

User and Administration Guide

LiveReport Extensions 3.0

March 2008

Contents


1	Purpose	1
2	Introduction	1
3	Glossary of Terms	2
4	Installation	3
	Unzip Installation Files.....	3
	Install the Module.....	3
5	Writing LiveReports for External Databases	4
	Requirements	4
	Creating Connections to External Databases	4
	Creating a LiveReport for an External Database.....	4
6	Enhanced Inputs and Substitution Parameters	5
	Overview.....	5
	Insert String	6
	Introduction	6
	Usage.....	6
	Configuration.....	7
	Creating Extra Inputs or Substitution Parameters.....	8
7	Turning Off Transaction / Cursor Modes	9
	Use Transaction mode.....	9
	Use Cursor Fetch mode	9
8	SQL Templates	10
9	Enhanced SQL editing	12
10	View SQL Source	13
11	View Query	14

1 Purpose

This document provides information on the installation, administration and use of the LiveReport Extensions module from Open Text Corporation. It is primarily intended for Livelink system administrators and LiveReport developers.

2 Introduction

LiveReport Extensions provides enhancements to standard Livelink LiveReports including

- The ability to report on external databases
- Greatly improved User Input and Substitution Parameter handling with virtually unlimited numbers of inputs and parameters available
- Ability to turn off transaction mode when running queries to help with lock related issues
- Ability to turn of “Fetch Cursor” mode to improve query performance
- The ability to create multiple pieces of conditional SQL code (called templates) for inclusion in the main SQL source based on defined conditions
- The ability to view the constructed SQL queries (including all parameter and template substitution) prior to running the query
- The ability to manage appropriate logic (WHERE,AND,OR) for optional SQL clauses
- The ability to manage comma separated lists when the number of items is not known at development time
- Enhanced SQL editing in the LiveReport complete with Undo and Redo buttons
- Useful Return buttons from Autoreport view
- Additional Save options to allow a variety of Save/Execute variations
- Ability to provide a read only view of a LiveReport setup
- User interface help call-outs when hovering over  symbols providing hints and tips on creating LiveReports.

These features are delivered as enhancements to LiveReports and can be used with or without Livelink WebReports.

LiveReport Extensions 3.0.1 supports Livelink 9.5, Livelink 9.6. and Livelink 9.7

3 Glossary of Terms

With respect to this document, the following terms are defined as such:

End Users: This refers to Livelink consumers who use the output of reports, but do not develop them themselves.

Developers: Any users who may be involved in creating new WebReports or editing existing ones (to change the look, feel and behavior). For any particular WebReport, Developers fall into two categories: Those who can edit or add a version to the reportview, and those who can only fetch or download the reportview. The latter case is useful in terms of enabling Developers to reuse an existing reportview in a new WebReport, without allowing them to change the original.

LiveReport: A Livelink item that allows users to query (and potentially modify) the Livelink database. The SQL statement in a LiveReport communicates directly with the database. The Livelink Administrator may create new LiveReports, and grant other users permission to run them. Several LiveReports appear on the LiveReports tab on the Reports page of each user's Personal Workspace.

System Administrators: Besides the role of managing WebReport permissions and access requirements, these users may also have access to the LiveReports used to supply data to the WebReports. The various tasks and functions described in this document are targeted to this role.

WebReports: The module that enhances Livelink reporting with a new WebReport node that offers comprehensive control of report formats, scheduling, and data exporting.

4 Installation

Unzip Installation Files

- From the ZIP file you have been provided, extract the directory called `reportext_3_0_x` into the Livelink staging directory located at `<livelink install directory>/staging`.

Install the Module

- Open Livelink's administration page in your Web browser and login as the Livelink Administrator user.
- Under the **Module Administration** section, click the **Install Modules** link. The URL to the administration page is listed below:

`http://<domain name>/<livelink service name>/livelink.exe?func=admin.index`

- Check the box beside **Livelink LiveReport Extensions** and click **Install**.
- Restart the Livelink service when prompted and click **Continue**.

5 Writing LiveReports for External Databases

Requirements

The optional module WebForms must also be installed in your Livelink instance in order to connect to external databases.





Creating Connections to External Databases

To report on an external database, a connection must be established in the WebForms Database Lookup function in the Livelink Administration pages. These connections can only be setup by a Livelink Administrator. Navigate to the **Web Forms Database Lookup Administration** section and click on **Add Database Connection Information**.

Complete the form for each external database required.

Creating a LiveReport for an External Database

When adding or editing a LiveReport a new option is presented (if WebForms is installed) providing a choice of database on which the report will run. The choices will reflect connections established in the WebForms Database Lookup function. The default choice is the database for the Livelink instance hosting the LiveReport.

Record Limit: 	<input type="text" value="1"/>
Database Selection: 	Current Livelink Instance 
Show Prompts: 	<input type="checkbox"/>

Note that when reporting on an external database, some LiveReport options such as **Allow Database Modification**, **Click-thru Sub-Reports** and **Display Columns**, are not relevant or supported. An alert to this effect is shown to the LiveReport Developer and these options are removed from the Add / Edit LiveReport form. Furthermore the **Report Format** choice of "LiveReport" is not supported for external databases.

6 Enhanced Inputs and Substitution Parameters

Overview

Parameters can be passed to LiveReports with user inputs. The inputs are defined in the LiveReport editor by selecting the type of input and providing corresponding prompt text. Alternatively, the prompting stage can be avoided if all the inputs are provided in the URL that executes the LiveReport. These are defined in URL parameters using the form: **inputlabelx** where x is a number. For example:

```
http://intranet/livelink/livelink.exe?func=ll&objId=1234
&objAction=RunReport&viewType=1&inputlabel1=treadstone
```

In this example, the string “treadstone” is passed into the user input 1. If this was the only input defined, the user would not be prompted and the report would run immediately.

LiveReport Extensions offers a number of important improvements to parameter passing. These are:

- The ability to insert an unquoted string of characters (including SQL terms) into the SQL query
- The ability to enable/disable a secure mode which restricts pre-defined terms from being passed into the SQL query
- An unlimited number of inputs can be prompted for or passed in the URL
- Unlimited substitution parameters in the SQL statement.

In addition there is a SQL Template feature that allows report designers to include or exclude segments of SQL depending on various pre-defined conditions such as the state of selected inputs or other URL parameters. This feature is discussed in a separate section of this document.

Insert String

Introduction

LiveReport Extensions provides a new Input Type called **InsertString**. This new Input Type allows any string to be passed as an input to the LiveReport and inserted directly into the SQL query without any conversion or modification (such as putting quotes around the string). Among other things this permits whole segments of SQL to be passed into the LiveReport as a parameter. This could, for example, be used to pass additional logical WHERE or AND conditions to the SQL query.

To ensure that this flexibility can not be used to pass undesirable SQL terms into the SQL query (by accident or otherwise), LiveReport Extensions provides a **Secure Mode** option to analyze and block pre-defined strings. This secure mode is the default mode, but can be turned off if desired. *Open Text does not recommend that the secure mode is turned off.* Report designers who turn the secure mode off must take extra care to ensure they place the input substitution parameter at a point in the SQL where no unwanted commands could be constructed by malicious users. Whether or not this feature or LiveReport Extensions itself is used, report designers must always take responsibility for the security of the LiveReports that they create.

(The SQL Template feature provides a secure and graceful alternative to controlling the make up of a SQL statement. See the SQL Template section - section 8 - later in this document.)

Usage

To use this feature select **InsertString** under **Type** in the inputs box and set the corresponding **Parameter %** to the appropriate user input. Note the **Secure Mode** checkbox appears once **InsertString** has been selected. You should only uncheck this if you have read the notes above and understand the implications.

If a user executing a LiveReport attempts to pass a term which is being disallowed by the secure mode feature, an error message will be displayed. E.g.

```
[** Disallowed SQL Term(s): {;}{DROP} passed via parameters]
```

Carefully select where the string should be inserted by placing the substitution parameter **%1** or **%2** etc. at that point in the SQL. Even in **Secure Mode** LiveReport designers should consider carefully where such substitutions should be allowed and should always avoid placing these at the start or the end of the SQL. Typically this feature should be used where the string inserted is part of a "where" clause or within a particular logical condition inside the where clause.

In the following example the Insert String substitution parameter **%1** is contained within a string and surrounded by % wildcards. The effect of this query is to list all objects in Livelink with a name containing the input string. Note that the need to escape % symbols by repeating them like this: %% is removed by LiveReport Extensions.

Also note that InsertStr does NOT add any quotes to the substituted parameter (unlike the String type).

#	Type	Prompt	Secure Mode
1	InsertString	Name Contains	<input checked="" type="checkbox"/>

SQL: `select * from DTree where Name like '%%%1%%'`

Allow Database Modification:	<input type="checkbox"/>
Use Cursor Fetch mode:	<input checked="" type="checkbox"/>
Use Transaction mode:	<input checked="" type="checkbox"/>

#	Type	Options
%1	User Input	UserInput Input # 1

Configuration


The secure mode actively screens the final SQL query (including any passed parameters), looking for any “banned terms”.. By default, the following terms are blocked when secure mode is enabled:

;(semicolon),UPDATE,UPDATETEXT,WRITETEXT,REMOVE,DROP,CREATE,ALTER,INSERT, COMMIT,EXECUTE,FETCH,REVOKE,ROLLBACK,SAVE,TRUNCATE

If desired, this list of banned terms can be altered using an entry in the opentext.ini file which overrides the default list above. The following example shows an INI entry which would duplicate the default behavior:

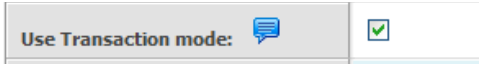
```
[reportext]
blockedSQLterms= " ; , UPDATE , UPDATETEXT , WRITETEXT , REMOVE , DROP , CREATE , ALTER , INSERT , COMMIT , EXECUTE , FETCH , REVOKE , ROLLBACK , SAVE , TRUNCATE "
```

Creating Extra Inputs or Substitution Parameters

Click the plus symbols  to create additional User Inputs or Substitution Parameters. When adding new User Inputs select **User Input** from the **Type** pop-up and then select the **Prompt** number from the **Options**. This Prompt number identifies which of the User Inputs defined in the **Inputs** section will be substituted.

7 Turning Off Transaction / Cursor Modes

Use Transaction mode

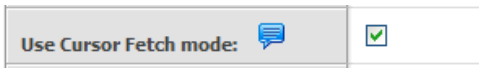


In standard LiveReports, and by default with LiveReport Extensions, all SQL queries run within a database transaction which locks the tables affected by the query. Where a query is altering the database it is essential that it runs within a transaction. (Note that generally it is not recommended to use LiveReports to make changes to the Livelink Database.)

It is most often the case that LiveReport queries are only reporting on data, and not modifying it. For queries which are only reading data, running the query within a transaction is less desirable as it locks the tables involved. (It is beyond the scope of this document to describe the concept of Transactions.)

Note, for the reasons already explained, certain functions such as **Allow Database Modification** are not available once **Use Transaction mode** is unchecked.

Use Cursor Fetch mode








In standard LiveReports and by default with LiveReport Extensions, queries are also executed using a "Cursor Fetch" to retrieve one row of data at a time so that Livelink can limit the total number of results returned. This limit can be set via the **Record Limit** field in both standard LiveReports and LiveReport Extensions. This approach protects the database from inappropriate queries returning excessively large result sets; however, this mode is slower than a standard SQL query (i.e. not Cursor Fetch mode). When queries are being developed and tested it is recommended to leave the Cursor Fetch mode selected and limit result sets with the **Record Limit** option. Once a query has been debugged and determined to be safe, this mode could be unselected in order to improve query performance. When this option is unselected an alert is generated to warn that the SQL query will now return unlimited results unless some constraints are specifically added to the SQL (e.g. **RowNum < 1000** in Oracle or **TOP 1000** in MS SQL).

8 SQL Templates

The template feature provides the ability to create SQL segments which can optionally be added to the main SQL query. This feature provides many new capabilities but in particular makes it easier to support applications where a variable number of WHERE/AND clauses may be required in a query.

This section of the user guide is not intended as a definitive guide to using this feature. Additional documentation for this feature is included in a separate document called “**Livelink LiveReport Extensions Using Templates**”.

For each template it is possible to define what conditions will determine whether the optional SQL Source is used or not. This is selected using the **Include IF** option (see below). Each template field is explained below. Note that each template includes a **View Template Behavior** icon  on the far right hand side which provides an illustration of how the template will be interpreted based on its current settings.

#	Auto-Where	SQL Source	Auto Comma	Include IF	Options	
~1	<input checked="" type="checkbox"/>	Name like '#1' and SubType = %2	<input type="checkbox"/>	URL Parameter	Param Name nameFilter	
~2	<input checked="" type="checkbox"/>	DataID = %3	<input type="checkbox"/>	URL Parameter	Param Name dataidFilter	
~3	<input checked="" type="checkbox"/>	CreateDate > %4	<input type="checkbox"/>	Not all inputs set to flag	Flag Value	

- **# (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 an empty 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 an empty 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 used input values have been set to the specified flag. For example, if the Flag Value is empty (considered an empty string) and there are two user inputs used in the template, then the template will be included unless both inputs are set to an empty 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 template 2 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.

9 Enhanced SQL editing

LiveReport Extensions provides some improvements to the SQL editing offered within the LiveReport.

- The size of the editing window has been increased and a vertical and horizontal scroll bar provided.
- The SQL query may be formatted with spaces and carriage returns to improve readability and aid maintenance of the query. This formatting is retained by Livelink when a developer returns to edit the LiveReport.
- Tab characters within the SQL editing field are inserted into the text rather than being interpreted (and moving the cursor out of the edit field).
- An Undo and a Redo button have been provided to aid in text editing.

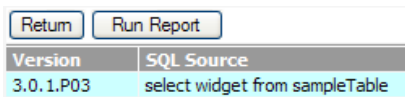
10 View SQL Source

This feature is only available in LiveReport Extensions 3.0.1 and above, though in version 3.0.0 there was a URL parameter called &SQLdebug which provided a similar feature.

This feature provides a way to review the SQL source prior to execution. The SQL source is shown following substitution of all parameters and templates. This is particularly useful when a LiveReport is being developed with templates as it becomes immediately obvious what the final compiled source looks like. This option can be invoked in two different ways:

- Using the Function menu **View SQL Source** option
- Using the **Save and View Source** button from the LiveReport Edit screen.

Once this option has been invoked using either of these two methods, the LiveReport is processed in exactly the same way as if it was being run but the actual query is never actually executed. Instead, a display screen is provided to show debug information made up of the module version and subversion plus the SQL source that would have been executed. This image illustrates an example of what this feature output looks like.



Version	SQL Source
3.0.1.P03	select widget from sampleTable

This simple example shows the version field (which is often of interest to support personnel) and then the SQL Source which shows the exact query that would be executed if this report had been run normally.

This example also shows that besides the **Return** button (that returns the user to the previous screen) there is a **Run Report** button that will actually execute the query shown in the SQL Source field.

11 View Query

This feature is designed for situations where it is necessary to allow certain users to see the LiveReport Query without giving them access to the actual edit screen. In some situations, access to the Edit screen can be potentially dangerous and insecure. If certain developers or users need access to the contents of these queries (for example to learn from other examples or to observe what a particular report does) it is possible through this feature to give them appropriate permissions to use this feature but not the Edit feature.

This image illustrates the format that this feature provides.

Return

LiveReport Query Details For: Demo

Record Limit: **1**

Database Selection: **currentivelink**

Database Options:

Transaction Mode	Fetch Cursor Mode	Allow Database Modification
Enabled	Enabled	Disabled

SQL:

**select ~1~2
from sampleTable**

Inputs:

#	Prompt Type	Prompt Text
1	InsertStr	name
2	InsertStr	type
3	InsertStr	subtype

Parameters:

%	Parameter Type
%1	User Input 1
%2	User Input 2
%3	User Input 3

Templates:

~	Where/And	SQL Source	Include IF	Options
~1	Disabled	%1, %2, %3	allnotset	Flag value: Empty String
~2	Disabled	#	tempnotset	Template Number: 1

Show Prompts: **Disabled**

Report Format: **autoreport**

There are two possible ways of configuring permissions for this feature. By default, this feature is only available to users who have at least **Modify** permissions for the LiveReport. If the user has not been setup to create LiveReports from the **Administer Object and Usage Privileges** then they will not have access to the **Edit** option but will have access to view the query.

Alternatively, it is possible to change this behavior so that users with **See and See Contents** can use this feature. This is done by adding the following entry to the **[ReportExt]** section of the **Opentext.ini** file:

QueryVisibleWithSeeContents=true

For any Livelink server that has this option set, users with **See and See Contents** permissions on a LiveReport will be able to use the **View Query** option.

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.

