|  |  |
| --- | --- |
|  | **Unidev** |
|  | **Unified Development, Inc.**  16690 Swingley Ridge Road  Suite 260  Chesterfield, MO 63017  (636) 532-4424 |

|  |
| --- |
|  |
|  |

**Duncan Solutions, Inc**.

Custom Grid Options

May 2014

Revision 1.00

**Prepared by:**

Ron Howard

Caleb Miller

Unified Development

**Table of Contents**

[1. Overview 4](#_Toc388339523)

[1.1 Goal 4](#_Toc388339524)

[2. Target Audience 5](#_Toc388339525)

[3. Database 6](#_Toc388339526)

[3.1 Tables 7](#_Toc388339527)

[3.2 Stored Procs 8](#_Toc388339528)

[3.3 Adding Grids 10](#_Toc388339529)

[3.4 Adding Grid Columns 12](#_Toc388339530)

[3.5 Steps 12](#_Toc388339531)

[3.5.1 Update Customer Grid “Customer Id = 0” 12](#_Toc388339532)

[3.5.2 Update the Entity 13](#_Toc388339533)

[3.5.3 Update the MVC View 13](#_Toc388339534)

[3.5.4 Update DB View (if applicable) 14](#_Toc388339535)

[3.5.5 Update Customer Grid Options 14](#_Toc388339536)

[3.5.6 Testing 15](#_Toc388339537)

[4. UI Update 16](#_Toc388339538)

[4.1 Current 16](#_Toc388339539)

[4.2 Updates – Duncan 16](#_Toc388339540)

[4.2.1 Actions 18](#_Toc388339541)

[5. Additional work 20](#_Toc388339542)

**Revision** History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Author | Description | Revision Date |
| 1.00 | Caleb Miller | Initial version | 05/13/14 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

## Goal

The purpose of this document is to explain the change to the customer grids in the system and how the administrators of a customer will define the column order and visibility on a per grid basis. Currently these options are inserted directly into the database via a master file stored procedure called sp\_InsertAllCustomGridOptions.

The application also currently uses versioning on each grid to define the available options for selection for each grid. This update will be removing versions from the system for customer grids, and the new interface will allow the administrator users to interact and update each customer’s grid on a case by case basis, therefore nullifying the need for a versioning system.

For a review of how the grids work currently, see the Grid Management document in TFS.

There are two main sections that will need to be updated

1. Database and data access
   1. This update removes a few tables and stored procedures, adds a few, and updates the data context of the application.
2. Customer Grids tab
   1. This tab will be updated to reflect the change to allow administrators to modify each grid for each customer.

This document does not cover how to add grids to the system, which is covered in the Grid Management and Kendo Grids document. This document will only be covering the updates to the data access of the Customer Grids section and the associated administration tab.

# Target Audience

The target audience of this document is person or persons who have:

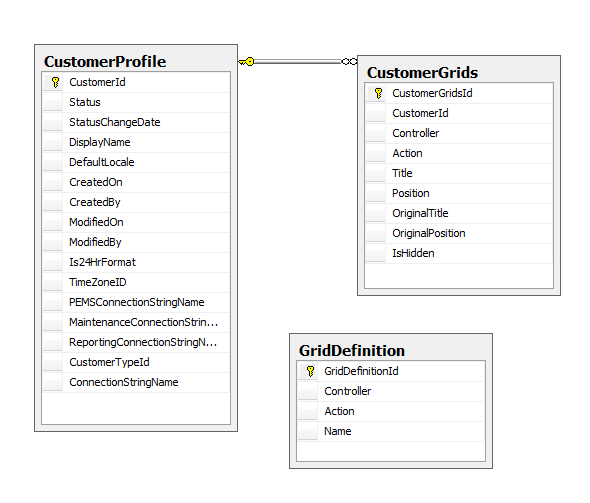
1. Experience in the following technologies:
   1. Javascript / Jquery. This is used heavily on each grid.
   2. C# /.Net / MVC
   3. Working knowledge of Telerik and Kendo MVC
   4. T-SQL, SSMS, SqlServer, Stored Procedures and Views
2. The user of this document has a full understanding of the Duncan PEMS project. This includes database and system architecture knowledge, desired business rules of the application, etc.
3. Microsoft SQL Server administration and understand rights, database creation and administration, and are able to use either SSMS or SQL command line interface. The user will need the ability to create and maintain stored procedures and views.
4. Visual Studio 2012

# Database

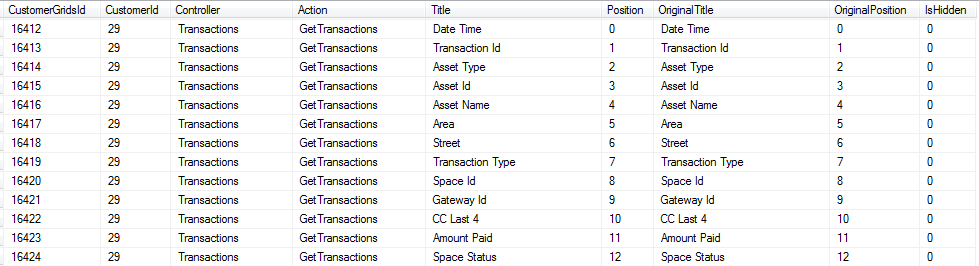
The PEMS RBAC database is used for storing all of the data needed. Here is the **current** database diagram:



And here is the updated DB diagram:

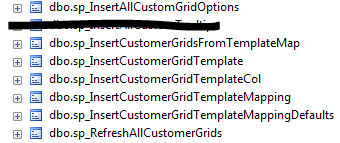


## Tables

1. **Customer Profile**: **THIS TABLE IS NOT CHANGING.**
2. **Customer Grids**: **IMPORTANT**: the table that holds the finalized grid data used for each grid on the site. The grids in the web application will use this table to pull the visibility, title, and order of the columns for each grid in the system. This table is the only table the Kendo grids use. The goal is to input the correct data to this table so the application works correctly. All other tables and stored procs are support items to facilitate keeping this table up to date with the correct data
   1. 
   2. **THIS TABLE STRUCTURE IS NOT CHANGING**
   3. This update will insert all of the available options for all of the grids in the system into this table with a Hidden = 0 and a customer ID of 0. This will allow us to reset a customer grid to a default state and also allow us to insert new customer data based off of the customerID = 0 data in this table.
3. **Customer Grid Template**: This table will be removed.
4. **Customer Grid Template Col**: This table will be removed.
5. **Customer Grid Template Map**: This table will be removed.
6. **Grid Definitions**: This update will be adding a table called Grid Definitions. The purpose of this table is to allow Developers to name a controller / action combination into something more user friendly. For example, instead of a grid being called “Asset GetItems”, we can give it a name of “Asset Inquiry” instead. This is the only purpose of the table and does not affect the customer grids at all.

## Stored Procs

The following is a list of current stored procedures the application uses to maintain all of the grid data in the system. We will be removing all of these and adding two new ones. See the Grid management document to get an idea of what the existing procedures do.



1. Insert all custom grid options: This will be removed.
2. Insert Customer Grids From Template Map: This will be removed.
3. Insert Customer Grid Template: This will be removed.
4. Insert Customer Grid Template Col: This will be removed.
5. Insert Customer Grid Template Mapping: This will be removed.
6. Insert Customer Grid Template Mapping Defaults: This will be removed.
7. Refresh all customer grids: This will be removed

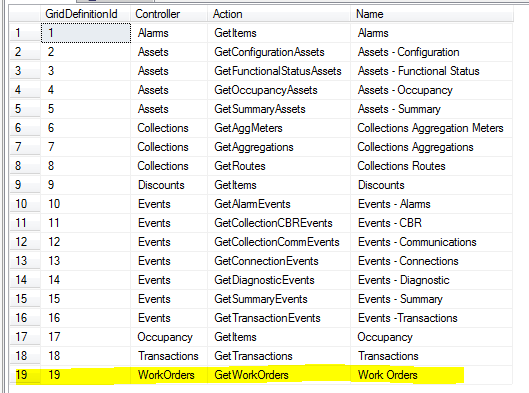
This update will be adding the following Stored Procedures:

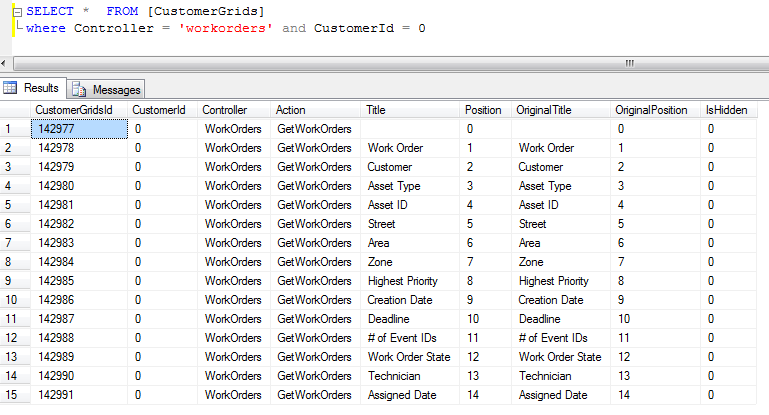
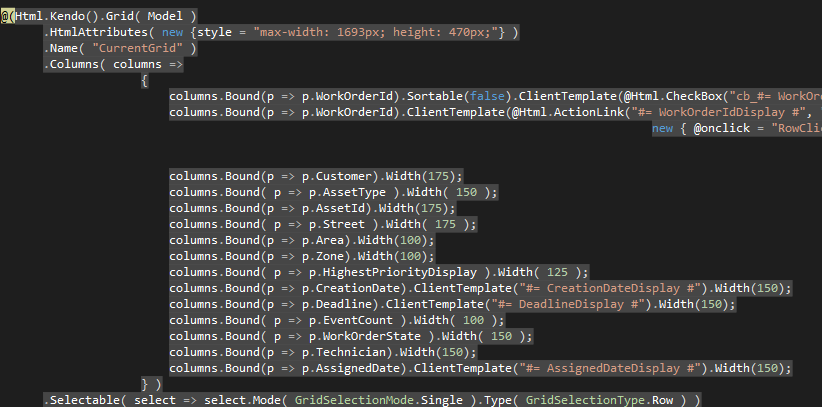
1. **Refresh Customer Grids**: This procedure will roll through all of the customers in the system, then roll through all of the Customer Grid data with a customerId = 0 and insert it for the customer. This allows the application to insert a set of default data for a customer, and also backfill any missing customer specific data. This procedure has been provided.
2. **Reset Customer Grid**: This procedure will accept three parameters (Customer ID, Controller, Action) and will remove all of the customer grid entries for the specified Customer’s Controller / Action values of the associated Grid Definition’s Controller / Action, then re-insert the values of the customerId = 0. This will allow the application to reset a specific customer grid to a default state. The UI will give users to “Reset” a customer’s grid options, and this is the procedure that will do it. This procedure has been provided
3. **UpdateCustomerGridEntry**: This procedure will update a specific grid entry for a customer and will accept the following parameters. This procedure will be invoked when the user is on the custom grids tab and is modifying a specific grid. For each row of data in that grid, the application will call this procedure and update the appropriate data. This procedure has been provided
   1. CustomerGridId – the ID of the unique entry in the customer grids table
   2. Title – the title to set for this entry
   3. Position – the position to set for this entry
   4. IsHidden – bit flag denoting if the entry is hidden or not

## Adding Grids

When adding new grids to the system, the following steps must be taken to ensure all of the grid options are available for all the customers in the system:

1. Add a new entry to the Grid Definition Table



1. Add the Customer ID = 0 entries for the associated grid in the Customer Grids table
   1. This will be the default values that are inserted for a customer upon creation as well as when the customer grid is “reset” via the application:
   2. 
   3. The order and position of these items will directly correspond to the order in which they were put onto the MVC View that renders this information to the user:
   4. 
   5. This is replacing the current functionality provided by the Customer Grid Template Col table.
2. Run the new “Refresh Customer Grids” stored procedure. This will insert all of these values for the new grid into each customer.
3. Now an administrator user can configure the grid columns, titles, and visibility for the new grid for a particular customer via the “Customer Grids” tab (Explained below).
4. Any entity creation, MVC file creation, stored procs and views for this grid, etc. are explained in the grid management and kendo grid documents.
5. NOTE: Any value that is associated with a customer id = 0, the title and original title should be the same, as well as the original position and position columns.

## Adding Grid Columns

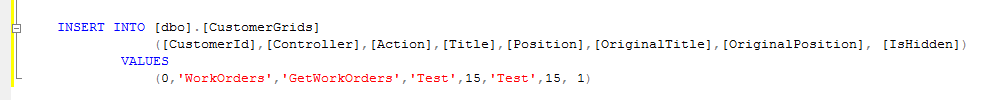
The method for adding grid columns to the system will change as well.

## Steps

These are the steps that will need to be taken in order to maintain data consistency and successfully add a column to an existing grid.

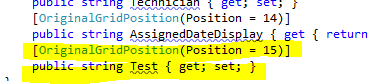
### Update Customer Grid “Customer Id = 0”

Insert the new column in the Customer Grids table with a customer ID of 0. The rules for adding columns are as follows:

1. When adding a column to an existing grid, it is important to note that the customer may have updated the ordering, visibility, and titles of the existing columns for that grid. This means that when inserting a new column for a grid, we will
   1. Add the entry into the customer grids table with a customer id of 0
   2. The hidden value MUST be set to 1. This will account for any existing hidden columns. Since the hidden columns must be last, by adding a column as the last one and NOT setting this to hidden will confuse the client side displaying and hiding of these columns and the application will not work correctly.
   3. The entry should be the last one for that grid. For our example, we will add a column named “Test” to the work order grid. Take the highest position and add one to it, and this will be the value used as the position and the original position.
      1. Since our highest position in work orders is 14, we will insert the value of 15 as position and original position
   4. 
   5. Now our default for work orders will be
   6. 

### Update the Entity

The class needs to be updated that represents the Grid data on the Index page. A Display version of this field will be added in order to correctly format the content on the site. When adding a field, it is important to make sure the OriginalGridPosition attribute of the property correctly lines up with where we add it to all versions in the MasterGridList. In this example, we will be adding the new field, and updating “Test”’s OriginalGridPosition to be 15, in order to match the changes made above.



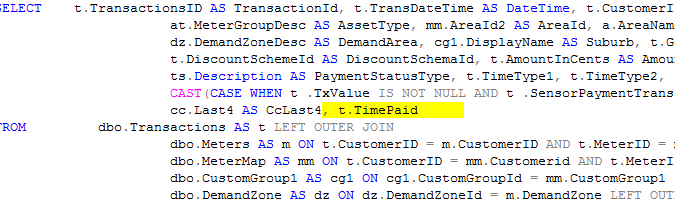
### Update the MVC View

The Kendo grid will need to be updated to display the value to the user:



### Update DB View (if applicable)

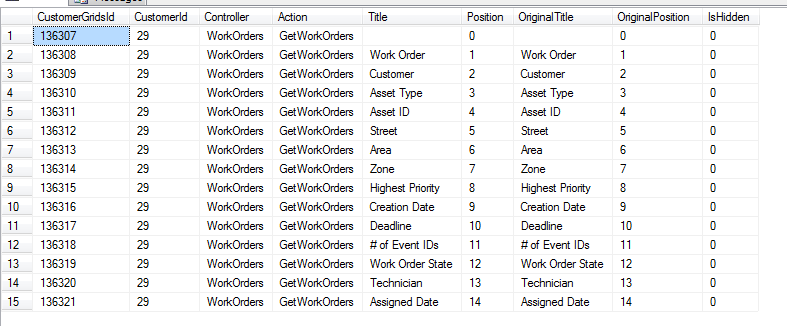
The data access method must be updated to populate our new property correctly (Test). If your example uses the stored procedure / view method, this will need to be updated to return the data requested. In this case t.TimePaid was added. If the data access is using the simple grid, then make sure to populate our new property in the respective Make method.



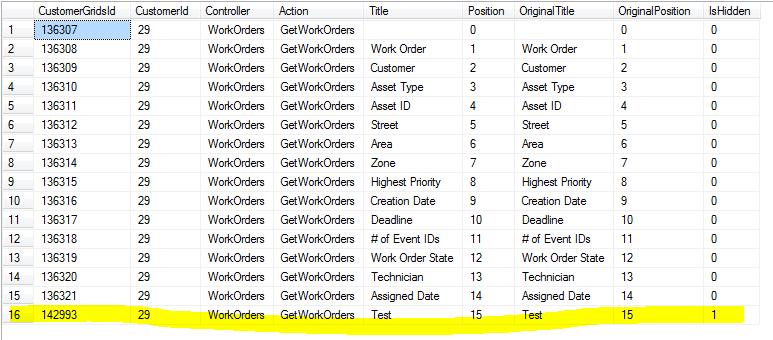
### Update Customer Grid Options

Now that the grids and their associated data access are updated, the data within the Customer Grids table needs to be updated to reflect the new column. This is done by running the newly created **Refresh Customer Grids** stored procedure within the RBAC DB. This will update each customers Customer Grid data to include the new field in the appropriate location for display on the site.

Here is a snapshot of the Work Order data for customer 29 before running the Refresh Customer Grids:



After:



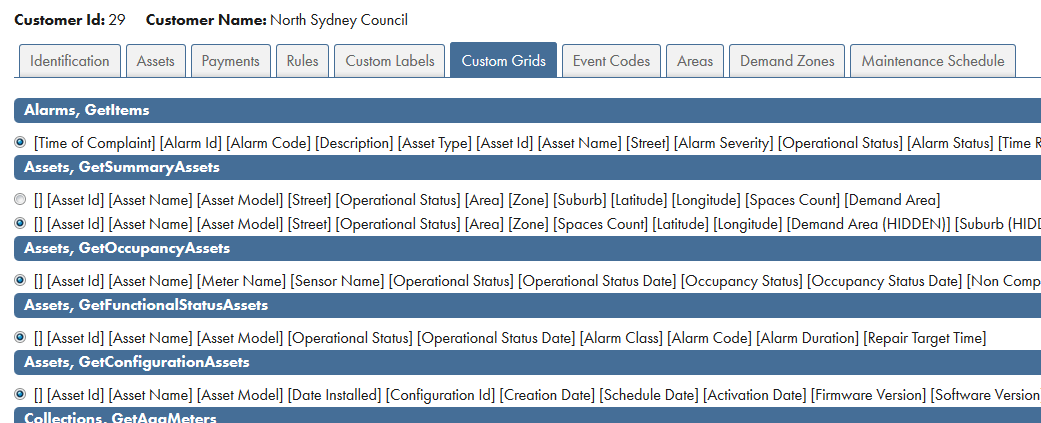
### Testing

The final step is to verify the data is coming back correctly and is displaying in the grid.

# UI Update

## Current

The current customer grid tab looks like:

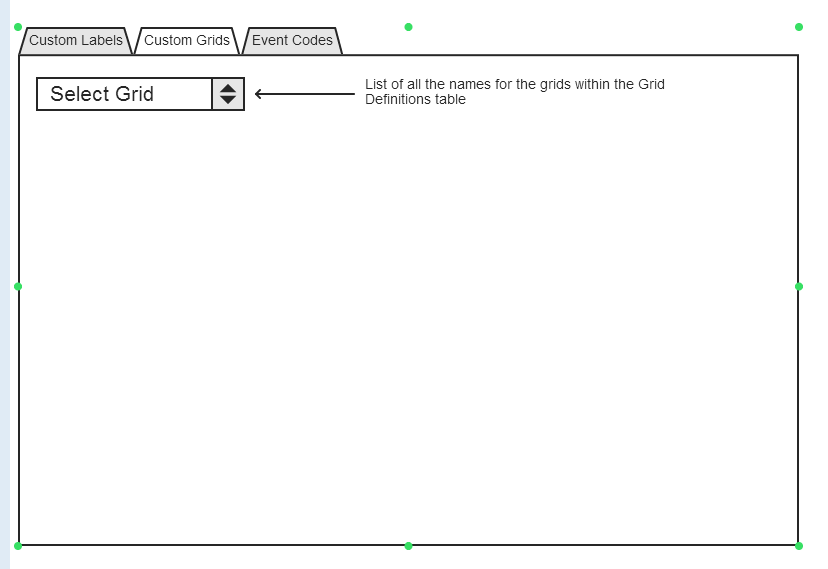


One of the goals of this update is to allow users to modify the layout and visibility of grids on a per customer, per grid basis. Since there will no longer be multiple versions of each grid, the UI for this tab will have to be updated to reflect the new functionality.

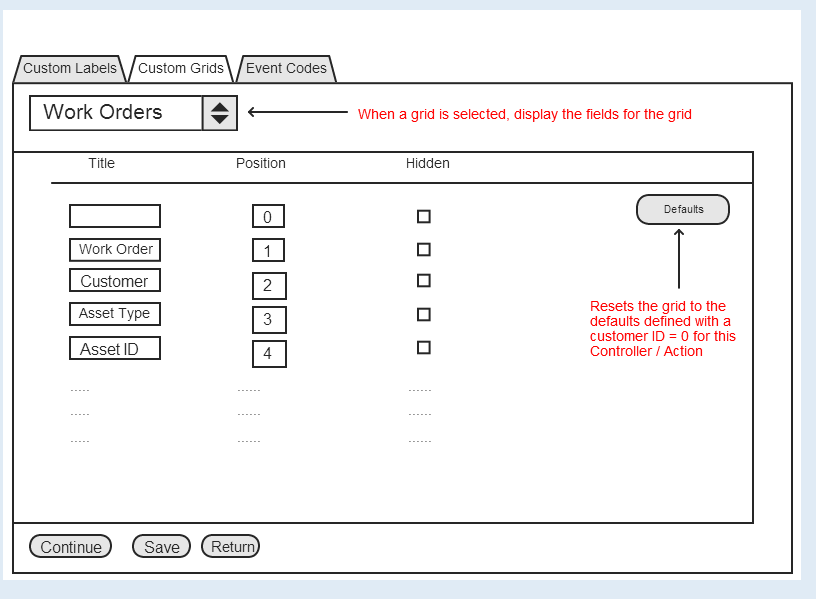
## Updates – Duncan

The custom grids tab will be completely revamped. These mockups are just guidelines on how the page should act and are NOT requirements, just suggestions. They can be changed or modified if needed.

When the user first navigates to this page, they will be given the option to select the grid they wish to modify:



Once they have selected the grid, the page will display a list of all of the fields in the system for that customer for that grid. This page will allow the user to edit the Title, Position, and Hidden attributes for a custom grid entry. Everything else will be used for sorting the data in the backend, but not displayed to the user (original title, original position).



### Actions

#### Loading the data

When loading the data for a grid, these rules must be followed:

1. The fields will be displayed and sorted by the Original Title
2. The model that is built for this page must contain the “Customer Grid Id” value for each row entry, as this will be needed when updating the data on save.

#### Validating the data

There are a couple of validation rules that need to be checked before allowing the user to save the data:

1. All of the positions must be valid and unique. If the Positions are 0,1,2,3,4,5 and 6, then whenever the user is finished, the positions must contain these values. They can be in a different order (0,1,6,4,5,3,2), but there MUST NOT be duplicates OR missing numbers
   1. (0, 1,6,4,5,5,2) – This has two 5’s and is missing the 3, so this is invalid on both counts.
2. The hidden fields MUST be only assignable to the highest (last) position. The way the grids interact with the customer grid table requires all of the hidden fields to be the very last columns after they are ordered. The reason for this is because the client side grid logic gets the grid data, re-titles all of the columns, then re-orders them, then counts how many columns are hidden and hides that number starting from the last column of the grid. So, if 3 columns are hidden, NO MATTER WHAT the last three columns will be hidden, even if those are not the intended hidden columns.
   1. The user can specify as many hidden fields as they wish, but they have to be the last N number.
   2. If a user sets a field as hidden:
      1. Logic will need to run to determine if it is a valid hidden field. Meaning if it is the last position, or the last non-hidden position (entries that have higher positions that are also hidden are acceptable)
      2. If the user sets a field as hidden, then changes the position of the field, validation logic will have to determine if that field should stay hidden

#### Save

This button will save the data to the custom grids table. The original position and original title will NOT be changed, just the Title, position, and Hidden fields. This will require the developer to re-write the data access for this page to determine what the user has selected, all of the custom validation, etc. Once the application knows what data the user wants to save, then the “Update Customer Grid Entry” stored procedure will be called for each row of data for this grid. All of the requirements and rules for the customer grid table must be respected, as this is what the index pages throughout the application will use.

#### Defaults

This button will reset the current grid values to the default. This will be done by calling the stored procedure “Reset Customer Grid” described above. The application will need to pass the current customer id and the controller / action of the selected item in the drop down list (Grid Definition). The procedure will then refresh the current grid with the data defined for the same grid and the customerId = 0. Once this is complete, the application will need to send the user back to this page so they can see the refreshed version of the data for this grid.

# Additional work

Besides full regression testing (Duncan) to make sure the application is respecting the customer grid data correctly, there are also a few tasks that need to be completed. The customer creation process will need to me modified since the original stored procs that populate the customer grid data are going to be removed. The following is the result of a quick search of the existing stored procedures and some items that will need to be updated to accommodate the new customer grid data structure

1. Customer Factory
   1. Create new customer – **Unidev : This has been complete**
      1. This method uses a few stored procedures to populate the customer grid data, so this will need to be removed and instead the “Refresh Customer Grids” stored procedure needs to be called. This will insert the newly created customer (id) into the customer grid table with the default data.
   2. Set Customer Grids Model – Duncan – this will be part of Section 4
      1. This method saves the data when the user hits save on the custom grids tab. This will need to be re-worked to correctly save the user defined grid data to the customer grids table.
2. Data Context Update – **Unidev: This has been complete**
   1. The data context will need to be updated to reflect the new table structure