Kiwiplan - ESP Crystal Reports Standards and Help Hints

Customised Reports:

In order to make sure that customised reports, and the SQL views or stored procedures behind them, do not get overwritten during ESP upgrades the following points should be observed:

1. A site specific reports directory should be set up and this must be the first directory on the path listed under the section Directories/Reports in ESP.
2. All customised reports (“.rpt” files) and “.ini” files should be saved in the site specific reports directory. To avoid confusion **ONLY** customised reports and ini files should be saved to this directory.
3. Any customised report should use a customised source for its data. If the report has been customised but uses a generic SQL view or stored procedure then the view or stored procedure should be copied and renamed. For example the stored procedure for the generic Cost Estimates Report is called “irsCostEstimateDetails”, at one site the stored procedure is called “irsACostEstimateDetails”, at another the procedure is called “irsWCostEstimateDetails”.
4. As a naming convention it is strongly suggested that, in cases where a customised view or stored procedure is derived from a generic one the customised view/stored procedure is given the same base name with a site specific suffix eg for a site called “AnyCartons” a customised version of the view “irsListofInvoices” could be named “irsListofInvoices_AnyCartons”.
5. Once the copy of the view or stored procedure has been made Crystal reports will allow you to “alias” the report fields to the new View/Stored Procedure with no changes required to the fields on the report. This is done through the “Database/Set Location” menu option in the Crystal reports designer.(**NOT THE Database/Set Alias” menu option**)..
6. Once the alias is set using the “Set Location” option it will be necessary in some case to perform a final step. This is done through the “Show SQL Query” option, also on the Database Menu. If the report is being driven from a View there will be a “FROM viewname” clause. If this has not been changed by step 4) above the following will need to be done:

- If the existing view is called “irsOldView” the new customised view is called “irsNewViw”
- If the from clause was “from irsOldView” this will need to be changed to “from irsNewView irsOldView”
- If the from clause reads “from irsOldView irsOldView” or “from irsOldView as irsOldView” then this needs to be changed to “from irsNewView irsOldView”. The “as” in the second example is optional

These points should minimise the chances of an ESP upgrade causing problems with customised views and reports.

As another safeguard any changes made to existing Crystal reports or ini files should have copies sent to Kiwiplan NZ. Any SQL views or stored procedures should be saved as sql scripts using the naming convention:

Yyymmdd_Updirsxxxxx.sql – where sxxxxxis is an existing view/stored procedure name

or

Yyymmdd_Addirsxxxxx.sql – where sxxxxxis is a new view/stored procedure name

And then a copy of the script sent to Kiwiplan NZ.

If small changes are required to a standard report it is worthwhile discussing the changes with Kiwiplan NZ before deciding to create a custom report. Sometimes small changes can be added to the existing standard report and made available to all customers.

SQL Naming and Coding Conventions

Traditionally all views and stored procedures that are created to be used with reports are named with the prefix “irs”

All scripts that create or modify views, stored procedures or SQL functions should follow the format:

If SQL object exists

Drop SQL object

GO

Create [dbo].[SQL Object]

<SQL to create the object>

GO

Grant permissions on SQL Object

GO

For a view the Grant statement should be:

GRANT SELECT on [dbo].[<view name>] to [PUBLIC]

For a stored procedure (or SQL function) the statement should read:

GRANT EXECUTE ON [dbo].[<stored procedure name>|<SQL function name>] TO [public]

Make sure that all objects are created with the “dbo” owner qualification to the name.

All character variables within stored procedures and functions should be defined as “nvarchar” to cater for Unicode based sites.

If creating temporary tables all character columns should be defined using the “COLLATE DATABASE_DEFAULT” option to cater for sites where multiple collations could be in use. This is shown below:

declare @table table

(

atextfield nvarchar(50) COLLATE DATABASE_DEFAULT null,

....more fields

)

This allows sorts and equality testing to happen without an error being raised.

A couple of points about performance:

- Use table variables for temporary tables rather than the traditional “#” temp tables
- Avoid cursors if possible
- Think in terms of set operations rather than individual row operations. That is if a query can be written to perform bulk operations then use this in preference to traversing a result set and working on individual rows

- Use Case statements in queries to cater for multiple options rather than repeating the query.
- Try to think of the way the indexes on the tables involved in a query are structured when building a “WHERE” clause, the closer the “WHERE” clause matches underlying indexes the more chance there is that the query optimizer will utilise the index.

Standard ESP SQL Functions

The following functions are part of the ESP database.

Note: For the date and financial period functions “zero hour” is “00:00:00” hours on the day, “midnight hour” is 23:59:59.99” on the day

Function name	Parameter(s)	Returns	Description
Date Functions -Calendar dates			
dbo.fn_getstartday	datetime	datetime	Given a datetime value, returns the “zero hour” value for that date
dbo.fn_getendday	datetime	datetime	Given a datetime returns the “midnight hour” value for that day
dbo.fn_getstartmonth	datetime	datetime	Given a datetime returns the “zero hour” value for the first of the month
dbo.fn_getendmonth	datetime	datetime	Given a datetime returns the “midnight hour” value for the last day of the month

Function name	Parameter(s)	Returns	Description
dbo.fn_getstartyear	datetime	datetime	Given a datetime returns the “zero hour” value for the first of January for that year
dbo.fn_getendyear	datetime	datetime	Given a datetime returns the “midnight hour” value for the 31st of December for that year
Date Functions - Financial dates			**These need the financial year and financial periods to be set up in ESP to work, otherwise they fall back to calendar months and years**
dbo.fn_getstartfinperiod	datetime	datetime	Returns the “zero hour” on the first date of the financial period that the given a date is contained in
dbo.fn_getendfinperiod	datetime	datetime	Returns the “midnight hour” on the last date of the financial period that the given a date is contained in
dbo.fn_getstartfinyear	datetime	datetime	Returns the “zero hour” on the first date of the financial year that the given a date is contained in
dbo.fn_getendfinyear	datetime	datetime	Returns the “midnight hour” on the last date of the financial year that the given a date is contained in
General Utility Functions			
dbo.fn_IsInList	@strlist nvarchar(1024) @strsearch nvarchar(100) @joinchar nvarchar(1)	smallint	Given a string containing a list of values (@strlist) separated by a single character(@joinchar) and a value to search for(@strsearch) returns 0 if the exact value is not found or 1 if it is.
dbo.fn_parsecriteria	@sValue nvarchar(200) @fieldname nvarchar(100) @datatype nvarchar(50)	nvarchar(500)	Given a string containing a value to include in a WHERE clause, the name of the field to search on and the data type of the field will return a string that can be concatenated to a SQL statement

Function name	Parameter(s)	Returns	Description
dbo.fn_as16s	Int	nvarchar(30)	Given an integer value returns the value formatted as 16ths ie Given 21 will return “1.05”, Given 30 returns “1.14” (Used on sites that use imperial units of measure)
dbo.fn_fixx	Float	Int	Given a number returns the largest integer less than or equal to the given number Given 1.5 returns 1 Given -2.4 returns -3
dbo.fn_MaskValue	@mask nvarchar(500) @pval int	nvarchar(500)	Returns the given number masked with the given mask
dbo.fn_StripMask	@mask nvarchar(500) @pval vnarchar(500)	Int	Returns the numeric value of a masked string
dbo.fnGetFKColumns	@constraintID int, @keyID int, @rf nchar(1)	varchar(2126)	This , and the next function are used to retrieve details on foreign key constraints and indexes. There were written to be used in the stored procedure “dbo.inf_ResetCollation” (see Utility Stored Procedures section below)which resets and rebuilds all constraints and indexes in a given database to a given collation.
dbo.fnGetIndexColumns	@objname sysname @objid int @indid int	varchar(2126)	
ESP Specific Function			
dbo.fn_numworkdays	@firstdate datetime @lastdate datetime @id int @idtype nvarchar(1)	int	Return the number of working days between a given start and end date for either a specific address or a plant if the @idtytpe is “P”(uses the plant base address). This uses the “opendays” field from the address.

Function name	Parameter(s)	Returns	Description
dbo.fn_getcoatings	@pdid int @coattype varchar(2)	varchar(500)	Returns a comma separated list that contains, for each colour/coating, the sequence, code and coverage. The parameter @coattype can be one of the following: IC – Inside colour OC – Outside colour II - Inside coating OI – Outside coating
dbo.fn_getPDCoating	(@pdid int @isColourorCoating int @inside smallint @coatorcover int	varchar(500)	Returns a comma separated list of either the coating/colour codes or the coverage percent. Data is selected from the table ebxproductdesigncoating based on the parameters: @inside equals the column inside (can be -1 or 0) @isColourOrCoating equals the column @isColourorCoating
dbo.fn_ShipPiecesPerUnit	@pd int @route int	int	For a given route for a product design returns the quantityperunit field from th euntizing data for the last step on the route.
dbo.fn_unitcost	@ceid int @ppid int	float	Returns the “per unit” full cost value from either a cost estimate or product price record.
dbo.fn_UnitSummaryQty	@unitsummary varchar(1000)	int	Returns the total quantity of items as held in the unit summary field
dbo.fnGetSetTotalPrice	@CostEstimateID int @CalculationQuantity int, @currdate datetime	money	Returns the total “freight inclusive” price from the product price table for all components (if any) that are part of the set defined by having the supplied cost estimate as their “master” cost estimate. They must also have the same calculation quantity and be active at the given time.

Function name	Parameter(s)	Returns	Description
Functions to access MAP/ULT data		**These functions have been re-written as stored procedures so that they would not be restricted to a “hard coded” linked server. The sored procedures are named under the function names and both take the linked server name as a parameter. Output remains a table/dataset with the same column names.**	
dbo.fn_GENPL() (Stored procedure dbo.irsGENPL)		plantno varchar(8) plantname varchar(30) defaultCorrugatorMachineNo int defaultDespatchMachineNo int boardTransferMachineNo int pickupGoodsMachineNumber int quarantineMachineNumber int fgsReworkStoreMachineNo int sbsFromStockMachineNumber int sheetBoardStrapperNumber int sheetBoardSlitterMachineNo int supplyFromStockMachineNo int defaultRssReceiptingStore int stockDespatchMachineNo int despatchNotificationMachineNo int defaultBoardReceivingmchNo int defaultOrderingMachineNo int	Returns a table giving, for each plant in the MAP dataset, the default machines as set up in the GEN/PL XLATEP parameter in MAP
dbo.fn_GENTU() (Stored procedure dbo.irsGENTU)		pallettype varchar(8) pallet varchar(30) width int null length int null height int null weight int null	Returns a table giving, for each pallet type, the width, length, height and weight. These are taken from the GEN/TU XLATEP parameter in MAP

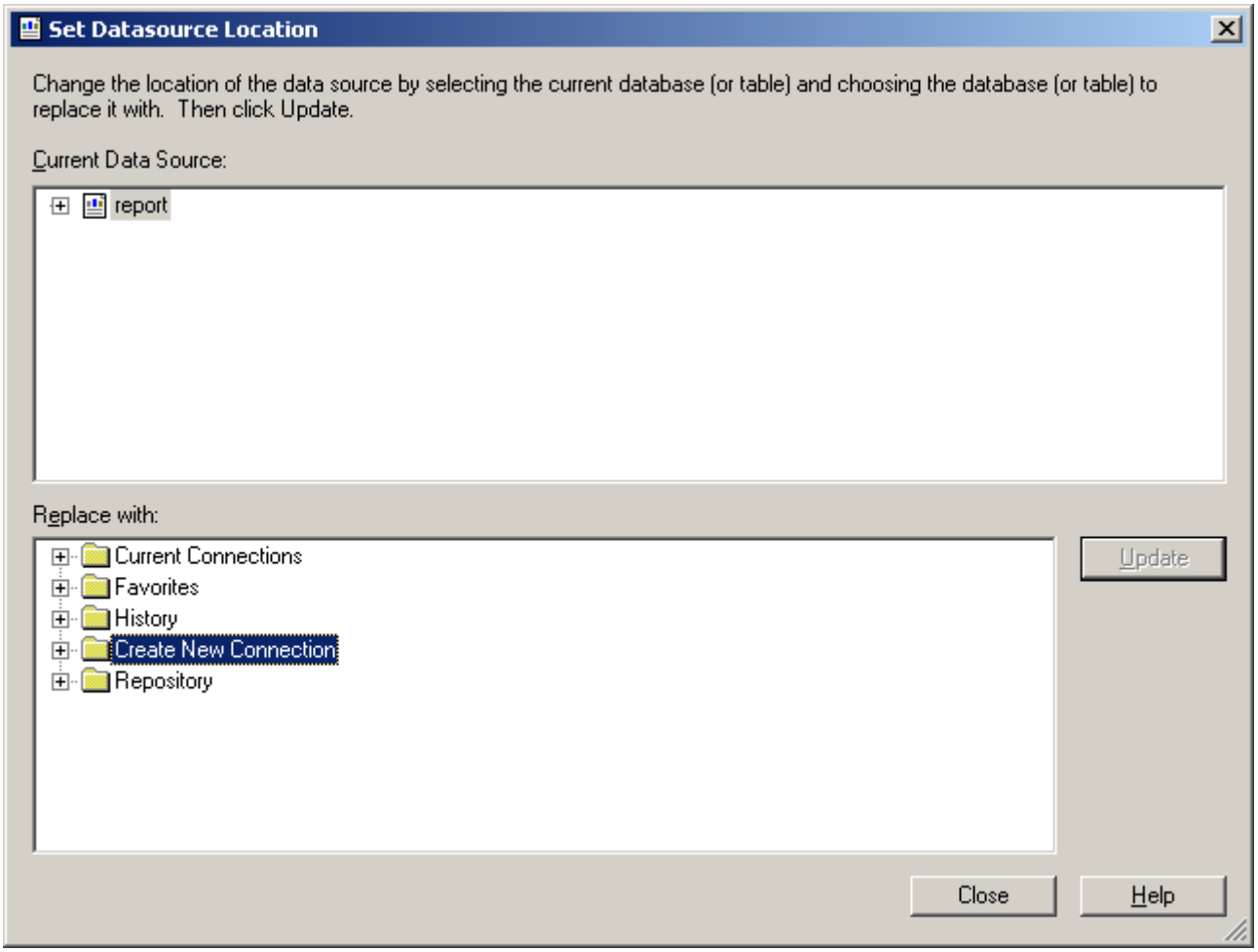
Utility SQL Stored Procedures

The following table lists stored procedures that provide functionality that is outside specific report requirements.

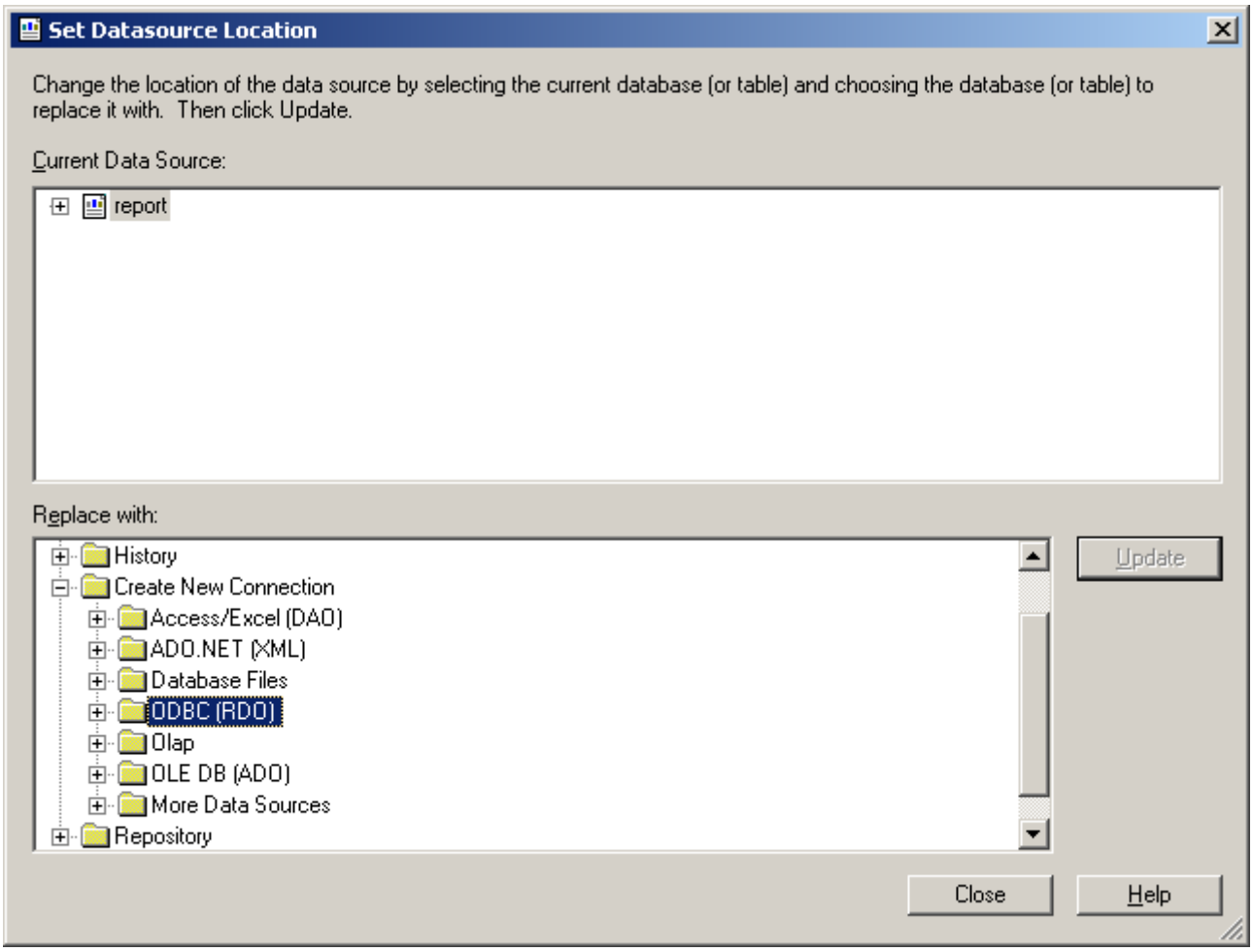
Stored Procedure	Input Parameters	Output	Description
Procedures to access MAP/ULT data		**These procedures assume a valid “linked server” to a MySQL database is set up. The syntax of the SQL is MySQL specific.**	
dbo.irsGENPL	nvarchar(500)	See entry above for dbo.fn_GENPL()	Given a linked server name returns the table described above in the Functions section
dbo.fn_GENTU	nvarchar(500)	See entry above for dbo.fn_GENTU()	Given a linked server name returns the table described above in the Functions section
dbo.irsCORRCP	nvarchar(500)	plant_number int, corrugatornumber int, corrugatorname nvarchar(30) machinenumber int	Given a linked server name, returns a list of all corrugators with information taken from the CORR/CP parameter in the XLATEP table and the CSCNTR table.
dbo.irsmachines	nvarchar(500)	Machineno interactionmachinename nvarchar(40) machinegroup int	Given a linked server name, returns a list of all machines held in the CSCNTR table.

Stored Procedure	Input Parameters	Output	Description

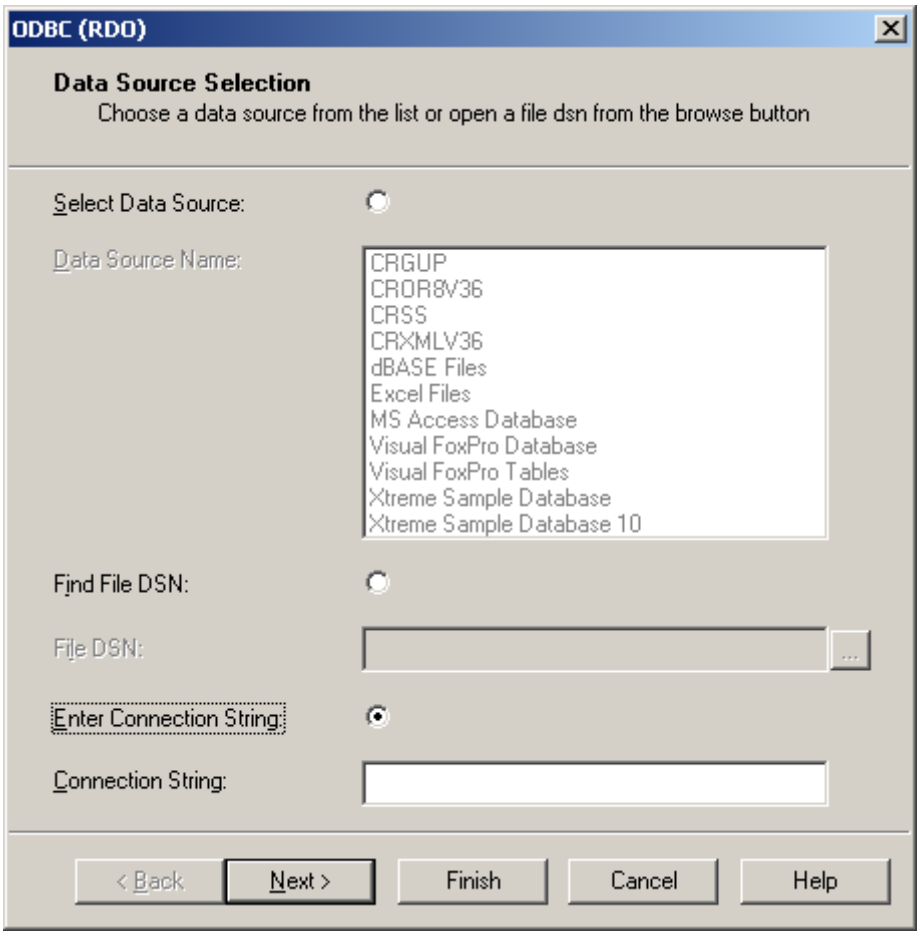
Crystal 10 Database Connections



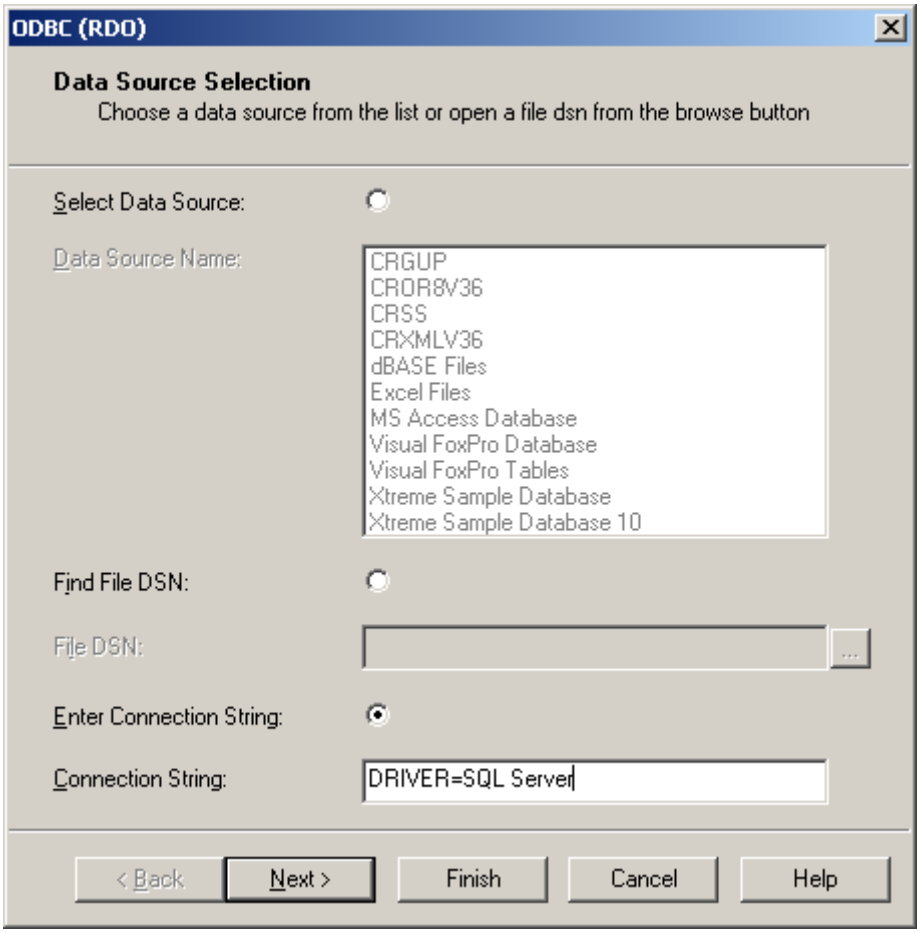
Select “Create New Connection”



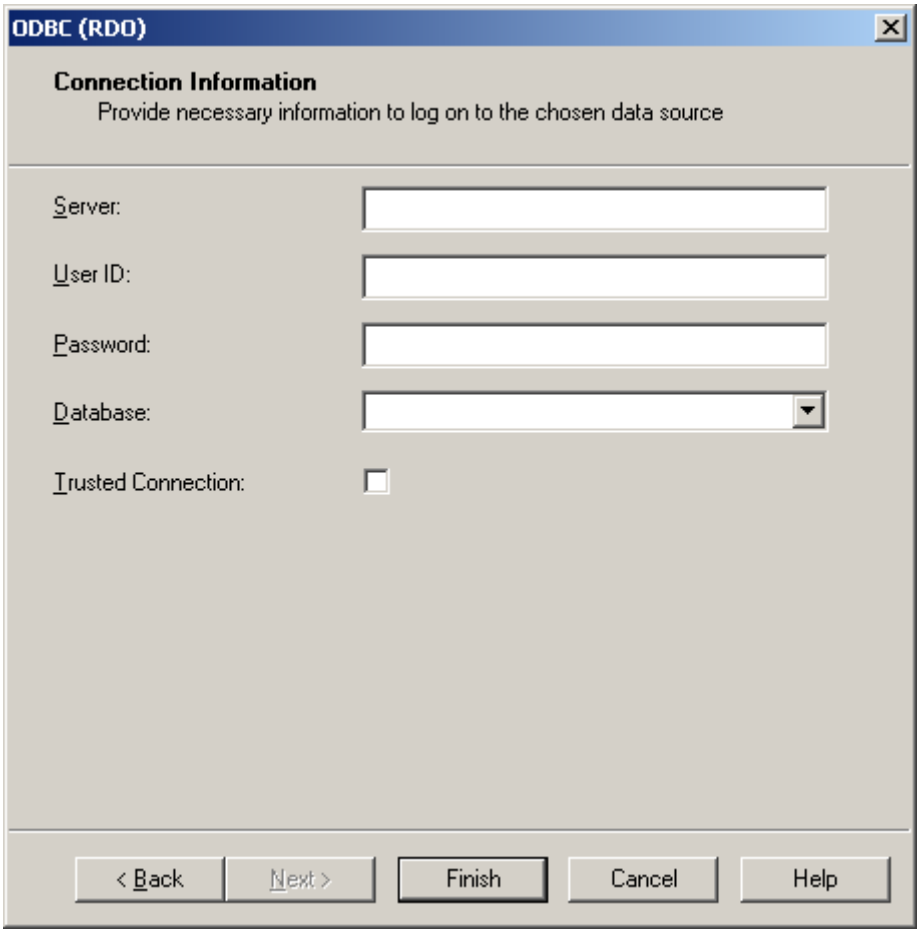
Select “ODBC (RDO)”



Select “Enter Connection String”



Enter the string “DRIVER=SQL Server”



Fill in the details:

1)Server is the database server

2)User ID and Password are the values held in the ESP Parameters under

Configuration/Application/InfClient(or Infrastructure)/Section/Security/Report Login Name

Configuration/Application/InfClient(or Infrastructure)/Section/Security/Report Login Password

3)Database is the ESP database name

DO NOT check the “Trusted Connection” check box

Click on “Finish” and if all details have been entered correctly you should be able to select tables, views or stored procedures from the database.

Crystal Reports and ESP

This document provides an outline of the interaction between ESP and Crystal Reports and covers the points to be aware of if you are planning on creating Crystal reports to be used from within ESP.

Overview

There are two major components involved in the integration of Crystal Reports with ESP.

- Ini files – these provide a specification for the construction of the report options form that allows the user to make criteria selections
- Rpt files – these are the actual Crystal reports files

These files MUST have the same base name for example if you have saved your report as

“My Report.rpt”

the ini file must be called

“My Report.ini”

Both files must exist and must reside in a directory that is specified in ESP in the “INFClient/Sections/Directories/Reports” parameter.

Sections below cover the following points in more detail:

- An overview of the processes involved when running a report
- Ini Files
- Points to take into consideration when writing Crystal reports
- Adding reports to ESP
- Report Data Sources

Running a Report

Once a report is selected from the reports menu, or one of the sub menus, the following process takes place:

- ESP looks for the ini file with the same name as the report. It searches for this file in the directories specified by the “INFClient /Sections/Directories/Reports” parameter, in the order that the directories are listed.
- Once found the ini file is read and the report options form is created with the controls specified by the various sections in the file (see Ini Files below). Default values are placed in fields where specified.

- If translation is turned on then all labels and constant text values are translated, this includes all hard coded lists (combo boxes of “listAbsValue” type or default text)
- The options form is displayed and the user makes their selection of criteria/parameters/sorting fields.
- The user also has the option of either exporting or viewing the generated report.

Once the user has made their selections and clicked the “OK” button (or pressed enter) ESP performs the following actions:

- ESP looks for the rpt file in the directories named in the “INFClient /Sections/Directories/Reports” parameter. NOTE: The rpt file does NOT have to be in the same directory as the ini file, as long as it can be found in one of the directories specified in the parameter.
- If translation is turned on for the site then the report is translated (see Crystal Reports – Translation below)
- Values selected or entered in the various parameter and criteria fields on the report options form are processed and the Crystal reports’ SQL is updated (if required) and report parameters are set.
- The reports data source is set to be the current database that ESP is running against unless the section “[DBConnection]” has been set in the Ini file. If it has the connection will either be set to the defined connection or left as is.
- If the “Export Report” option has been selected the Crystal Reports “Export” dialogue box where the type and location of the export can be chosen. The report is then generated and output to the specified location.
- If the “View Report” option has been selected the report is generated and displayed in the Crystal reports viewer window.

Ini Files

Ini files provide information to ESP to allow it to construct a report options form that acts as the interface between the end user and the Crystal report being run. They follow the standard Windows ini file format with section headers, indicated by a keyword enclosed in square brackets alone on a line:

[Criteria]

and section bodies, generally a keyword followed by an equal sign followed by various values separated by semi-colons:

Report Date=date;{irsRepView.datefield}

Sections recognised by ESP are:

- [General] – This must always be the first section and always consists of the one line body:

Form=frmReportOptions

- [Criteria] – This section specifies values which will be used to build a, or add to an existing, “where clause”. This section is NOT used if the data for the report is supplied by a stored procedure.
- [Default Criteria] – This section provides default values for items in the “Criteria” section. Labels must match exactly with a Criteria Label in the Criteria section.
- [Parameters] – This section is used to supply values to Crystal report “Parameter” fields. There are two types of parameter, displayed or hidden. Either type of parameter can be either used internally in the report of passed through as input parameters to stored procedures.
- [Default Parameters] – Values in this section provide default values to items in the “Parameters” section. Labels must match exactly with a Parameter Label in the Parameters section.
- [DBConnection] – Specifies an alternate data source to be used for the current report.
- [Sortable Fields] – Any items in this section are used to structure the sort order of the report. If a report has pre-defined grouping the sorting will be defined at the detail level, within the lowest level of grouping.
- [Default Sorting] – Values in this section allow the default sort options to be offered, including setting the sort direction.
- [Formulas] – Formulas and Formula Fields allow for a degree of end user selection of fields to appear on the report. These fields can be calculated, concatenations of multiple fields or can be individual fields. This section defines the labels and available formulae.
- [Formula Fields] – This section lists the formula field names that are available on the report and defines the link between the formula field used to display a selected formula value and the formula field which will display the formula label.
- [Default Formulas] – This section provides default formula selections.
- [Show Form] – This section is used to tell ESP that the report options form should not be displayed
- [Default Mode] – This section is used to set the default mode for processing the report (View or Export)

The following is the contents of an example ini file. Most ini files will not be anywhere near as complicated as this. For the purposes of documentation every possible section and option has been used. Sections may be left blank, for example a report that relies on a stored procedure for its data will have no values in either the Criteria or Default Criteria sections and many reports have no Sort or Formula sections defined. Following the sample is an explanation of the various sections and detail lines of the file.

[General]

FormName=frmReportOptions

[Criteria]

Plant=Long;{espInvoice.plantID};listQryID=plant,,name,ID;

Customer=String;{orgcompany.name};listqryvalue=company,id in(select id from orgcompany where iscustomer = -1),name

Invoice Number=String;{espInvoice.InvoiceNumber};

Transaction Date=Date;{espInvoice.TransactionDate};

OrderType=String;{espOrder.ordertype};listAbsID=0,Make and Deliver,1,Call Off,2,Top Up

(NP)Status=String;{irsInvoice.InvoiceStatus}

Product Design Number=String;{ebxproductdesign.designnumber}

[Parameters]

Application Name=Application Name

Application Version=Application Version

Username=User

Copyright=Copyright

Criteria=Criteria

(NP)ReportType=String;

ReportCustomer=Customer

Invoice Type=String;;listAbsValue=Product,Non-Product

Report Date Range=Date;

[DBConnection]

Server=Access1

DB=db11

Login=mylogin

Password=no password

[Sortable Fields]

Invoice Number={espInvoice.InvoiceNumber}

Transaction Date={espInvoice.TransactionDate}

Customer Name={orgCompany.Name}

[Formulas]

Description LxW=ToText({espOrder.width}) + " X " + ToText({espOrder.Length})

Status={espOrder.OrderStatus}

Contact Name={orgContact.FirstName} + " " + {orgContact.LastName}

Special Instructions={espOrder.SpecialInstructions}

[Formula Fields]

extraField1=extraField1Label

extraField2=extraField2Label

extraField3=extraField3Label

[Show Form]

[Default Mode]

Export

[Default Criteria]

(NP)Status=<>Cancelled

Transaction Date=[mtd]

Product Design Number=[class#productdesign.designnumber]

[Default Parameters]

(NP)ReportType=Detail

Invoice Type=Product

[Default Sorting]

Invoice Number=ascending

[Default Formulas]

extraField1=Description LxW

All the first two lines in ini files used for reports must start with:

[General]

FormName=frmReportOptions

The other sections can appear in any order, the only requirement is that the section header appear as the only value on a line, it is not necessary to have blank line between the last item in a section and the next section header, however it does make the ini file easier to read.

The usage of each section is detailed below. The section name is given, followed by a line showing the syntax of the detail items, under this is an example line from the sample ini file, then a description of the parts of the line.

[Criteria]

Field Label=Data Type;{Data Field};Combo Type(optional);Required?(optional)

Plant=Long;{espInvoice.plantID};listQryID=plant,,name,ID;Yes

- Field Label / Plant -The value to the left of the equal sign is the name that will appear on the options form as the label for the field. If a value is selected in the field the label is also formatted into the “Criteria” field value displayed in report headers. If the Field Label is prefaced with the string “(NP)” then the criteria will not be displayed on the options form although if a default value for it is specified in the “[Default Criteria]” section the value will be used in building the SQL where clause.
- Data Type / Long – The value immediately to the right of the equal sign indicates the data type of the field. This can be one of the following:
  - Long – The data entry will be a text box unless a Combo Type is specified
  - String – The data entry will be a text box unless a Combo Type is specified
  - Date – The data entry will be a text box unless a Combo Type is specified
  - Boolean – Data entry will be via a check box
- {Data Field} / {espInvoice.PlantID} – This is the field against which the entered criteria is applied in the where clause. It must be enclosed in curly braces “{}” and must match exactly the name of a field that is available to the query in the report. This can be either a table column or a field from a SQL view (if that is the data source for the report. In this example if the user had entered the value 5 in this field on the options form, then ESP would format this as:

(espinvoice.plantid = 5)

If the query already had a where clause would append this as

and (espinvoice.plantid = 5)

If the query did not have a where clause this would be added

where (espinvoice.plantid = 5)

- Combo Type / listQryID=plant,,name,ID – The “Combo Type” part of the section detail item is optional and is used when a drop-down combo box is to be displayed. There are several different combo box types available depending on the source of the data.
  - ListQryId – Data is supplied by a query and the “ID” field from the selected row is used.
  - ListQryValue – Data is supplied by a query and the value of the selected row is used
  - ListAbsId – Data is supplied by a hard coded list and the “ID” value is used
  - ListAbsValue – Data is supplied by a hard coded list and the value is used.

The syntax of the two “ListQry…” combos is

Table Name,Filter Clause(optional),Display Value,ID(not used if the combo type is ListQryValue)

Where

- - Table Name is the business class name (the database table name without its 3 letter prefix)
  - Filter Clause is a filtering condition and should take the form of

Field=condition

For example, in the example being used, the customer combo will only display names from the orgcompany table where the “IsCustomer” flag is true (-1),

id in(select id from orgcompany where iscustomer = -1)

If no filter condition is used then the space must be delimited with a trailing comma (as in the sample ini file)

- - Display Value is the name of the field to display in the list, in the case of a “ListQryValue” this is the field used to build the where clause.
  - ID is specified when the combo type is “ListQryID”

The syntax of the combo type “ListQryID” is as follows

listAbsID=ID1,Value1,ID2, Value2,….

listAbsID=0,Make and Deliver,1,Call Off,2,Top Up

In other words, it consists of a list of pairs of values, a numeric “ID” value followed by a string to be displayed, all values separated by commas.

The syntax of the “ListQryValue” combo consists of a simple list of values separated by commas.

- Required?(optional) - This is an optional value, that, if set to one of the following values will cause the field background to be displayed with the colour defined in the ESP parameter “InfClient/FORMS/Required controls colour”. It will also cause ESP to check that a value has been entered in the field when the user clicks the “OK” button. If there is no value in the field then the user is warned and asked to enter a value.
  - 1
  - -1
  - true
  - required
  - mandatory
  - yes

[Default Criteria]

Criteria Label=Default Value

(NP)Status=<>Cancelled

There are three types of default values. In the example above the value is literal, in that it will be displayed exactly as it is (<>Cancelled) against the specified Criteria. The second type is a keyword enclosed in square brackets as in the line below.

Criteria Label=[Keyword]

Transaction Date=[mtd]

The possible keywords are explained in the following list

- [now] – The current date and time formatted as a "General Date"
- [datenow] – The current date formatted as a “Short Date”
- [date] – The current date formatted as a “Long Date”
- [time] – The current date formatted as a “Long Time”
- [mtd] – A date range with the first of the current month as the first value and “today” as the second value, both formatted as “Short Date”
- [ytd] – A date range with the first of the current year as the first value and “today” as the second value, both formatted as “Short Date”
- [yesterday] – The day before the current date formatted as a “Short Date”
- [lastmonth] – A date range with the first of the previous month as the first value and the last of the previous month as the second value. Both formatted as “Short Date”
- [lastyear]– A date range with the first of January of the previous year as the first value and the thirty first of December of the previous year as the second value. Both formatted as “Short Date”

The third type of default value is also enclosed in square brackets and has the form:

Criteria Label=[Default Value]

Product Design Number=[class#productdesign.designnumber]

In this type of default value the keyword class# indicated that we want ESP to use the current context of the user and to

use as a default the specified property of the class. In this example, if the user had the Product design form open then the current Product designs design number would be placed as a default value against the Criteria “Product design Number”

[Parameters]

There are two major types of Parameters, displayed and hidden. In both types the Parameter Label that appears to the left of the equal sign must exactly match a Parameter field defined in the Crystal report. If the parameter does not exist (or is misspelled) then any value derived from the Parameter from the ini file is ignored.

Displayed Parameters

Displayed parameters have a similar syntax and usage to Criteria.

Parameter Label=Data Type;;Combo Type(optional);Required?

Invoice Type=String;;listAbsValue=Product,Non-Product;Mandatory

The major differences between Displayed Parameters and Criteria are:

- The “Data Field” that appears on the Criteria is missing from the Parameter line.
- If no “Combo Type” is specified then the Data Type must be terminated with a semi-colon.
- No “where clause” is built using any entered value. Apart from date parameters all values are passed through to the equivalent report parameter
- Date values can be either a single date or a date range (two dates separated by two dots). When ESP comes to process the value in a Date parameter it performs the following actions:
  - If the value is a single date, say 21/06/04, then two values are created, one that is for the start of the day, formatted as “yyyymmdd hh:mm:ss”, (“20040621 00:00:00”) and one for the end of the day (“20040621 23:59:59”).
  - If a range has been entered, say 20/06/04..30/06/04, then the first value is formatted as the start of that day (“20040620 00:00:00”) and the second as the end of that day (“20040630 23:59:59”)
  - It looks for a parameter in the Crystal report with a name that matches the parameter label but with the word “start” added to the end. In the sample ini file the parameter from the ini file is “Report Date Range” and the expected parameter from the report would be “Report Date Rangestart”. The first date value is assigned to this parameter.

    - Next ESP looks for a report parameter with the same base name but ending with the word “end” (“Report Date Rangeend”) and assigns the second value to this parameter if it is found.
    - If no report parameter ending with the word “start” is found ESP will look for a report parameter that exactly matches the Parameter Label and if found will assign the literal value of the field to it.
- Ranges (two values separated by .. (two dots) are only supported for date parameters. With Criteria ranges are supported for String and Numeric data types as well.
- The label shown as black text on a white background (when using the standard Windows colour scheme). Criteria labels are black text on a background the same colour as the form background.

Hidden Parameters

Hidden parameters fall into three categories, Keyword, Criteria Value and Hidden Value

Keyword Parameters

Keyword=Keyword

Application Name=Application Name

There are eight possible keyword parameters. They follow the syntax of two instances of the keyword, separated by an equals sign. The possible keywords are:

- Application Name – This is the application that the report is listed as a parameter under. In the standard reports this will be “InfClient”
- Application Version – This is the current major version of ESP e.g. 4.028 or 4.036
- Username – The logged on user running the report
- Copyright – Kiwiplan Copyright
- Criteria – This is a “readable” version of the where clause built from selected Criteria values. It follows the format of :

(Criteria Label = Selected Value) And (Criteria Label 2 = Selected Value 2)

- Now – This is the current date and time that the user presses “OK” on the options form
- Date – The current date at the time that the user presses “OK” on the options form
- Time – The current time that the user presses “OK” on the options form

Criteria Value parameters

Parameter Name=Criteria Label

ReportCustomer=Customer

These are Parameters that are not displayed but which are loaded with the value selected by the user for the Criteria. This is useful as the individual Criteria values are not available within the body of the report.

Hidden Value Parameters

(NP)Parameter Label=DataType;

(NP)ReportType=String;

Hidden Value parameters have their Parameter Label prefixed by the string “(NP)”. As with all parameters the name of the Crystal Report parameter must match exactly (including the “(NP)”) the label in the ini file. The parameter line in the ini file must be terminated with a semi-colon. This type of parameter would be used to pass a default value through to the report that the report designer did not want the user to see, possibly to set site specific values in a report that was otherwise generic.

[Default Parameters]

Default Parameters follow the same rules and have the same processing as Default Criteria. However they are only applied to Displayed Parameters and Hidden Value Parameters. Keyword Parameters are already their own default and to apply a default to a Criteria Value Parameter you would apply it to the Criteria that the parameter was pointing to.

[DBConnection]

Connection Parameter Name=Value

Server=Access1

DB=db11

Login=mylogin

Password=no password

This section is used by ESP to set the connection details for a report if it has been defined to use a different data source than the ESP database. There are two ways of using this section,

- The first as shown in the sample ini file, specifies the values required to set the database connection. The example is for a system ODBC DSN.
  - Server=Access1 – In the case of an ODBC connection this is the DSN name, if the report was being pointed at a SQL Server database (using the Crystal native SQL driver) then the server value would be the name of the SQL Server.
  - DB=db11 – This is the database that holds the tables that the report is looking for..
  - Login=mylogin – this is the username to use to login on to the database. If this option is not specified or is blank the default “report user” as defined for ESP will be

used.
    - Password=no password – This indicates that no password is to be used on this connection. If this option is not specified or left blank the default “report user password” as defined for ESP will be used.
- The second way is to specify a single line:
    - UpdateTables=no – This tells ESP that you want it to use the connection details from the report itself. However there is a limitation here in that the user and password on the report connection **MUST** be the same as the default defined as the ESP “report user”.

[Sortable Fields]

Sort Field Label={Sort Data Field}

Invoice Number={espInvoice.InvoiceNumber}

The Sort Field Label is displayed in the list box of available sort fields. Selecting any of the sort fields will cause the report to be sorted on the Sort Data Field assigned the Sort Field Label Name.

[Default Sorting]

Sort Field Label=Sort Direction

Invoice Number=ascending

Any entries must exist as Sort Field Labels in the Sortable Fields section, and will be automatically selected when the form is displayed. The value to the left of the equals sign is optional and can be either “ascending” or “descending”. If omitted the equals sign must be used and the default sort direction will be descending. Multiple default sort fields can be specified below each other in this section. The order that they are placed in determines the precedence of sorting on the report.

[Formulas]

Formula Label=Formula

Description LxW=ToText({espOrder.width}) + " X " + ToText({espOrder.Length})

The Formula Label is what is displayed in the drop down list of available formulae.. It is also the value that is assigned to the Label Formula Field. The Formula is assigned to an empty formula field on the Crystal report.

The syntax of the Formula must be acceptable to Crystal Reports. (Default syntax used in the KiwiPlan generic reports is “Crystal” not “Basic”)

[Formula Fields]

Formula Field Name=Label Formula Field Name

extraField1=extraField1Label

This section holds a list of pairs of empty formula fields from the Crystal report. The Formula Field Name is the formula to which a selected formula is assigned. The Label Formula Field Name has the Formula Label assigned to it.

[Default Formulas]

Formula Label=Formula Field Name

Description LxW=extraField1

The Default Formulas section allows the assignment of formula to specific formula fields.

[Show Form]

This section is optional if used and the [Show Form] section header is followed on the next line by the word “No” the report options form will not be displayed. This option can be used where a report can have the various criteria and /or parameters set with default values that should not be altered. For example a sales report that is always run using the last calendar month as the date range.

[Default Mode]

Export

This section is optional. It is used to specify that the “Export Report” option is selected by default on the report options form. If omitted or any other value than “Export” is placed under the heading the default mode will be set to “View Report”

Crystal Reports

This section outlines the points that need to be taken into consideration when either writing new reports or modifying existing reports for use from ESP. Most of these have to do with making sure that reports are handled correctly when language translation is turned on for Crystal reports from within ESP. There are three conditions that must be met to enable language translation of Crystal reports when they are run from ESP.

- The Crystal runtime dlls, CRAXDRT.dll (version 8.5.0.674) and CRPE32.dll (version 8.5.3.979), must exist and be registered. Both of these ship with ESP. DLL
- The parameter INFClient/Section/Language Translation/Enable Translation must be set to “True”
- The parameter INFClient/Section/Language Translation/Translate Crystal Reports must be set to “True”

If these three conditions are met then ESP will translate the report using the process outlined below.

In order for any values to be translated there must be an entry in the “InfClient” language table for the value and there must be a “Site Text” value for that entry. If there is no entry in the language table, or if the “Site Text” field is empty for the value then it will be displayed on the report “as is”.

Overview of the Translation process

Translation of a report is done in several steps.

- A copy of the report is made
- All formula field in the report are processed
- All formulae associated with groups and sections (conditional formula) are processed
- All text objects on the report are translated
- All conditional formula attached to formulae, fields and text objects are translated.

Formula Field Translation

- Each formula field on the report is examined and any values enclosed in double quotes are translated.
- If the string “//func” is encountered then all text from that point to the end of the formula is not processed
- If a sting beginning with a double at sign “@@” is encountered it is assumed to be a placeholder for one of the ESP Crystal functions as in the following list:

Placeholder	
@@smallareafull()	Returns the full text of the small area unit from the Unit table
@@smallarea()	Returns the abbreviation of the small area unit from the Unit table
@@largeareafull()	Returns the full text of the large area unit from the Unit table
@@largearea()	Returns the abbreviation of the large area unit from the Unit table
@@smallweightfull()	Returns the full text of the small weight unit from the Unit table
@@smallweight()	Returns the abbreviation of the small weight unit from the Unit table
@@largeweightfull()	Returns the full text of the large weight unit from the Unit table
@@largeweight()	Returns the abbreviation of the large weight unit from the Unit table

@@smallvolumefull()	Returns the full text of the small volume unit from the Unit table
@@smallvolume()	Returns the abbreviation of the small volume unit from the Unit table
@@largevolumefull()	Returns the full text of the large volume unit from the Unit table
@@largevolume()	Returns the abbreviation of the large volume unit from the Unit table
@@smalllinearfull()	Returns the full text of the small linear unit from the Unit table
@@smalllinear()	Returns the abbreviation of the small linear unit from the Unit table
@@largelinearfull()	Returns the full text of the large linear unit from the Unit table
@@largelinear()	Returns the abbreviation of the large linear unit from the Unit table
@@currencysymbol()	Returns a dollar sign. To enable translation of this into the local symbol the “$” symbol will need to be defined and have a “Site Text” value in the translation table

In this case the returned value from the function will replace the placeholder in the string.

if {?UseTranslation} then

"Avg area per order (@@smallarea())"

//func

else

"Avg area per order " + EspUOMgetUnitInfo("abbreviation","area","small")

If the site in question had “sq m” as their small area abbreviation and translation was turned on the above would result in the following string being passed through to be translated:

"Avg area per order (sq m)"

And the translated value of the string would be displayed on the report. The line “//func” would cause the translation process to ignore the remainder of the formula.

Note that if translation is turned off then the first part of the formula would be ignored and the “else” portion would be executed while the report was being generated.

Group Name Translation

The process used here is identical to that followed by ESP when translating Formula Fields.

Text Object Translation

- All text objects in the report are translated
- It is not possible, with a text object, to conditionally translate part of the text. If there is text that is required to be displayed un-translated then one of the following methods can be used:
  - Do not have a “Site Text” value for the text in the language table”
  - Place the text in a formula field and place the string “//func” on the first line of the field
- DO NOT use Text objects with fields embedded in them. Due to limitations in the 8.5 version of Crystal, ESP will strip the embedded fields out and replace them with square brackets.

Conditional Formula Translation

The process used here is identical to that followed by ESP when translating Formula Fields.

Adding Reports to ESP

Reports are made available to users in two steps

- Placing the report and ini file in the appropriate directory. These directories are specified in the parameters section of ESP under “InfClient/Section/Directories/Reports”. The directories are searched in the order that they are listed in this parameter, this allows for site specific reports (or ini files) to be placed in a folder that will not be overwritten during an upgrade process. The report and ini files do not have to both be in the same directory if, for example, a site had customised defaults on a standard report they could place the customised ini file in the site specific directory (normally first in the parameter) and leave the report in the standard ESP reports directory.
- Making a parameter entry in the InfClient/Section/Reports area of Configuration. This can be done either manually or through running the SQL outlined below.

if (select count(id) from infparameter where value like 'report=Chep Pallets%') <= 0

BEGIN

EXECUTE infHelperInsParameter 'InfClient',

'Reports',

'all=Chep Pallets',

'report=Chep Pallets|app=INF|group=Pallet Reports|caption=Chep Pallets', 0,

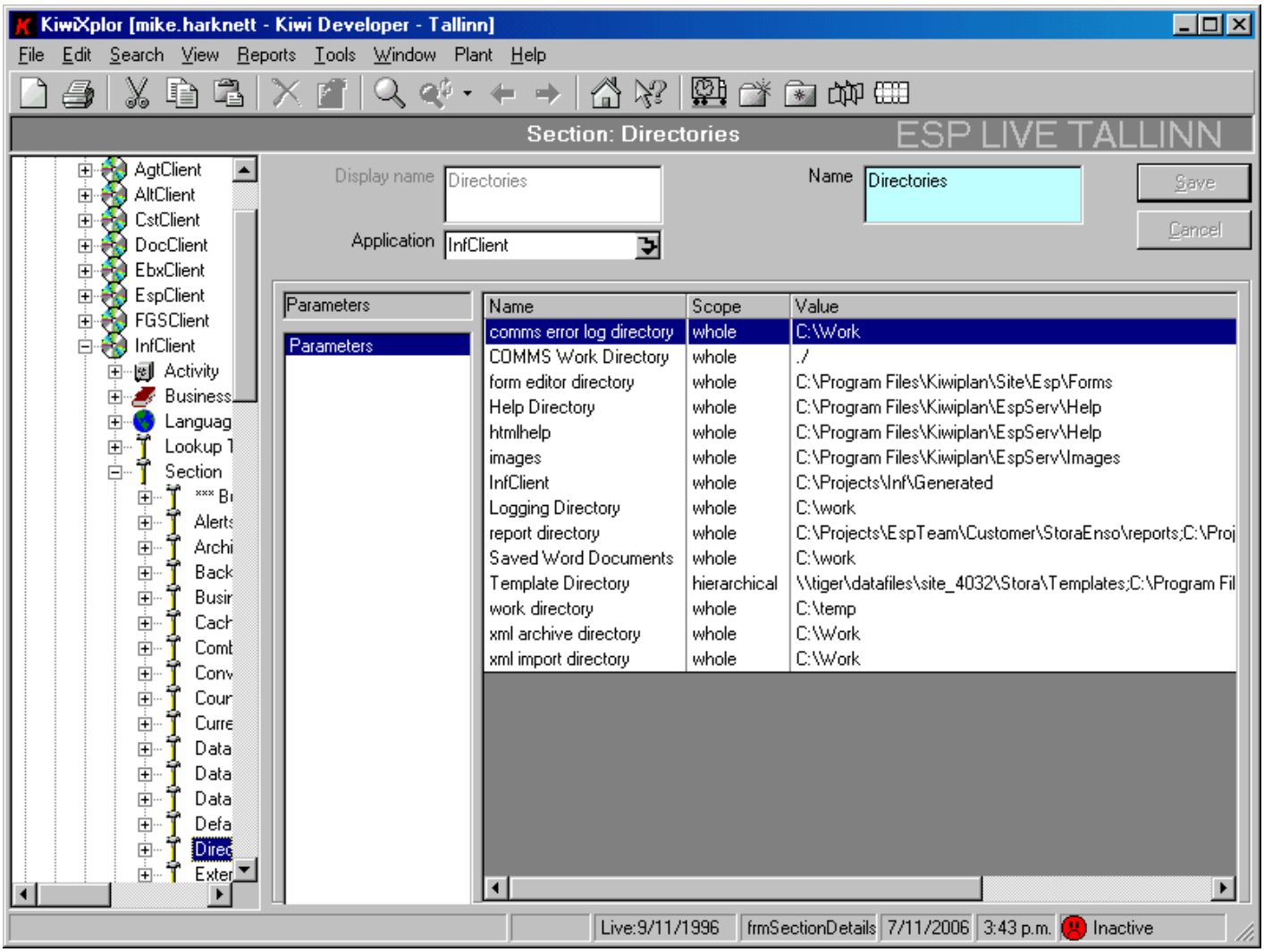
NULL

END

go

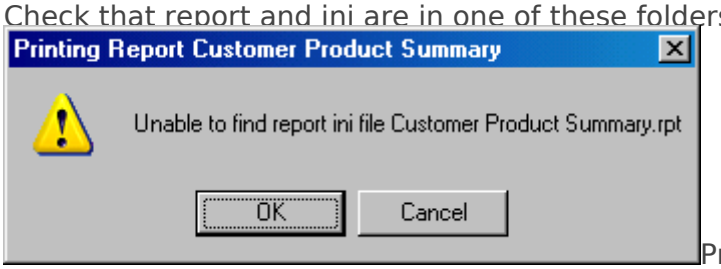
This SQL checks to see if the report already exists, if not it calls the stored procedure “infHelperInsParameter”, passing the following parameters:

- ‘InfClient’ – This is the Application under which the report is to be listed, all ESP reports should use ‘InfClient’
- ‘Reports’ – This is the parameter branch that reports are stored on
- ‘all=Chep Pallets’ – this is the “name” field value and indicates that the report is available whatever the current users context in ESP.
- ‘Report=Chep Pallets|App=INF|group=Pallet Reports|caption=Chep Pallets’ – this parameter consists of label=value pairs separated by “|” symbols
  - Report=Chep Pallets – the value to the right of the equal sign is the name of the rpt and ini files without their extensions
  - App=INF – the application group, should always be ‘INF’
  - Group=Pallet Reports – this is an optional value, if included it indicates that the report should be placed on a sub-menu called (in this example) “Pallet reports”
  - Caption=Chep Pallets – this is the value which will be displayed on the menu, it may differ from the true report name.
- ‘0’ - this is scope of the report parameter, 0 indicating a report that is globally visible.
- ‘NULL’ – This parameter is not used and should be left set as ‘NULL’



Support Scenarios:

Check that report and ini are in one of these folders



Problem:Can't find rpt or ini:

Solution

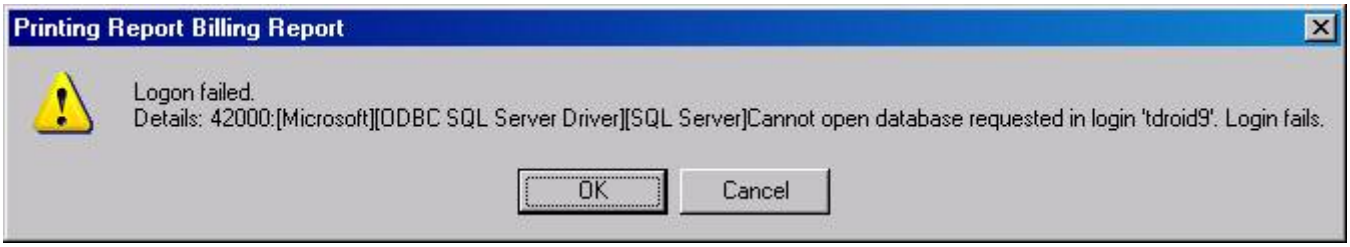
Check that the report and ini files are in one of the folders listed under

Configuration/Application/InfClient/Section/Directories/Report Directory

The rpt and ini do NOT have to be in the same folder, but files are used based on the ordering of the

folders.

Problem Login failed:

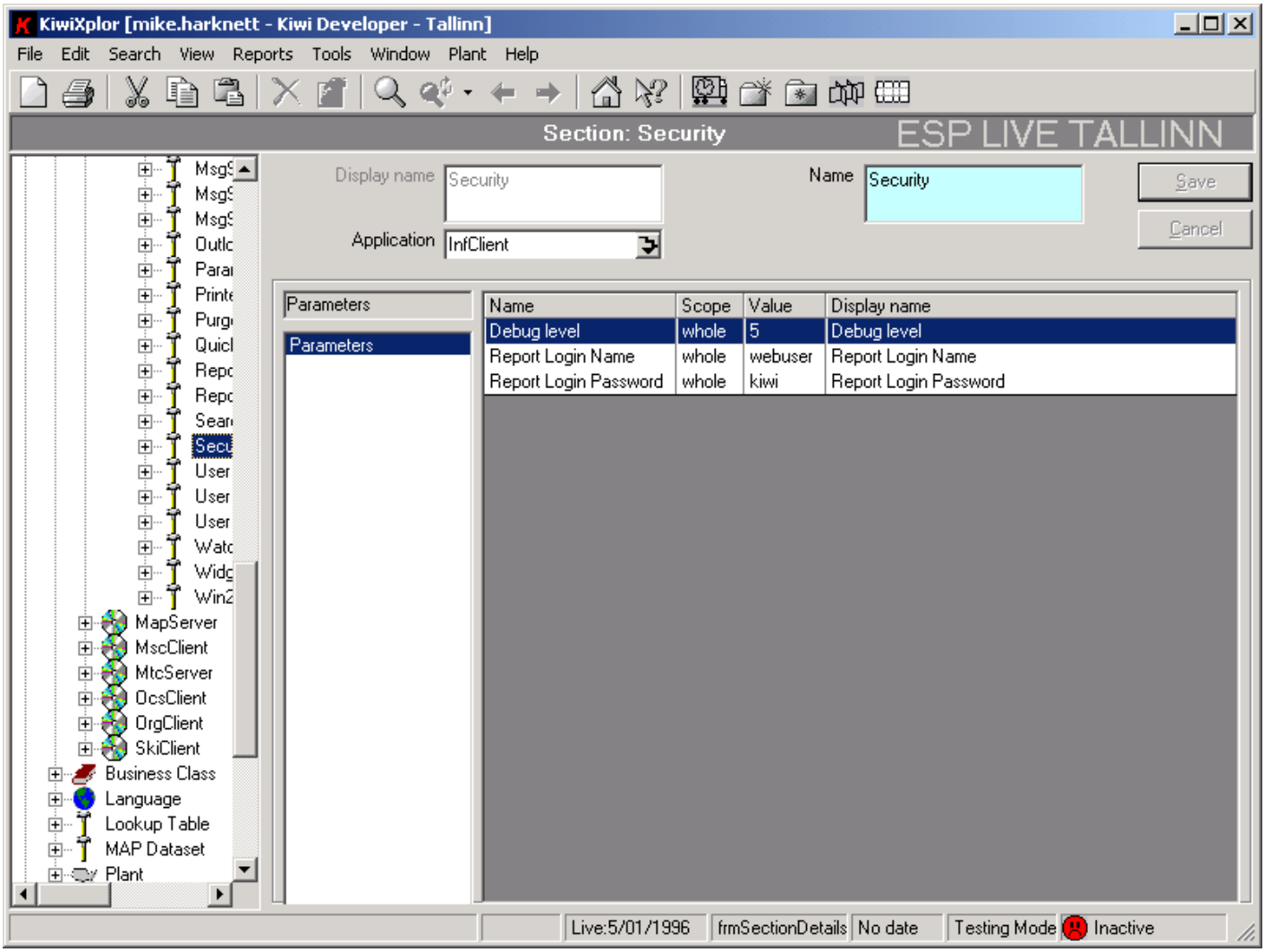


Solution: This is caused by the report user and/or password set in ESP not having correct permissions in SQL Server to access the ESP database. Ensure that the values set in the parameters

Configuration/Application/InfClient/Section/Security/Report Login Name

Configuration/Application/InfClient/Section/Security/Report Login Password

match a valid user in the ESP database.

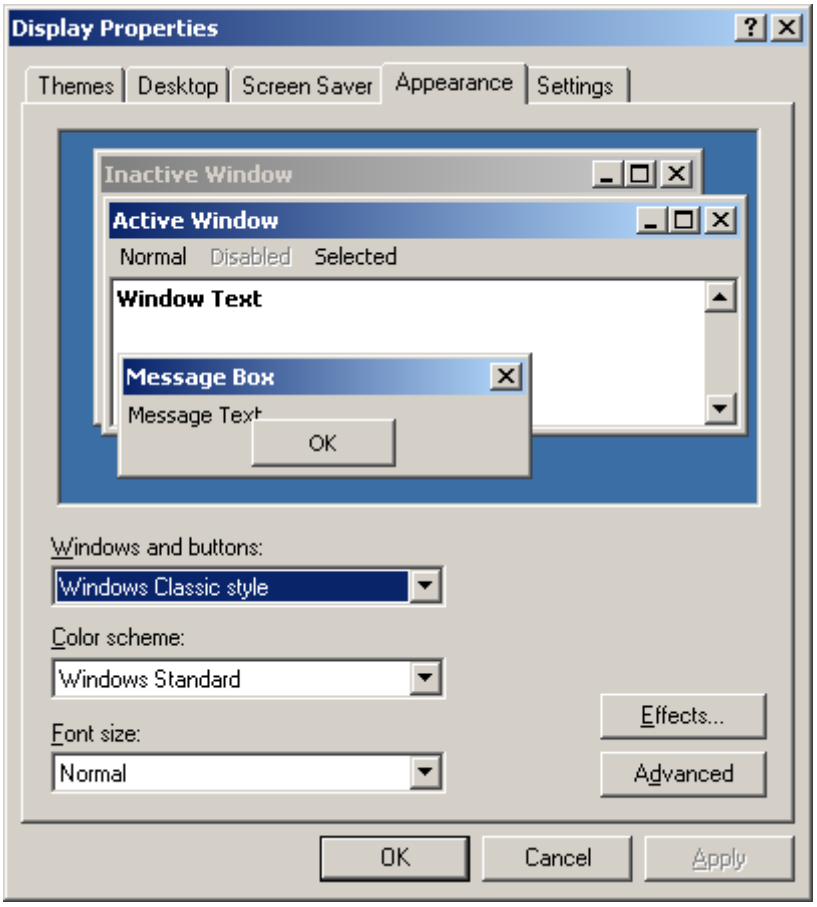


Ensure that this is a valid user in the ESP database and the correct password for that user

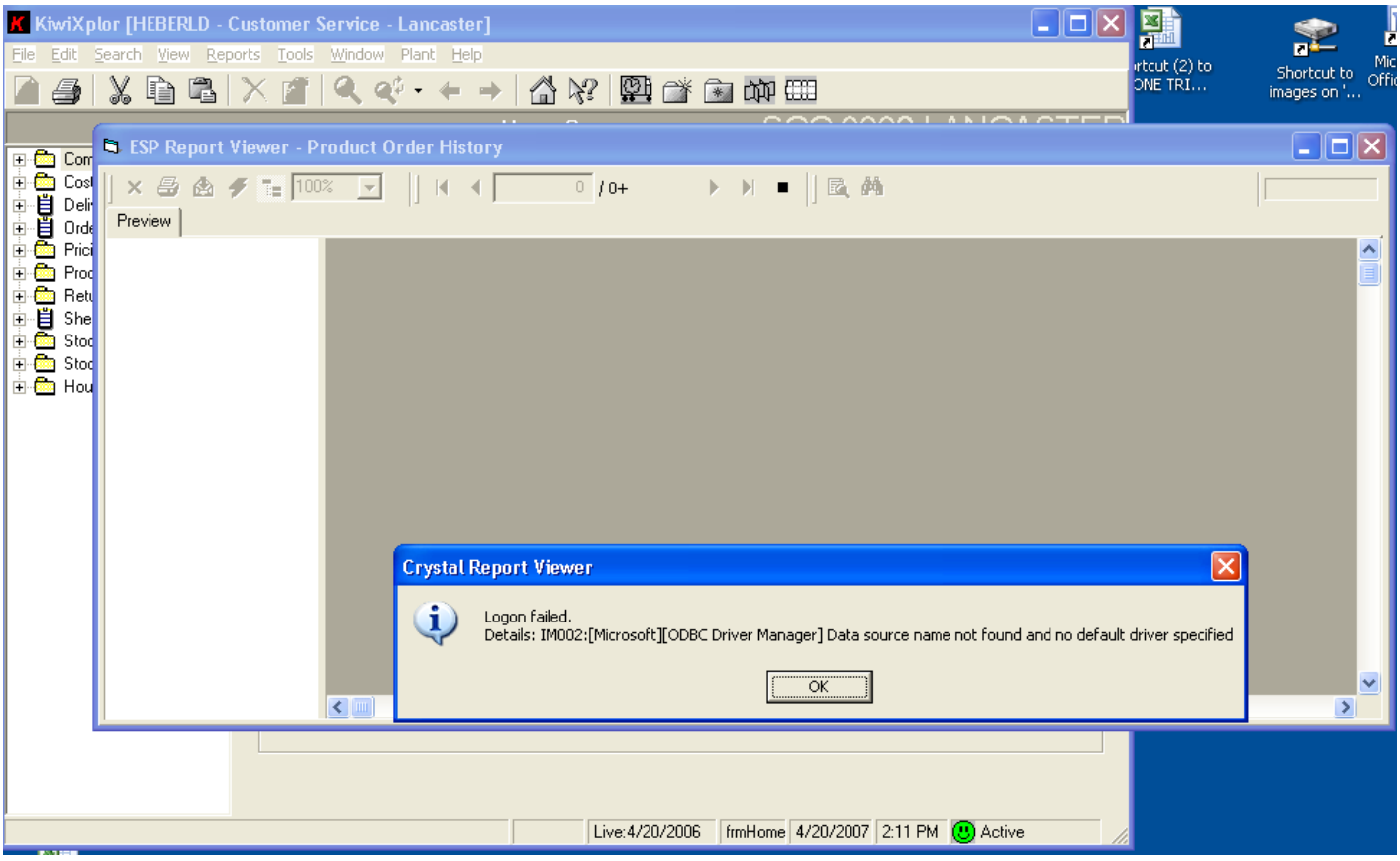
Problem: Unable to see labels on report options form Labels and background both set to the same colour

Solution:

This is caused by the colour settings in the Display Properties reached by right clicking on the desktop and selecting the “Appearance” tab. The sfest option is the “Windows Classic” or “Windows Standard” style, however other styles can be fine, it just may take some experimentation to find the correct one.



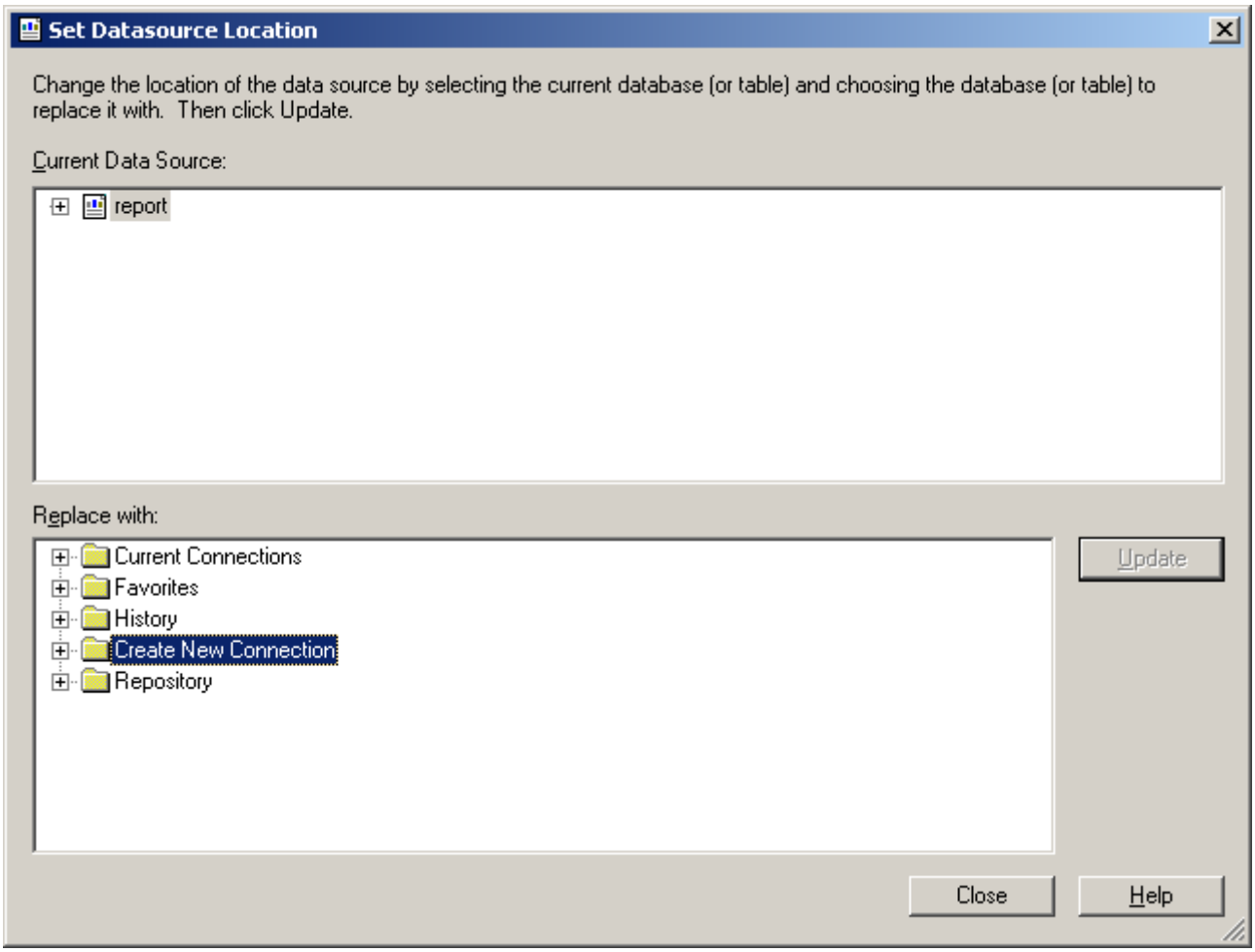
Problem: Unable to connect to database



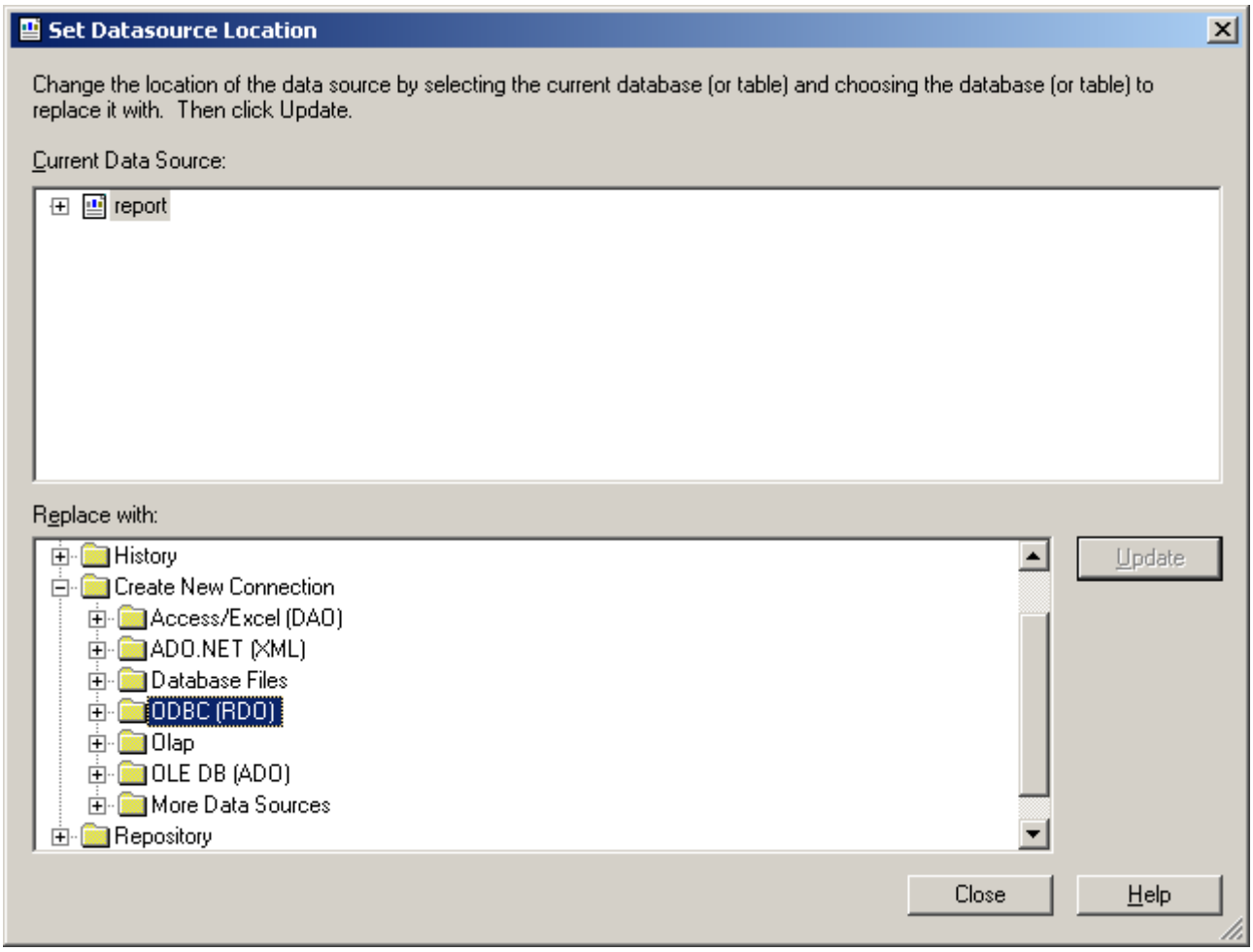
S

Solution: This is caused by an incorrect datasource used when creating the report. The following procedure should be followed to ensure that ESP can correctly set the datasource on the report at runtime.

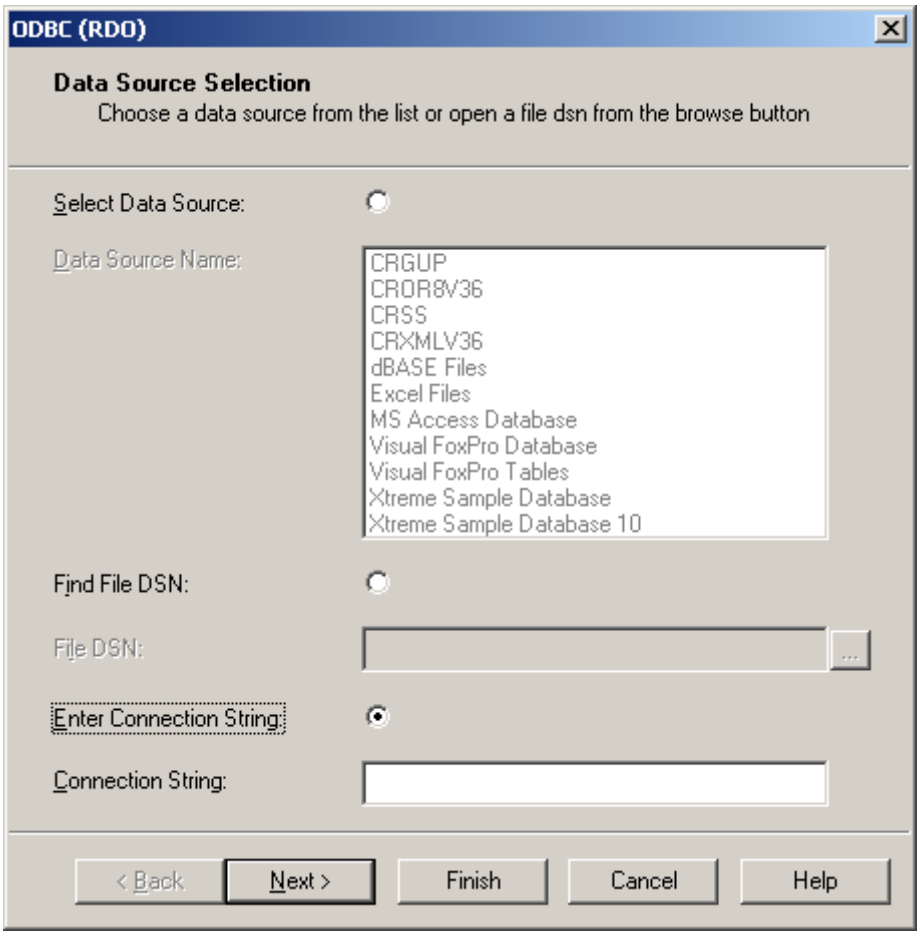
In Crystal Reports open the Database/Set Database Location menu option



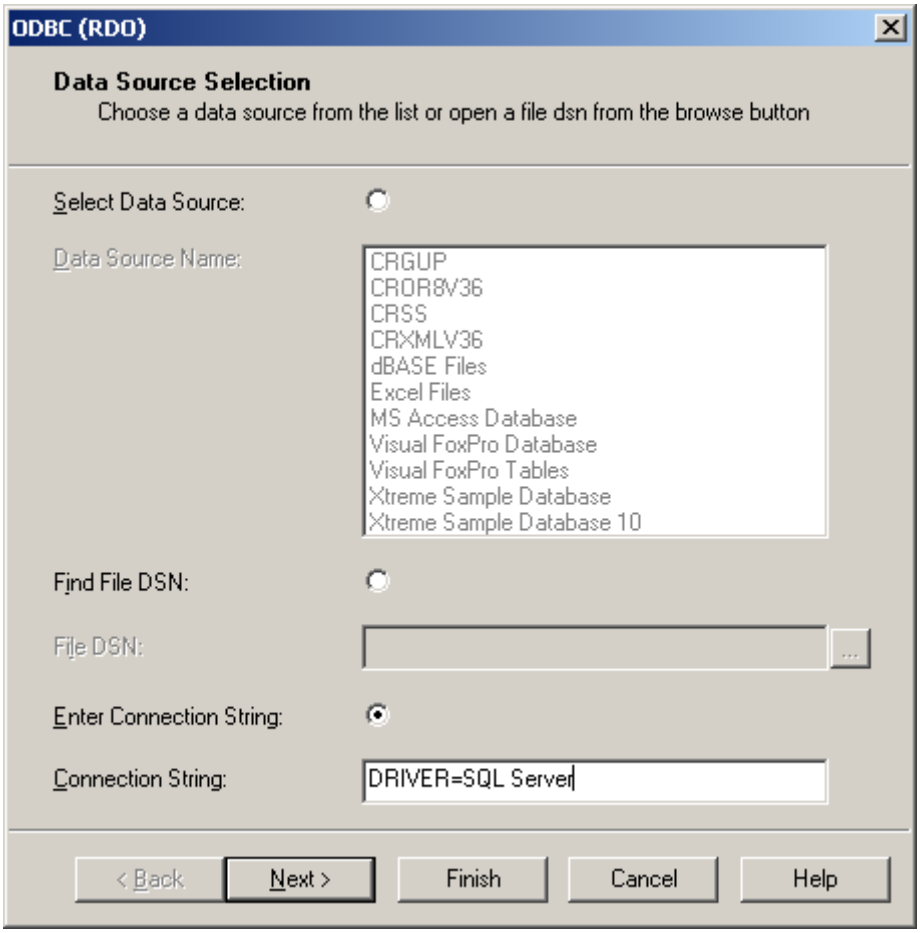
Select “Create New Connection”



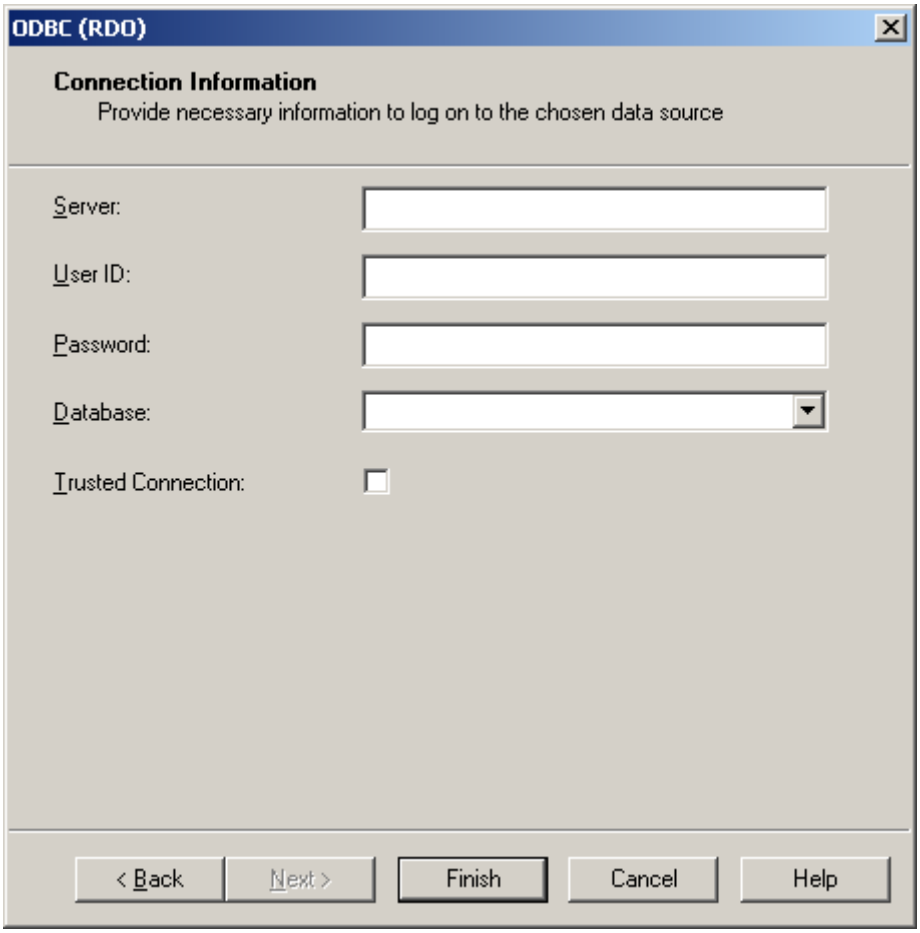
Select “ODBC (RDO)”



Select “Enter Connection String”



Enter the string “DRIVER=SQL Server”



Fill in the details:

1)Server is the database server

2)User ID and Password are the values held in the ESP Parameters under

Configuration/Application/InfClient(or Infrastructure)/Section/Security/Report Login Name

Configuration/Application/InfClient(or Infrastructure)/Section/Security/Report Login Password

3)Database is the ESP database name

DO NOT check the “Trusted Connection” check box

Click on “Finish” and if all details have been entered correctly you should be able to select tables, views or stored procedures from the database.

Linked Server

The target audience for this document is technical and support personnel.

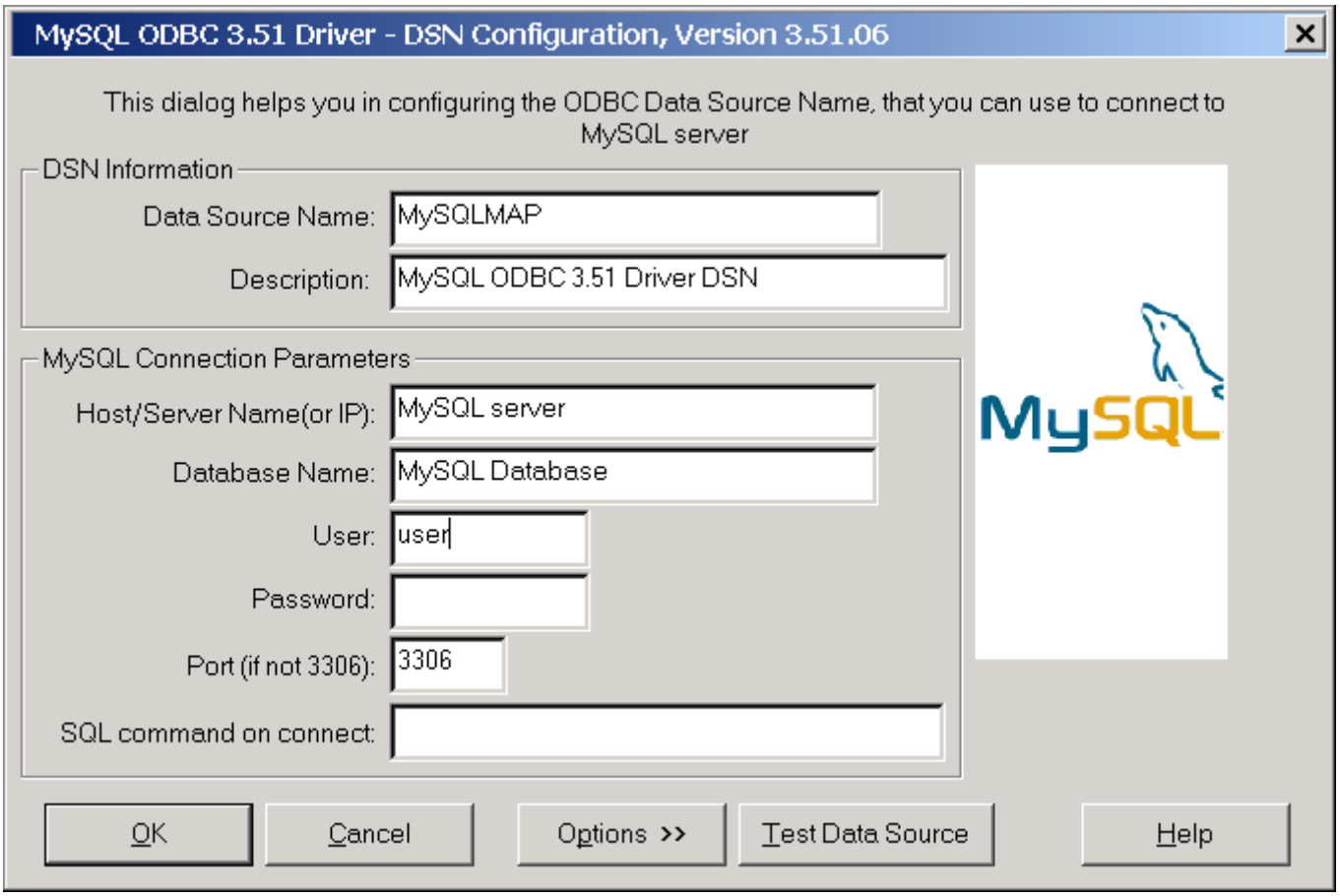
Manual Setup

ODBC DSN

On the SQL Server machine

1) Install the MySQL ODBC Driver

2) Set up a System DSN



Name of MySQL server

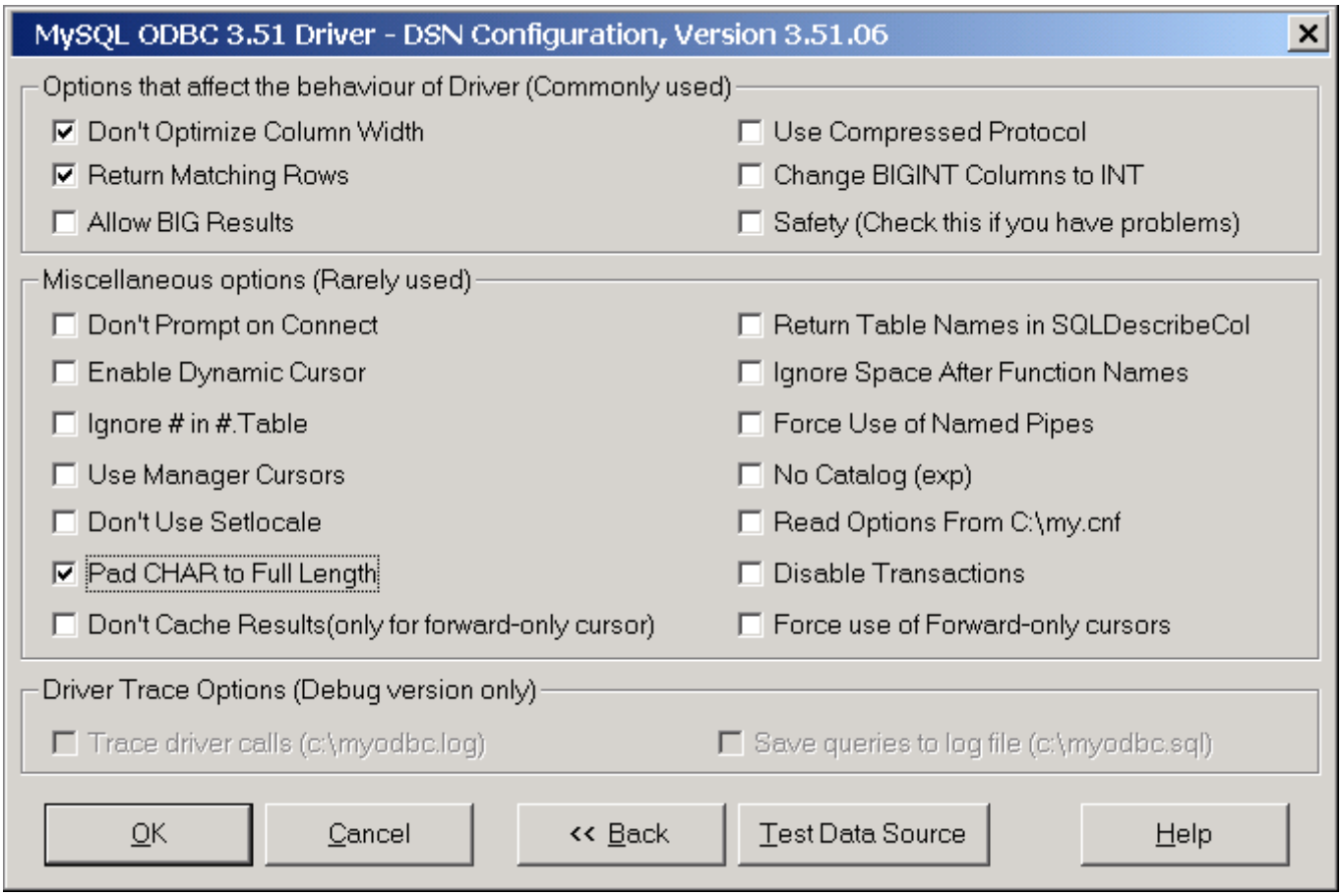
Name of database to link to

Name of user, must have privileges on the database being linked to

Leave this

3)Set options:

These are defaults-leave them



Set this

Once these details click on “Test data Source” if the test is successful click on OK if not there is a chance that the user has not had access granted on the MySQL database.

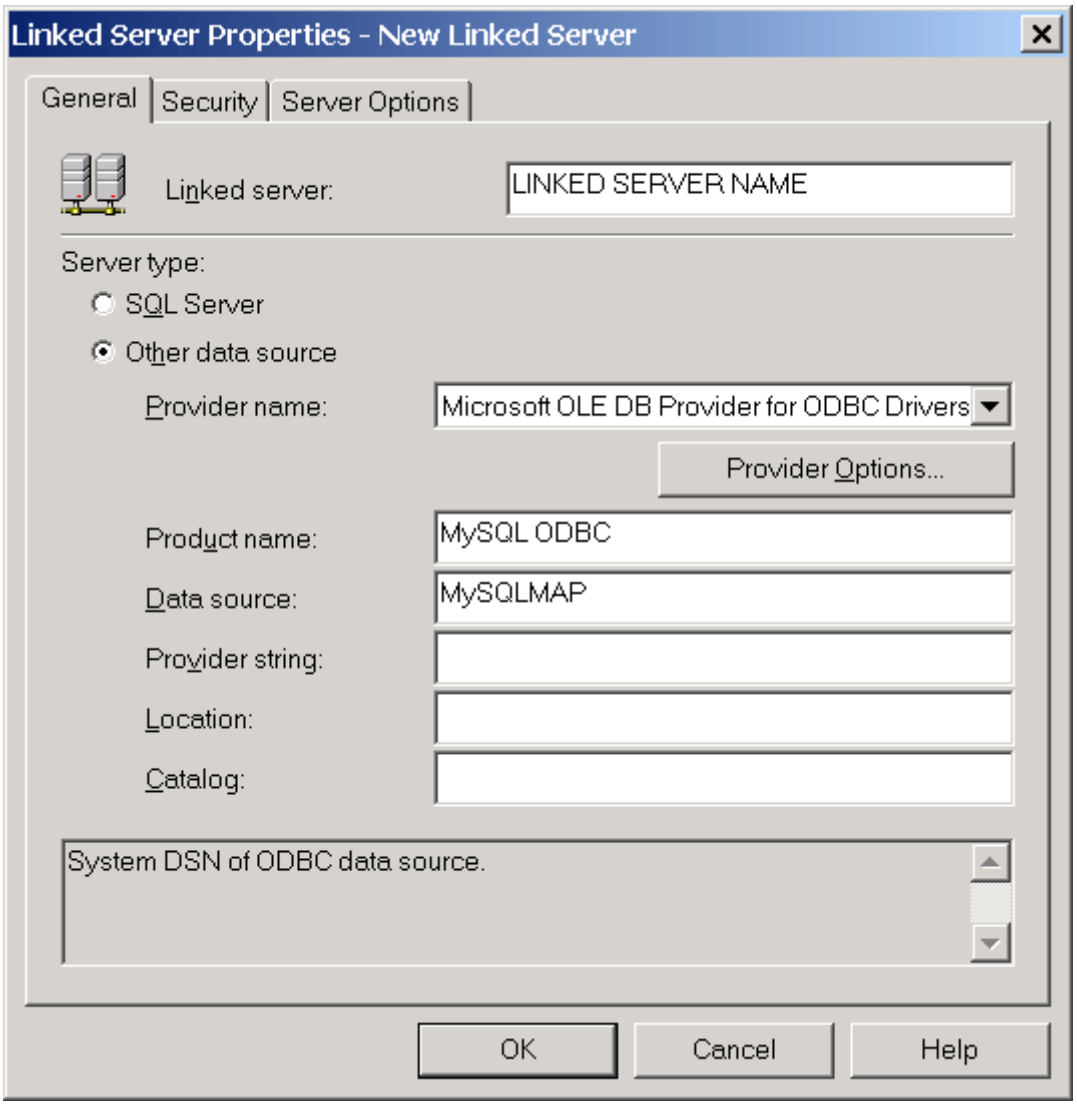
Linked Server Setup in SQL Server

After this is done the Linked Server needs to be setup. There are two ways of doing this, both must be carried out on the SQL server machine.

Manual Setup

Expand the Security branch of the Explorer tree in Enterprise Manager, then right click on the “Linked Servers” node. Select “New Linked Server” and the following dialog will be displayed:

This can be anything

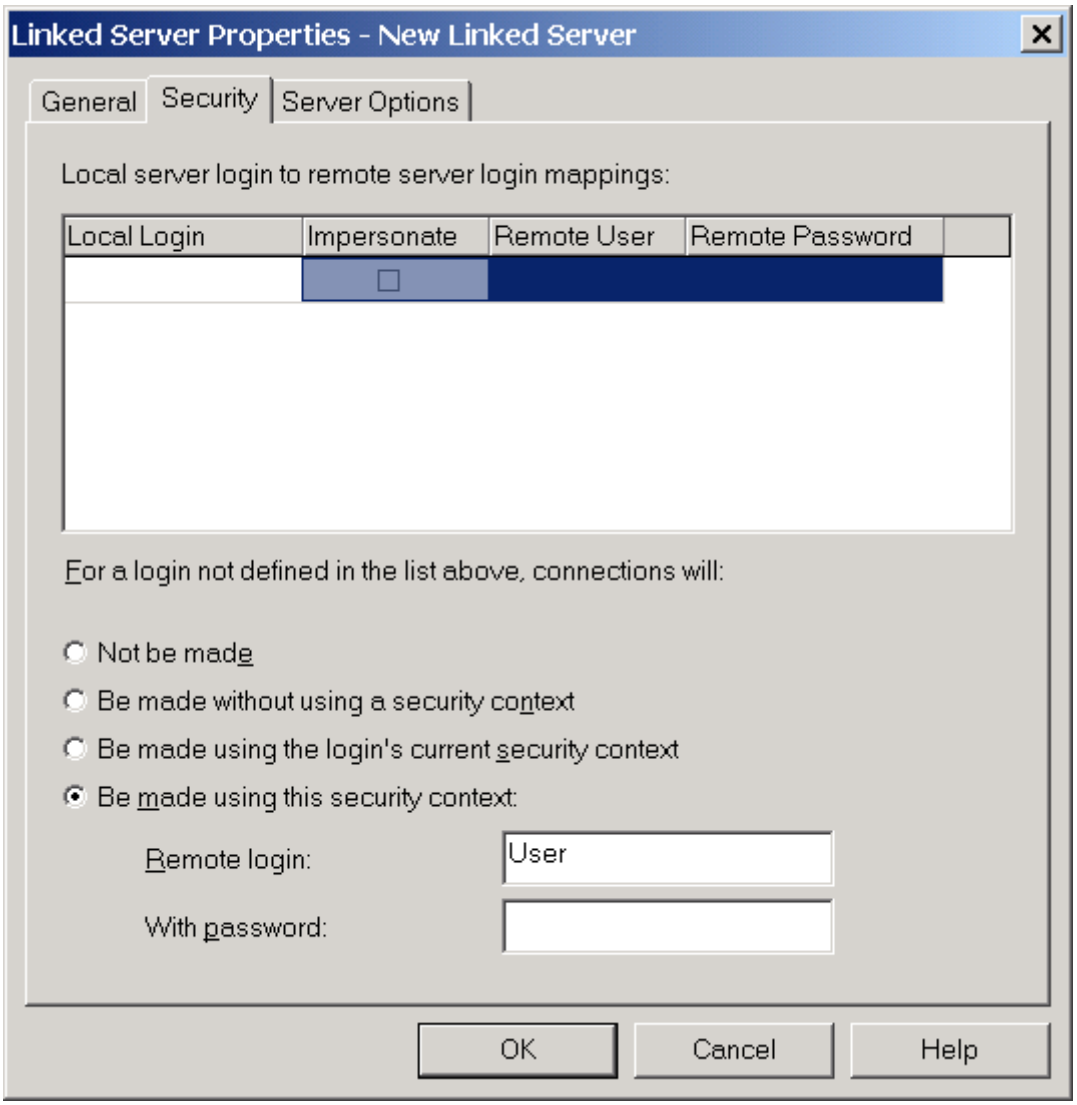


Select from drop down list

This can be anything

Name of System DSN set up above

Then open the “Security” tab and select the “Be made using this security context” option.



Enter the name of the user defined in the DSN as the “Remote login" as well as any password defined in the DSN.

Click on OK and that should be it.

Setup Using Scripts

ODBC DSN

It is possible to create a registry script that will create a system DSN, however it is not recommended as

the possibility of corrupting the registry exists.

Linked Server Setup

The linked server in SQL Server can be created using the system stored procedure “sp_addlinkedserver”. The following SQL commands will accomplish this. See notes after the SQL for more explanation.

/*

Script to add linked server, requires that a ODBC System DSN be set up first on the actual SQL Server

machine with the name "MAPMYSQL"

*/

EXEC sp_addlinkedserver 'MAPMYSQL', 'MySQL', 'MSDASQL', Null, Null,

'Driver={MySQL ODBC 3.51 Driver};DB=mysqldb;

SERVER=mysqlserver;option=512;uid=mysqluser;pwd=mysqlpassword;'

EXEC sp_addlinkedsrvlogin 'MAPMYSQL', 'false', NULL, 'mysqluser', 'mysqlpassword'

EXEC sp_serveroption 'MAPMYSQL', 'data access', 'true'

EXEC sp_serveroption 'MAPMYSQL', 'use remote collation', 'true'

EXEC sp_serveroption 'MAPMYSQL', 'connect timeout', 0

EXEC sp_serveroption 'MAPMYSQL', 'query timeout', 0

/*

End script

*/

NOTES:

1. The value (MySQL ODBC 3.51 Driver) will need to be changed if a later version of the MySQL ODBC driver is used.
2. The values for DB, SERVER, uid, pwd in the sp_addlinkedserver line will need to be modified for each site. These should match the values used to set up the ODBC System DSN.
3. The values 'mysqluser' and 'mysqlpassword' in the sp_addlinkedsrvlogin will need to be edited to

match the values used for the 'uid' and 'pwd' parameters in the previous line

Once this is done data is accessed through the SQL Server “OpenQuery()” function. In theory linked servers support the 4 part naming convention for sql , however it appears that the 3.511 version of the MySQL ODBC driver does not. So the “OpenQuery” function is the only way of accessing the MySQL data from within SQL server.

The “OpenQuery” function can be used to create views so that the fact that the data is from a linked server is transparent to the users.

Create view MAP_ULOADC as

select u.* from openquery(MAPMYSQL, 'select * from ULOADC') as u

go

Be aware that in the MAP dataset the following tables have blob or tinyblob fields and views for them should be created by explicitly selecting each column and casting the blob column as char

Table blob column

COMTRN body

LABELS body

LBMISC misc_data

SCHCOM body

XLATEP xl_body

XLDEFN body

The example below shows how this is done with the XLATEP table.

create view dbo.MAP_XLATEP as

select x.* from openquery(MAPMYSQL,

'SELECT xl_prefix, xl_key,

cast(xl_body as char) as xl_body,

xl_system, xl_group

FROM XLATEP') as x

Linked Server Name Parameter

A parameter called “Linked Server Name” has been added to the section “Report Parameters” under the Inf Client application in ESP. This is to allow individual sites the freedom to name their linked servers however they wish.

The code below shows how the linked server name can be retreived from the parameter table then used to build an openquery sql statement.

```
declare @linkedservername nvarchar(50)

--get the linked server name

select @linkedservername = isnull(value, 'MAPMYSQL') from infparameter where name = 'Linked Server Name' and sectionid in (select id from infsection where name = 'Report Parameters')

--build the sql statement using the retrieved linked server name

set @sql = 'declare qcurs cursor for select isnull(qty,0) from openquery(' + @linkedservername +

','select sum(quantity_good_out) as qty from FACTRY where job_number = ''' + @job +

''' and machine_number in (1180,2180,3180)

and finish_datetime < ''' + convert(nvarchar(30),@dte,120) +

''' group by job_number'')'

-- now exec the statement

exec(@sql)
```

This functionality can be used in stored procedures but not in SQL functions or Views.

Revision #2
Created 4 June 2025 23:57:19 by Steve Ling
Updated 5 June 2025 00:30:16 by Steve Ling