|  |
| --- |
| Ultimate Software |
| Using Platform Configurable Fields in Interfaces |
| Interface Team |

|  |
| --- |
| Last Updated:  6-21-2017 |

Contents

[Purpose 3](#_Toc485829646)

[Platform Configurable Fields (PCF) 3](#_Toc485829647)

[Functional Guide 3](#_Toc485829648)

[PCF Table-Value Functions 3](#_Toc485829649)

[Example: PCF Table-Value Function & EmpComp 4](#_Toc485829650)

[Deeper Dive: What are the underlying system tables accessed by the Table-Value functions? 5](#_Toc485829651)

[PCF Audit View 7](#_Toc485829652)

[Example: PCF Audit View (*vw\_dsi\_CfgAuditData)* 7](#_Toc485829653)

[Example: Using the PCF Audit View in a Changes-Only Export 8](#_Toc485829654)

[PCFs Imports 10](#_Toc485829655)

[Example: PCF Imports 10](#_Toc485829656)

[Sample SPs & Specs 14](#_Toc485829657)

[Applications in Interfaces 15](#_Toc485829658)

[Benefit Plan/Group Mappings 15](#_Toc485829659)

[Example 15](#_Toc485829660)

[FAQ 17](#_Toc485829661)

[How can I get a list of all the Configurable Fields currently setup in a customer’s database? 17](#_Toc485829662)

[The Platform Configuration Field I am working with is setup as a dropdown selection of descriptions, but the corresponding codes are being stored in SQL. How do I map these codes to the descriptions that display on the web? 17](#_Toc485829663)

[I can see a Platform Configurable Field setup on the web, but I do not see the field in SQL via the ***fn\_MP\_CustomFields\_[TableName]\_Export*** function. Where is it at?? 17](#_Toc485829664)

# Purpose

The purpose of this document is to provide the interface team with documentation of Platform Configurable Fields (PCF) and how they are used for the purposes of interfaces.

# Platform Configurable Fields (PCF)

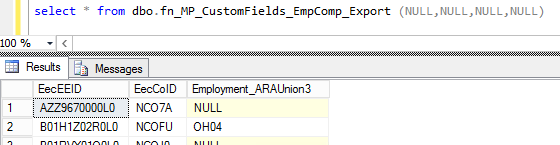
## Functional Guide

For detailed documentation re: Platform Configurability from a functional perspective, please see the documentation at <http://documentation.ultimatesoftware.com/Onlineguides/PlatformConfig/PlatformConfig-Guide.pdf>

# PCF Table-Value Functions

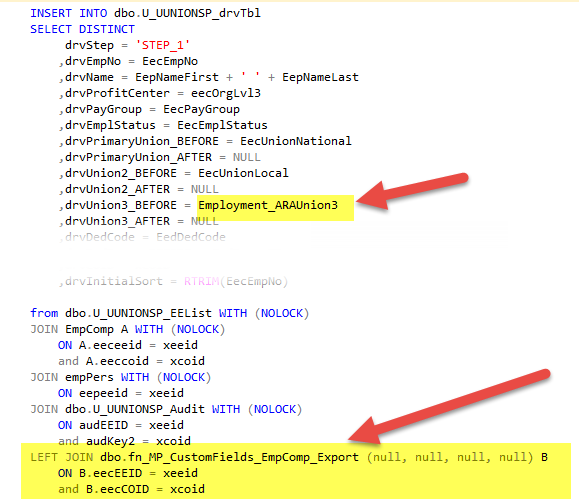
Table-Value functions have been provided by the Development Team for the purpose of writing SQL queries based on PCF data. You will find a Table-Value function for each Class in which there is a PCF setup.

* The Table-Value Functions for PCFs use the naming convention: ***dbo.fn\_MP\_CustomFields\_[TableName]\_Export (NULL,NULL,NULL,NULL)***.
* For example: PCF data in the Employment class would be accessible using ***dbo.fn\_MP\_CustomFields\_EmpComp\_Export (NULL,NULL,NULL,NULL).***



### Example: PCF Table-Value Function & EmpComp

In the example below, the function ***dbo.fn\_MP\_CustomFields\_EmpComp\_Export*** is being joined using EecEEID & EecCOID.



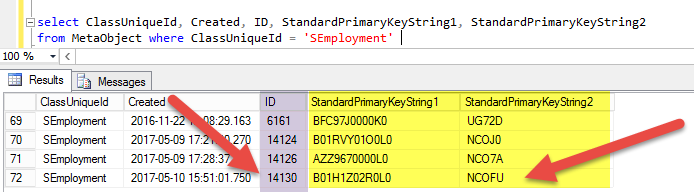
### Deeper Dive: What are the underlying system tables accessed by the Table-Value functions?

For the majority of export interfaces, accessing the data via the Table-Value functions will be sufficient. However, if there is a need to dive deeper for the purposes of further research/troubleshooting, the ***MetaObject*** & ***MetaFieldValue*** tables can be reviewed. These tables contain the data for each PCF.

#### **MetaObject**

The ***MetaObject*** table contains the primary key fields that can be used to identify the object to which the PCF data belongs (e.g. which Employee, Contact, OrgLevel, Location, etc.)

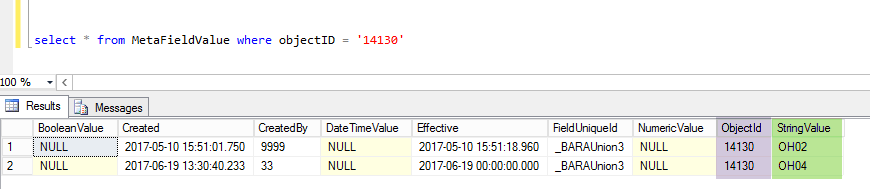
* PCF Class names have the prefix “**S**” when in the ***MetaObject*** table.
  + For example, the Employment class has a ClassUniqueID = **SEmployment**
* There is 1 record in the ***MetaObject***for each object within the class.
  + For example, for the Employment Class, there is 1 record in ***MetaObject*** for each EEID (*StandardPrimaryKeyString1*)/COID (*StandardPrimaryKeyString2*) combination.
* In the example below, the **14130** is the ObjectID that will help us identify the PCF values in the Employment class for EEID/COID shown in Row 72.
  + This *ID* field will be found in the ***MetaFieldValue*** table to view the actual data that has been stored in this PCF object.



#### **MetaFieldValue**

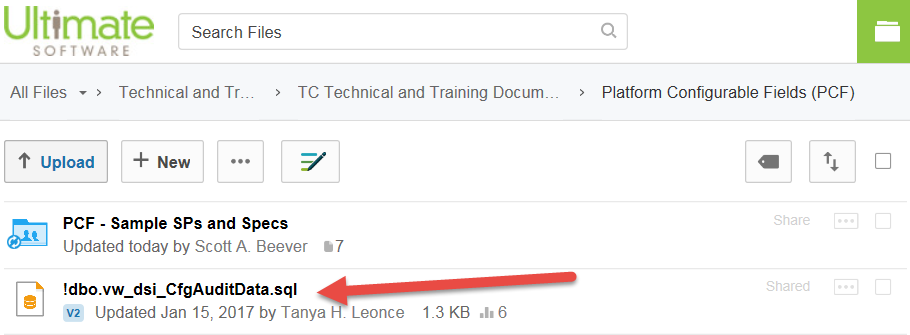
The ***MetaFieldValue*** table contains the PCF values for each ObjectID, by Field Name & Effective Date. Since the data is stored effective date, a history of values for each object’s PCFs can be accessed from this table.

* PCF Field Names have the prefix “**\_B**” in the ***MetaFieldValue*** table.[[1]](#footnote-1)
  + The example in the screenshot below shows all of the PCF’s for ObjectID = **14130**.
* In this example, **ARAUnion3** is the only PCF setup in the Employment Class. If there were other PCF’s in the Employment class for this employee, they would also be visible when reviewing the ***MetaFieldValue*** table.
* The ***MetaFieldValue*** table will display a history of values for each ObjectID, by Field Name & Effective Date.
  + In the example below, the current value of ARAUnion3 = **OH04** for this employee, effective 6/19/2017. Previously, ARAUnion3 = **OH02**.
* The PCF data will be stored in one of the following fields in the ***MetaFieldValue*** table, depending on the data type definition for the PCF at the time the data was populated:
  + BooleanValue (0 or 1)
  + DateTimeValue
  + NumericValue
  + StringValue
    - In the example below, the **ARAUnion3** field data is defined as a String Value.



# PCF Audit View

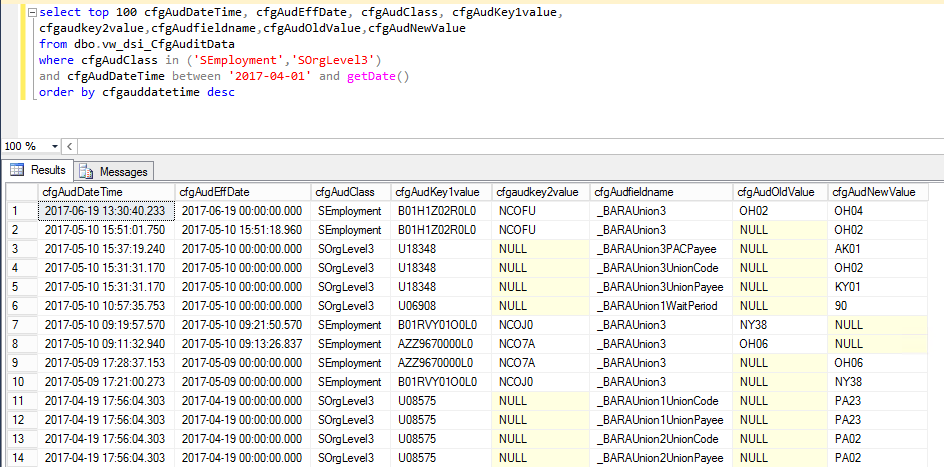
A view was been developed (*vw\_dsi\_CfgAuditData*) to assist with viewing changes to data in each PCF. The view can be retrieved from the PCF folder in Box & deployed to the customer’s environment as needed (<https://ulti.box.com/s/expib2vyrdr6h7fyottnnon66zbz84ld>).



* Once deployed to the customer’s environment, this audit view can be used in a similar manner as the standard UltiPro audit view.
* Remember that the Class names in the PCF audit view will have the prefix “**S**” & the field names will have the prefix “**\_B**”.

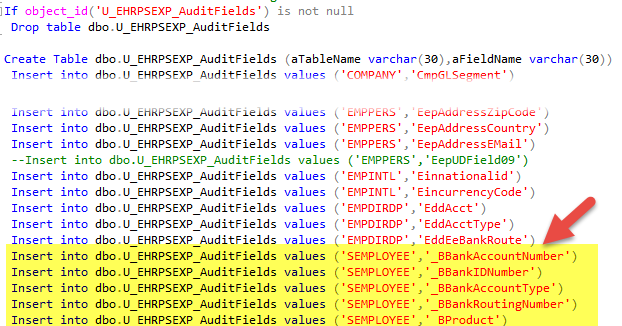
## Example: PCF Audit View (*vw\_dsi\_CfgAuditData)*

This example query displays the recent audit changes for PCF’s in the Employment & OrgLevel3 classes.

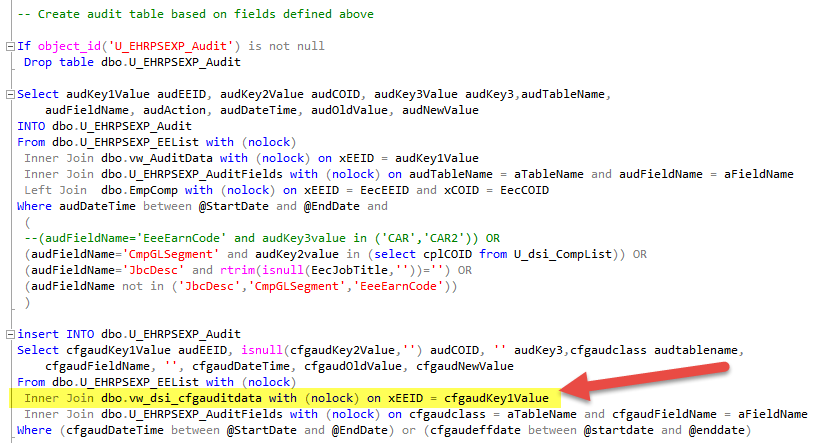


## Example: Using the PCF Audit View in a Changes-Only Export

1. Insert the fields you wish to include in the audit selection into an “Audit Fields” table, as you would with UltiPro Core fields.
   * Remember that the Class names in the PCF audit view will have the prefix “**S**” & the field names will have the prefix “**\_B**”.



1. When creating the audit table in the export, ensure you are joining on *vw\_dsi\_CfgAuditData* for the PCF’s, rather than the standard UltiPro Core audit view.
   * The example below is gathering audit change both from the UltiPro Core audit view and the PCF audit view.



# PCF Imports

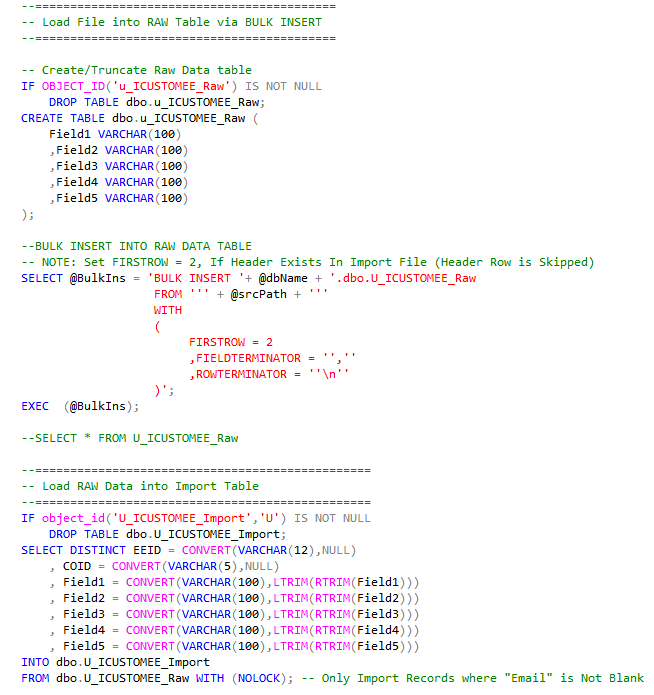
Importing data into PCFs can be accomplished by inserting records directly into the ***MetaObject*** & ***MetaFieldValue*** tables.

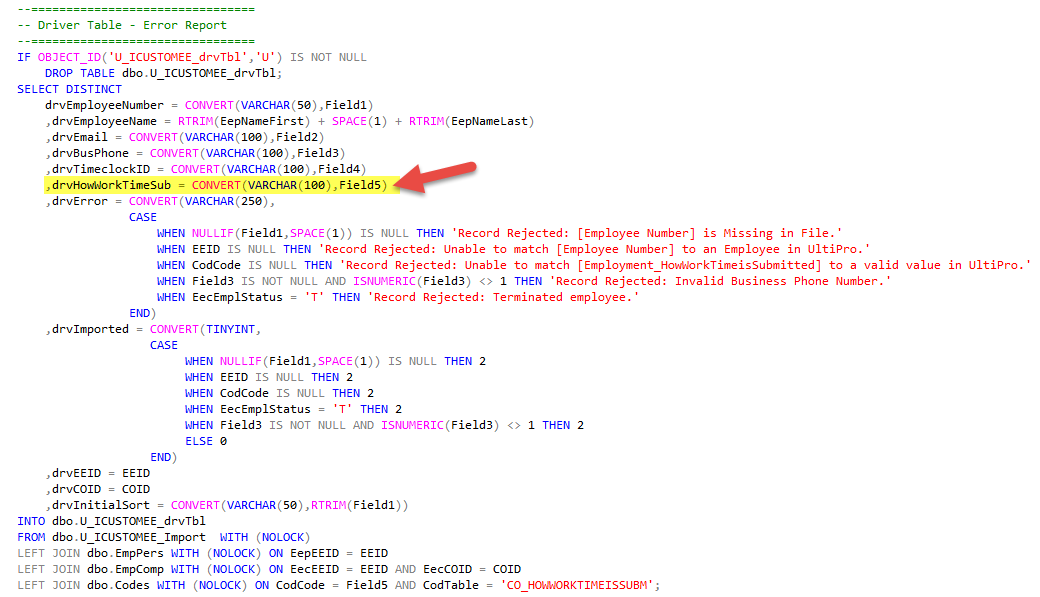
## Example: PCF Imports

In the example below, data is being imported into the **HowWorkTimeIsSubmitted** field in the **Employment** class.

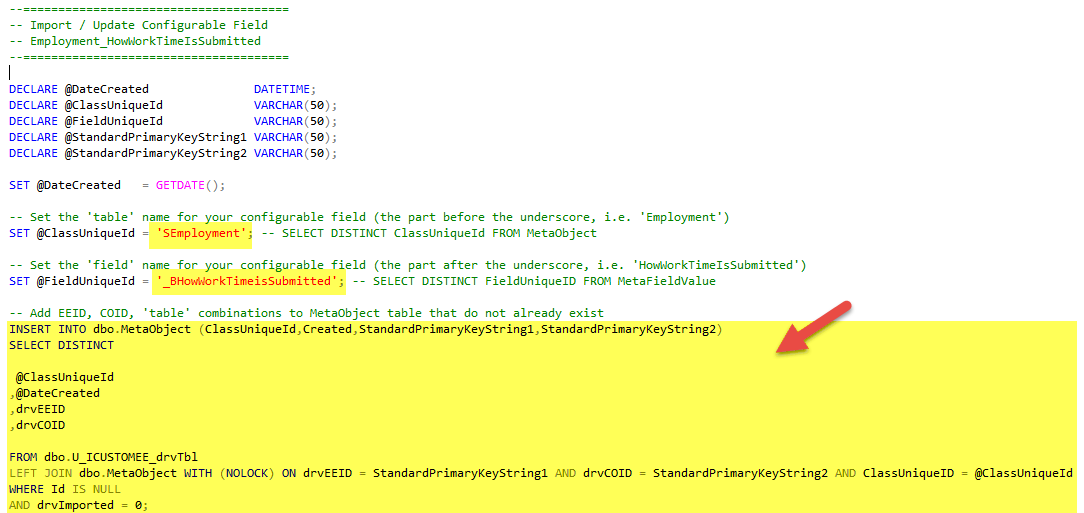
1. Similar to an standard SQL import:

* Load & parse the import file using Bulk Insert.
* Use the driver table to perform any translations/error checking/validations necessary, based on the import requirements.

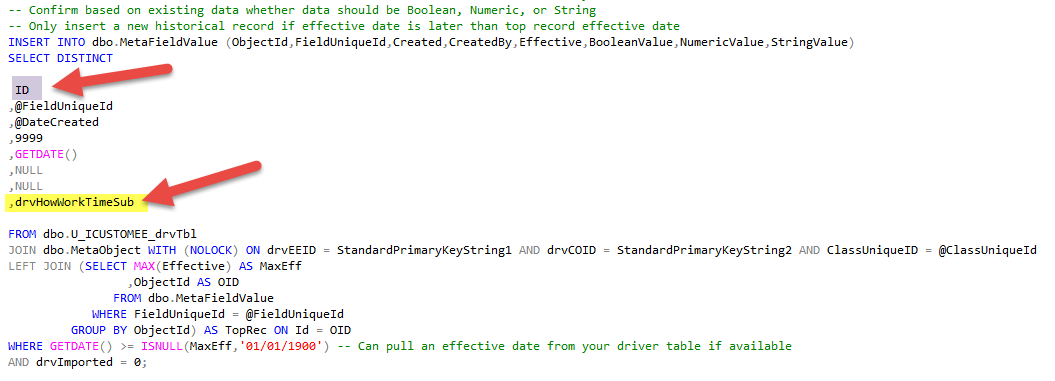




1. Insert records into the **MetaObject** table for objects do not already exist in this table.
   * Remember that the Class names will need to have the prefix “**S**” & the field names will have the prefix “**\_B**”.
   * In this example, **MetaObject** records are being inserted for records where the EEID/COID combination does not already exist in the **Employment** class.



1. Insert records into **MetaFieldValue** table for the PCF field that is being updated.
   * There is a new record in **MetaFieldValue** for the PCF each time it is changed/updated, so we will always be inserting rows into **MetaFieldValue** with a new effective date, not updating existing rows.
   * The data must be inserted into the correct field in **MetaFieldValue**, based on the data type: *BooleanValue*, *NumericValue*, *StringValue*, *DateTimeValue* (not shown in the screenshot below since it was not needed).
     + For the example below, the field drvHowWorkTimeSub is being stored in the *StringValue* field.
   * This insert is populating the *ID* from the **MetaObject** table into each **MetaFieldValue** record. This *ObjectID* would allow us associate this record with the employee’s EEID/COID.

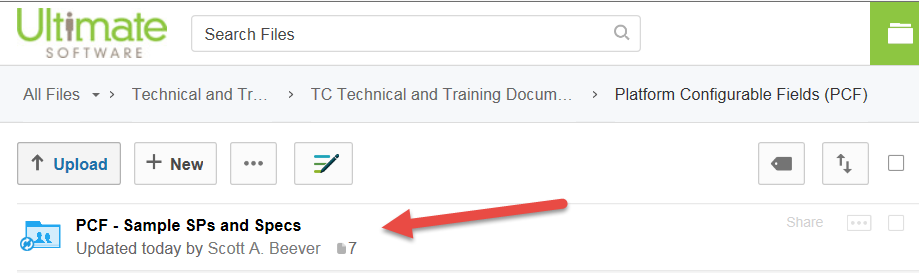


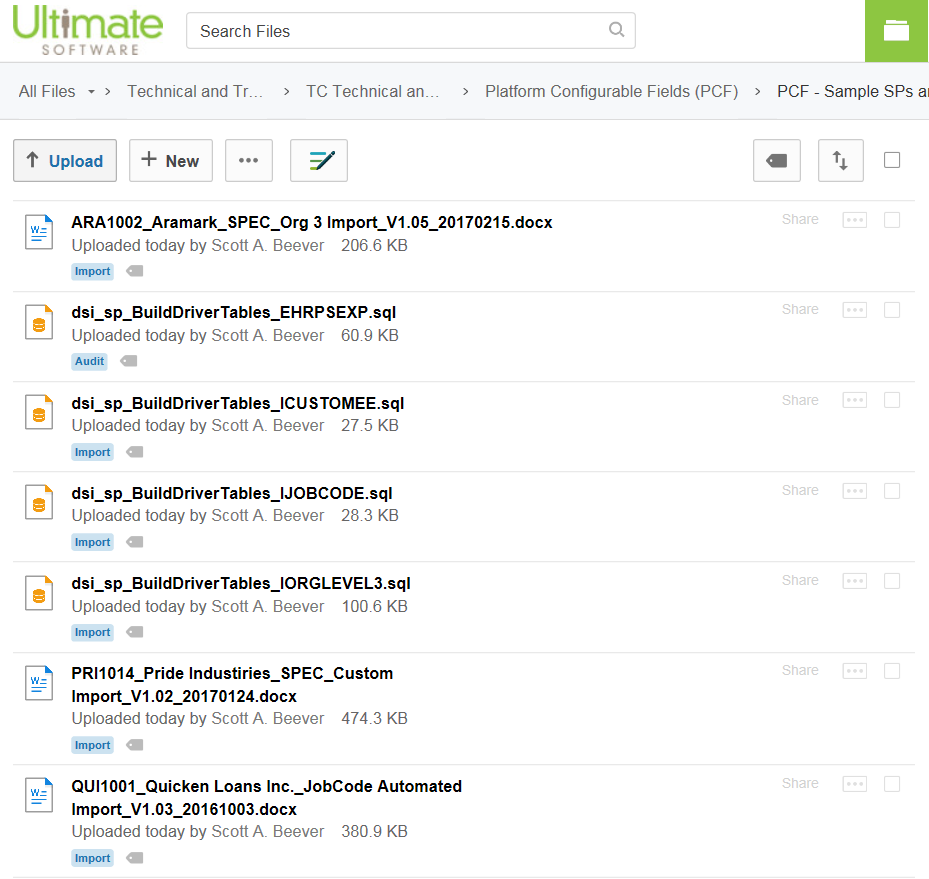
1. For inserts into the **Employment** Class only:
   * There is an **EmploymentHistory** class that goes along with the **Employment** Class.
     + **Employment** is similar to **EmpComp**
     + **EmploymentHistory** is similar to **EmpHJob**
   * As such, we need to update the **EmploymentHistory** PCF when the **Employment** PCF is updated. Luckily, there is a SP in the product (sp\_usg\_CopyMetaFieldValuesToEmploymentHistory) that will perform these updates for us.
     + We just need to loop through the EEID/COID’s that were updated and call this function in order to have **EmploymentHistory** updated for these employees.
     + The example below shows the cursor that was declared to loop through & update **EmploymentHistory** for the employees in the driver table.
   * This update is needed only for the **Employment** Class. This would not be needed for updates to other classes that do not have “History” tables (e.g. JobCode, OrgLevel, Location, …)



# Sample SPs & Specs

Example SPs & specs have been compiled under the **Platform Configurable Fields** 🡪 **PCF – Sample SPs and Specs** folder in Box: <https://ulti.box.com/s/expib2vyrdr6h7fyottnnon66zbz84ld>





# Applications in Interfaces

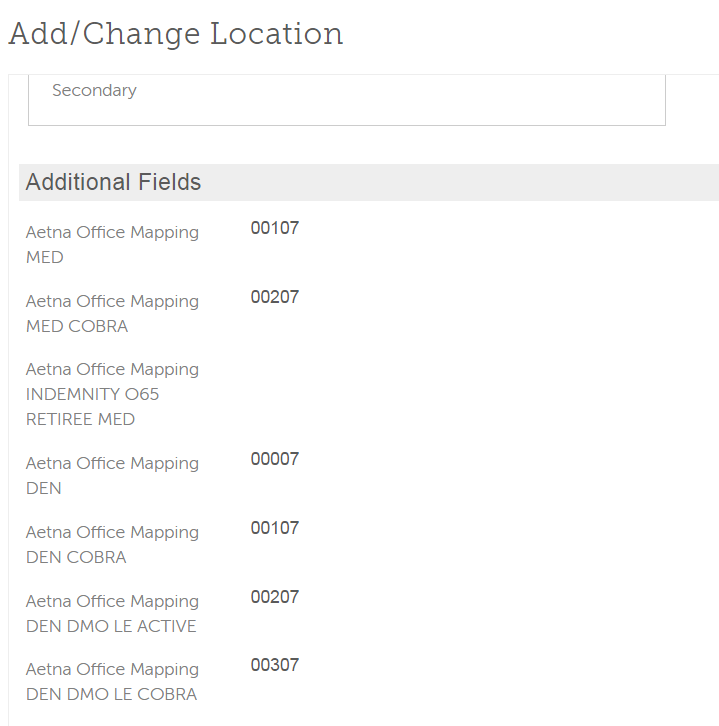
## Benefit Plan/Group Mappings

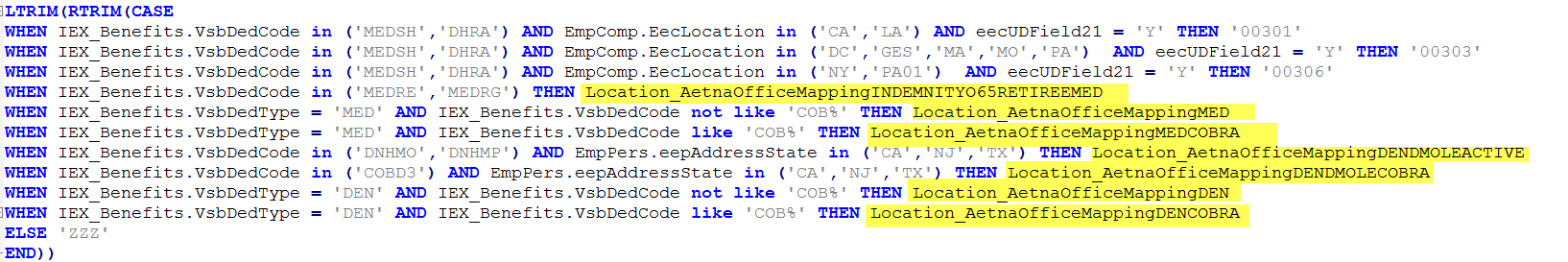
Often Benefit Plan/Group # mappings for a particular vendor are based upon a structure that relates back to the customer’s Location Codes, Org Levels, Component Companies, etc. If so, it may be worth investigating PCFs as a solution where the customer could maintain the mappings themselves (rather than needing an SR for simple changes).

### Example

In this example, the customer’s plan mappings were based (in part) on the employee’s location code.

* They decided to add a series of PCFs in the **Location** class, so that these can easily be updated by the end user when a new Location is added or needed to be updated.
* The interface programming is then looking to these PCF values, rather than hardcoding these mappings
  + This will save the customer from opening a SR each time there is an update to the Location code mappings.





* CAUTIONS/Limitations:
  + Currently, it is not possible to setup PCFs that would be associated with the **DedCode** table itself. If/when that functionality is added to PCFs, this would add extensive additional flexibility when working with Benefit Plan mappings.
  + While PCFs could be added in the **Employee** or **Employment** classes for Benefit Plan mappings, it is probably **not** best practice to do so:
    - Maintaining the PCF benefit plan mapping on the employee level would require quite a bit of overhead maintenance by the customer.
    - Managing these PCF benefit plan mappings on the employee level would likely be a mess during OE!

# FAQ

## How can I get a list of all the Configurable Fields currently setup in a customer’s database?

The following query can assist with this research:

select distinct classuniqueID, fielduniqueId

FROM dbo.MetaFieldValue

    JOIN dbo.MetaObject

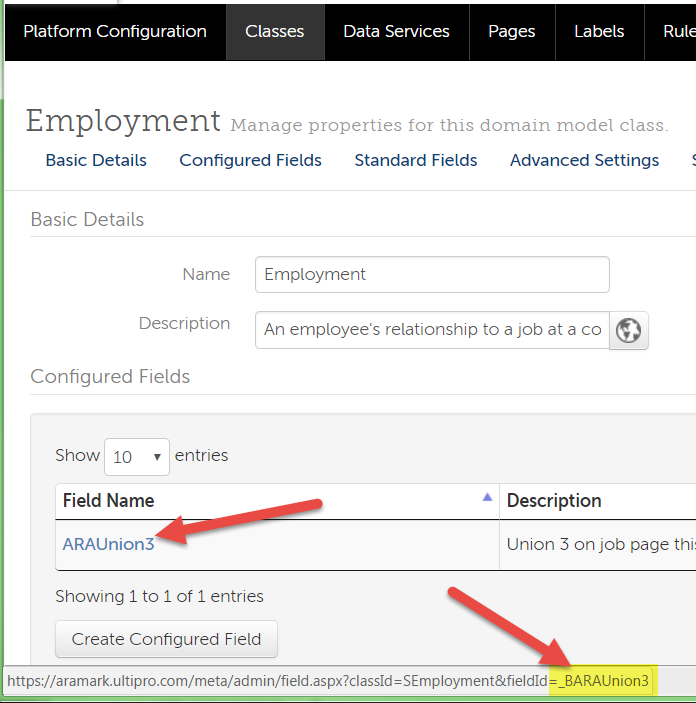
        ON ObjectID = ID

## The Platform Configuration Field I am working with is setup as a dropdown selection of descriptions, but the corresponding codes are being stored in SQL. How do I map these codes to the descriptions that display on the web?

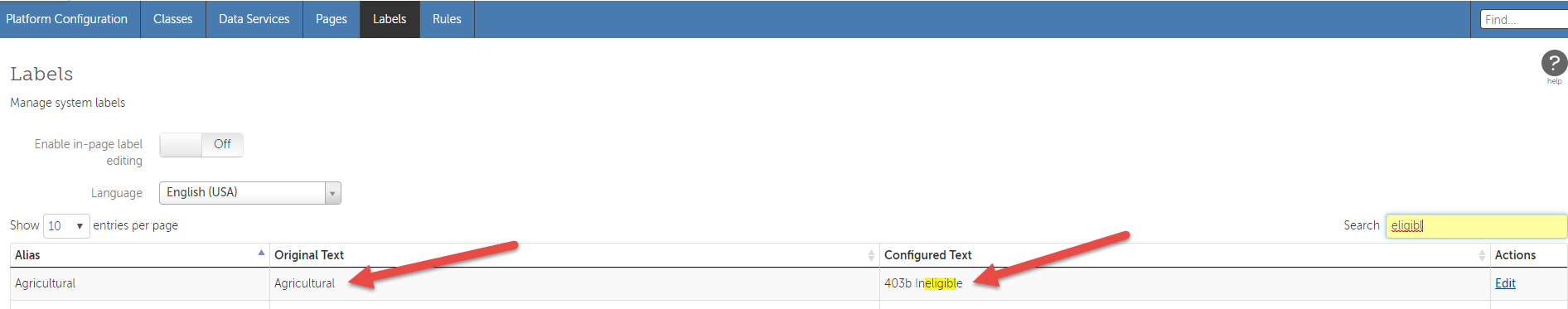
If you need the Platform Configuration Field codes to their descriptions, Tanya put together an UltiHome article on this topic with screenshots: <https://ultihome.ultimatesoftware.com/spaces/419/Article/cc765e03-98cc-4a31-a418-c572e45c33e2/35821>

## I can see a Platform Configurable Field setup on the web, but I do not see the field in SQL via the ***fn\_MP\_CustomFields\_[TableName]\_Export*** function. Where is it at??

1. It is possible that the setup for the Configurable Field was changed after it was originally created. If a configurable field was renamed on the web, it will NOT rename the field in SQL.
   * This is purposefully done so that any existing code that leverages the configurable field does not break if it is changed on the web.
   * In order to change the name on the back end, a new configurable field would need to be created.
   * In order to confirm the SQL name, you can hover over the Field Name in the Configurable Field Classes setup, as shown in the screenshot below. The URL that displays at the bottom of the screen will provide information about the SQL name that is associated with this Configurable Field.



1. Although not considered best practice, it is also possible that a standard field was “repurposed” in UltiPro by changing the label that displays on the web with Platform Configuration. In the example shown below, the label ***Agricultural*** was changed to ***403b Ineligible*** by the customer. In this example, the data will still be stored in **EecIsAgricultural**, but the label on the web would display this field with the label that was configured in the screen below.
   * A better practice would be to establish a new Configurable Field for this purpose, but this example is provided since it has been seen previously.



1. If the Platform Configurable Field still cannot be located via the ***fn\_MP\_CustomFields\_[TableName]\_Export*** function, the function for that class may need to be updated. Based on the following article, there have previously been issues where configurable fields have been imported, but the functions were not updated.

*Workaround: Go to the affected classes and add a dummy field (then go back and delete it). This will force the function for that class to be updated.*

<https://ultimatesoftware.my.salesforce.com/articles/Known_Issue/Platform-Configuration-Customer-has-created-configurable-fields-but-the-database-functions-e-g-fn-MP-CustomFields-EmpPers-Export-are-not-reflecting-these-fields>

1. You may also find records in the MetaFieldValue table with the prefix “**\_F**”. These are not technically PCFs, although they use the same data structure as the PCFs. These are “standard” fields are related to Global payroll functionality in UltiPro. [↑](#footnote-ref-1)