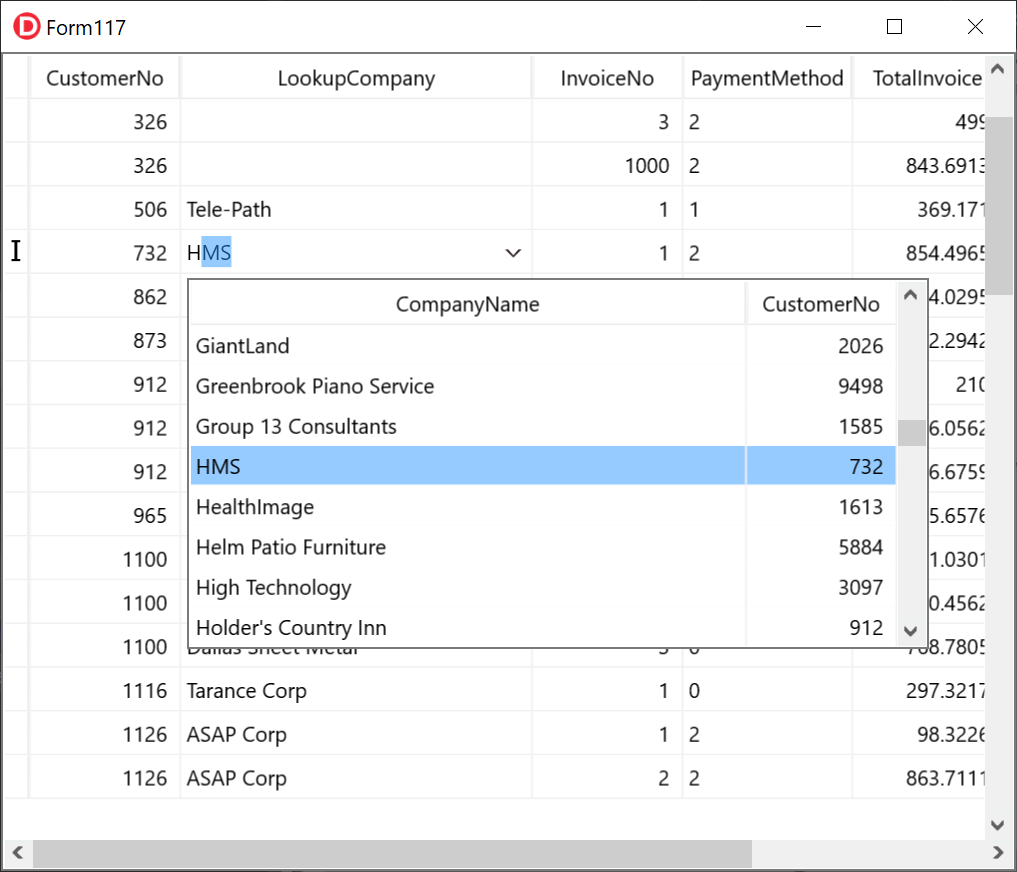
**Using a FirePower LookupCombo in a DataGrid**

Today we are going to talk about one of the most useful components in FirePower, the TwwLookupComboEdit. This component allows you to display values generated from a related table in a dropdown list, and then after the user makes a selection the value is filled into the edited data source. Here is a screenshot of what we are going to accomplish today. We will embed the LookupCombo in a FirePower grid.



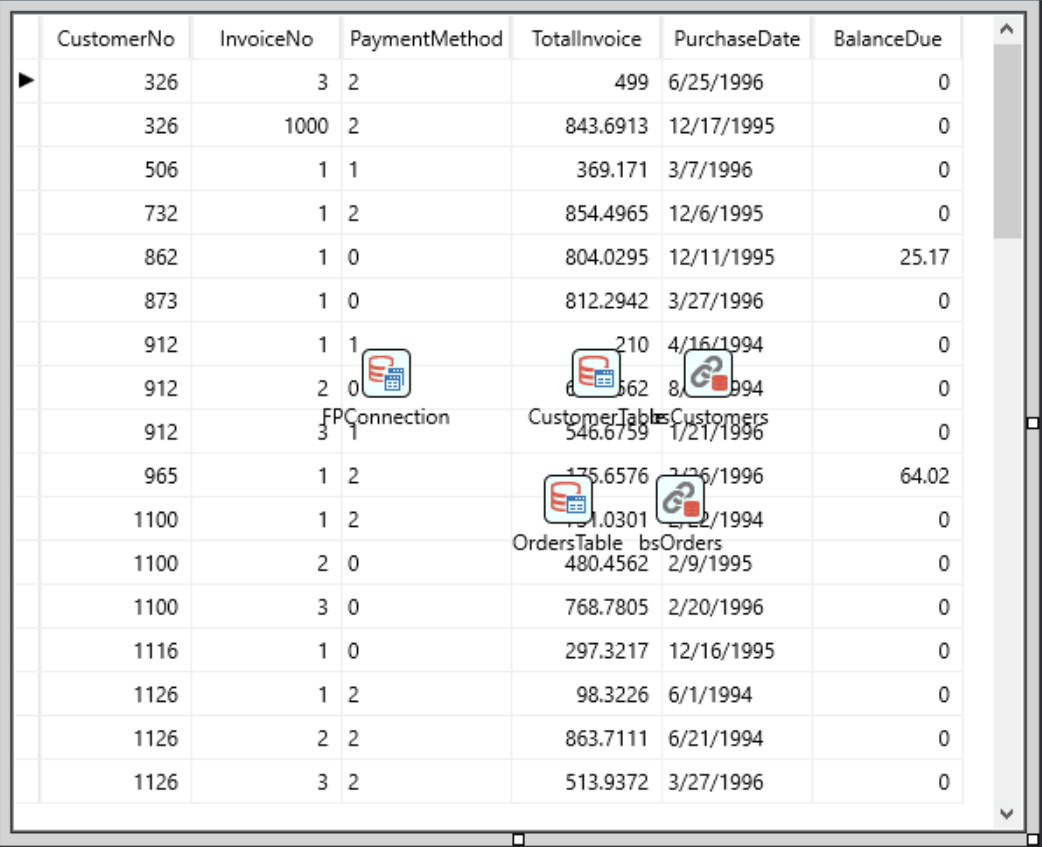
Since we are going to do this in a grid, it will be a 3 step process. Here is a summary of the steps we will be going over.

1. Setting up the grid to use the Orders table in our database
2. Setting up the grid to display the descriptive text (Company Name) from our CustomerTable
3. Associating a LookupComboEdit control with the lookupfield in the grid.

# Setting up the grid to use the Orders table in our database

To see this, let us quickly set up a demo using a TwwDataGrid. We are going to use our FirePower sqlite demo database for this.

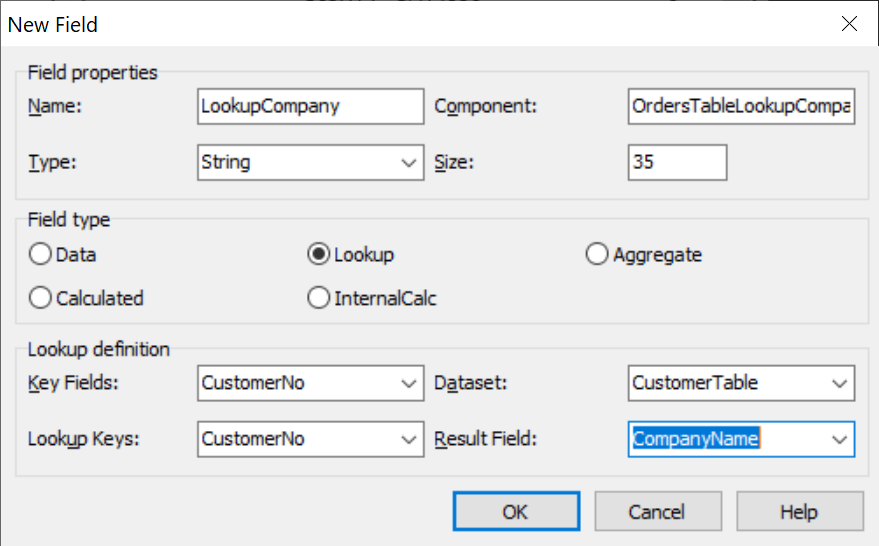
1. Drop in a TFDConnection, and 2 TFDTables. Name the first TFDTable as OrdersTable, and the second TFDTable as CustomersTable, and the TFDConnection as FPConnection
2. Set up the database connection
   1. Dbl-click the FPConnection object and dropping down the combo labelled *Connection Definition Name* and selecting FirePowerSQLLite. Please note that this selection is only available after you have installed FirePower Trial or Release version.
   2. Now click the Test Connection button to make sure the connection is working. After confirming, then click ok to close the dialog.
   3. Click on FPConnection and set LoginPrompt to false so we don’t have to keep entering our credentials. Also you can set FPConnection.Connected to True.
3. Connect your tables we will be using
   1. Select OrdersTable and select the *TableName* as *Orders.*
   2. Select CustomersTable and select the *TableName* as *fdCustomer*. Also set the FormatOptions.SortLocale to 0. Otherwise you may get duplicate records during incremental searching.
4. Create bindsources for your tables by right-clicking each one and selecting *Add BindSource*. We need these in order to connect our grid to the tables. Name the bindsource connected to the *OrdersTable* as *bsOrders*, and the bindsource connected to the *CustomersTable* as *bsCustomers*
5. Activate your tables by setting their Active property to true.
6. Drop in a TwwDataGrid and then set its DataSource property to bsOrders. Now let us align the grid, by setting Align to alClient. After this step, our form should like like this.

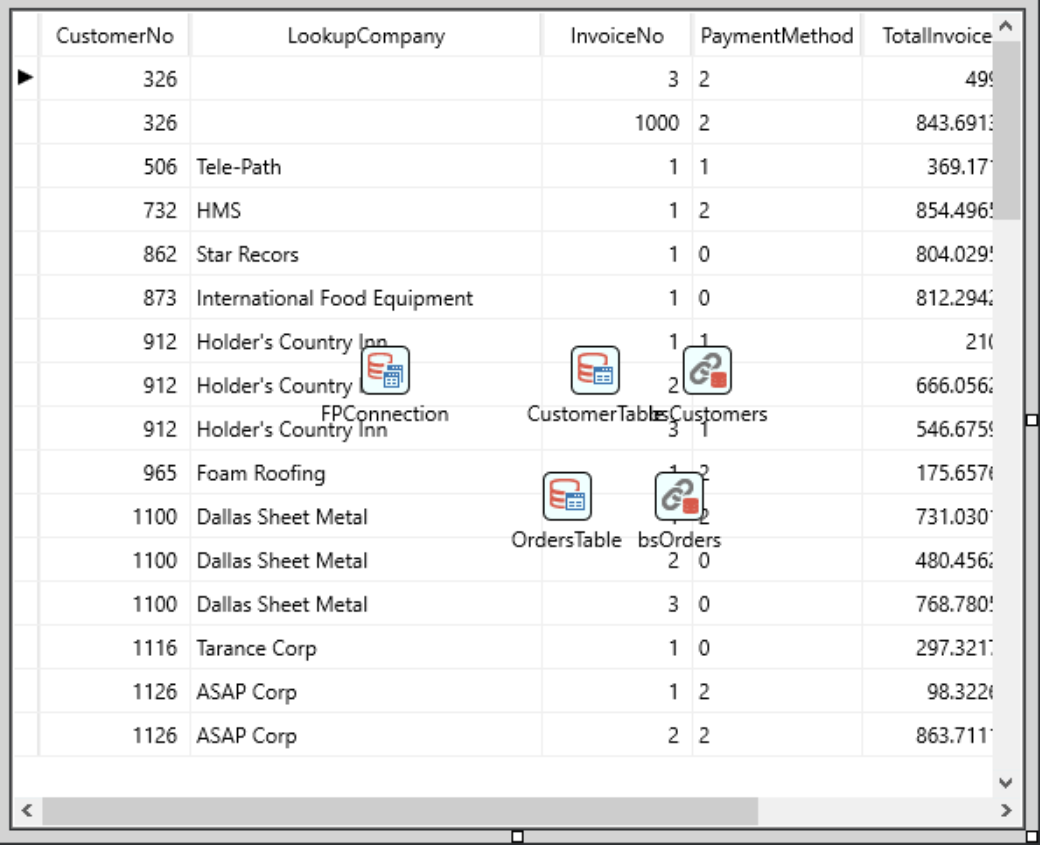


In this grid we are displaying a list of orders. We assume our OrdersTable is normalized, meaning it does not contain redundant information from a related table. What we would like to do is to lookup an existing customer and fill the Customer No into the OrdersTable. Even better, we would like to lookup and display using a more meaning descriptive name (such as Company Name), but then fill in the CustomerNo value internally without requiring the user to enter this. This allows us to use a normalized database.

# Setting up the grid to display the descriptive text

We would like to have the grid display the CompanyName from the CustomersTable instead of the CustomerNo from the OrdersTable. This can be accomplished by creating a lookupfield in our OrdersTable. Here are the steps to do this.

1. Dbl-click your OrdersTable and then right click the blank are and select *Add All Fields*. We need to do this before we can create our lookup field.
2. Now right click the field list again, and now select *New Field*. This will bring up dialog where we can create our lookup field. You can assign the properties as in the dialog below. Click OK to save your changes.
3. Now let us add this newly created lookup field to our grid
4. Dbl-click the grid, and select *Add Field*, followed by selecting the field *LookupCompany*. Click OK to save your selection. Now your grid should show this field from your CustomersTable



# Associating a LookupComboEdit control with the lookupfield in the grid

Our last step is to use the FirePower TwwLookupComboEdit to display the list of companies that the user can select from when editing within the grid. Here are the steps to accomplish that.

Drop in a TwwLookupComboEdit from the component palette and set the following properties for it.

Name = ‘*lkCompanyName’*

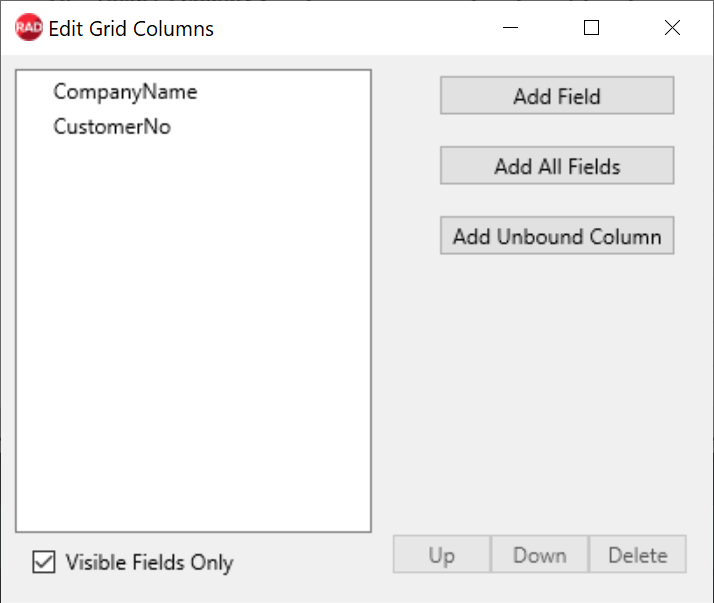
LookupSource = bsCustomers

LookupField = 'CustomerNo'

When used outside a grid, you would need to right click the component and select *bind visually* to bind to the CustomerNo field in the OrdersTable. However we are going to use the grid to bind the lookupcombo instead so we can skip this step here. Here is how we do that.

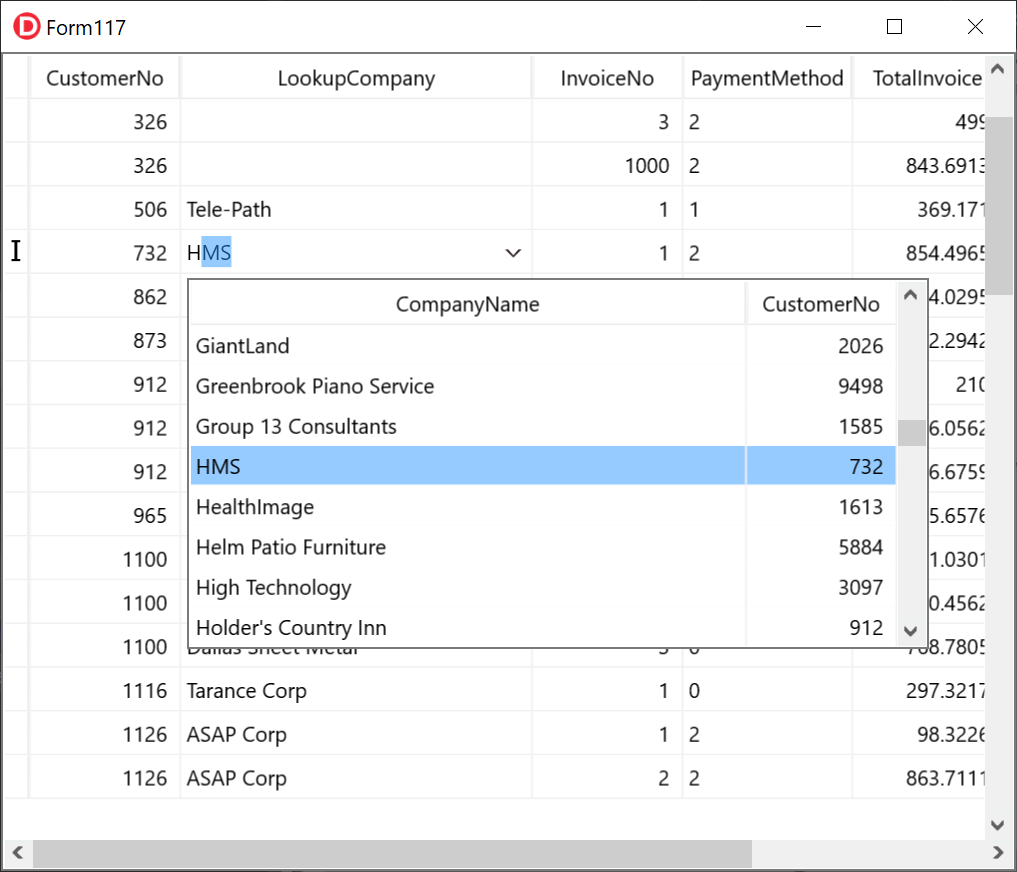
1. Dbl-click the grid to bring up the columns editor
2. Select the *LookupCompany* column and go to the object inspector and assign the *CustomControl* property to *lkCompanyName*

Our final step is to define which fields of the lookupcombo we wish to display in the dropdown list as well as the edit portion of the control. To configure what is in the dropdown list we can dbl-click the lookupcombo (lkCompanyName) and select the columns we wish to include. For this demo, let us include the columns *CompanyName* and *CustomerNo*. To make this easy, let us first select all the fields, and then click the Delete button to remove them. Now click add fields and select CompanyName and CustomerNo in that order. **Please note that the first column in our list is the column that is displayed in the edit portion of the lookupcombo.**



Now close the dialog, and we are ready to run our program. It should appear now exactly as the screenshot at the top of this document. Note the following behavior we inherit.

1. When a selection is made, the CustomerNo value is filled into our OrdersTable’s CustomerNo field.
2. Incremental searching on the lookuptable is automatically supported



\*\* Please note that when running the program and editing with SQLite, you should close delphi first as it will only allow once instance when editing the table. Your other alternative is to change your application so that the tables are opened at runtime instead of both design and runtime. If you choose this route, you will need to make your TConnection’s Connected property false in the IDE, and then add the following code in your form’s OnCreate event.

FPConnection.active:= true;

CustomerTable.active:= true;

OrdersTable.active:= true;

# Customizing the dropdown list’s appearance of the LookupComboEdit

The LookupCombo has great versatility in controlling the appearance of the dropdown list. We will highlight a few of these

Remove the titles from the list – Set DropDownGrid.Options.dgTitles to false

Changing the number of rows in the dropdown list – Set DropDownCount

Changing the list selection to be from a dialog instead of a dropdown list – Set the lookupcombo’s *LookupOptions.PopupType* to *UseDialog*