

Using the Templates Feature

LiveReport Extensions 3.0

March 2008

Contents

Purpose	1
Introduction	1
Setting up a Template	2
Using Templates	4
Dynamic filtering with the Auto-Where feature	4
Using a URL Parameter to Control Template Insertion	5
Using User Input Values to Control Template Insertion.....	9
Making Templates Dependent on Other Templates	13
Using the Auto Comma feature	14
Additional Template Features	15
Using Templates within Templates	15
Using the Template View Icon	15

Purpose

This document provides an in-depth description of the new “Templates” feature introduced in LiveReport Extensions versions 3.0 and above. This document supplements the documentation found in the online help for this module as well as the user and administration guide.

The Introduction and Setting up a Template sections which follow summarize and review some of the main features of templates in LiveReport Extensions. Some readers who have already read the User and Administration Guide may wish to go straight to the [Using Templates](#) section of this document where many examples provide effective illustrations of how to use Templates.

Introduction

LiveReport Extensions 3.0 introduces a brand new concept to the traditional LiveReports object, called “templates”. Templates provide the ability to setup pre-defined blocks of SQL code that can be included or excluded at run time based on pre-defined conditions. This concept is particularly useful when setting up ad-hoc reporting type applications. The ability to dynamically add or remove data filters or to dynamically change the returned data set is very important in ad-hoc reporting. For customers using LiveReports without Report Extensions the problem is that only data values can be passed to the LiveReport via the URL. There is no way to add additional filter clauses (such as AND NAME = 'test') once the LiveReport has been created. If all of the possible filter clauses are added when the LiveReport is first created, then the SQL query may fail unless valid values are passed for each of these clauses. Normally this would require wild cards to be passed for any of these clauses that were not required in order to ensure the SQL query returned the correct number of results.

As an initial step to deal with this problem, the InsertStr type parameter added in previous versions of Report Extensions helped with this requirement by allowing variable amounts of SQL to be inserted into the SQL source. The main problem with this approach is that in some cases large amounts of SQL are being passed via the URL. Typically in this approach an entire SQL clause (e.g. AND Name = 'test') is passed to the LiveReport.

The template feature allows that any optional SQL clauses can be predefined in the actual LiveReport. Each template is setup so that only the parts of the data that need to be variable (e.g. = 'test' in the previous example) are passed in the URL. Each template can be setup so that a predetermined condition must be met in order for the template to be included. These conditions are described in depth within this document but in general template inclusion is managed according to whether certain variables are set to pre-determined “flag values” or whether a predefined parameter is set in the URL.

For example, using the simple example above, a template could look like this:


AND Name = %1

Assuming that %1 is a user input type parameter of type String, the template could be setup so it is only included if %1 is not set to a blank string. This would ensure that the text “AND Name = %1” is only included in the SQL query if %1 were passed with a non blank value in the URL.

Setting up a Template

A template in LiveReport extensions looks like this:

# Where/And	SQL Source	Auto Comma	Include IF	Options
~1		<input type="checkbox"/>	Not all inputs set to flag	Flag Value

Additional templates can be created using the  icon. Templates can be removed by clearing any text in the SQL source box and saving the LiveReport.

The fields used in a template are described in the online help but are also described here for completeness. When described in general terms, Templates can seem quite abstract. They are best understood by reviewing real examples. Readers are advised to review this section briefly and then move on to the section called Using Templates where many examples illustrate how to use these fields.


- **# (template number)** - This field shows the reference which is used to insert a template in the main SQL query. The tilde symbol (~) is used with the template number to indicate where the template is to be inserted. E.g. ~1 specifies where template 1 should be inserted (assuming that various conditions are met). Note, if no template reference is found in the SQL query then the template will be inserted at the end of the SQL
- **Auto-Where** - Checking this box provides the capability to automatically add the correct filter word (either WHERE or AND or OR) to the beginning of a template. This is useful when there is no WHERE clause in the main SQL and the order of template insertion is not known. This option will always use WHERE for the first template being added and either AND or OR (depending on which of these words is in the original template source) for each subsequent template. If the template source does not contain one of these words, then the default will be either WHERE or AND. The simplest way to use this feature is to place either OR or AND as the first word in the template source and then the feature will replace this word with WHERE if the template becomes the first one to be inserted into the SQL
- **SQL Source** - The content of the template which will optionally be added to the SQL is defined in this field. This source can include normal parameters in the form: %1, %2, as well as markers in the form #1, #2 which are used in conjunction with the URLParameter, **Include IF** condition (described below)
- **Auto Comma** - Checking this box enables a feature that will ensure that any comma separated variables (e.g. %1, %2, %3) have the correct number of commas in place after the variables have been resolved. For example if %2 resolves to a blank string the list would automatically be converted to remove any empty commas. Thus **12,,24** would be converted to **12,24**. Note: the original list must be syntactically correct
- **Include IF** - The conditions for template inclusion are selected using this option. For the purposes of the descriptions below, the term "User Input" refers to any parameters that have been passed to the LiveReport in the form &inputlabel1=somedata. Many of the template conditions are dependent on the contents of one or more of these user inputs and these user inputs can be used flexibly within the template source. The available conditions are:
 - **Mandatory (no condition)** - The template will always be inserted when this option is selected. This option is useful for breaking the SQL down into smaller pieces.

- **URL Parameter** - This option specifies that this template will be inserted if a named parameter is found in the URL. When this option is selected a **Parm Name** text field is revealed. This is used to specify the name of the URL parameter which will cause the template to be included. This parameter in the URL can also be used to specify which user inputs will be inserted into the template when markers in the form #1 have been used in the template source. These markers specify where in the source each user input will be inserted. For example, a template could be enabled (included) by a URL parameter in the form: **&addtemplate=U2,U3**. This specifies a list of user inputs which can be used by this template. Each item in this list represents a user input. For example, U2 = USERINPUT 2. The user input specified by the first item in the list is inserted in place of marker 1 (#1) so in this example, #1 is replaced by the &inputlabel2 value. The second item references U3 so the &inputlabel3 value is substituted in place of marker 2 (#2), etc. Several examples of this abstraction are included later in this document. **Note:** The URL parameter must specify at least as many user inputs as are included in the template source in order for the template to be included.
- **No Inputs set to flag** - This option specifies that the template's inclusion is dependent on the settings for any user inputs which are used in the template source using the normal parameter form (e.g. %1). When this option is selected a **Flag Value** text field is revealed. This is used to specify what value will be used as a flag in passed user inputs. The most commonly used value will be a blank string but other likely examples are 'none' and 'all'. This option specifies that the template will only be included if none of the user inputs have been set to the specified flag. For example, if a template is using two input parameters and one of them has been set to a blank string then the template will be excluded.
- **Not all inputs set to flag** - This option is very similar to the previous option; however, with this option set, the template will be included, unless all User Input values have been set to the specified flag. For example, if the Flag Value is empty (considered a blank string) and there are two user inputs used in the template, then the template will be included unless both inputs are set to a blank string.
- **Template condition true** - With this option set, the template will be included based on whether another template evaluates to true or not. The Template # option is used to specify which preceding template's condition should be used. For example, if the template # is set to 2, then the template with this option set will only be included if the template 2 **Include IF** condition evaluates to true.
- **Template condition false** - With this option set, the template will be included based on whether another template evaluates to false or not. The Template # option is used to specify which preceding template's condition should be used. For example, if the template # is set to 2, then this template will only be included if template 2 evaluates to false. This option is useful when you have two templates that are mutually exclusive.

Using Templates

This section provides some simple examples of how templates can be used to solve reporting problems. Examples and detailed explanations are provided for each feature in this section.

Notes:

- In all the following examples simple “GET” URLs have been used; however, the equivalent parameters could be formed using form POST syntax.
- For any of these template setups the view template feature can be used to help explain how the template will behave. This feature is executed using the:  icon.

Dynamic filtering with the Auto-Where feature

Each template includes a check box called “Auto-Where”. This feature is provided to make it easier to use templates for dynamically adding multiple filter clauses using either WHERE, AND or OR. Although templates can be used to store any part of a SQL query, one of the most obvious uses of templates is to allow a varying number of these filter clauses to be added to the original query. For example, here are some examples of SQL statements that are used for filtering

- WHERE Name = 'test'
- AND DATAID > 2000
- OR Name LIKE '%new%'

Unless the main SQL source includes the first (WHERE) filter clause, the first template to be added to the source needs to be preceded by WHERE, instead of AND or OR. The problem is that when the template source is written, it is not necessarily possible to determine which template will be the first one as the conditions for inclusion/exclusion are determined when the report is being run. In particular, when a single template is being used repeatedly to provide filter clauses (this is possible with the URL parameter condition) we need to be able to specify that the first copy of the template must be preceded by WHERE.

The Auto-Where feature deals with all of these scenarios and is controlled subject to the following rules:

- Any templates that do not have Auto-Where selected will be ignored by the Auto-Where feature, even if one of the reserved words WHERE, AND, OR have been included in the template source
- For each template that has this feature enabled, the first word in the source is checked and treated as follows:
 - If the first word is WHERE, AND or OR and this is the first template to be included, then the word is converted to or left as WHERE in the template source for insertion.
 - If neither WHERE, AND or OR are found and this is the first template to be included, then the word WHERE is added to the front of the template source before insertion.
 - If the first word is AND or OR and this is NOT the first template to be included, then the word is left as is in the template source for insertion.
 - If the first word is WHERE and this is NOT the first template to be included, then WHERE is replaced with AND before the template source is inserted.
 - If none of these words is found and this is NOT the first template to be included then the word AND is added to the front of the template source before insertion.

This feature is illustrated in many of the examples in this document.

Using a URL Parameter to Control Template Insertion

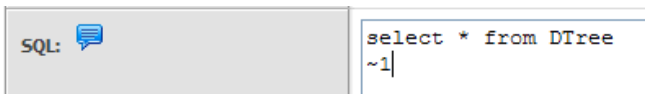
The URL Parameter condition is used to specify the name of a parameter in the URL that will cause the template to be inserted. There are a couple of ways that this can be used. In the simplest example, the URL parameter only needs to be set to true. For example, **&someparam=true**. The URL Parameter option for inserting templates can also be used to specify which of any input parameters in the URL should be used in the template. This feature can be useful where a few inputs might be used to supply data to more than one template and or the same template may be re-used with different inputs inserted. Note that in this usage, even though the word **true** is not being passed through the parameter, the template will be included provided the number of user inputs specified in the URL match with any required parameters in the template source. This is explained more fully in the following examples.

Example 1. Adding an order clause to the SQL query

Description:

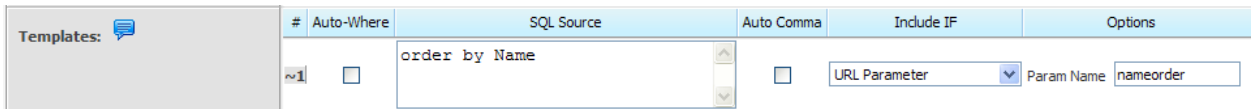
In this simple example, we have defined a template with a fixed ORDER BY clause that will be inserted if the parm **&nameorder=true** is included in the URL. The **~1** marker in the SQL source specifies that if the template is added, it should be added after the main query. Note, if **&nameorder** is not included in the URL, or the URL includes **&nameorder=false**, then the **~1** marker is replaced with an empty string (nothing).

Main SQL Source:



```
SQL: select * from DTree
~1
```

Template Setup:

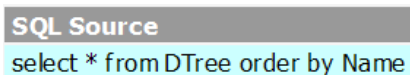


#	Auto-Where	SQL Source	Auto Comma	Include IF	Options
~1	<input type="checkbox"/>	order by Name	<input type="checkbox"/>	<input type="checkbox"/>	URL Parameter Param Name: nameorder

Sample URL:

```
.../livelink.exe?func=ll&objAction=RunReport&objId=32666&nameorder=true...
```

Resulting SQL:




```
SQL Source
select * from DTree order by Name
```

Example 2. Re-using a template with a variable input

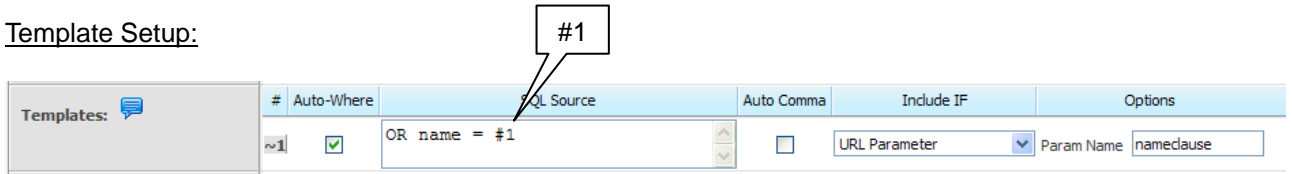
Description:

In this example, the same template will be added repeatedly using a different user input parameter each time. This has the effect of constructing multiple OR clauses. The use of the Auto-Where feature allows the template feature to automatically add WHERE instead of OR when the first template is inserted, and then to use OR for each subsequent template that is inserted.

Main SQL Source:

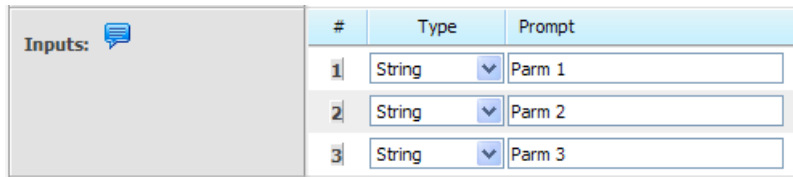
```
SQL: 
select * from DTREE
~1
```

Template Setup:



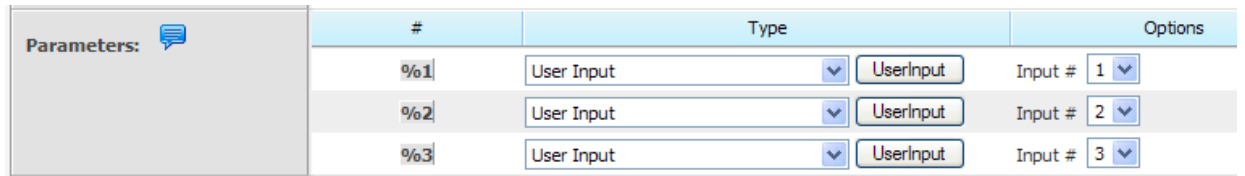
#	Auto-Where	SQL Source	Auto Comma	Include IF	Options
~1	<input checked="" type="checkbox"/>	OR name = #1	<input type="checkbox"/>	URL Parameter	Param Name nameclause

Inputs (for illustration – different types could be used):



#	Type	Prompt
1	String	Parm 1
2	String	Parm 2
3	String	Parm 3

Parameters:



#	Type	Options
%1	User Input	Input # 1
%2	User Input	Input # 2
%3	User Input	Input # 3

Sample URL (Only Parameters shown):

... [&nameclause=U1](#) [&nameclause=U2](#) [&nameclause=U3](#)
[&inputlabel1=cat](#) [&inputlabel2=dog](#) [&inputlabel3=frog](#)

Resulting SQL:

```
SQL Source
select * from DTREE WHERE name = 'cat' OR name = 'dog' OR name = 'frog'
```

Example 3. Re-using a template with multiple variable inputs

Description:

This example is similar to the previous one but the template specifies two variables to be inserted. This means that if the URL parameter specifies less than two user inputs (as in the previous example) then the template will not be included. This example shows two different templates, with slightly different purposes. One supports a direct match using equals (=) and the other supports a partial match using LIKE. Either one can be selected using the appropriate Parameter Names (either **likecompare** or **equalscompare**). These two parameters are passed along with a list of which User Inputs to use in the SQL, e.g. U1,U2 means, use inputlabel1 and inputlabel2. A graphic explanation of this example follows.

Main SQL Source:

```
SQL:
select * from DTREE
~1
~2
```

Template Setup:

#	Auto-Where	SQL Source	Auto Comma	Include IF	Options
~1	<input checked="" type="checkbox"/>	AND #1 LIKE #2	<input type="checkbox"/>	<input type="checkbox"/>	URL Parameter (dropdown) Param Name likecompare
~2	<input checked="" type="checkbox"/>	AND #1 = #2	<input type="checkbox"/>	<input type="checkbox"/>	URL Parameter (dropdown) Param Name equalscompare

Inputs (for illustration – parameters are not shown):

#	Type	Prompt
1	InsertString	fieldname
2	String	fieldvalue
3	InsertString	fieldname2
4	String	fieldvalue2

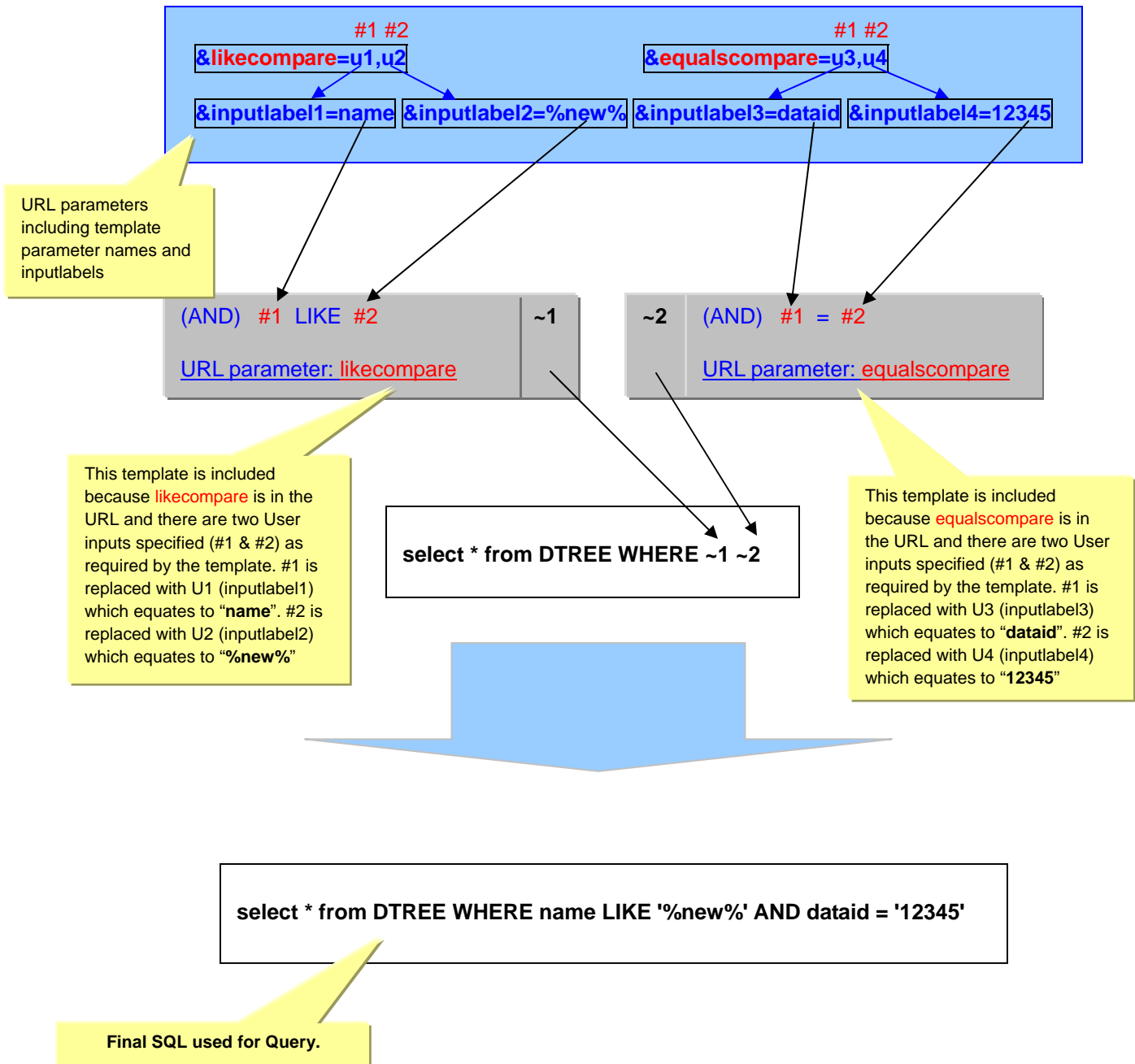
Sample URL (Only Parameters shown):

**&likecompare=u1,u2&equalscompare=u3,u4
&inputLabel1=name&inputLabel2=%new%&inputLabel3=dataid&inputLabel4=12345**

Resulting SQL:

```
SQL Source
select * from DTREE WHERE name LIKE '%new%' AND dataid = '12345'
```

Example 3: Graphic Explanation:



Using User Input Values to Control Template Insertion

Besides using unique URL parameters to control the insertion of templates, it is possible simply to specify that a template will be inserted (or not) based on the value of one or more User Input parameters that are used in the template. Consider the following template:

NAME = %1 AND DATAID > %2

When the template is setup we can specify a flag value that will be used to determine whether the template is included or not. For example, a value of empty string could be set.

We could either specify that if one of the flags is set to empty string then the template is not used, or we could specify that the template will be excluded if both of the parameters are set to empty string.


Most commonly the specified flag will be an empty string. This allows templates to be excluded if one or more of the parameters used are not used.

Example 4. Excluding a template if one of the parameters is empty

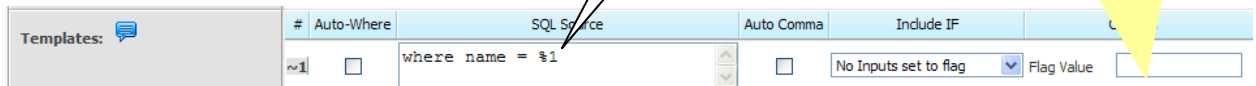
Description:

For this example, two different URLs are shown to illustrate how different input settings affect the SQL source that is executed. The template is setup to use a single User input (%1). The **Include IF** condition is set to **No Inputs set to flag** which means that if the user input is set to the specified Flag Value then the template will NOT be included. The Flag Value in this case is an empty string so in specific terms, if &inputlabel1 is not included in the URL or &inputlabel1= is set to nothing (i.e. an empty string) then the template will not be included. If inputlabel1 is set to anything else then the template will be included. In URL 1 a value has been specified for &inputlabel1 so the template is included. In URL 2 &inputlabel1 has been left blank so the template is not included. (Note, the parameters section of this LiveReport has not been shown but %1 has been set to coincide with User Input 1.)

Main SQL Source:

```
SQL: 
select * from DTREE
~1
```

Template Setup:



#	Auto-Where	SQL Source	Auto Comma	Include IF
~1	<input type="checkbox"/>	where name = %1	<input type="checkbox"/>	No Inputs set to flag

Flag Value:

Inputs (for illustration – parameters are not shown):

#	Type	Prompt
1	String	name

Sample URL 1 (Only Parameters shown):

(Note, with this type of condition it is not necessary to include any additional parameters as the template's inclusion is purely based on the contents of normal User Input parameters.)

&inputLabel1=New WebReport&nextURL=...

Resulting SQL:

SQL Source
 select * from DTREE where name = 'New WebReport'

Sample URL 2:

Note, when no text is specified between the equals (=) sign and the next parameter start (&) or the end of the URL then the parameter is set to an empty string

&inputLabel1=&nextURL=...

Resulting SQL:


SQL Source
 select * from DTREE

Example 5. Excluding a template if any of multiple parameters are set to the flag

Description:

This example is similar to the previous one except that there are multiple User Input parameters in this template. In this case, we want the template to be excluded if any of these User Inputs are set to the flag ("null" in this case). To achieve this we again use the **Include IF** condition of **No inputs set to flag** but for this example we are using the text characters: **null** as a flag. In this case if either &inputlabel1, &inputlabel2 or &inputlabel3 are set to the text string **null** then the template will not be included. In our sample URL &inputlabel1 is set to: **null** so this template is not included in the resulting SQL.

Main SQL Source:

```
SQL: 
select * from sampleTable
~1
```

Template Setup:

#	Auto-Where	SQL Source	Auto Comma	Include IF	Options
~1	<input type="checkbox"/>	<pre>where color LIKE %1 and type LIKE %2 and subtype LIKE %3</pre>	<input type="checkbox"/>	No inputs set to flag	Flag Value: null

%1
%2
%3

Sample URL:

[&inputLabel1=null&inputLabel2=widget&inputLabel3=grommet&nextURL=...](#)

Resulting SQL:

```
SQL Source
select * from sampleTable
```

Example 6. Including a template if any of the parameters are not empty

Description:

This example demonstrates one of the lesser used conditions. In this simplistic example, we have a situation where the number of selectable columns is variable. If at least one of these User Input parameters is not empty, then we want to include the template. By using the **Include IF** condition: **Not all inputs set to flag** we specify that the template will be included unless all of the User Input parameters are set to the flag value (which is an empty string in this example). Note that we are also using the Auto Comma feature in this example to ensure that the number of commas in the select list is syntactically correct. This feature is explained more fully in the section: Using the Auto Comma feature. In this example we also use a second template that will only be inserted if the first template is not included. This **Include IF** condition is explained further in the next section.

Main SQL Source:

```
SQL:
select ~1~2
from sampleTable
```

Template Setup:

Templates:	#	Auto-Where	SQL Source	Auto Comma	Include IF	Options
	~1	<input type="checkbox"/>	§1, §2, §3	<input checked="" type="checkbox"/>	Not all inputs set to flag	Flag Value <input type="text"/>
	~2	<input type="checkbox"/>	*	<input checked="" type="checkbox"/>	Template condition false	Template # <input type="text" value="1"/>

Inputs (for illustration):

Inputs:	#	Type	Prompt
	1	String	name <input type="text"/>
	2	String	type <input type="text"/>
	3	String	subtype <input type="text"/>

Sample URL:

[&inputLabel1=Parts&inputLabel2=&inputLabel3=grommet&nextURL=...](#)

Resulting SQL:

```
SQL Source
select Parts,grommet from sampleTable
```

Making Templates Dependent on Other Templates

There are two **Include IF** conditions that are simply dependent on other templates. These two conditions are: **Template Condition True** and **Template Condition False**. In each case a Template # field is provided in order to input the number of whichever template's condition will be used to control this one.

Template Condition True – This condition is useful when pieces of SQL need to be inserted in different parts of the main SQL source, based on the same condition. One of the SQL pieces might have the actual conditions defined for inclusion while the other one(s) would just use Template Condition True to reference the other template.

Template Condition False – This condition is particularly useful whenever two templates are mutually exclusive. Using this condition saves having to work out the appropriate set of conditions for one template only to be included when another one is being excluded. For an example of this condition see Example 6 above.

Example 7. Adding two templates based on a single condition

Description:

This example shows two templates where one template is conditional on &inputlabel1 not being set to an empty string, and the second template is dependent on the first template. In this example we show a situation where the developer has decided that if the subtype has not been included as a filter clause, then it will not be required as a column. If &inputlabel1 is not a blank string then both the Where template and the column Select template are included.

Main SQL Source:

```
SQL:
select name~2
from sampleTable
~1
```

Template Setup:

#	Auto-Where	SQL Source	Auto Comma	Include IF	Options
~1	<input type="checkbox"/>	where subtype = %1	<input type="checkbox"/>	No Inputs set to flag	Flag Value <input type="text"/>
~2	<input type="checkbox"/>	, subtype	<input type="checkbox"/>	Template condition true	Template # <input type="text" value="1"/>

Inputs (for illustration):

#	Type	Prompt
1	String	Subtype

Sample URL:

```
&inputLabel1=grommet&nextURL=...
```

Resulting SQL:

SQL Source

```
select name , subtype from sampleTable where subtype = 'grommet'
```

Using the Auto Comma feature

The Auto Comma feature is designed to help with the building of comma delimited lists using User Input parameters. This feature is designed to help with the following scenario. Imagine that a template looks like this:

```
%1, %2, %3
```

When these User Input parameters are resolved, if one of them resolves to an empty string, e.g. &inputlabel2, then the output could end up looking like: **apple,,pear**. If the **Auto Comma** feature is enabled then the output would be corrected to: **apple,pear**.

Note: this feature does not currently support comma resolution when another template is used to insert values into a comma list.


Additional Template Features

This section describes a few additional features of templates that warrant further description.

Using Templates within Templates

Besides specifying templates for inclusion in the main SQL field, it is also possible to use templates within other templates. The syntax is the same, e.g. ~2; however, a template can only use a template with a lower number. For example, template ~2 can include a reference to template ~1 **but** not to itself or template ~3, or any number above 2.

Using the Template View Icon

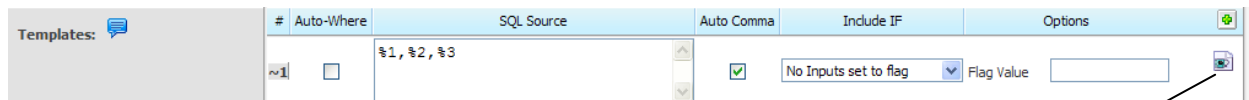
Each template includes a special icon:  which can be used to View Template Behavior. Clicking on this icon opens a window that contains a series of statements about the template in question. These statements attempt to clarify for each template how it will behave based on the selected condition and the source that has been specified.

Example 8. Viewing behavior for the “Not all inputs set to flag” condition

Description:

The conditions: **Not all inputs set to flag** and **No Inputs set to flag** both include a table of conditions which is designed to help understand how User Input parameters in the template will affect the inclusion or exclusion of the template. In the example below, the table shows that if all User Input parameters are set to an empty string, the template is excluded but if any User Input parameter is not set to an empty string then the template will be included.

Template:



Explanation Window:

The combination of fields in this row are interpreted as follows:

- The parameters: %1,%2,%3 were found in the SQL source field for this template
- The Condition Type is set to "No inputs set to flag". This means that: These parameters will be tested to see if the flag: (An empty string) has been passed as a value.
- If none of the parameters in the SQL source field are set to An empty string then this template will be inserted anywhere in the SQL where the ~1 symbol is found
- Conversely, if ANY of the parameters in the source field (%1,%2,%3) are set to the flag value (An empty string) then template ~1 will not be included in the SQL source

This table indicates the expected results for different settings of parameters:

Parm 1	Parm 2 etc.	Result
An empty string	An empty string	EXCLUDE
(any other value)	An empty string	EXCLUDE
(any other value)	(any other value)	INCLUDE

Copyright 2003-2008 Resonate KT Limited

www.resonatekt.com

The information in this document is subject to change without notice. All rights reserved.

Open Text Corporation is the owner of the trademarks Open Text, 'Further Faster', HyperInnovation, Accelerating Innovation, Livelink, myLivelink, BASIS, Techlib, OnTime among others. This list is not exhaustive. All other products or company names are used for identification purposes only, and are trademarks of their respective owners.

