQuerySource class calls the AdhocToSQL function to do the actual adhoc handling. AdhocToSQL is a 'C' function accessed using a Declare Function statement, so I was unable to trace into it using e.Report Designer Professional.

In the "Resulting SQL Where Clause Expression" columns in the following tables, "DBCOL" represents the database column with which the adhoc parameter is associated.

In the following tables, the single quote and double quote characters are used in the example parameter values and expressions. To avoid confusion, when a reference is made to a specific character string that is matched by a parameter value or expression, the string is delimited with the { and } bracket characters. The { and } characters should not be interpreted as being a part of the string.

How to Report Errors and Omissions

If you find any errors or omissions in this document, or have any suggestions, please email them to chris_geiss@yahoo.com so I can update this document accordingly.

Adhoc Parameter Processing for a String Type Database Column

Parameter Expression Entered	Resulting SQL Where Clause Expression	Data Filter Resulting from the Parameter Value	Comments
A	DBCOL LIKE 'A%'	Wildcard. Matches all values that start with {A}.	A single string value is automatically treated like a wildcard expression.
A%	DBCOL LIKE 'A%'	Same as above.	You can provide the % explicitly.
A\%	DBCOL LIKE 'A%'	Same as above.	The backslash character does not escape the meaning of the %.
‘A%’	DBCOL LIKE 'A%'	Same as above.	Using % forces wildcard handling.
‘A\%’	DBCOL LIKE 'A%'	Same as above.	Same as above.
%A	DBCOL LIKE '%A'	Wildcard. Matches all values that end with {A}.	You can change the default location of the % in a wildcard expression.
A%B	DBCOL LIKE 'A%B'	Wildcard. Matches all values that start with {A} and end with {B}.	Same as above.
%A%B%	DBCOL LIKE '%A%B%'	Wildcard. Matches all values that have an {A} and {B} within them.	You can include more than one % wildcard character.
A*	DBCOL LIKE 'A*%'	Wildcard. Matches all values that start with {A*}.	The asterisk does not act like a wildcard character. Only use ‘*’ if you are searching for the asterisk character.
‘A’	DBCOL = 'A'	Equality. Matches all values that equal {A}.	Placing parameter values in single quotes forces them to be treated literally instead of as wildcard expressions.
‘A*’	DBCOL = 'A*'	Equality. Matches all values that equal {A*}.	Only use ‘*’ if you are searching for the asterisk character.
“A”	DBCOL LIKE '"A"%'	Matches all values that start with {“A”}.	The double quote character does not have any special meaning. Double quote marks are treated as characters you are trying to match.
‘”A”’	DBCOL = '"A"'	Equality. Matches all values that equal {“A”}.	Enclosing the parameter value above in single quotes takes away the wildcard behavior.
A,B,C	DBCOL LIKE 'A%' OR DBCOL LIKE 'B%' OR DBCOL LIKE 'C%'	List of wildcards. Matches any value that starts with {A} or {B} or {C}.	

Adhoc Parameter Processing for a String Type Database Column (cont.)

Parameter Expression Entered	Resulting SQL Where Clause Expression	Data Filter Resulting from the Parameter Value	Comments
A\,B,C	DBCOL LIKE 'A,B%' OR DBCOL LIKE 'C%'	Matches all values that start with {A,B} or {C}.	The backslash character escapes the meaning of the comma.
‘A,B’,’C’	DBCOL = 'A' OR DBCOL LIKE 'B'%' OR DBCOL = 'C'	SQL syntax error.	Do not use. Comma has precedence over the single quote. Use \ to escape the meaning of the comma.
‘A\,B’,’C’	DBCOL = 'A,B' OR DBCOL = 'C'	Matches all values that equal {A,B} or {C}.	This provides the result desired above. Use \ to escape the meaning of the comma.
%A,%B,%C	DBCOL LIKE '%A' OR DBCOL LIKE '%B' OR DBCOL LIKE '%C'	Matches all values that end in {A} or {B} or {C}.	
‘A’,’B’,’C’	DBCOL = 'A' OR DBCOL = 'B' OR DBCOL = 'C'	Matches all values that equal {A} or {B} or {C}.	List of exact values to match.
“A”,”B”,”C”	DBCOL LIKE '"A"%' OR DBCOL LIKE '"B"%' OR DBCOL LIKE '"C"%'	Matches all values starting with {“A”} or {“B”} or {“C”}	The comparison includes the quote marks.
A-P	DBCOL BETWEEN 'A' AND 'P'	Match all values that are alphabetically between {A} and {P}.	Dash is used to specify a range of values and removes the default wildcard behavior.
A\-P	DBCOL LIKE 'A-P%'	Matches all values that start with {A-P}.	The backslash escapes the range meaning of the dash.
‘A\-P’	DBCOL = 'A-P'	Matches all values that equal {A-P}.	The single quotes take away the wildcard behavior.
‘A-P’	DBCOL BETWEEN 'A' AND 'P"	SQL syntax error.	Do not use. Must use a backslash to escape the meaning of the dash.
“A-P”	DBCOL BETWEEN '"A' AND 'P"'	Match all values that are alphabetically between {“A} and {P”}.	
‘A’-‘P’	DBCOL BETWEEN 'A' AND 'P'	Matches the same values as: A-P	The single quotes have no effect.
“A”-“P”	DBCOL BETWEEN '"A"' AND '"P"'	Match all values that are alphabetically between {“A”} and {“P”}.	
=A	DBCOL = 'A%'	Matches all values that equal {A%}. Does not perform a like.	Using the equal sign avoids the conversion to the like predicate, but includes a %.
=A%	DBCOL = 'A%'	Same as above.	
=%A	DBCOL = '%A'	Matches all values that equal {%A}. Does not perform a like.	

Adhoc Parameter Processing for a String Type Database Column (cont.)

Parameter Expression Entered	Resulting SQL Where Clause Expression	Data Filter Resulting from the Parameter Value	Comments
=%A%B%C%	DBCOL = '%A%B%C%'	Matches all values that equal {%A%B%C%}.	
=’A’	DBCOL = 'A'	Same as: ‘A’	Using the equal sign here does not make a difference.
=”A”	DBCOL = '"A"%'	Matches all values that equal {“A”%}.	
=A,B,C	DBCOL = 'A%' OR DBCOL LIKE 'B%' OR DBCOL LIKE 'C%'	Matches all values that equal {A%} or that start with {B} or {C}.	Using the equal sign results in the % character being automatically inserted, but the % does not act like a wildcard.
‘A’,B, C	DBCOL = 'A' OR DBCOL LIKE 'B%' OR DBCOL LIKE 'C%'	Matches all values that equal {A} or that start with {B} or {C}.	
=A,=B,=C	DBCOL = 'A%' OR DBCOL = 'B%' OR DBCOL = 'C%'	Matches all values that equal {A%} or {B%} or {C%}.	Use = if you want to search for strings containing %.
=’A’,=’B’,=’C’	DBCOL = 'A' OR DBCOL = 'B' OR DBCOL = 'C'	Matches all values that equal {A} or {B} or {C}.	
=A-P	DBCOL BETWEEN 'A' AND 'P'	Same as: A-P	Use A-P instead.
=’A-P’	DBCOL BETWEEN 'A' AND 'P"	SQL syntax error.	Do not use.
ABC-DEF	DBCOL BETWEEN 'ABC' AND 'DEF'	Matches all values that are alphabetically between {ABC} and {DEF}.	The range is inclusive.
‘A’,B,C-F	DBCOL = 'A' OR DBCOL LIKE 'B%' OR DBCOL BETWEEN 'C' AND 'F'	Matches all values that are equal to {A} or start with {B} or are alphabetically between {C} and {F}.	The list, equality and range features can be combined.
A-C,X-Z	DBCOL BETWEEN 'A' AND 'C' OR DBCOL BETWEEN 'X' AND 'Z'	Matches all values that are alphabetically between {A} and {C} or {X} and {Z}.	
3/1/02	DBCOL LIKE '3/1/02%'	Matches all values that start with {3/1/02}.	Date values used with character parameters are handled differently than when used with date parameters.
‘3/1/02’	DBCOL = '3/1/02'	Matches all values that equal {3/1/02}.	Same as above.

Adhoc Parameter Processing for a String Type Database Column (cont.)

Parameter Expression Entered	Resulting SQL Where Clause Expression	Data Filter Resulting from the Parameter Value	Comments
<ABC	DBCOL < 'ABC%'	Matches all values alphabetically less than {ABC%}.	The added % may cause matching problems.
<’ABC’	DBCOL < 'ABC'	Matches all values alphabetically less than {ABC}.	Placing the text in single quotes solves the added % problem.
<=’ABC’	DBCOL <= 'ABC'	Matches all values alphabetically less than or equal to {ABC}.	
>’ABC’	DBCOL > 'ABC'	Matches all values alphabetically greater than {ABC}.	
>=’ABC’	DBCOL >= 'ABC'	Matches all values alphabetically greater than or equal to {ABC}.	
<’ABC’,>’XYZ’	DBCOL < 'ABC' OR DBCOL > 'XYZ'	Matches all values alphabetically less than {ABC} or greater than {XYZ}.	
<=’ABC’,DEF-XYZ	DBCOL <= 'ABC' OR DBCOL BETWEEN 'DEF' AND 'XYZ'	Matches all values alphabetically less than or equal to {ABC} or alphabetically between {DEF} and {XYZ}.	

Adhoc Parameter Processing for a Numeric Type Database Column

Parameter Expression Entered	Resulting SQL Where Clause Expression	Comments
10	DBCOL = 10	Single value.
=10	DBCOL = 10	Equal sign is not required.
10,20,30	DBCOL = 10 OR DBCOL = 20 OR DBCOL = 30	List of values.
10-30	DBCOL BETWEEN 10 AND 30	Range of values, inclusive.
10-30,100-120	DBCOL BETWEEN 10 AND 30 OR DBCOL BETWEEN 100 AND 120	List of value ranges.
<10	DBCOL < 10	Less than.
<=10	DBCOL <= 10	Less than or equal to.
>10	DBCOL > 10	Greater than.
>=10	DBCOL >= 10	Greater than or equal to.
<10,>=100	DBCOL < 10 OR DBCOL >= 100	List of comparisons.
<10,100-500	DBCOL < 10 OR DBCOL BETWEEN 100 AND 500	Comparison combined with a range.
<10,100-500,1000	DBCOL < 10 OR DBCOL BETWEEN 100 AND 500 OR DBCOL = 1000	Compound list.
-300,-200	DBCOL = -300 OR DBCOL = -200	List using negative values.
\-300,\-200	DBCOL = -300 OR DBCOL = -200	\ does not have any effect here.
\-300-\-200	n/a	Does not work.
>=-300,<=-200	DBCOL >= -300 OR DBCOL <= -200	Negative values can be problematic when using the dash notation for ranges. Use inequality expressions instead.
3/1/02	DBCOL = 3	Everything after first / is ignored.

Adhoc Parameter Processing for a Date Type Database Column

NOTE: these parameters were used with an ODBC database connection which is why the dates in the SQL expressions are expressed in {d ‘yyyy-mm-dd’} format.

Parameter Expression Entered	Resulting SQL Where Clause Expression	Comments
3/1/02	DBCOL = {d '2002-03-01'}	Single value; do not have to include leading zeroes.
03/01/02	DBCOL = {d '2002-03-01'}	Leading zeroes are OK.
3/1/02,3/10/02	DBCOL = {d '2002-03-01'} OR DBCOL = {d '2002-03-10'}	List of dates.
3/1/02-3/10/02	DBCOL BETWEEN {d '2002-03-01'} AND {d '2002-03-10'}	Range of dates, inclusive.
3/1/02,3/10/02-3/20/02	DBCOL = {d '2002-03-01'} OR DBCOL BETWEEN {d '2002-03-10'} AND {d '2002-03-20'}	Single value and a list.
=3/1/02	DBCOL = {d '2002-03-01'}	Equal sign is superfluous.
<3/1/02	DBCOL < {d '2002-03-01'}	Less than.
<=3/1/02	DBCOL <= {d '2002-03-01'}	Less than or equal to.
>3/10/02	DBCOL > {d '2002-03-10'}	Greater than.
>=3/10/02	DBCOL >= {d '2002-03-10'}	Greater than or equal to.
<3/1/02,>=3/10/02,3/20/02-3/24/02	DBCOL < {d '2002-03-01'} OR DBCOL >= {d '2002-03-10'} OR DBCOL BETWEEN {d '2002-03-20'} AND {d '2002-03-24'}	Ranges can be combined with comparison expressions.
3-1-02	n/a	Invalid date format.
2002-3-1	n/a	Invalid date format.

Adhoc Parameter Value and Expression Syntax Rules Summary

This section summarizes the adhoc parameter value and expression syntax rules that were discovered based on the experiments outlined in the preceding tables.

String Parameters

1) If you enter a string value that is not enclosed in single quote marks, Actuate assumes you want a wildcard character (%) at the end of the string so that the query matches any strings that start with the text you entered. This helps deal with database columns that are defined as char instead of varchar and are thus right padded with spaces. The user does not have to enter the trailing spaces to get a match.
2) You can explicitly include wildcard characters (%) within the text you provide and Actuate will use the wildcards you provide and will not automatically place one at the end of the text you entered.
3) If you want to search for an exact text value, and do not want wildcard handling, then enclose the text you are searching for in single quote marks, for example 'ABC'. If you specify a value that is not contained within single quotes, then Actuate will make it into a wildcard, unless the text you enter starts with =.
4) The asterisk is not treated as a wild card, but instead as a literal character for which you are searching. For example, A*B only matches the text "A*B", and not "ACB" or "ACDB".
5) Double quotation marks are treated as literal characters, and do not have any special meaning; thus entering a double quote mark searches for the character ".
6) If you want to search for more than one item, then you can separate items with commas. This works for wildcard and literal searches. For example: A,B,C or A%,B%,C% are wildcard searches and search for any string that starts with A or B or C. 'A','B','C' does a literal search for the exact strings A or B or C. 'A', B, C% is also acceptable syntax.
7) If you want to search for a range of items, then you can separate them with a dash (-). For example, ABC-DEF matches all values that are alphabetically between the strings "ABC" and "DEF", inclusive. When you use the dash syntax, wildcard handling is disabled for the strings you specify.
8) Backslash acts as an escape character that removes the special meaning for some characters. For example A\-P searches for all items starting with "A-P" instead of looking for items between "A" and "P".
9) The escape character does not work with %. Therefore, you cannot search for strings containing the % character by specifying \%.
10) The = character can be used to search for strings containing literal % characters. For example, =A%B searches for strings that equal the text "A%B", and does not treat the % as a wildcard. Basically, if the text you enter starts with =, then Actuate treats it as a literal string to search for and not as a wildcard expression. However, unless you place any text after the = between single quotes, or you place a % somewhere in the text, Actuate will place a % at the end of the string.
11) The comma (,) takes precedence over the single quote mark ('). A comma basically becomes "OR" unless it is escaped with the \ character.
12) The various syntaxes can be combined. For example, a list of ranges can be specified as: A-C, G-I, X-Z.
13) The inequality operators (<, <=, >, >=) work as expected so long as you place the string values you are looking for within single quotes. For example, <'ABC',>'XYZ'.

Numeric and Date Parameters

1) These adhoc parameters are straightforward and behave the way you would expect since there is no default wildcard behavior as there is with strings.
2) These parameters support the comma (,) to specify a list of values or ranges.
3) These parameters support the dash (-) to specify a range of values.
4) These parameters support the equality and inequality operators.
5) With numeric parameters, negative signs can be confused as range dashes, so use the inequality operators to avoid problems. For example, instead of -300--200, use >=-300,<=-200.
6) The only date formats permitted for date type adhoc parameters are mm/dd/yy and mm/dd/yyyy. However, note that 3/1/02 can be used instead of 03/01/02. Please note that these comments apply to US locales only, I have not tested this with non-US installations or localized versions of Actuate products.